Adaptive Backpressure: Efficient Buffer Management for On-Chip Networks

Daniel U. Becker, Nan Jiang, George Michelogiannakis and William J. Dally
Department of Electrical Engineering, Stanford University, Stanford, CA, USA
{dub,qtedq,mihelog,dally}@cva.stanford.edu

***Abstract*—This paper introduces Adaptive Backpressure, a novel scheme that improves the utilization of dynamically managed router input buffers by continuously adjusting the stiffness of the flow control feedback loop in response to observed traffic conditions. Through a simple extension to the router's flow control mechanism, the proposed scheme heuristically limits the number of credits available to individual virtual channels based on estimated downstream congestion, aiming to minimize the amount of buffer space that is occupied unproductively. This leads to more efficient distribution of buffer space and improves isolation between multiple concurrently executing workloads with differing performance characteristics.**

Experimental results for a 64-node mesh network show that Adaptive Backpressure improves network stability, leading to an average 2.6× increase in throughput under heavy load across traffic patterns. In the presence of background traffic, the proposed scheme reduces zero-load latency by an average of 31 %. Finally, it mitigates the performance degradation encountered when latency- and throughput-optimized execution cores contend for network resources in a heterogeneous chip multi-processor; across a set of PARSEC benchmarks, we observe an average reduction in execution time of 34 %.

I. Introduction

Moore's Law continues to drive forward integration levels in the semiconductor industry, providing chip designers with ever increasing transistor budgets. As improvements to single-threaded performance have become limited by power constraints, this has caused industry focus to shift towards design approaches that scale performance by leveraging chip-level parallelism. With future microprocessors expected to integrate hundreds of execution cores on a single die, on-chip communication will have a significant impact on chip-level performance and power efficiency [1], [2]. Networks-on-Chip (NoCs) are widely considered to be a promising approach for addressing the scalability and complexity challenges inherent in such designs [3].

Current NoC research largely utilizes input-queued routers; in such designs, packets that arrive at a router's input and cannot be forwarded immediately are temporarily held in First-In, First-Out (FIFO) buffers until they can proceed to the next hop. Buffer space is typically allocated at the granularity of fixed-size units called *flits*—one or more of which comprise a *packet*—and is logically divided into multiple Virtual Channels (VCs) in order to avoid deadlock and to reduce Head-of-Line (HoL) blocking [4]. Proper buffer sizing and organization are essential to achieving optimal network performance [5]–[7]. At the same time, input buffers account for a large fraction of the overall area and power budget of typical NoC routers [8], [9]. Consequently, buffer resources must be utilized efficiently in order to achieve good cost-performance trade-offs. To this end, prior research has proposed dynamic buffer management schemes in which a pool of buffer slots is shared between VCs [7], [10].

In the present contribution, we first show that sharing buffer space among VCs without further restrictions can lead to performance pathologies when multiple types of traffic with different performance characteristics share access to the network. Specifically, it causes VCs which experience heavy downstream congestion to monopolize buffer space at the expense of other VCs. This in turn can allow an adversarial workload to significantly degrade the performance of a concurrently executing well-behaved workload, even when both operate on strictly disjoint sets of VCs. In light of increasing Quality-of-Service (QoS) requirements and continuing industry trends towards workload consolidation and virtualization, such interference effects are highly undesirable.

To address these issues, we then propose Adaptive Backpressure (ABP), a mechanism that heuristically limits credit availability for each VC based on its observed performance characteristics. In doing so, our scheme aims to assign buffer space to those VCs that carry well-behaved traffic and to limit the amount of buffer space that is held unproductively by VCs experiencing congestion. ABP is readily implemented as a simple, low-cost extension to the existing flow control mechanism.

We present simulation results for a 64-node mesh network that show that our proposed scheme improves network stability for adversarial traffic, increasing throughput under heavy load by an average of 2.6× across six synthetic traffic patterns. We further demonstrate that ABP can effectively reduce performance coupling between two workloads competing for network resources; in particular, it improves zero-load latency in the presence of background traffic by an average of 31 %. Lastly, we investigate performance isolation in a heterogeneous Chip Multi-Processor (CMP) and show that ABP reduces the performance degradation incurred by a set of parallel application benchmarks from the PARSEC suite [11]

This work was supported in part by the National Science Foundation under Grant CCF-0702341, the National Security Agency under Contract H98230-08-C-0272-P007, and the Prof. Michael J. Flynn, P. Michael Farmwald and Robert Bosch Stanford Graduate Fellowships.

due to streaming background traffic by 34 %.

The remainder of this paper is organized as follows: In Section II, we present a brief overview of buffer management in NoC routers and show how unrestricted buffer sharing among VCs can lead to pathological performance. Section III introduces the proposed ABP mechanism. Using the experimental setup described in Section IV, we evaluate the efficacy of our scheme in Section V. Section VI briefly discusses related work, and we conclude the paper in Section VII.

II. Motivation

A. Dynamic Input Buffer Management

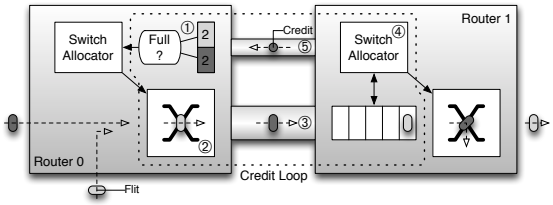
As a result of the stringent timing constraints that the NoC environment typically imposes, much prior work has opted to implement low-complexity buffer management schemes which statically divide the available buffer space among all VCs. While such schemes minimize control logic complexity, they are prone to buffer under-utilization when network load is not evenly distributed across VCs. Furthermore, in order to avoid credit stalls, each VC individually must be assigned at least enough buffer space to be able to cover the credit round-trip delay, causing buffer requirements to scale linearly with the number of VCs while further reducing the average utilization factor.

Dynamic buffer management schemes [7], [10], [12], on the other hand, organize the buffer space available at each router input port as a shared pool of slots which are dispensed to individual VCs on demand. At the cost of more complex control logic, this mitigates the adverse effects of uneven load distribution across VCs. Additionally, by allowing VCs to use slots from the shared pool to cover the credit round-trip delay, the incremental cost for each VC is substantially reduced. This enables the buffer to support more VCs than would be feasible in a statically partitioned design, which in turn reduces HoL blocking at high load [6], [7]. Overall, prior research has shown that dynamic buffer management can either improve performance by up to 25% for a given buffer size or reduce the amount of buffer space required to achieve a desired level of performance by 50% compared to a statically partitioned design [7].

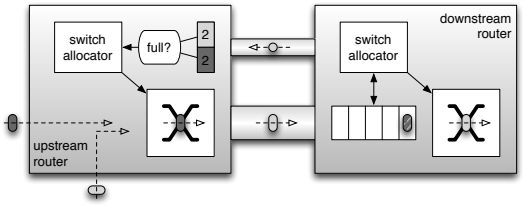
B. Performance Pathologies

While sharing buffer space among multiple VCs improves utilization, it also introduces an additional degree of coupling between VCs, as they now have to compete for both channel bandwidth and buffer space. As we will demonstrate in the remainder of this section, such coupling can lead to severe performance degradation when buffer space is shared freely among multiple types of traffic with different performance characteristics. For clarity, we illustrate this effect using a simplified example with two VCs; however, similar examples are readily constructed for configurations with any number of VCs that are divided into multiple traffic classes, e.g. for purposes of deadlock avoidance, QoS or workload isolation.

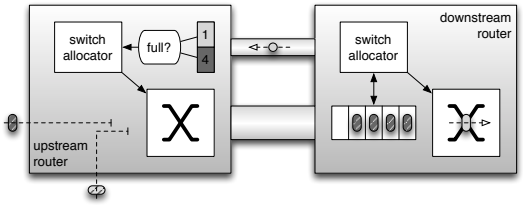
Figure 1a shows a snapshot of the steady state of flits from two different VCs, shaded in light gray and dark gray,



(a) Uncongested steady-state operation.



(b) Onset of downstream congestion.



(c) Congestion propagates upstream.

Fig. 1. Unrestricted sharing causes congestion to spread across VCs.

arriving at router 0 from the bottom and left, traversing a network channel, and leaving router 1 at the right and bottom, respectively. The dotted outline highlights the credit loop that implements flow control between the two routers: A credit is consumed when the switch allocator at router 0 generates a grant ①. In subsequent cycles, the flit traverses the upstream crossbar ② and the channel ③. It then arrives at router 1, where it is buffered until it receives a grant from the switch allocator ④. Once a grant is generated, the credit is sent back upstream ⑤, allowing the switch allocator at router 0 to assign the buffer slot to a different flit in the next cycle. Thus, in the congestion-free steady state, four credits are outstanding, and a total of at least five buffer slots is required to ensure that the upstream router never exhausts its credit supply, enabling it to forward a new flit in every cycle. Note that only one of the outstanding credits actually corresponds to an occupied downstream buffer slot, while the remainder account for in-flight flits as well as the necessary delay for propagating credits back upstream. Overall, in the absence of congestion, all credits are used productively, as they directly support data movement.

In Figure 1b, the bottom output at router 1 becomes congested. As a result, the flit at the head of the buffer becomes blocked—indicated by a pattern of stripes—and remains stationary. Because there are still only four credits outstanding, the congestion is not yet visible to the upstream

router, which thus continues to forward flits from both VCs as they arrive. The light gray VC's destination output remains uncongested, and its flits therefore continue to be forwarded from the downstream buffer immediately after arrival. As such, any credits it consumes continue to be returned upstream after one round-trip delay. In contrast, once flits from the dark gray VC reach router 1, they are unable to make further progress due to persisting congestion. This causes the flits to accumulate in the buffer and prevents their credits from being returned upstream.

Assuming that one buffer slot is reserved for each VC to prevent interleaving deadlock and starvation [13], the congested VC eventually fills up all of its available downstream buffer space as shown in Figure 1c. By exhausting the VC's credit supply, this causes backpressure to reach router 0, prompting it to exclude the VC from switch allocation and thus propagating congestion *within the VC* upstream. For the uncongested VC, on the other hand, flits continue to be forwarded immediately after arrival at router 1. However, because only its reserved buffer slot remains available to it, the VC can only forward a single flit per credit round-trip interval across the channel; additional flits that arrive at router 0 during this interval become blocked, as shown in Figure 1c. Therefore, monopolization effects in dynamically managed buffers also cause congestion to spread *across VCs*, potentially degrading the performance of otherwise well-behaved traffic.

In contrast to Figure 1a, the majority of the outstanding credits in Figure 1c correspond to flits that remain stationary in the downstream buffer; only one of the outstanding credits is actually used to support data movement. Since the stationary flits do not actively contribute to network performance, this represents inefficient use of buffer resources, as it causes overall throughput to be reduced and leaves network channels severely under-utilized.

As demonstrated in this example, unrestricted buffer sharing allows congestion to spread across VCs. This increases the network's susceptibility to tree saturation [14]; furthermore, it allows a misbehaving workload that makes inefficient use of network resources to significantly degrade the latency and throughput of other workloads, even if the latter generate well-behaved traffic themselves.

III. Adaptive Backpressure

A. Overview

In order to mitigate the adverse effects described in Section II-B without sacrificing the benefits of dynamic buffer management under benign conditions, we introduce a mechanism that regulates sharing by heuristically limiting the number of outstanding credits for each VC based on its observed performance characteristics. The goal is to assign these credit quotas in a way that provides individual VCs with enough credits to sustain their observed throughput while minimizing the amount of buffer space that is occupied unproductively as in the example in Figure 1c.

When the number of outstanding credits for a given input VC's downstream destination VC reaches the current quota, the input VC can no longer forward flits downstream until additional credits return upstream or its quota increases. However, once occupied by a flit, a buffer slot's corresponding credit can only be returned upstream once the flit is forwarded on. As a result, VCs can temporarily exceed their quota if an update reduces it below the current number of outstanding credits. Since quota checks are performed in addition to the conventional checks for credit availability, this does not interfere with the correct operation of the router; instead, it simply causes the *actual* distribution of credits between VCs to deviate from the *desired* distribution as determined by the quotas. In the absence of network deadlock, any blocked downstream VC will make progress eventually and return its credits upstream. As these newly returned credits can only be consumed by those VCs that have not exhausted their current quota, such deviations from the desired credit distribution tend to self-correct over time.

B. Quota Computation

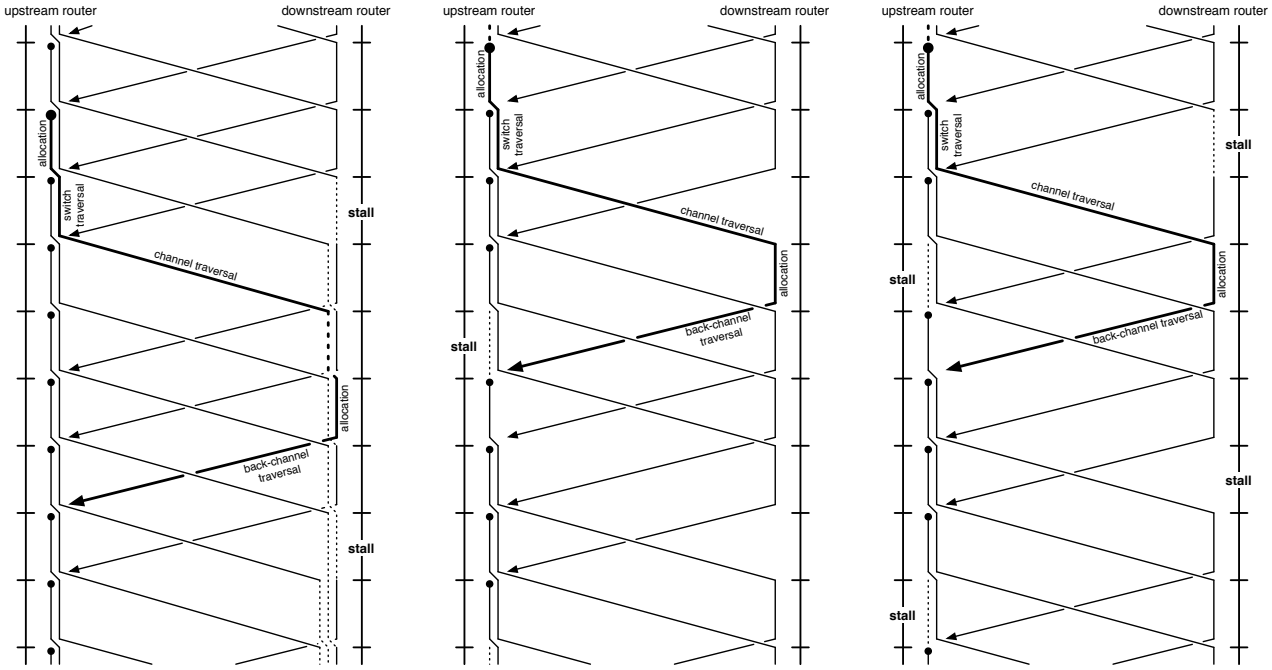
In determining quota values for individual VCs, we aim to make credits freely available to those VCs that utilize them efficiently, while being more restrictive in cases that are prone to the previously described performance pathologies. To this end, we take advantage of the realization that the number of credits that a given output VC can utilize productively is effectively limited by the throughput it achieves:

Based on our earlier example in Figure 1a, we know that if a VC achieves a steady-state throughput of one flit per cycle, the number of outstanding credits required to support this throughput is equal to the basic credit round-trip latency determined by router pipeline and channel length. In the absence of congestion, any additional credits available to the VC beyond the required number will simply remain unused.

If congestion causes stalls at the downstream router, as shown in Figure 2a, both the number of outstanding credits at the upstream router and the number of occupied buffer slots at the downstream router increase in response to each stall if no quota is imposed. This represents inefficient use of buffer space in the same way as in Figure 1c, as the additional flits accumulating in the downstream router's input buffer are not needed to support the resulting effective throughput, and the corresponding credits are unavailable to other VCs.

On the other hand, limiting the number of outstanding credits to less than the amount required to cover the basic credit round-trip latency leads to idle cycles in which the downstream router's input buffer is empty, as shown in Figure 2b: In the given example, once four flits are in flight—and hence, all four allowed credits are outstanding—, the upstream router must suspend transmission until one of the outstanding credits returns. Thus, by imposing a quota on the number of outstanding credits, we can effectively regulate throughput.

We can exploit this ability to regulate throughput by matching the credit quota value to the level of downstream congestion: Figure 2c shows the result of applying the downstream stall pattern from Figure 2a, which causes throughput to be reduced by 20 % at the downstream router, to Figure 2b, where



(a) 20 % congestion, no quota. (b) No congestion, 4-credit quota. (c) 20 % congestion, 4-credit quota.

Fig. 2. Matching quotas to congestion levels avoids unproductive use of downstream buffer space.

the credit quota leads to a 20 % reduction in throughput at the upstream router. The resulting effective throughput is the same as in Figure 2a; however, in contrast to the latter, there is no unproductive accumulation of flits in the downstream router's input buffer. As a result, the congested VC uses fewer credits, leaving more available to other VCs.

In the general case, we can avoid inefficient use of credits and buffer resources in this way by setting a VC's quota value to the product of its effective throughput and the basic credit round-trip latency $T_{crt,base}$. However, in practice, the upstream router cannot easily measure throughput directly. Instead, we compute quota values based on the observed round-trip time $T_{crt,obs}$ for individual credits:

If $T_{crt,obs}$ for a given credit is equal to $T_{crt,base}$, we know that the corresponding flit must have been forwarded immediately at the downstream router, suggesting that the VC it was assigned to is able to achieve unimpeded throughput; consequently, we set its quota value to $T_{crt,base}$ credits.

On the other hand, $T_{crt,obs}$ for a credit may exceed $T_{crt,base}$ if the corresponding flit experienced one or more stall cycles at the downstream router directly, or if it incurred queueing delay as an indirect result of previous stalls as shown by the highlighted transition in Figure 2a. Each cycle by which $T_{crt,obs}$ exceeds $T_{crt,base}$ requires a subsequent idle cycle in order to avoid unproductive increases in buffer occupancy. By subtracting the difference from the quota value for the congestion-free case, we can ensure that the appropriate number of idle cycles will be generated over the course of the next credit round-trip interval.

Noting that we must allow each VC to use at least one credit in order to guarantee that quota values can continue to be updated, we derive the overall equation for quota updates based on the observed credit round-trip time:

$$\begin{aligned} Q &= max(T_{crt,base} - (T_{crt,obs} - T_{crt,base}), 1) \\ &= max(2 \times T_{crt,base} - T_{crt,obs}, 1) \end{aligned} \quad (1)$$

Quotas for all VCs are updated independently, and no explicit effort is made to ensure that the sum of all quotas does not exceed the total buffer capacity. However, as explained in Section III-A, quota values merely represent a *desired* distribution of buffer space among VCs and thus are not required to be conservative in order to ensure correct operation.

C. Implementation

From an implementation perspective, ABP comprises two main components: A facility that measures credit round-trip times and updates quota values in response, as well as a mechanism for preventing flits from being forwarded to downstream VCs that have exhausted their quota.

To minimize overhead, we measure credit round-trip delay for at most one outstanding credit per VC at a time. This allows us to conduct the measurements using a timer and a countdown register per VC as follows: A given VC's timer is started whenever a flit is forwarded to that VC and the timer is not already running. At the same time, the countdown register is initialized to the number of credits that are already outstanding for this VC; this register is subsequently decremented whenever a credit is received. When the first credit arrives after the countdown register has reached zero, the timer is stopped

TABLE I
STORAGE OVERHEAD FOR ABP.

Description	Cost
Quota registers	$V \times \lceil log_2(T_{crt,base}) \rceil = 12\,bits$
Round-trip timers	$V \times \lceil log_2(2 \times T_{crt,base}) \rceil = 16\,bits$
Countdown registers	$V \times \lceil log_2(B) \rceil = 16\,bits$
Total storage overhead per port	$44\,bits$

and its value is used to update the VC's quota. The width of the timer must be chosen such that it can accommodate twice the minimum credit round-trip time; in case of overflow, the quota is simply set to its minimum value of one as per Equation 1. Since measurements and quota updates can be performed off the critical path, they do not affect the router's cycle time.

With credit-based flow control, routers must track the number of outstanding credits for each output VC and block any switch allocator request whose destination VC has no credits available. We can implement quota enforcement for ABP as a simple extension to this mechanism, with no further changes to the router pipeline required.

D. Overhead

When using a dynamically managed input buffer with a capacity of $B = 16$ flits per input buffer, $V = 4$ VCs and a basic credit round-trip latency of $T_{crt,base} = 8$ cycles, we can compute the total number of registers required for implementing ABP using Table I. Assuming a flit width of 64 bits, the resulting 44 registers represent an overhead of 4.2 % relative to the cost of the flit buffer and its associated management logic.

IV. EXPERIMENTAL SETUP

We evaluate the efficacy of ABP using a customized version of the BookSim 2.0 interconnection network simulator. Simulations are performed on an 8×8 2D mesh network. All network channels are 64 bits wide and have a delay of one cycle. Packets are routed using Dimension-Order Routing (DOR).

We model input-queued routers with credit-based flow control and two pipeline stages. The first stage performs combined VC and switch allocation [15] and generates updated lookahead routing information for the next hop [16], while the second pipeline stage is reserved for crossbar traversal. We use a separable input-first switch allocator design with round-robin arbiters. After arriving at a router, credits incur a processing and signal propagation delay of two cycles before the corresponding downstream buffer slot becomes available for allocation.

Each input buffer has a total capacity of 16 flits which is shared among 4 VCs. One buffer slot is reserved for each VC in order to avoid interleaving deadlock and starvation [13]. The *baseline* configuration does not otherwise restrict sharing, while the *adaptive* configuration implements ABP as described in Section III. For experiments with two traffic classes, two VCs are statically assigned to each class.

Network terminals maintain a separate, unbounded injection queue for each traffic class. Each terminal can can inject a single flit into the network in any given cycle. Unless otherwise noted, all reported latencies include source queueing delay.

We first present simulation results for synthetic traffic. Packet arrival times are generated by a Bernoulli process. Destination addresses are either chosen randomly or selected according to one of five permutation patterns. Packet lengths follow a bimodal distribution, with half the packets comprising two and six flits, respectively.

To evaluate traffic isolation for realistic application workloads running on a heterogeneous CMP, we further simulate a network where each node injects two different types of traffic in separate classes:

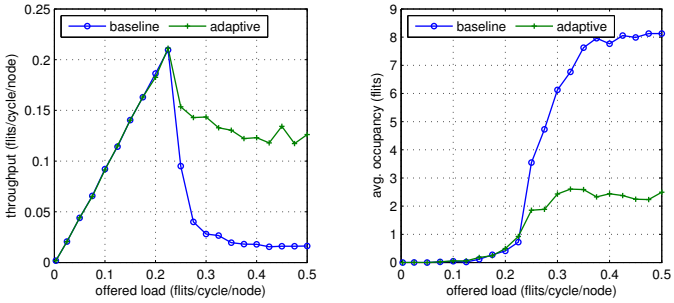
We leverage Netrace [17] to generate traffic representative of latency-optimized CPU cores using PARSEC benchmarks [11]. By tracking and enforcing dependencies between packets in a trace, Netrace enables us to perform closed-loop simulations without incurring the overhead of using a full-system simulator. The target system for the PARSEC traces models in-order RISC cores that run at four times the network's clock frequency and have private L1 instruction and data caches of 32 kB each. The L1 caches are 4-way associative, have a 3-cycle access time and implement MESI coherence. 16 MB of L2 cache are organized as a 64-bank fully shared S-NUCA cache with 8-way associativity and an 8-cycle bank access time. Access times in both cases are reported relative to the core clock. Cache lines are 64 bytes wide across the hierarchy; with 64-bit wide network channels, this leads to a bimodal packet length distribution where long packets comprise a head flit, one flit carrying a 64-bit memory address and eight payload flits, while short packets only include the first two. There are a total of eight memory controllers distributed along the four edges of the chip. Memory accesses incur a latency of 150 core clock cycles.

In addition to application traffic, each node generates a second class of traffic in which data is streamed to the memory controllers at the maximum admissible rate (12.5 % per node). This is intended to model an array of throughput-optimized stream processing cores with a large aggregate number of outstanding memory transactions that is co-located with each latency-optimized core. Data is streamed at cache line granularity with the same bimodal packet length distribution as application traffic. Destination addresses are interleaved such that load is spread uniformly over all memory controllers.

V. EVALUATION

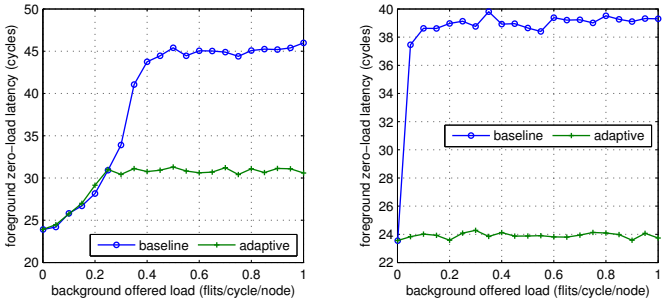
A. Network Stability

Figure 3a shows the *effective* throughput—i.e., the minimum throughput observed across all source-destination pairs—for the *tornado* traffic pattern as a function of the injection rate. In this adversarial traffic pattern, once the saturation point is reached, throughput for the baseline configuration decreases as offered load continues to increase. This instability is a result of starvation effects in the network that start to manifest as contention increases.



(a) Effective throughput. (b) Buffer occupancy.

Fig. 3. Network stability measurements for tornado traffic.



(a) Uniform background. (b) Hotspot background.

Fig. 5. Perceived foreground zero-load latency with background traffic.

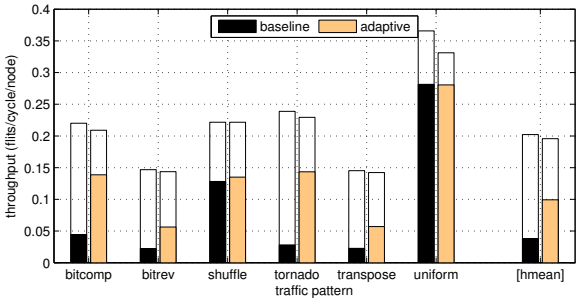


Fig. 4. Effective throughput at 30 % load. Outlines show saturation rate.

By promoting efficient use of buffer resources, ABP significantly reduces throughput degradation past the saturation point: At 50 % injection rate, throughput is 7.76× higher than that of the baseline configuration. We can explain this performance increase by studying the average buffer occupancy for the two configurations as shown in Figure 3b. Once the injection rate passes the saturation point, buffer occupancy for the baseline configuration quickly grows to a value that is on the same order as the credit round-trip time. This is indicative of the monopolization behavior shown in Figure 1c and causes saturation to quickly spread throughout the network. With ABP, on the other hand, buffer occupancy remains low even once the network reaches saturation, leaving more credits available and thus reducing network congestion.

While practical systems with finite injection queues are inherently self-throttled, and therefore cannot operate in the post-saturation region in steady state, it is possible for the injection rate in such systems to exceed the saturation rate temporarily, e.g. as a result of bursty traffic or the formation of a transient hotspot. In such cases, maintaining high post-saturation throughput enables the network to recover and return to steady-state operation more quickly.

Figure 4 shows the effective throughput at 30 % injection rate for different synthetic traffic patterns, as well as the harmonic mean across patterns. With the exception of uniform random, this places the network in saturation; consequently, ABP is able to outperform the baseline implementation by 2.6×. This improvement comes at the cost of an average of 3 % reduction in saturation rate, shown as outlines in Figure 4. For uniform random traffic, the saturation rate decreases by 10 %; this is because correlation between packets is reduced due to their randomly selected destinations, and consequently quota values are less likely to be applicable to successive packets in a given VC. However, it should be noted that this disadvantage only applies for uniform random traffic at injection rates exceeding 33 %. With other traffic patterns, on the other hand, ABP provides better performance starting at significantly lower rates.

B. *Traffic Isolation*

Performance in many CMPs workloads is primarily limited by latency rather than throughput [2]. In evaluating traffic isolation, we thus focus our investigation on how packet latency for a given foreground traffic is affected by a secondary background traffic.

Figure 5 demonstrates how the *perceived* zero-load latency observed by a foreground workload that consists of uniform random traffic varies with different background injection rates. We consider uniform random background traffic in Figure 5a and background traffic comprising a hotspot at the center of the network in Figure 5b. In both cases, latency initially increases up to the point where the background workload becomes saturated. Beyond this point, buffer occupancy in the baseline network continues to increase, resulting in fewer credits being available to the foreground traffic and thus a marked increase in latency. With ABP, on the other hand, average buffer occupancy stops increasing once the background load reaches saturation, enabling the foreground traffic's packets to traverse the network with less delay.

Figure 6 illustrates the resulting latency-throughput characteristics for uniform foreground traffic when uniform background traffic is injected at a rate of 50 %. ABP significantly improves both the latency perceived by the foreground traffic and its effective saturation throughput.

We similarly measure the perceived foreground zero-load latency for the remaining traffic patterns; the results are shown in Figure 7. The white segment at the bottom of each bar corresponds to the true zero-load latency measured in the absence of background traffic. As in Figures 5 and 6, when going from 30 % to 50 % background injection rate, latency

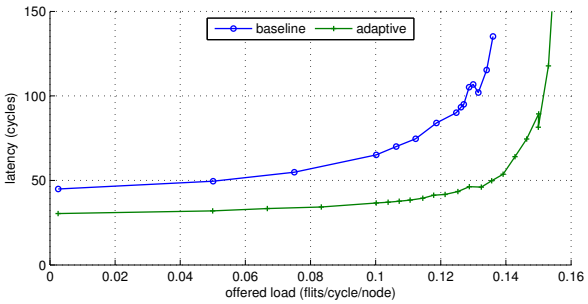


Fig. 6. Average foreground packet latency with 50 % background traffic.

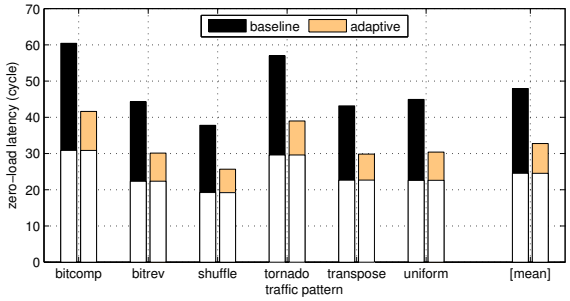


Fig. 7. Foreground zero-load latency increase with 50 % background traffic.

increases for the baseline configuration, but remains virtually unchanged for the configuration using ABP.

The correspond simulation results with hotspot background traffic are largely in line with those from Figure 5b, with about 35 % latency degradation for the baseline configuration and virtually no change in latency when using ABP; we omit the corresponding graphs for brevity.

C. Application Performance

Figure 8 shows the execution time increase observed for individual PARSEC benchmarks that run on the general-purpose cores in the presence of streaming background traffic as described in Section IV, as well as the geometric mean across all benchmarks. For each PARSEC application, we measure the time it takes to deliver the first one million packets from the benchmark's Region of Interest (ROI) across the network. All results are normalized to the execution time

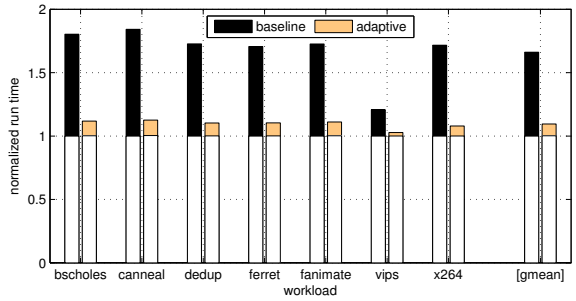


Fig. 8. Application slowdown for latency-optimized cores.

measured for the *base* configuration when no background traffic is injected into the network.

The white segment at the bottom of each bar in Figure 8 corresponds to the execution time without background traffic for a particular combination of configuration and benchmark. As in our earlier experiments with synthetic traffic, we see that using ABP does not adversely affect the performance in the benign case.

Once streaming background traffic is injected, all three configurations experience significant performance degradation. This slowdown is primarily a result of increased packet latency: As the simple in-order cores modeled in our experiment support only a single outstanding memory transaction, any additional network delay incurred by memory traffic directly results in stall cycles.

The reasons for the latency increase are two-fold: On the one hand, additional contention delay is incurred as a result of failed allocation attempts as both types of traffic compete for channel bandwidth. On the other hand, dynamically managed input buffers can allow the streaming background load to monopolize buffer space as described in Section II, effectively limiting the application traffic's credit supply. In contrast to contention-induced delay, the effects of this *indirect* throttling cannot be mitigated by prioritizing latency-sensitive traffic during allocation.

In limiting buffer occupancy for adversarial traffic, ABP reduces packet latency and improves throughput for the PARSEC traffic generated by the general-purpose cores, resulting in a 34 % reduction in execution time compared to the baseline configuration. The benefit of employing ABP is least pronounced (15 %) for the *vips* benchmark; this is because *vips* uses coarse-grain parallelism with comparatively little sharing and data exchange between different nodes, making this benchmark less sensitive to network performance. In contrast, the *canneal* benchmark generates significantly higher network load and is thus much more sensitive to network performance, resulting in improvements of 39 %.

VI. Related Work

The *ViChaR* scheme [7] regulates the number of active VCs at each router input based on network load; however, it does not impose limits on the amount of buffer space that individual VCs can occupy. While the use of atomic VC allocation and a fixed, short packet length prevent *individual* VCs from monopolizing buffer space in this scheme, *groups* of VCs with similar performance characteristics—e.g. those assigned to a particular traffic class—can in aggregate still significantly degrade performance for other VCs.

Banerjee and Moore [18] show that resource utilization in NoCs can be improved by performing allocation for flows rather than for individual packets. Shim et al. [19] propose a similar approach that statically binds flows to specific VCs at design time. Both approaches prevent blocked flows from acquiring more than a single VC at each input buffer, but neither limits the amount of buffer space occupied by that VC. As such, they are complementary to our proposed mechanism.

Lai et al. [20] propose a scheme in which each router predicts congestion levels at neighboring routers' output ports and prioritizes those packets during switch allocation that will be forwarded to uncongested outputs. Similarly to ABP, this causes fewer flits to be sent to VCs which are subject to downstream congestion; however, because congestion levels are estimated at port granularity, this approach cannot prevent interference between multiple flows of packets destined for the same output.

Network-level congestion control schemes based on source throttling [21], [22] inherently mitigate buffer monopolization effects by reducing the incidence of congestion in the network. However, because such schemes only perform coarse-grained traffic regulation at the network boundary, they tend to be pessimistic and slow to react to localized changes in network behavior. Furthermore, they typically only consider the aggregate behavior across all VCs and thus do not address interference between concurrent workloads.

Finally, prior research has explored various schemes for providing QoS guarantees and isolation between workloads in NoCs [23]–[25]. These mechanisms generally assume that buffer space is statically partitioned and include no provisions to avoid interference effects caused by buffer sharing; consequently, such approaches are complementary to the ABP scheme introduced in the present contribution.

VII. Conclusions

In this paper, we have introduced ABP, a novel scheme for regulating buffer space usage in dynamically managed router input buffers. By heuristically limiting each VC's credit supply based on its observed performance characteristics, the proposed scheme aims to minimize unproductive use of buffer space and to prevent VCs that experience downstream congestion from monopolizing shared buffer space at the expense of other VCs' performance. These benefits are achieved while maintaining the key utilization and performance benefits that buffer sharing provides under benign load conditions. ABP can be implemented as a simple, low-overhead extension to the existing flow control logic.

We evaluate our proposed scheme on a 64-node mesh network and show that it improves network stability and increases throughput under heavy load by 2.6× on average for a set of synthetic traffic patterns. Furthermore, we demonstrate that it is effective at reducing performance coupling in the presence of background traffic, yielding an average improvement in zero-load latency of 31 % compared to a state-of-the-art implementation with unrestricted sharing. Finally, we present simulation results for PARSEC benchmarks running on a heterogeneous CMP and show that ABP can reduce the performance impact of sharing the network with arrays of stream processing cores by an average of 34 % across benchmarks.

Overall, ABP enables networks to satisfy more stringent quality-of-service requirements while preserving the benefits of unrestricted buffer sharing under benign load conditions.

References

[1] P. Gratz *et al.*, "Implementation and Evaluation of On-Chip Network Architectures," in *Proc. of the IEEE Int'l Conf. on Computer Design*, 2006.

[2] D. Sanchez *et al.*, "An Analysis of Interconnection Networks for Large Scale Chip-Multiprocessors," *ACM Trans. on Archictecture and Code Optimization*, vol. 7, no. 1, 2010.

[3] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Inteconnection Networks," in *Proc. of the 38th Design Automation Conf.*, 2001.

[4] W. J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, 1992.

[5] J. Hu and R. Marculescu, "Application-Specific Buffer Space Allocation for Networks-on-Chip Router Design," in *Proc. of the IEEE/ACM Int'l Conf. on Computer-Aided Design*, 2004.

[6] M. Rezazad and H. Sarbazi-azad, "The Effect of Virtual Channel Organization on the Performance of Interconnection Networks," in *Proc. of the 19th IEEE Int'l Parallel and Distributed Processing Symp.*, 2005.

[7] C. Nicopoulos *et al.*, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *Proc. of the 39th Int'l Symp. on Microarchitecture*, 2006.

[8] X. Chen and L.-S. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks," in *Proc. of the Int'l Symp. on Low Power Electronics and Design*, 2003.

[9] H. Wang *et al.*, "Power-driven Design of Router Microarchitectures in On-chip Networks," in *Proc. of the 36th Int'l Symp. on Microarchitecture*, 2003.

[10] Y. Tamir and G. L. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communications Switches," *SIGARCH Computer Architecture News*, vol. 16, no. 2, 1988.

[11] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, 2011.

[12] J. Park *et al.*, "Design and Evaluation of a DAMQ Multiprocessor Network With Self-Compacting Buffers," in *Proc. of the ACM/IEEE Conf. on Supercomputing*, 1994.

[13] J. Liu and J. G. Delgado-Frias, "DAMQ Self-Compacting Buffer Schemes for Systems with Network-On-Chip," in *Proc. of the IEEE Int'l Conf. on Computer Design*, 2005.

[14] G. F. Pfister and V. A. Norton, "Hot Spot Contention and Combining in Multistage Interconnection Networks," *IEEE Trans. on Computers*, vol. 34, no. 10, 1985.

[15] A. Kumar *et al.*, "A 4.6Tbits/s 3.6GHz Single-cycle NoC Router with a Novel Switch Allocator in 65nm CMOS," in *Proc. of the IEEE Int'l Conf. on Computer Design*, 2007.

[16] M. Galles, "Spider: A High-Speed Network Interconnect," *IEEE Micro*, vol. 17, no. 1, 1997.

[17] J. Hestness and S. W. Keckler, "Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation," The University of Texas at Austin, Dept. of Computer Science, Tech. Rep. TR-10-11, 2011.

[18] A. Banerjee and S. W. Moore, "Flow-Aware Allocation for On-Chip Networks," in *Proc. of the 3rd ACM/IEEE Int'l Symp. on Networks-on-Chip*, 2009.

[19] K. S. Shim *et al.*, "Static Virtual Channel Allocation in Oblivious Routing," in *Proc. of the 3rd ACM/IEEE Int'l Symp. on Networks-on-Chip*, 2009.

[20] M. Lai *et al.*, "A Dynamically-Allocated Virtual Channel Architecture with Congestion Awareness for On-Chip Routers," in *Proc. of the 45th Design Automation Conf.*, 2008.

[21] M. Thottethodi *et al.*, "Self-Tuned Congestion Control for Multiprocessor Networks," in *Proc. of the 7th Int'l Symp. on High-Performance Computer Architecture*, 2001.

[22] E. Baydal *et al.*, "A Family of Mechanisms for Congestion Control in Wormhole Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 9, 2005.

[23] J. W. Lee *et al.*, "Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks," in *Proc. of the 35th Int'l Symp. on Computer Architecture*, 2008.

[24] B. Grot *et al.*, "Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip," in *Proc. of the 42nd IEEE/ACM Int'l Symp. on Microarchitecture*, 2009.

[25] ——, "Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees," in *Proc. of the 38th Int'l Symp. on Computer Architecture*, 2011.