Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip

Boris Grot Stephen W. Keckler Onur Mutlu[†]

Department of Computer Sciences
The University of Texas at Austin
{bgrot, skeckler@cs.utexas.edu}

[†]Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu

ABSTRACT

Future many-core chip multiprocessors (CMPs) and systems-on-a-chip (SOCs) will have numerous processing elements executing multiple applications concurrently. These applications and their respective threads will interfere at the on-chip network level and compete for shared resources such as cache banks, memory controllers, and specialized accelerators. Often, the communication and sharing patterns of these applications will be impossible to predict off-line, making fairness guarantees and performance isolation difficult through static thread and link scheduling. Prior techniques for providing network quality-of-service (QOS) have too much algorithmic complexity, cost (area and/or energy) or performance overhead to be attractive for on-chip implementation. To better understand the preferred solution space, we define desirable features and evaluation metrics for QOS in a network-on-a-chip (NOC). Our insights lead us to propose a novel QOS system called Preemptive Virtual Clock (PVC). PVC provides strong guarantees, reduces packet delay variation, and enables efficient reclamation of idle network bandwidth without per-flow buffering at the routers and with minimal buffering at the source nodes. PVC averts priority inversion through preemption of lower-priority packets. By controlling preemption aggressiveness, PVC enables a trade-off between the strength of the guarantees and overall throughput. Finally, PVC simplifies network management through a flexible allocation mechanism that enables per-application bandwidth provisioning independent of thread count and supports transparent bandwidth recycling among an application's threads.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors—*Interconnection architectures*

General Terms

Design, Measurement, Performance

MICRO'09, December 12–16, 2009, New York, NY, USA.
Copyright 2009 ACM 978-1-60558-798-1/09/12 ...$10.00.

1. INTRODUCTION

Power limitations of aggressive monolithic cores, design complexity considerations, and growing transistor budgets have recently pushed the semiconductor industry toward chip multiprocessors (CMPs) and complex Systems-On-a-Chip (SOCs). These single-die systems integrate execution cores, accelerators, custom IP blocks, and memories, providing an unprecedented degree of functionality on a single piece of silicon. Current commercial offerings in this space range from Intel's 8-core superscalar CMP [18] to a 64-core network and media processor from Tilera [21] to a 256-element reconfigurable processor array from Rapport, Inc. [16]. With continuing technology scaling, CMPs with hundreds to thousands of general and special-purpose cores are likely to appear in a variety of application domains in the near future.

As the number of compute elements grows, so will the number of intra- and inter-application threads executing concurrently on a given substrate. These threads will compete for shared resources, such as cache space, specialized accelerators, on-chip network bandwidth, and off-chip memory bandwidth. As a result, ensuring application stability, scalability, and isolation in the face of increased resource sharing will become more important and more difficult.

Furthermore, a user or operating system may wish to increase the performance of one application at the expense of another by giving the former a greater share of some system resource. Thus, to ensure fairness and provide differentiated services, future CMP and SOC substrates will require an integrated quality-of-service (QOS) mechanism.

Until recently, on-chip QOS research has focused on individual system end-points, such as cache banks or memory controllers, seeking to balance fairness, performance and cost when these resources are shared among multiple threads [12, 13, 11]. Unfortunately, such work ignores the shared interconnect used to access the individual resources. By ignoring the access medium, fairness at the end-points cannot be guaranteed. QOS research in conventional networks, on the other hand, has yielded elegant service disciplines that provide hard guarantees and good performance, but at high cost in terms of storage required at each routing node, buffer access energy, and in some cases, a scheduling algorithm's computational complexity. These costs can be prohibitive in area-, energy- and latency-sensitive on-chip networks. Recent work on Globally Synchronized Frames (GSF) introduced a method to move much of the complexity from network routers into the source nodes [9]. Unfortunately, GSF suffers from several shortcomings, including reliance on large source buffers, low throughput under some unstructured traffic patterns, and an inflexible bandwidth allocation mechanism.

In this work, we seek to understand the qualities of an ideal QOS solution for networks-on-a-chip (NOCs). We draw on traditional QOS literature and supplement it with our own observations to enumerate the attribute set of an ideal NOC QOS scheme. We also consider the metrics for evaluating the different approaches. Our insights lead us to propose Preemptive Virtual Clock (PVC), a novel QOS scheme specifically designed for cost- and performance-sensitive on-chip interconnects. Unlike all prior approaches for providing network quality-of-service, PVC requires neither per-flow buffering in the routers nor large queues in the source nodes. Instead, PVC provides fairness guarantees by tracking each flow's bandwidth consumption over a time interval and prioritizing packets based on the consumed bandwidth and established rate of service. PVC avoids priority inversion by preempting lower-priority messages. The system provides guarantees and low latency for preempted messages via a dedicated ACK/NACK network and a small window of outstanding transactions at each node. Unique to this approach is the ability to trade the strength of throughput guarantees of individual flows for overall system throughput. Finally, PVC simplifies network management by enabling per-thread, per-application, or per-user bandwidth allocation.

The rest of the paper is structured as follows. Section 2 motivates the work by outlining the requirements and the metrics for NOC QOS techniques and presents an overview of prior approaches for network quality-of-service. Section 3 introduces PVC and compares it to prior schemes based on the attributes from Section 2. Section 4 covers the evaluation methodology, while Section 5 presents the results of the evaluation. Section 6 concludes the paper.

2. MOTIVATION

2.1 NOC QOS Requirements

An ideal NOC QOS solution should possess a number of attributes with regard to guarantees, performance and cost. In this section, we draw on traditional QOS literature and supplement it with our own observations to detail the desirable feature set. Items *a, b, c, e, i, j* are taken from or inspired by a similar list compiled by Stiliadis and Varma [20], while *f* comes from Demers et al. [3].

a) Fairness: Link bandwidth must be divided among requesting flows equitably based on individual reserved rates for both guaranteed and excess service.

b) Isolation of flows: Rate-observing flows should enjoy the illusion of a private network with bandwidth proportional to the specified rate, regardless of the behavior of other flows.

c) Efficient bandwidth utilization: Flows should be free to claim idle network bandwidth regardless of their reserved rate or bandwidth usage history.

d) Flexible bandwidth allocation: It should be possible to allocate bandwidth at granularity of a core, a multi-core application, or a user. Coarser granularities simplify provisioning and improve bandwidth utilization.

e) Low performance overhead: Compared to a similarly provisioned network with no QOS support, a QOS-enabled network should enjoy approximately equal latency and overall throughput.

f) Delay proportional to bandwidth usage: Flows that observe their bandwidth share should enjoy faster service than bandwidth hogs.

g) Low area overhead: Per-flow buffering at each network node may be too expensive for on-chip networks that typically feature wormhole switching and a small number of virtual channels.

h) Low energy overhead: Energy may be the biggest constraint in future CMPs and SOCs [14]. Minimizing buffering is one way to reduce the energy overhead of a QOS subsystem.

i) Good scalability: As the network is scaled up in size, the QOS subsystem should be easy and cost-effective to scale proportionately, without compromising performance or guarantees.

j) Simplicity of implementation: Design and verification time are important contributors to overall system cost, and a simpler QOS solution is generally preferred to one with greater complexity.

2.2 Metrics

In addition to the standard area and energy metrics used for evaluating on-chip systems, QOS disciplines require dedicated metrics. Key among these are *relative throughput, latency, and jitter.*

Relative throughput: The fairness criterion dictates that link bandwidth should be allotted equitably, in proportion to the specified rate. Given the mean throughput of a set of flows with the same reserved rate, request rate and measurement interval, *relative throughput* can be measured by assessing the minimum, maximum, and standard deviation from the mean in the flow set. A system provides strong throughput fairness when each node's bandwidth consumption is close to the mean.

Latency: The end-to-end latency of a flow should be proportional to its hop count, reserved rate, and contention from other flows. In the absence of contention, the delay imposed by the QOS mechanism should be minimal. On the other hand, when two or more flows with the same specified rate converge on an output link, the QOS mechanism must ensure equal per-hop delay for the affected flows. As above, the key metrics are minimum, maximum, and standard deviation from the mean hop latency for a set of flows sharing a port.

Jitter: The variation in delay for a pair of packets in a flow is commonly called jitter. Low jitter in the face of contention provides a strong illusion of a private network for each flow, desirable for performance stability and isolation. QOS schemes that feature rate-proportional per-hop latency guarantees, as opposed to just end-to-end delay bounds, may naturally reduce jitter. The metric for jitter is termed *packet delay variation (pdv)*, defined for IP performance measurement as "the difference in end-to-end delay between selected packets in a flow with any lost packets being ignored" [17]. The maximum pdv and standard deviation from the mean pdv within a flow, as well as across flows, are more important than the minimum observed jitter value.

2.3 QOS Service Disciplines

A number of distinct disciplines have emerged over the years for providing fair and differentiated services at the network level. We partition these into three classes based on their bandwidth allocation strategy – *fixed, rate-based, and* *frame-based* – and cover the most important representatives of each class.

2.3.1 Fixed bandwidth allocation

Approaches such as Weighted Round Robin use a static packet schedule to deliver hard guarantees at low implementation complexity. The cost, however, is potentially poor network utilization, as resources held by idle flows cannot be rapidly redistributed to flows with excess demand.

2.3.2 Rate-based approaches

Rate-based service disciplines aim to allocate bandwidth to contending packets based on the provisioned rate. Idle bandwidth due to under-utilization by one or more flows is instantaneously redistributed among the competing flows. Service order is determined dynamically based on the set of active flows and their respective reserved rates by computing the service time for each flow and granting the flow with the earliest deadline. In general, rate-based approaches excel at maximizing throughput and providing strong isolation, but necessitate per-flow queueing and may require computationally expensive scheduling algorithms.

Fair Queueing (FQ) is a well-known rate-based approach that emulates a bit-by-bit round-robin service order among active flows on a packet basis [3]. Its generalized variant, Weighted Fair Queueing (WFQ), enables differentiated services by supporting different service rates among the flows. Both schemes offer provably hard fairness guarantees at a fine granularity and excellent bandwidth utilization. Unfortunately, computing the service time in FQ has O(N) complexity, where N is the number of active flows at each scheduling step, making the algorithm impractical for most applications.

In contrast, Virtual Clock [22] offers a simple deadline computation that emulates a Time Domain Multiple Access (TDMA) scheduler but with ability to recycle idle slots. Packets are scheduled using virtual time slots, computed based on the assigned service rate. Packet service time is simply its flow's virtual clock value, which is incremented every time the flow is serviced. In flows that respect the reserved rate, termed *rate-conformant flows*, virtual time tracks the service time under TDMA. Flows that exceed the specified rate "run ahead" of schedule by incrementing their virtual clock beyond the current round. Problematically, flows that run ahead are subject to starvation by rate-conformant flows until the rate-scaled real time catches up with their virtual clock. Both Fair Queueing and Virtual Clock require per-flow queues and a sorting mechanism to prioritize flows at each scheduling step, resulting in high storage overhead and scheduling complexity in networks with a large number of flows.

2.3.3 Frame-based approaches

Whereas rate-based disciplines aim for tight guarantees at a fine granularity by scheduling individual packets, frame-based approaches seek to reduce hardware cost and scheduling complexity by coarsening the bandwidth allocation granularity. The common feature of these schemes is the partitioning of time into epochs, or frames, with each flow reserving some number of transmission slots within a frame. A disadvantage of frame-based disciplines lies in their coarse throughput and latency guarantees, which apply only at the frame granularity. Coarse-grained bandwidth allocation can cause temporary starvation of some flows and high service rate for others, making jitter guarantees impossible. Frame-based approaches also require per-flow buffering at each routing node, necessitating enough storage to buffer each flow's entire per-frame bandwidth allocation. Schemes such as Rotating Combined Queueing (RCQ) [7] that support multiple in-flight frames to improve network bandwidth utilization incur additional area and energy overheads in the form of even greater buffer requirements.

Globally Synchronized Frames (GSF) is a frame-based QOS approach recently proposed specifically for on-chip implementation [9]. GSF also employs a coarse-grained bandwidth reservation mechanism. However, it moves the buffering and much of the scheduling logic from the network routers into the source nodes, thereby reducing the routers' area and energy overhead. Source nodes in GSF tag new packets with a frame number and slot them into their source queue. GSF supports bursts by allowing packets from future frames to enter the network, up to a maximum allowed burst size. A fast barrier network synchronizes the frames over the entire chip by detecting when the head frame has been drained and signaling a global frame roll-over. To ensure fast frame recycling, injection of new packets into the head frame is prohibited. Packets from multiple frames may be in the network at the same time, and age-based arbitration on the frame number is used to prioritize packets from older frames over younger ones. GSF does not specify the service order within a frame, preventing priority inversion by reserving a single virtual channel (VC) at each input port for the head frame; however, in-flight packets from future frames may be blocked until their respective frames become the oldest.

Although GSF significantly reduces router complexity over prior approaches, it suffers from three important shortcomings that limit its appeal: performance, cost, and inflexible bandwidth allocation.

The performance (throughput) limitations of GSF arise due to its source-based bandwidth reservation mechanism. With only limited support for excess service, bound by the finite number of in-flight frames, GSF is inherently restricted in its ability to efficiently utilize network bandwidth. Once a source node has exhausted its burst quota, it is immediately throttled and restricted to its reserved allocation in each frame interval.

Figure 1 highlights a scenario that compromises a node's throughput despite idle network capacity. A set of nodes, in grey, all send traffic to a common destination, colored black. The combined traffic causes congestion around the black node, exerting backpressure on the sources and impeding global frame reclamation. As frame reclamation slows, an unrelated node, striped in the figure, in a different region of the network suffers a drop in throughput. The striped node is only sending to its neighbor, yet is throttled upon exhausting its burst quota, letting the requested link go idle. We simulated this scenario on a 64-node network with an aggressive GSF configuration (2000 cycle frame, 6-frame burst window, and 8 cycle frame reclamation) and equal bandwidth allocation among nodes, under the assumption that the actual communication pattern is not known in advance. We observed that throughput for the striped node saturates at around 10%, meaning that the link is idle 90% of the time. Increasing both the size of the frame and the burst window ten-fold made no difference in actual throughput once the striped node exhausted its burst capacity.

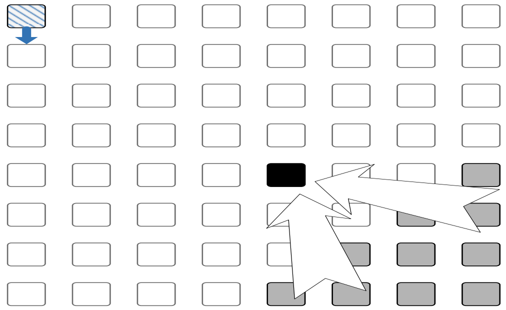


Figure 1: Scenario demonstrating poor bandwidth utilization with GSF. The grey nodes congest the center of the mesh, slowing down frame reclamation and compromising striped node's throughput.

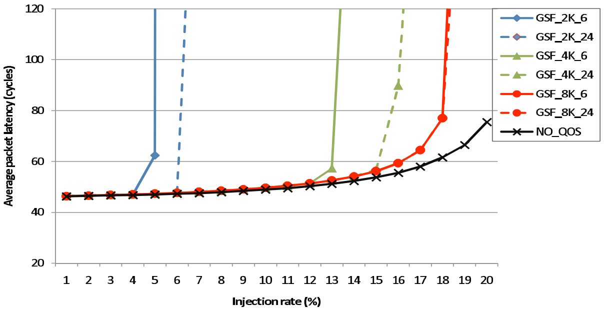


Figure 2: Performance of GSF with various frame (first number in legend) and window (second number) sizes versus a similarly provisioned network without QOS support.

Another drawback of GSF is the cost associated with the source queues, where packets are slotted to reserve bandwidth in future frames. Longer frames better amortize the latency of barrier synchronization and support bursty traffic, but necessitate larger source queues. Our experiments, consistent with results in the original paper, show that in a 64-node network, a frame size of 1000 flits or more is required to provide high throughput on many traffic patterns. To support asymmetric bandwidth allocation, whereby any node may reserve a large fraction of overall frame bandwidth, source queues must have enough space to buffer at least a full frame worth of flits. Assuming a frame size of 1000 flits and 16-byte links, GSF requires a 16 KB source queue at each network terminal. Scaling to larger network configurations requires increasing the frame size and source queues in proportion to the network size.

Figure 2 shows the performance of GSF under the *uniform random* traffic pattern on a 256 node network with different frame lengths and window sizes (number of in-flight frames). To reach a level of throughput within 10% of a generic NOC network with no QOS support, GSF requires a frame size of 8000 flits, necessitating 128 KB of storage per source queue.

Finally, GSF is inflexible in its bandwidth allocation, as bandwidth may only be assigned at the granularity of individual nodes, complicating network management. For instance, a parallel application with a fluctuating thread count running on multiple cores can cause a network to be reprovisioned every time a thread starts or ends, placing a burden on the OS or hypervisor.

3. PREEMPTIVE VIRTUAL CLOCK

Our motivation in designing a new QOS system is to provide a cost-effective mechanism for fairness and service differentiation in on-chip networks. Primary objectives are to minimize area and energy overhead, enable efficient bandwidth utilization, and keep router complexity manageable to minimize delay. Another goal is to simplify network management through a flexible bandwidth reservation mechanism to enable per-core, per-application, or per-user bandwidth allocation that is independent of the actual core/thread count. This section details the resulting scheme, which we term Preemptive Virtual Clock (PVC).

3.1 Overview

Bandwidth allocation: As the name implies, PVC was partly inspired by Virtual Clock due its rate-based nature and low scheduling complexity. Each flow in PVC is assigned a rate of service, which is translated into a certain amount of *reserved* bandwidth over an interval of time. Routers track each flow's bandwidth utilization, computing a packet's priority based on its respective flow's bandwidth consumption and assigned rate. The packet with the highest priority at each arbitration cycle receives service. Similar to Virtual Clock, flows may consume bandwidth beyond the reserved amount, potentially subjecting them to subsequent starvation from rate-conformant flows. This problem arises as a result of relying on past bandwidth usage in priority computation.

To reduce the history effect, PVC introduces a simple framing strategy. At each frame roll-over, which occurs after a fixed number of cycles, bandwidth counters for all flows are reset. Thus, PVC provides bandwidth and latency guarantees at frame granularity but uses rate-based arbitration within a frame. Note that because flows are free to consume idle network bandwidth, PVC does not require multiple in-flight frames or sophisticated frame completion detection to achieve good throughput. Figures 3 and 4 compare the framing schemes of GSF and PVC, respectively. GSF supports multiple in-flight frames whose completion time is determined dynamically via a global barrier network that detects when all packets belonging to a frame have been delivered. In contrast, PVC has only one fixed-duration frame active at any time. Packets in PVC are not bound to frames, and a given packet may enter the network in one frame interval and arrive in the next.

Freedom from Priority Inversion: PVC uses relatively simple routers with a small number of virtual channels per input port. Without per-flow queueing, packets from flows that exceed their bandwidth allocation in a frame may block packets from rate-conformant flows. Similarly, flows that greatly exceed their assigned rate may impede progress for flows that surpass their allocation by a small margin. Both situations constitute *priority inversion.* PVC uses a preemptive strategy to deal with such scenarios, removing lower priority packets from the network, thus allowing blocked packets of higher priority to make progress.

To support retransmission of dropped packets, PVC re-

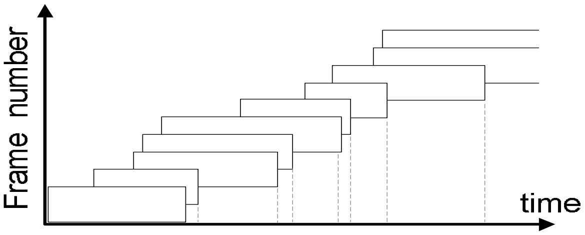


Figure 3: GSF framing strategy: multiple in-flight frames; length determined dynamically via a global barrier network.

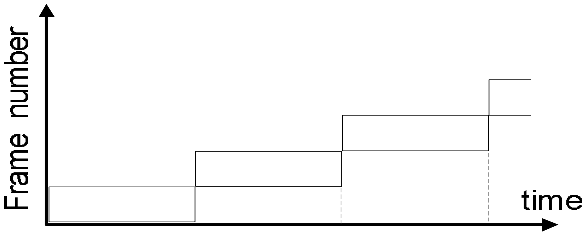


Figure 4: PVC framing strategy: single in-flight frame; fixed frame length.

quires a preemption recovery strategy. One option for preemption recovery is a timeout. Establishing a safe timeout interval is often difficult, however. Additionally, timeouts necessitate large source buffers to support a sufficient number of outstanding transactions to cover the timeout delay. Instead, we choose to use a dedicated non-discarding ACK network for signaling packet completion and preemption events. The cost of such a network is low as its width is small compared to the wide on-chip data network. In addition, this cost may be amortized by integrating the network with the chip's fault-tolerance logic to provide end-to-end data delivery guarantees, which may be required as substrates get less reliable due to technology scaling.

As packets are subject to discard, they must be buffered at the source until an acknowledgement from the destination is received. In the case of dropped packets, preemption of the header flit generates a NACK message to the source node. Once received at the source, the NACK triggers a retransmission of the dropped packets. Thus, PVC requires a small source window to buffer outstanding transactions. Advantageously, a *small* window size acts as a natural throttle, or rate-controller, preventing individual nodes from overflowing the network's limited buffering. The window only needs to be big enough to support high throughput when the interconnect is congestion-free and allows for prompt ACK return. In our experiments, a 64-node network sees little benefit from source windows larger than 30 flits on most traffic patterns. As the network size is scaled up, the window size must increase in proportion to the network diameter to cover the longer ACK round-trip time. In a mesh topology, the diameter is proportional to the square root of the mesh size; thus, quadrupling a PVC network from 64 to 256 nodes requires doubling the source window to 60 flits.

Researchers have previously studied the use of preemption to overcome priority inversion in interconnection networks. Knauber and Chen suggest its use in wormhole networks for supporting real-time traffic [8]. Their work, however, does not consider impact on fairness, overall throughput, and recovery mechanisms. Song et al. also propose using preemption for real-time traffic [19]. Their scheme requires a dedicated FIFO at each router node where preempted packets are stored. The FIFO must have enough buffering to store a full-sized packet for each supported priority level, except the highest, requiring a significant storage overhead in systems with a large number of priority levels. Their work also does not consider fairness and other QOS-related issues.

Flow Tracking and Provisioning: Finally, PVC routers must track each flow's bandwidth utilization for scheduling and preemption purposes. While this requires additional storage, the presence of per-flow state at each router offers important advantages in network provisioning and bandwidth utilization. For instance, several threads from an application running on multiple cores can share the same flow identifier. The ability to combine multiple flows into one enables per-application bandwidth allocation, reducing management overhead when the thread count changes over the lifetime of the application. In addition, coarser bandwidth allocation granularity enables better bandwidth utilization by allowing communication-intensive threads of an application to recover idle bandwidth from less active threads.

3.2 QOS Particulars

3.2.1 *Preemption Throttling*

A common definition of priority inversion in a network is the presence of *one or more* packets of lower priority at a downstream node, impeding a higher priority packet's progress. A PVC system based on this definition experiences very high preemption rates under congestion, considerably degrading throughput as a result. To address this problem, we use an alternative definition that potentially sacrifices some degree of fairness in exchange for improved throughput. Specifically, priority inversion in a PVC network occurs when a packet cannot advance because *all* buffers (virtual channels) at the downstream port are held by packets of lower priority. Thus, as long as one or more downstream VCs belong to a packet of same or higher priority as the current one, preemption is inhibited. In addition, PVC employs three mechanisms for further controlling preemption aggressiveness and balancing fairness with throughput.

The first mechanism is the allocation of some *reserved bandwidth* per flow per each frame interval. The amount of reserved bandwidth, in flits, is a function of the frame size and the flow's reserved rate. Any flit within the reserved envelope is not subject to preemption, forming the basis for PVC's bandwidth guarantee.

The second mechanism for preemption throttling is based on reducing the resolution of bandwidth counters by masking out some number of lower-order bits via a programmable *coarsening mask*. Doing so reduces the resolution of the computed priority values, effectively forming coarser priority classes. Packets that map to the same priority class may not preempt each other.

The final preemption control technique built into PVC addresses a pathological case in which multiple retransmissions of a packet reduce a flow's priority by incrementing the bandwidth counters up to the preemption point. With each unsuccessful transmission attempt, the flow's priority is further reduced, compromising throughput. To avoid this pathology, PVC transmits the hop count up to the preemption point as part of the NACK sent back to the source node. In turn, the source embeds the count in a dedicated field of the retransmitted packet. This counter is decremented at each hop until it reaches zero and inhibits the update of the flow's bandwidth counter as long as it is non-zero.

3.2.2 *Guarantees*

PVC is able to make four important guarantees: minimum bandwidth, fairness, worst-case latency, and non-preemption within a flow. By combining the last guarantee with a deterministic routing function, the system can provide in-order delivery within a flow. In order for these guarantees to be met, a PVC system must comply with the following requirements:

1. No link in the system is overbooked. Thus, for every link, the sum of provisioned rates across all flows does not exceed 100%.
2. The number of reserved flits for each flow is no less than the size of the source window used for buffering outstanding transactions.
3. Resource arbitration collisions (multiple requesters with the same priority) are broken fairly (e.g., randomly). Similarly, when multiple packets at an input port have the same priority and one must be preempted, the selection mechanism is fair.

The OS or hypervisor must satisfy the first two requirements whenever the network is configured and rates are assigned to flows. The last requirement is ensured at design time. Note that the first requirement does not prevent flows from exceeding the assigned rate whenever idle network bandwidth is available, as rate enforcement occurs only under contention.

Minimum bandwidth: Each PVC flow gets a certain number of reserved flits, computed as a fraction of the frame size based on the flow's negotiated rate. These flits are not preemptable. They also cannot be blocked by packets from other flows that have exhausted their bandwidth reserve in the current frame, as preemption guarantees freedom from priority inversion. Finally, per the first requirement above, no link in the system is overbooked. Thus, all reserved flits that enter the system by the start of the frame are guaranteed to be delivered by the end.

Fairness: A PVC network distributes excess bandwidth in a fair, rate-proportional manner, choosing the flow with the lowest *relative throughput* (rate-adjusted bandwidth utilization) at each arbitration event. To resolve resource conflicts, PVC uses fair (per requirement 3) priority arbiters, described in Section 3.3. Note that the strength of this guarantee is a function of the resolution of the bandwidth counters used in priority computation.

Worst-case Latency: Once a packet enters a source window, PVC guarantees its delivery by the end of the following frame interval. The guarantee is a direct outcome of requirement 2 and the minimum bandwidth guarantee. Basically, any packet in the source window will be within the reserved bandwidth cap in the new frame, thus assuring its delivery in that frame.

Non-preemption within a flow: In PVC, two packets belonging to the same flow may never preempt each other. Monotonicity of the priority function guarantees freedom from priority inversion within a flow. Priority is computed by rate-scaling the flow's bandwidth utilization, a monotonically increasing function within a frame over a given link. Priorities, which are inversely related to bandwidth utilization, are thus monotonically decreasing. Therefore, for any pair of packets within a flow at a common arbitration point, the younger packet's priority is guaranteed to be no greater than that of the older packet, inhibiting preemption. The relation holds across frame boundaries, since clearing the bandwidth counters does not violate the monotonicity property, as long as the counters are cleared synchronously across the chip.

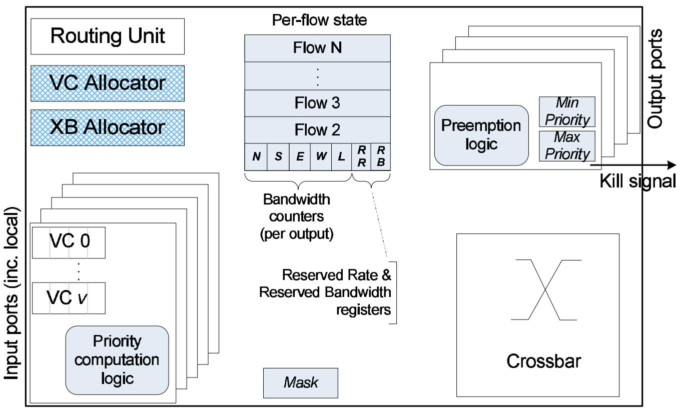


Figure 5: PVC router microarchitecture. Highlighted structures are new; crosshatched structures are modified relative to the baseline. *Italics* indicate a register name.

3.3 Microarchitecture

As a baseline, we adopt a generic NOC router architecture detailed by Peh and Dally [15] that has no QOS support. The router has a three-stage pipeline consisting of virtual channel allocation (VA), crossbar allocation (XA), and crossbar traversal (XT). The router uses route-lookahead to remove route computation from the critical path by performing it in parallel with virtual channel allocation [4].

Figure 5 shows the modifications to the baseline NOC router required to support PVC. Compared to the baseline, a PVC router needs priority computation logic, which includes the per-flow bandwidth counters and reserved rate registers. It also requires priority arbiters for virtual channel and switch arbitration instead of priority-oblivious matrix arbiters in the baseline router. Finally, a PVC router needs a preemption mechanism.

Priority computation logic: To support per-flow bandwidth tracking and rate-based arbitration, PVC routers must maintain per-flow bandwidth counters for each output port. In addition, each flow needs a reserved bandwidth register and a rate register, which can be shared across the ports. Finally, one *mask* register per router stores the bandwidth counter coarsening mask.

When a packet header arrives at a router's input port, priority computation logic uses the packet's flow identifier to access the bandwidth counter at the requested output port (computed at the previous hop). The access is a read-modify-write operation that increments the counter by the size of the packet, in flits. Concurrent with the update, the pre-incremented counter value is masked using the bandwidth counter coarsening mask and scaled by the flow's rate register. The resulting priority value is used for virtual channel and crossbar arbitration in subsequent cycles.

Unfortunately, the above approach adds a new pipeline stage for priority computation, increasing router delay. To remove priority calculation from the critical path, we propose using the priority value computed at the previous hop for virtual channel arbitration in the first cycle at a given node. Concurrent with VA, the flow updates its bandwidth counter and priority, using the updated priority for any subsequent VA retries and switch arbitration requests.

The resulting approach is safe if each source node has a unique flow identifier, as the flow's bandwidth utilization at the previous node is guaranteed to be no less than its usage through any output port at the current node. Thus, the new priority can never be lower than that of the previous hop. However, this technique is not safe if multiple sources share the same flow identifier, as the guarantee breaks down under a convergent traffic pattern. Fortunately, we can still use this approach with a minor modification: if a flow wins virtual channel arbitration in its first cycle but the computed priority is lower than the value used for arbitration, the winning request is not granted and must rearbitrate with the updated priority.

Priority arbiter: Allocation delay frequently determines the router's clock frequency in conventional networks, necessitating fast arbiters. PVC benefits from not requiring per-flow buffering, which keeps arbitration complexity modest even as the network size is scaled up. At the core of our arbiter is a low-latency comparator design proposed by Harteros and Katevenis, which uses a binary comparison tree with several acceleration techniques based on fast adder circuits [5]. We anticipate that a single-cycle priority arbiter based on this comparator design can be realized for NOC networks that have up to 64 virtual channels per router.

Preemption mechanism: To support preemption, PVC requires a modification to the virtual channel allocator that enables it to assign a VC to a requester even when none of the VCs at a downstream port are free. For that purpose, PVC maintains *Min priority* and *Max priority* registers at each output port, corresponding to the downstream virtual channel with the minimum and maximum priority value, respectively. In parallel with virtual channel arbitration, each requester's priority is compared to the value of the *Max priority* register. If the requester's priority exceeds *Max priority*, the virtual channel corresponding to *Min priority* is tentatively marked for preemption. VA logic assigns this virtual channel to the winning requester if none of the legal VCs are free. Of course, any packet within the reserved bandwidth envelope is not eligible for preemption.

In the next cycle, while the winning VC arbitrates for crossbar access, the resources associated with the preempted packet are released at the current node. If some part of the preempted packet has already been transferred, preemption logic sends a *kill* signal to the downstream node over a dedicated wire. The process of releasing resources held by the packet is repeated at each downstream hop until the header flit is encountered. Preemption of the header flit generates a NACK message to the source, which triggers a retransmission of the message.

3.4 Comparison to Prior Approaches

Table 1 compares three QOS schemes – WFQ, GSF, and PVC – on the feature set presented in Section 2.1. WFQ has excellent fairness guarantees and strong performance isolation that scale well with network size. However, it requires per-flow queueing and complex scheduling, resulting in large area and energy cost, with potentially high per-hop latency.

Table 1: Feature comparison of QOS schemes. '+' indicates *good*, 'o' is *fair*, and '-' is *poor*.

Feature	WFQ	GSF	PVC
a) Fairness	+	+	+
b) Isolation	+	o	o
c) Bandwidth utilization	+	o	+
d) Flexible bandwidth allocation granularity	+	-	+
e) Performance overhead	-	+	+
f) Delay proportional to bandwidth usage	+	-	+
g) Area overhead	-	-	+
h) Energy overhead	-	o	o
i) Performance scalability	+	o	o
j) Implementation complexity	o	+	o

GSF, on the other hand, has simple routers and modest frame management hardware, yielding low router delay and low implementation complexity. However, by pushing much of the scheduling responsibility into the terminals, GSF sacrifices throughput and has no flexibility in its bandwidth allocation. GSF's other shortfall lies in its poor suitability to fine-grained communication, as our experimental evaluation in Section 5 confirms. Because injection into the head frame is disallowed, this scheme introduces additional latency under contention. Thus, delay is unrelated to bandwidth usage. In fact, aggressive senders can temporarily block network access to sources with low injection rates, making the scheme susceptible to a denial-of-service attack.

PVC has good bandwidth efficiency, modest router complexity and low area overhead. A shortcoming of PVC compared to GSF is PVC's higher implementation complexity, which stems from the distributed protocols associated with preemption and ACK/NACK handling, as well as the logic for per-flow bandwidth tracking at each router node.

Both PVC and GSF provide only *fair* isolation of flows, which stems from their lack of per-flow buffering at each router node. They also have some undesirable energy overheads. In PVC, the overhead results from re-transmission of packets, flow table lookups, and the ACK network; in GSF, it is from source queue accesses. Finally, both approaches leave room for improvement with regard to performance scalability. As the network size is scaled up, GSF becomes increasingly prone to bandwidth coupling and other efficiency overheads that reduce its throughput. In PVC, more nodes increase the likelihood of contention which can cause preemptions and reduce throughput as a consequence.

4. METHODOLOGY

We use a custom cycle-precise simulator to evaluate three QOS schemes – WFQ, GSF, and PVC – on performance and fairness using the metrics from Section 2.2. As a baseline, we use a generic NOC with no QOS support. Details of the simulation infrastructure are summarized in Table 2.

Experiments: To evaluate the ability of different schemes to meet fairness guarantees while maximizing throughput, we use *hotspot* and *uniform random* synthetic traffic pat-

Table 2: Simulation methodology details; 64-node network (256 nodes).

Network	64 and 256 nodes, 16 byte link width, dimension order routing
Synthetic benchmarks	*hotspot* and *uniform random*. 1- and 4-flit packets, stochastically generated
PARSEC traces	*blackscholes, bodytrack, ferret, fluidanimate, vips, x264*: sim-medium datasets.
Baseline network	6 VCs per network port, 5 flits per VC; 1 injection VC, 2 ejection VCs
WFQ network	Per-flow queueing at each router node: 64 (256) queues, 5 flits per queue
GSF network	2K (8K) frame size, 6 (24) frames in-flight, 8 cycle frame reclamation delay; 6 VCs per network port with 1 reserved VC, 5 flits per VC; 1 injection VC, 2 ejection VCs
PVC network	50K cycle frame, 30 (60) flit source window 6 VCs per network port with 1 reserved VC, 5 flits per VC; 1 injection VC, 2 ejection VCs

terns whose network behavior is easy to understand, simplifying analysis. The *uniform random* pattern is also used to understand how well the different approaches scale when the network size is increased from 64 to 256 nodes.

Additionally, we assess the ability of GSF and PVC, the two schemes without per-flow buffering, to provide efficient fine-grained communication and performance isolation in the face of a denial-of-service attack. For this experiment, we dynamically combine traffic from PARSEC [1] application traces with synthetic "attack" traffic. The traces were collected using the M5 full-system simulator [2] executing PARSEC benchmarks in their entirety. Our infrastructure supports the six applications in Table 2; of these, we present results for *blackscholes, fluidanimate* and *vips* as a representative subset.

We also demonstrate PVC's ability to provide differentiated services by specifying a custom bandwidth allocation on a hotspot traffic pattern. Finally, we evaluate energy and storage overheads of different schemes. For energy analysis, we use modified versions of CACTI 6 [10] and ORION 2 [6].

For all configurations except PVC's *differentiated services* experiment, we assume that the actual traffic pattern is not known ahead of time and allocate all flows an equal share of network bandwidth.

WFQ configuration: Weighted Fair Queueing represents our ideal QOS solution with respect to fairness, performance isolation, and bandwidth utilization efficiency. Although we believe that WFQ is a poor fit for most NOC substrates due its high buffer requirements and complex schedule computation, we use it as a yard-stick for evaluating the two other QOS schemes. We idealize the WFQ routers by endowing them with an unrealistically low 3-cycle pipeline latency in the contention-free case – the same latency enjoyed by GSF and PVC routers that have simple schedule computation and no per-flow queueing.

GSF configuration: The baseline GSF configuration in the 64-node network features a 2000-cycle frame, 6 in-flight frames and an 8-cycle frame reclamation delay. The routers have 6 VCs per input port, with one reserved VC for the head frame. This configuration is similar to the default setup in the original paper by Lee et al. [9], except that we use a shorter frame reclamation delay and larger frame size, both of which improve GSF's performance. For the scalability experiment, we quadruple both the frame and window size to 8000 cycles/frame and 24 frames, ensuring good performance (as shown in Figure 2).

PVC configuration: In a PVC network, the choice of the frame size has important implications for both throughput and fairness. Larger frames are desirable to amortize various protocol overheads and minimize the effect of gently relaxed fairness settings. On the other hand, longer frames may result in greater drift among the different flows' bandwidth consumption, increasing the likelihood of preemption for flows with high bandwidth utilization. Empirically, we found 50,000 cycles to be a good frame length for balancing these conflicting requirements. We compute each flow's reserved bandwidth quota by multiplying its rate by 95% of the frame size. Five percent of frame bandwidth is uncommitted, allowing PVC to tolerate various overheads, such as router delays and ACK return latencies, without compromising bandwidth guarantees.

Our PVC baseline is configured to maximize fairness, potentially at the expense of throughput, using unmasked bandwidth counter values for priority computation. We also show the effect of relaxed fairness settings on select experiments by increasing the bandwidth counter coarsening mask to 8 and 16 bits. The latter configuration completely eliminates all preemptions by effectively masking out the full value of the bandwidth counter.

PVC's router configuration is similar to that of GSF with 6 VCs per port, including one for reserved flits. Unlike GSF, PVC does not require a reserved VC, since preemption guarantees freedom from priority inversion. However, we found that reserving a VC can eliminate some preemptions, reducing energy and latency cost of retransmissions. PVC uses 30-flit source windows for buffering outstanding packets for possible retransmission. In the 256-node network, we double the source window to 60 flits.

For the ACK network, we assume a simple design with single-flit messages and a single 10-flit buffer per input port. Message size is 16 bits in the 64 node network (20 bits with 256 nodes), which is sufficient to cover the address, index of the acknowledged packet, hop count to the preemption point (if applicable), and status (ACK or NACK).

5. EVALUATION

5.1 Quality-of-Service Results

First, we evaluate the QOS schemes on their ability to provide fair bandwidth allocation in a highly congested network and compare them to a system without QOS support. We use a *hotspot* traffic pattern with a corner node as the epicenter, and simulate 5 million cycles after the warm-up interval. Per Section 2.2, we are interested in *relative throughput* of different nodes. A tight distribution of throughput values across all flows is desirable, indicating a fair allocation of bandwidth to all nodes.

Table 3 shows the mean, minimum, and maximum throughput across the flows for each configuration. Both absolute throughput, in flits, and relative, as a percentage of the

Table 3: *Relative throughput* of different QOS schemes, in flits. Maximum aggregate throughput in the measurement interval is 5 million flits.

	total throughput (% of max)	mean	min (% of mean)	max (% of mean)	std dev (% of mean)
No QOS	4,999,972 (100%)	79,364	1,645 (2.1%)	100,966 (127.2%)	36,237 (45.7%)
WFQ	4,999,907 (100%)	79,363	79,333 (100.0%)	79,379 (100.0%)	10 (0.01%)
GSF	4,763,217 (95.3%)	75,607	75,433 (99.8%)	75,737 (100.2%)	56 (0.07%)
PVC	4,916,383 (98.3%)	78,038	77,042 (98.7%)	79,351 (101.7%)	607 (0.78%)

Table 4: Packet delay variation (jitter).

	mean (cycles)	max (cycles)	std dev
No QOS	264	20,675	214
WFQ	63	63	0
GSF	63	1,949	239
PVC	63	1,645	30

mean, are depicted. We also include the standard deviation from the mean as well as aggregate system throughput (first column). The latter is useful for assessing the efficiency of different schemes in utilizing available bandwidth.

In general, we see that all three QOS schemes are capable of fair bandwidth allocation. WFQ achieves the tightest distribution of bandwidth to nodes, benefiting from per-flow queueing and a sophisticated scheduling policy. GSF also performs very well, as source-based bandwidth reservation ensures equitable bandwidth allocation within each frame. However, GSF has the lowest aggregate throughput of any scheme, exposing inefficiencies in its bandwidth allocation. PVC has the most slack in its bandwidth distribution, but still offers good fairness with little deviation among nodes and standard deviation of just 0.8% of the mean throughput. Finally, a network with no QOS support offers high aggregate throughput but no fairness, with the node farthest from the hotspot receiving just 2.1% of the mean bandwidth.

Slack in PVC's throughput fairness has two primary causes. The first is due to fixed frame length, which allows some flows to be slightly ahead of their peers in bandwidth consumption by frame rollover. This favors nodes closer to the hotspot, as flits from different nodes progress in wavefronts. We attribute the second source of diminished fairness to our definition of priority inversion, described in Section 3.2.1, which inhibits preemptions whenever a downstream VC is held by a packet of same or higher priority as that of a requester upstream. Thus, multiple packets of lower priority can occupy other VCs at a given downstream port and make progress whenever the VC held by the higher priority packet experiences a stall.

We also measure the packet delay variation, or jitter, associated with different QOS approaches. We modify our experimental setup to generate only single-flit packets, thus simplifying analysis. During the measurement phase, we compute the delay difference for each pair of consecutive packets within a flow. We record all such differences, and use them to compute the metrics for each flow. The aggregate mean, max and standard deviation across *all* flows is presented in Table 4.

As expected, WFQ has the tightest distribution of jitter values, with virtually no variation across the flows or within any flow, benefiting from per-flow queueing coupled with a powerful scheduling function. GSF, on the other hand, shows the worst distribution of jitter values among QOS schemes due to unordered packet service within a frame. In contrast, PVC's standard deviation of jitter values is nearly eight times lower than GSF's, thanks to PVC's rate-based scheduling within a frame. Like GSF, PVC does not provide any jitter guarantees, as it is ultimately a frame-based approach. However, PVC's rate-based features can reduce packet delay variation in many cases, as this example shows.

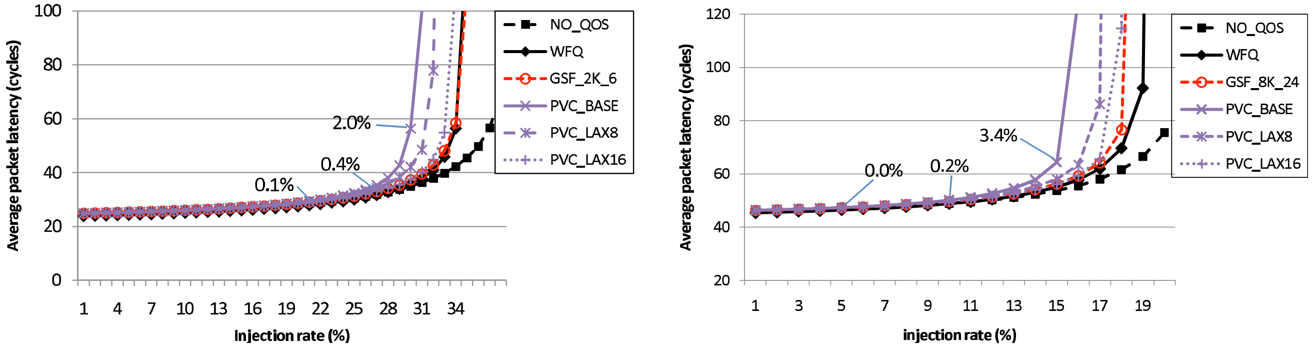
5.2 Throughput and Performance Scalability

We use a *uniform random* traffic pattern to assess the performance of the different QOS approaches in terms of latency and maximum throughput. This all-to-all workload is self-balancing, loading all bisection links uniformly and not favoring any particular node. In fact, no fairness mechanism is necessary to achieve equal bandwidth distribution among the network nodes. Thus, this pattern is effective at exposing the performance overheads associated with the respective QOS approaches.

Figure 6(a) shows the latency-throughput curves for the various schemes. Three PVC curves show the difference in throughput between our baseline (conservative) fairness setting and two relaxed configurations. Labels on the baseline PVC curve show the number of wasted hops due to dropped flits as a percentage of all hop traversals at 20%, 25%, and 30% injection rates. The drop rate peaks at 35% injection rate with 5.9% of all hop traversals resulting in a preemption (not shown in the figure).

The best throughput is achieved by the generic NOC due to high VC buffer utilization. In comparison, our WFQ implementation binds each flow to a dedicated queue, causing head-of-line blocking within each flow. GSF and the most lax PVC configuration (PVC_LAX16) have similar performance, but fall short of a QOS-oblivious network on throughput due to restrictions on VC utilization. In both of these schemes, multiple packets are not allowed to share a given virtual channel to avoid priority inversion. The NO_QOS configuration is not hampered by this restriction, allowing multiple packets to queue up behind each other in a VC, thereby improving buffer utilization and boosting throughput. The PVC network with the strictest fairness setting (PVC_BASE) degrades throughput by 10% relative to the laxest configuration (PVC_LAX16) due to preemptions.

Figure 6(b) shows the effects of scaling the network size to 256 nodes. The relative performance of different schemes remains unchanged. The fairest PVC configuration again exhibits some throughput loss due to dropped packets, which result in 3.4% of hops wasted at a 15% injection rate and saturate near 30% injection rate (not shown in figure) with 9.5% of all hop traversals leading to a preemption. One way to combat the performance overhead of packet drop is through relaxed fairness settings, which the figure confirms to be an effective way to improve throughput.



(a) 64-node mesh

(b) 256-node mesh

Figure 6: Performance of WFQ, GSF and PVC on uniform random traffic. Labels on the PVC_BASE curve show the number of retried hops as a percentage of total hop traversals.

5.3 Performance Isolation

To test the ability of NOC QOS schemes to provide performance isolation without per-flow queueing, we orchestrate a denial of service (DOS) attack against multi-threaded applications from the PARSEC suite. Figure 7 shows the configuration for this experiment. Black nodes in the left-most column are "aggressors" which send packets to the striped node in the lower-right corner of the mesh at an average rate of 20%. The rest of the nodes, including the striped node, belong to PARSEC threads. The aggressors may be a virus intentionally trying to disrupt network performance or may be benign threads accessing a shared cache bank at the noted location. We compare the average latency of PARSEC packets in this configuration to their latency executing alone on a substrate without any interference.

Our PVC baseline maps each core to a different flow with a distinct bandwidth allocation. However, PVC offers the capability to map all threads of an application to a common flow, allowing idle bandwidth from one application thread to be transparently used by another. This feature maximizes bandwidth utilization and reduces the likelihood of preemptions for communication-intensive threads. To evaluate the performance of PVC that maps all PARSEC threads to a single flow, we provisioned the flow with 7/8-ths of the network capacity, which is the sum of rates of individual PARSEC threads in our PVC baseline.

The results of the evaluation are presented in Figure 8. Five bars for each of the three benchmarks show the average latency of PARSEC packets. The first bar corresponds to a network with no QOS support; the second and third are for GSF and PVC baselines, respectively; the fourth bar shows the PVC configuration with PARSEC threads aggregated into a single flow; the last bar marks the performance of each PARSEC application executing with no attack traffic.

Without QOS support, "aggressor" threads overwhelm network's limited buffering, effectively preventing PARSEC packets from entering the network. The rate at which PARSEC packets are able to acquire network resources is lower than their injection rate; as a result, their delays grow very large due to our open-loop simulation methodology.

By comparison, both GSF and PVC offer some degree of performance isolation. In a PVC network, the maximum latency increase for an average PARSEC packet over an isolated execution is 22%. This is significantly better than the protection that GSF is able to offer, which increases the latency over 500% in the worst case. The reason for GSF's poor performance is its scheduling mechanism. Because GSF does not allow injection into the head (oldest) frame to accelerate frame reclamation, new packets are assigned to a future frame. This forces newly generated PARSEC packets to compete for buffer space with packets from aggressor threads that may belong to a more future frame, exposing PARSEC traffic to priority inversion. Importantly, GSF violates property (f) from Section 2.1, which states that delay should be proportional to bandwidth usage and explains GSF's poor performance in this scenario.

Table 5: Differential bandwidth allocation in PVC.

	min throughput	max throughput	standard deviation
10% allocation	98.8%	101.2%	1.6%
1% allocation	98.0%	104.5%	1.3%

Finally, we note that the aggregated PVC configuration (PVC_1FLOW) shows even better resilience to the attack than the PVC baseline, increasing PARSEC's average packet latency by just 6-7% over stand-alone execution. The improvement comes as a result of improved bandwidth utilization among PARSEC threads, as bandwidth reserved for threads that rarely communicate can be recycled among remaining threads.

5.4 Differentiated Services

To better support concurrent execution of multiple applications on a single substrate, PVC allows for differential bandwidth allocation to satisfy applications' diverse runtime requirements. To demonstrate PVC's ability to enforce a differential bandwidth allocation, we modify our *hotspot* configuration by provisioning each of four nodes with 10% of the bandwidth. These well-provisioned nodes are the three corners other than the hotspot, as well as a node in the center of the network at location $\langle 3, 3\rangle$. The rest of the nodes each get 1% of the bandwidth. The packet generators at the nodes exceed the provisioned rate, ensuring the relevance of the QOS mechanism.

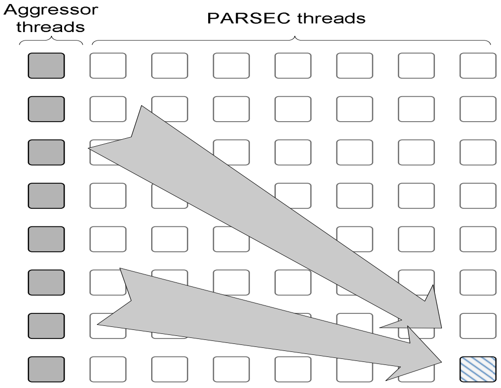


Figure 7: PARSEC setup

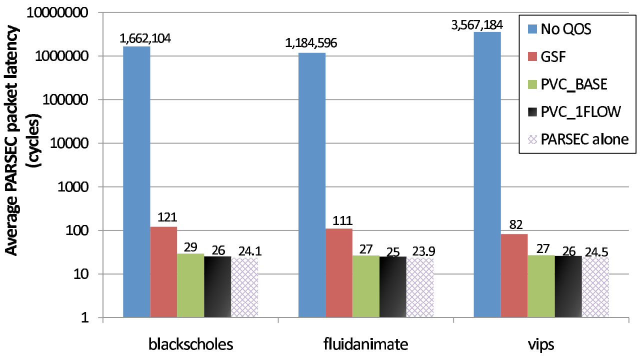


Figure 8: PARSEC results

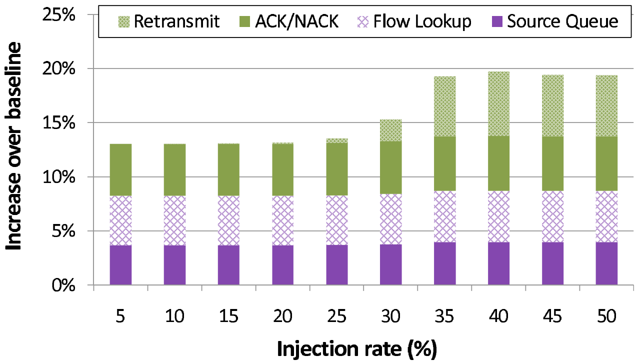


Figure 9: PVC Energy Overhead over a Generic NOC with no QOS support.

Table 6: Per-node storage requirements. Absolute values and relative to a generic NOC without QOS.

	64 nodes		256 nodes	
	bytes	relative	bytes	relative
No QOS	1,920	1	1,920	1
WFQ	5,120	2.7	20,480	10.7
GSF	33,920	17.7	129,920	67.7
PVC	3,376	1.8	6,564	3.4

Table 5 shows the standard deviation, minimum, and maximum throughput relative to the provisioned bandwidth for the two allocations. PVC is successful in differentiated bandwidth provisioning with a standard deviation of under 2% for both allocations. The 6.5% difference between the minimum and maximum throughput among nodes with a 1% allocation is a result of preemptions that arise due to certain nodes in the path of flows with high provisioned bandwidth. The fewer hops a flow with the low allocation shares with a high-allocation flow, the less likely it is to experience preemptions, resulting in higher throughput.

5.5 Energy

Figure 9 shows the energy expended in a 64-node PVC network relative to a baseline NOC with no QOS support on a *uniform random* traffic pattern. Four primary components of PVC's energy overhead are the source buffers, flow table lookups, ACK network, and retransmission of preempted messages.

Prior to saturation, PVC expends 13% more energy than the baseline due to source queue writes, flow table look-ups and updates, and ACK message overhead. As few preemptions occur before saturation, retransmissions incur very little energy overhead. As discussed in Section 5.2, the preemption rate peaks when the injection rate reaches 35% and holds steads thereafter, which Figure 9 confirms. In saturation, retransmissions are responsible for an additional 6% of the energy consumed. Other components of PVC's energy overhead also increase by 5-8% in saturation, contributing an insignificant amount to the overall energy budget.

WFQ, GSF, and PVC each have energy advantages and disadvantages. WFQ requires large per-flow buffers within each router, and a message must be written into and read from each of these as it traverses the network. GSF eliminates these buffers, but instead requires large source queues. Additionally, the large buffer capacity in both WFQ and GSF incur a non-trivial leakage energy penalty. PVC requires only small source buffers and also eliminates the per-flow buffers, giving it a potential storage energy advantage relative to the other two schemes.

5.6 Storage Overhead

We compare the storage requirements of different QOS schemes in 64- and 256-node networks in Table 6. For each configuration, both the absolute amount of storage, in bytes, and relative increase over a generic baseline with no QOS support is specified. For simplicity, we ignore the area overhead of the packet scheduling and buffer management logic, as well as the buffering at the local interfaces.

In WFQ, the primary source of storage overhead are the per-flow queues at each routing node. In contrast, GSF does not require per-flow buffering at the routers, instead necessitating large queues at the source nodes. PVC has three primary sources of area overhead: per-flow state in each router, buffering for outstanding transactions at each source interface, and flit buffers in the ACK network.

To store per-flow state, PVC needs bandwidth counters (one per flow) for each output port, as well as a reserved rate register and a reserved bandwidth register that may be shared across the ports, for a total of seven registers per flow. With a frame duration of 50,000 cycles or less, PVC requires 16 bits of storage per register.

In the 64-node network, PVC has 1.5 times less buffering than WFQ and 10 times less than GSF. In the larger network, PVC's storage footprint is 3 times smaller than WFQ's and 20 times smaller than GSF. Although the difference between WFQ and PVC may not appear significant, WFQ's scheduling and buffering overheads are in the critical path of each router node, which is undesirable in latency and energy sensitive on-chip interconnects.

6. CONCLUSION

Future CMP and SOC substrates will integrate hundreds or thousands of compute and memory elements on a single die. These elements will be connected by an on-chip network, which will shoulder the responsibility of providing fair access to shared resources while meeting performance, area, and energy targets. Prior network QOS schemes suffer from high buffer overheads, complex scheduling functions or poor bandwidth utilization, motivating us to propose Preemptive Virtual Clock, a novel QOS scheme specifically designed for on-chip interconnects. By combining features of frame-based and rate-based approaches, PVC provides strong guarantees, enforces flow isolation, and enables efficient bandwidth utilization with modest hardware cost and complexity. PVC does not require per-flow buffering, reducing router area and energy footprint. Priority inversion in a PVC network is averted through preemption of lower-priority packets. To support preemption, PVC requires a dedicated low-bandwidth ACK network and a small window of outstanding transactions at each node. Finally, PVC enables flexibility in network provisioning by allowing bandwidth to be allocated at any granularity from a single thread to an application to a user.

An evaluation of PVC in a 64-node network shows that it can guarantee fairness and provide differentiated services with low latency and good throughput. PVC also delivers strong performance isolation, demonstrated in a denial-of-service scenario against three PARSEC benchmarks. Results confirm that the average latency of PARSEC packets increases by less than 22% with PVC over their execution in isolation. In comparison, a previously proposed NOC QOS scheme called GSF causes latency to increase by up to 500%.

Acknowledgments

This research is supported by NSF CISE Infrastructure grant EIA-0303609 and NSF grant CCF-0811056.

7. REFERENCES

[1] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *International Conference on Parallel Architectures and Compilation Techniques*, October 2008.

[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The M5 Simulator: Modeling Networked Systems. *IEEE Micro*, 26(4):52–60, 2006.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, pages 1–12, New York, NY, USA, 1989. ACM.

[4] M. Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI Spider Chip. In *HOT Interconnects IV*, pages 141–146, 1996.

[5] K. Harteros and M. Katevenis. Fast Parallel Comparison Circuits for Scheduling. Technical Report TR-304, FORTH-ICS, March 2002.

[6] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design, Automation, and Test in Europe*, pages 423–428, April 2009.

[7] J. H. Kim and A. A. Chien. Rotating Combined Queueing (RCQ): Bandwidth and Latency Guarantees in Low-Cost, High-Performance Networks. In *International Symposium on Computer Architecture*, pages 226–236, 1996.

[8] K. Knauber and B. Chen. Supporting Preemption in Wormhole Networks. In *COMPSAC '99: 23rd International Computer Software and Applications Conference*, pages 232–238, 1999.

[9] J. W. Lee, M. C. Ng, and K. Asanovic. Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks. In *International Symposium on Computer Architecture*, pages 89–100, 2008.

[10] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. *International Symposium on Microarchitecture*, pages 3–14, December 2007.

[11] O. Mutlu and T. Moscibroda. Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors. In *MICRO*, pages 146–160, 2007.

[12] K. J. Nesbit, N. Aggarwal, J. Laudon, and J. E. Smith. Fair Queuing Memory Systems. In *MICRO*, pages 208–222, 2006.

[13] K. J. Nesbit, J. Laudon, and J. E. Smith. Virtual private caches. In *International Symposium on Computer Architecture*, pages 57–68, 2007.

[14] J. D. Owens, W. J. Dally, R. Ho, D. J. Jayasimha, S. W. Keckler, and L.-S. Peh. Research challenges for on-chip interconnection networks. *IEEE Micro*, 27(5):96–108, 2007.

[15] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *International Symposium on High-Performance Computer Architecture*, pages 255–266, January 2001.

[16] KC256. http://en.wikipedia.org/wiki/Kilocore.

[17] IP Packet Delay Variation Metric for IP Performance Metrics. RFC 3393. http://www.ietf.org/rfc/rfc3393.txt.

[18] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli. A 45nm 8-Core Enterprise Xeon Processor. In *International Solid-State Circuits Conference*, pages 98–99, February 2009.

[19] H. Song, B. Kwon, and H. Yoon. Throttle and Preempt: A New Flow Control for Real-Time Communications in Wormhole Networks. In *International Conference on Parallel Processing*, pages 198–202, 1997.

[20] D. Stiliadis and A. Varma. Design and Analysis of Frame-Based Fair Queuing: A New Traffic Scheduling Algorithm for Packet Switched Networks. In *SIGMETRICS*, pages 104–115, 1996.

[21] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, September/October 2007.

[22] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. In *SIGCOMM*, pages 19–29, 1990.