Appears in the Proceedings of the 6th Annual Workshop on the Interaction between Operating Systems and Computer Architecture

Topology-aware Quality-of-Service Support in Highly Integrated Chip Multiprocessors

Boris Grot
The University of Texas at Austin

Stephen W. Keckler
NVIDIA Research
The University of Texas at Austin

Onur Mutlu
Carnegie Mellon University

Abstract

Current design complexity trends, poor wire scalability, and power limitations argue in favor of highly modular on-chip systems. Today's state-of-the-art CMPs already feature up to a hundred discrete cores. With increasing levels of integration, CMPs with hundreds of cores, cache tiles, and specialized accelerators are anticipated in the near future. Meanwhile, server consolidation and cloud computing paradigms have emerged as profit vehicles for exploiting abundant resources of chip-multiprocessors. As multiple, potentially malevolent, users begin to share virtualized resources of a single chip, CMP-level quality-of-service (QOS) support becomes necessary to provide performance isolation, service guarantees, and security. This work takes a topology-aware approach to on-chip QOS. We propose to segregate shared resources, such as memory controllers and accelerators, into dedicated islands (shared regions) of the chip with full hardware QOS support. We rely on a richly connected Multidrop Express Channel (MECS) topology to connect individual nodes to shared regions, foregoing QOS support in much of the substrate and eliminating its respective overheads. We evaluate several topologies for the QOS-enabled shared regions, focusing on the interaction between network-on-chip (NOC) and QOS metrics. We explore a new topology called Destination Partitioned Subnets (DPS), which uses a light-weight dedicated network for each destination node. On synthetic workloads, DPS nearly matches or outperforms other topologies with comparable bisection bandwidth in terms of performance, area overhead, energy-efficiency, fairness, and preemption resilience.

1 Introduction

Complexities of scaling single-threaded performance have pushed processor designers in the direction of chip-level integration of multiple cores. Today's state-of-the-art commercial general-purpose chips integrate anywhere from four to one hundred cores [27, 24, 26], while GPUs and other specialized processors often contain hundreds of execution resources [21]. In addition to the main processors, these chips typically integrate cache memories, specialized accelerators, memory controllers, and other functional entities. As the degree of integration increases with each technology generation, chips containing hundreds and even thousands of discrete execution and storage resources will be likely in the near future.

The abrupt emergence of multi-core chips and their rapid proliferation have left researchers and industry scrambling for ways to exploit them. Two notable paradigms have arisen for monetizing CMPs – server consolidation and cloud computing. The former allows businesses to reduce server costs by virtualizing multiple servers on a single chip, thereby eliminating dedicated hardware boxes for each individual server. The latter enables delivery of various client services from remote (i.e., "cloud") servers. Since a single CMP can serve multiple users concurrently, hardware, infrastructure, and management costs are reduced relative to a model where each user requires a dedicated CPU.

Unfortunately, these novel usage models create new system challenges and vulnerabilities. For instance, in a consolidated server scenario, different priorities may be assigned to different servers. Thus, web and database servers for external customers could have a higher priority than intranet servers. But as multiple virtualized servers may be executing concurrently on a multi-core chip, traditional OS-level preemptive scheduling policies can fail at properly enforcing priorities of different VMs competing for shared resources.

In a cloud setting, multiple users may be virtualized onto a common physical substrate, creating a number of new concerns, including inadvertent interference among the different users, deliberate denial-of-service attacks, and side-channel information leakage vulnerabilities. Researchers have recently demonstrated a number of such attacks in a real-world setting on Amazon's EC2 cloud infrastructure, highlighting the threat posed by chip-level resource sharing on a public cloud [23].

Today's CMPs lack a way to enforce priorities and ensure performance-level isolation among the simultaneously-executing threads. Inter-thread interference may occur in any of the shared resources present on a CMP, including caches, memory controllers, and the on-chip network. Researchers have suggested using on-chip hardware quality-of-service (QOS) mechanisms to enforce priorities, limit the extent of interference, and provide guarantees for threads sharing a substrate [10, 15, 18]. While various shared resources have been studied as potential targets for QOS protection, little attention has been paid to the scalability of these techniques in CMPs with hundreds of cores, cache banks, and other discrete entities. In these highly-integrated CMPs, shared caches and the on-chip network emerge as potential scalability bottlenecks for chip-wide QOS support. Both are latency-sensitive distributed structures with a large number of nodes, requiring a light-weight, coordinated approach to fair capacity and bandwidth allocation.

In this work, we take a network-centric, topology-aware approach to chip-level quality-of-service. To reduce performance, area, and energy overheads of network-wide QOS support, we propose to isolate shared resources, such as memory controllers and accelerator units, into dedicated regions of the chip. Hardware QOS support in the network and at the end-points is provided *only* inside these regions. As shown in Figure 1(b), a richly-connected MECS topology [8] is used to connect each node to the shared region via a dedicated point-to-multipoint channel, ensuring physical isolation of memory traffic outside of the QOS-protected shared region. The majority of nodes on the chip, encompassing cores and cache memories, have no QOS support and enjoy significant savings in router cost and complexity.

The focal point of this paper is the organization of the shared region. Specifically, we consider the interaction between network topology and quality-of-service – a first such study, to the best of our knowledge. We evaluate three network topologies with preemptive QOS support [9] to understand their respective performance, fairness, and overheads. The topologies are mesh, MECS, and Destination Partitioned Subnets (DPS), a new topology we propose in this work. DPS uses a dedicated subnetwork for each destination node, enabling complexity-effective routers with low delay and energy overhead. All topologies show good fairness and experience little slowdown in the face of adversarial workloads with high preemption rates. On synthetic workloads, DPS consistently matches or outperforms mesh-based topologies in terms of performance, energy-efficiency, and preemption resilience. MECS has lower latency and better energy efficiency on long-distance communication patterns, but is inferior to DPS on shorter transfers.

In the remainder of this paper, Section 2 describes our proposed system architecture; Section 3 focuses on the organization of the shared region from the network perspective; Section 4 details the evaluation methodology; Section 5 presents the experimental results; Sections 6 and 7 describe related work and conclude the paper.

2 Topology-aware Quality-of-Service

2.1 Preliminaries

Our target system is a 256-tile CMP. Figure 1(a) shows the baseline organization, scaled down to 64 tiles for clarity. To reduce the number of network nodes, we employ four-way concentration as proposed by Balfour and Dally [1]. This organization reduces the number of network nodes to 64 by integrating four terminals[1] at a single router via a fast crossbar switch. The nodes are interconnected via a richly connected MECS topology [8]. MECS uses point-to-multipoint channels that fully connect a given node to other nodes along each of four cardinal directions. The topology leverages abundant wire resources found on a chip and is further aided by concentration, as the bandwidth across a concentrated node edge is greater than across a single tile.

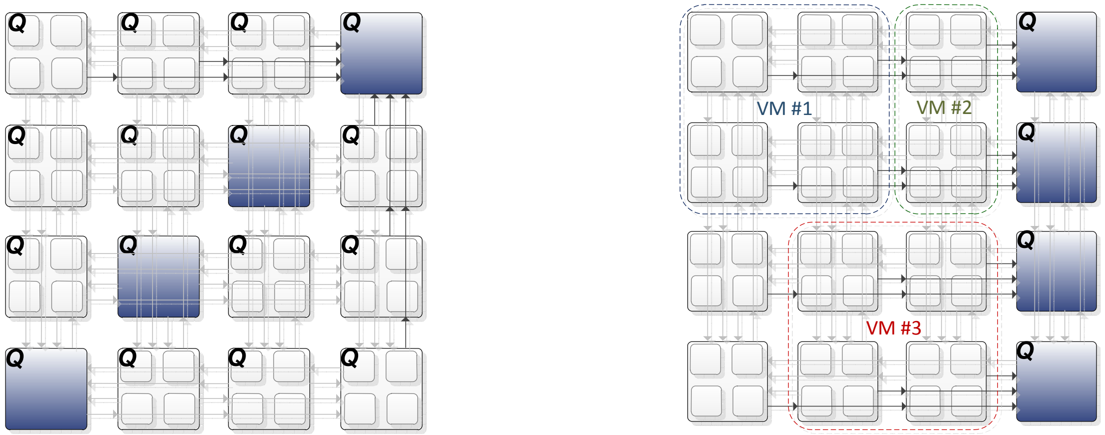
In the figure, shaded nodes correspond to shared on-chip memory controllers (MCs). The rest of the nodes integrate core and cache tiles. Cores can be identical or heterogeneous, and the ratio of core to cache tiles can vary. Assuming private last-level caches (an assumption we will later relax), the memory controllers and the on-chip network are the only shared resources in the figure. To ensure fair access to memory bandwidth for all tiles, each of the 64 on-chip routers needs to provide some QOS support, in addition to a QOS mechanism at the memory controllers. Assuming XY dimension-order routing, the routers must fairly provision row link bandwidth among the four terminals and regulate access to the shared column links, which become a contended resource as multiple tiles in a given row send packets to the same MC tile.

Unfortunately, per-hop QOS support is a weighty proposition. Traditional network QOS schemes require per-flow buffering at each router, which is undesirable in an on-chip setting due to the associated area and energy overheads. Recent work in on-chip QOS proposed relaxing the buffer requirements by using *preemption* to guarantee freedom from priority inversion [9]. While the scheme, called Preemptive Virtual Clock (PVC), significantly reduces cost over prior work, it nevertheless incurs certain overheads and preemption-induced performance degradations that may limit its ability to scale to large on-chip networks.

2.2 Topology-aware Architecture

In this work, we take a topology-aware approach to on-chip QOS. We observe that network QOS support is required to ensure fairness of only the *shared* links. In a

[1] A *node* refers to a network node, while a *terminal* is a discrete system resource, such as a core, cache tile, or memory controller, that has a dedicated port at a network node.



(a) Baseline QOS-enabled CMP

(b) Topology-aware QOS approach

Figure 1. 64-tile CMP with 4-way concentration and MECS interconnect. Light nodes integrate core and cache tiles; shaded nodes show memory controllers; 'Q' indicates hardware QOS support at the node.

MECS network, most sharing occurs within the column links, as row links are shared by just the four terminals of a source node. We can eliminate row-link QOS support by co-scheduling only "friendly" threads (i.e., those belonging to the same application or virtual machine) onto a node. To reduce the extent of QOS support in columns, we propose to isolate shared resources by placing them into one or more dedicated columns, called *shared regions*, and only provision QOS support inside these regions, eliminating cost and performance overheads of QOS in the rest of the network. Our approach relies on richly-connected MECS channels to ensure single-hop access into shared regions, thereby bypassing intermediate nodes and eliminating them as sources of interference in unprotected regions of the network. Figure 1(b) shows the proposed scheme.

Supporting efficient on-chip data sharing requires internode communication, which again raises the possibility of interference among the different network streams. To avoid re-introducing QOS support outside of the shared resource regions, we require the operating system (hypervisor) to place all threads of a given application (VM) in a convex region, called a *domain*, also shown in Figure 1(b). The resulting organization permits data sharing among the set of nodes making up the domain, as the convex shape ensures that all cache traffic stays within the allocated region. The scheme combines the benefits of increased capacity of a shared cache with physical isolation that precludes the need for cache-level hardware QOS support. An access to the shared region, such as a cache miss traveling to a memory controller, first traverses a non-shared MECS channel along the row in which the access originated before switching to a QOS-protected column containing shared resources.

Summarizing, our approach to chip-wide quality-of-service requires three components: a richly-connected topology that eliminates the need for QOS in non-shared regions, hardware QOS logic inside the shared regions, and operating system support. We now provide additional details on the role of each of these.

Topology: A topology with a high degree of connectivity is integral to our scheme, as it provides physical isolation for traffic between non-adjacent routers. We exploit the connectivity to limit the extent of hardware QOS support to a few confined regions of the chip, which can be reached from any node without going through any other node. With XY dimension-order routed MECS topology, the shared resource regions must be organized as columns in the two-dimensional grid of nodes to maintain the single-hop reachability property. We chose the MECS topology due to its attractive scalability properties and low router complexity; other topologies, such as the flattened butterfly [13], could also be employed.

Shared regions: The one or more regions containing shared resources serve two purposes. The first is to ensure fair access to shared resources, which requires hardware QOS support at both routers and end-points within each column. The second is to support inter-process or inter-VM communication, which also necessitates QOS protection at the routers, and is easily accommodated by our scheme.

To understand why inter-process/VM traffic must flow through shared regions, consider the case of VM #1 and VM #3 in Figure 1(b) sharing content. If the data originates at the top-left node of VM #1 and is destined for the bottom-right node of VM #3, packets will route in dimension-order toward VM #2, switching dimensions at its top node. With no QOS support at the turn node, inter-VM traffic can cause considerable interference with respect to local traffic of VM #2. To avoid such scenarios, we require all inter-process and inter-VM communication to transit via the QOS-equipped shared columns. While the requirement may result in non-minimal routes, as is the case in the example above, the expected frequency of such transfers is relatively low and they are typically not performance critical. As such, we anticipate latency and energy overheads incurred by additional network hops of inter-domain transfers to be modest.

OS support: We rely on the operating system to provide the following three services:

- Schedule threads from only the same application or virtual machine to run on a given node.
- Allocate compute and storage resources (core and cache tiles) to an application or virtual machine, ensuring that the domain complies with the convex shape property.
- Assign bandwidth or priorities to flows, defined at the granularity of a thread, application, or virtual machine, by programming memory-mapped registers at QOS-enabled routers and resources in shared regions.

As existing operating systems already provide scheduling services and support different process priorities, the additional requirements are very modest, requiring little developer effort and negligible run-time overhead.

3 Shared Region Organization

3.1 QOS support

Historically, high-performance network QOS schemes have required per-flow queues at each router node to isolate and schedule packets from different flows [7, 14, 5, 28]. As a result, these schemes had considerable router buffer requirements and scheduling complexity in larger network configurations, resulting in area, energy, and latency overheads that are undesirable in an on-chip setting. In response, researchers have recently proposed new approaches that try to address QOS requirements of on-chip networks [15, 9].

In this work, we adopt Preemptive Virtual Clock (PVC) as our preferred QOS mechanism [9]. PVC does not require per-flowing queuing, necessitating just enough virtual channels to cover the round-trip credit latency of a link. PVC features a light-weight packet prioritization function evolved from the Virtual Clock scheme [28]. Routers track each flow's bandwidth consumption, which is scaled by a flow's assigned rate of service to yield packet priority. To limit the extent to which a flow's present priority is affected by its past bandwidth consumption, all bandwidth counters are periodically cleared. The interval between two successive flushes is called a *frame*, whose duration determines the granularity of the scheme's guarantees.

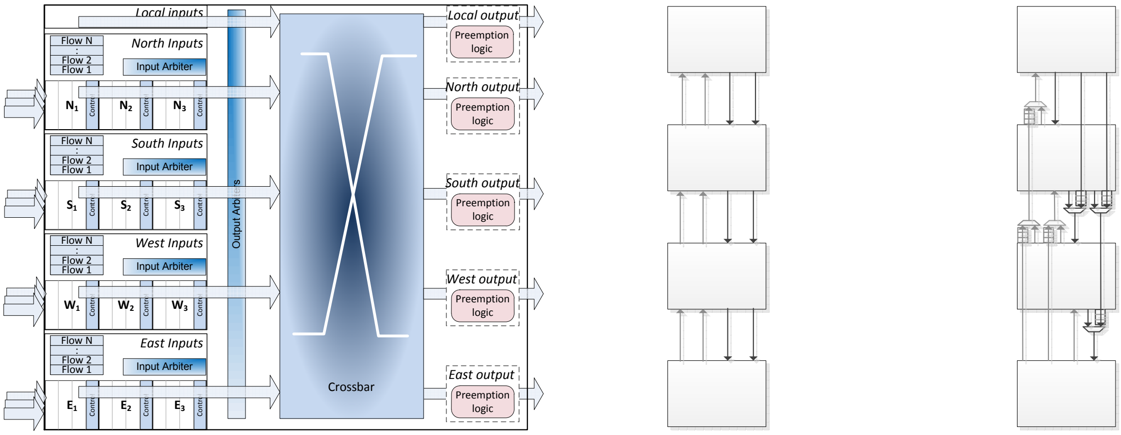
Since routers in a PVC network do not have dedicated buffer resources for each flow, lower priority packets may block packets with higher dynamic priority, a situation termed *priority inversion*. PVC detects priority inversion situations and resolves them through preemption of lower-priority packets. Discarded packets require retransmission, which is supported through a combination of a per-source window of outstanding packets and a dedicated ACK network used to acknowledge every delivered and discarded packet. In addition to the low-bandwidth, low-complexity ACK network, PVC requires per-flow state at each router node, whose size is proportional to the number of nodes on a chip and the number of router ports. Key limitation to PVC's performance scalability is its preemptive nature, which can reduce throughput at high packet discard rates. Although our work targets large CMP configurations, we only require QOS support in shared resource columns, thereby limiting resource and potential performance overheads of PVC.

3.2 Topologies

MECS is a natural consideration for the shared region as it is already deployed in the rest of the chip. To reduce complexity, MECS employs asymmetric routers with the number of row or column inputs equal to the number of nodes in a dimension (eight, in our case), but just two network outputs per dimension (one in each direction). Multiple input ports share a crossbar port, thereby reducing switch degree and arbitration complexity.

Figure 2(a) shows a PVC-enabled MECS router. While the MECS topology is a good fit for the main network, it may be less than ideal in the shared region. The reason is the increased complexity of a MECS router once QOS support is added to it. Long channel spans and the need to segregate flows necessitate considerable buffer and virtual channel (VC) resources to cover large round-trip credit latencies, even in the absence of per-flow queuing. In addition, scheduling complexity is quite high due to the presence of multiple VCs at each input port, multiple input ports from each direction, and multiple directions that may all be competing for a given output port.

In comparison, a mesh router has fewer input ports and short channel spans, leading to lower buffer requirements and simpler arbitration than its MECS counterpart. However, a basic mesh topology does not fully leverage the rich wire resources available on a chip, motivating researchers



(a) MECS preemptive router microarchitecture

(b) Mesh *x2* topology

(c) DPS topology

Figure 2. Shared region details.

to propose *replicated* networks [1]. Replication increases the channel count by the degree of replication at the cost of additional router resources. Although prior work argues that full replication that requires new router instances for the additional channels can significantly reduce crossbar complexity, it ignores the buffering, wire, and arbitration overhead of bypass paths to and from the terminals [8, 1]. In this work, we consider a variant of the approach that replicates the channels and associated router ports, but maintains a single monolithic crossbar at each node. Figure 2(b) shows an example network with twice the channel count of a basic mesh.

In addition to mesh and MECS topologies, we also consider a new network organization called Destination Partitioned Subnets (DPS). DPS uses a dedicated low-complexity network for each destination node; a 4-node DPS topology is shown in Figure 2(c). The motivation behind DPS is to combine low router complexity of the mesh topology with improved wire utilization and long-distance communication efficiency found in MECS.

A packet in a DPS network goes through routing, priority computation, and crossbar traversal only at source and destination nodes. Because each subnet maps to a unique destination node, once a packet enters a subnet, it does not need to be routed or switched to a different output port with respect to other packets until it reaches the destination – a subnet's end-point. Intermediate hops require only two input ports (network and local) and a single output port; as a result, a simple 2:1 mux suffices as a switch (Figure 2(c)). Flow tracking and priority computation are not required at intermediate hops either, eliminating the overheads of flow state queries and updates. In all, these simplifications enable a single-cycle router traversal for packets at intermediate DPS hops.

DPS source and destination nodes look similar to those in a mesh. Source-side, a larger number of output ports in a DPS router (one port per subnet) results in increased crossbar complexity. Tables containing flow state also have to be scaled up, since bandwidth utilization is maintained for each output port separately. Arbitration complexity is not affected, since the different output arbiters operate independently. Destination-side, DPS and mesh routers are virtually identical.

4 Experimental Methodology

We use an in-house simulator to evaluate the different topologies for the QOS-enabled shared region, which is embodied within a single column of an 8x8 grid of nodes in a large scale CMP. Figure 1(b) shows a diagram of a scaled down 4x4 grid with a similar organization. One column in the middle of the grid is devoted to shared resources with one terminal per node; the rest of the network employs 4-way concentration and a MECS interconnect. To focus our study, we examine only the network within the shard column. Our target is 32-nm process technology with on-chip voltage of 0.9 V.

Each router in the shared region has some topology-dependent number of column inputs, in addition to a terminal port and seven row inputs from MECS channels in east and west directions. Up to four MECS inputs (those arriving from the same direction) share a single crossbar port. Similarly, three ports are devoted to east, west, and terminal outputs, in addition to the north/south ports.

Table 1 summarizes the simulated configurations, consisting of mesh, MECS, and DPS topologies. We consider three mesh variants with 1x (baseline), 2x, and 4x replication. In all cases, we assume 128-bit channel width; thus, MECS, DPS, and mesh_x4 have equal bisection bandwidth, while mesh_x1 and mesh_x2 topologies feature more com-

Table 1. Shared region topology details

Network	8 nodes (one column), 16-byte links, 1 cycle wire delay between adjacent routers, DOR routing, virtual cut-through flow control
QOS	Preemptive Virtual Clock (50K cycle frame)
Benchmarks	*hotspot*, *uniform random*, and *tornado*; 1- and 4-flit packets, stochastically generated
Topologies	mesh_x1, mesh_x2, mesh_x4, MECS, DPS
mesh	6 VCs per network port, 2 stage pipeline (VA, XT)
MECS	14 VCs per network port, 3 stage pipeline (VA-local, VA-global, XT)
DPS	5 VCs per network port, 2 stage pipeline as source/dest (VA, XT), 1 stage pipeline at intermediate hops
common	4 flits/VC; 1 injection VC, 2 ejection VCs, 1 reserved VC at each network port

pact routers in exchange for less bisection bandwidth. Wire delay is one cycle between adjacent routers. All topologies use PVC for QOS support.

We faithfully model all aspects of each topology's router pipeline, which consists of virtual channel allocation (VA) and crossbar traversal (XT). We use virtual cut-through flow control [12], transferring an entire packet upon its acquisition of a virtual channel, and eliminating the crossbar arbitration stage as a result. Arrival of higher-priority packets does not interrupt an on-going transfer, but a preemption does. Due to the large number of ports and virtual channels, MECS routers require 2 cycles for arbitration; mesh and DPS arbitrate in a single cycle. All topologies enjoy a single-cycle crossbar traversal. As explained in Section 3.2, intermediate hops in a DPS network have just one cycle of router latency due to elimination of crossbar traversal. In all topologies, source hops require an additional cycle for route and priority computation; look-ahead routing and priority reuse [9] are subsequently employed to remove these stages from the critical path.

We assume two packet sizes, corresponding to request and reply traffic classes, but do not specialize the input buffers. With virtual cut-through switching, each virtual channel needs to hold the largest possible packet, which is four flits in our network. Worst-case traffic consists of a stream of single-flit request packets, each of which requires a separate VC. We use this as a guideline for provisioning the number of VCs at each input port based on the topology-specific round-trip credit latency; Table 1 lists the buffer organizations for each topology. In a MECS network, we do not employ location-dependent buffer sizing, provisioning buffers uniformly in all routers. In all topologies, we reserve one VC at each network port for rate-compliant traffic to reduce preemption incidence [9].

For area and energy evaluation, we use a combination of analytical models, Orion [11], and CACTI [17]. We modify Orion to more accurately model our crossbar configurations. In CACTI, we add support for modeling small SRAM arrays with data flow typical of a NOC router. We assume both input buffers and flow state tables are SRAM-based.

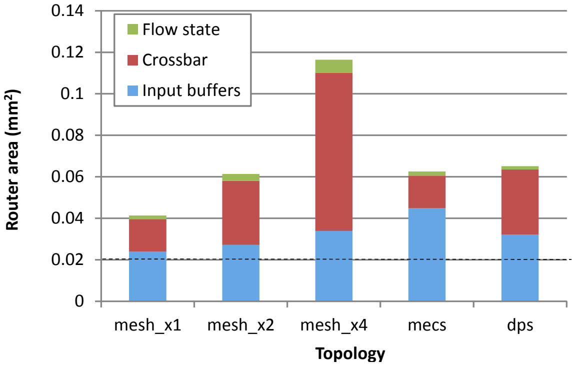


Figure 3. Router area overhead.

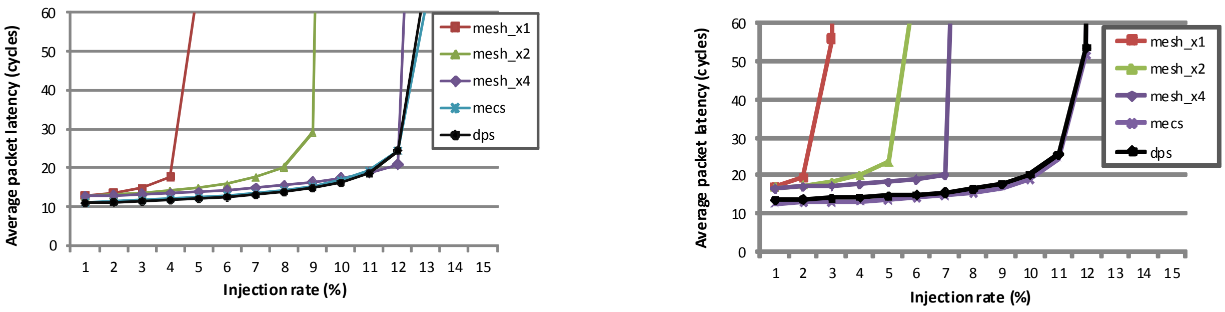
5 Evaluation Results

We evaluate the different topologies for the shared region (column) on area efficiency, latency and throughput, fairness, susceptibility to preemptions, and energy efficiency.

5.1 Area

Our area model accounts for three primary components of overhead: input buffers, crossbar switch fabric, and flow state tables. Figure 3 shows the router area overhead of different topologies. The dotted line shows buffer capacity allocated to row inputs, identical across all topologies.

Mesh_x1 is the most area-efficient topology as its routers have fewer ports than those in other organizations, leading to a compact crossbar and small buffer overhead. Mesh_x4, on the other hand, has the largest footprint, mostly due to a crossbar that is roughly four times larger than that in a baseline mesh. Crossbar area is proportional to the product of input and output port counts, which is 5x5 in mesh_x1 and 11x11 in mesh_x4, explaining the overhead of the 4-way replicated mesh. As expected, the MECS topology has the largest buffer footprint, but a compact crossbar thanks to just one switch port per direction (Figure 2(a)). DPS router's area overhead is comparable to that of MECS, as DPS has smaller buffer requirements but a larger crossbar due to the large number of column outputs at each node. Mesh_x2 has a similar router footprint to MECS and DPS but supports just half the bisection bandwidth. In all networks, PVC's per-flow state is not a significant contributor to area overhead.



(a) Uniform Random

(b) Tornado

Figure 4. Latency and throughput on synthetic traffic.

5.2 Performance

Figure 4 shows the performance of different schemes on two traffic patterns: *uniform random* and *tornado*. The former is benign, with different sources stochastically spreading traffic across different destinations, while the latter is a challenge workload for rings and meshes which concentrates the traffic from each source on a destination half-way across the dimension [3]. While these workloads are not directly correlated to expected traffic patterns in the shared region of a CMP, they stress the network in different ways and provide insight into the behavior of different topology options. Each curve corresponds to a topology and shows the load on the X axis and average packet latency on the Y axis.

Not surprisingly, the baseline mesh and mesh_x2 topologies show the worst throughput due to lower bisection bandwidth relative to the other network configurations. Mesh_x4 has competitive performance on random traffic, but is unable to load balance the tornado pattern. Both MECS and DPS show good scalability on tornado thanks to their ability to isolate traffic between each source-destination pair. On both workloads, DPS matches the throughput of MECS with just a fraction of the latter's buffer resources. In general, throughput is constrained in these topologies by asymmetry between the number of input and output ports at each node. Adding switch ports would improve throughput at the cost of additional switch area and energy.

On both traffic patterns, MECS and DPS enjoy lower average packet latency than mesh topologies. Meshes are slower due to the multi-cycle router traversals at each intermediate hop. A MECS network has deeper router pipelines than a mesh, but avoids all intermediate hops. The DPS topology has shallow mesh-like router pipelines at source and destination nodes with single-cycle intermediate hop traversals. On random traffic, MECS and DPS have nearly identical latency and are 13% faster than any mesh variant. On tornado, the longer communication distance favors MECS, as it is able to amortize larger router delay over longer flight time for a 7% latency advantage over DPS (24% versus mesh). While longer path lengths favor MECS, shorter communication distances favor DPS, which has lower router delay.

Preemptions rate (not shown) was measured to be quite low for all topologies. In saturation, the baseline mesh had the highest discard rate with nearly 7% of all packets replayed under random traffic; MECS had the lowest rate of just 0.04%. Mesh_x2, mesh_x4, and DPS replayed 5%, 0.1%, and 2% of their packets, respectively. By comparison, tornado traffic generated fewer preemptions for each topology. In general, topologies with greater channel resources show better immunity to preemptions on these permutations.

5.3 QOS and Preemption Impact

To measure the impact of the topology on fairness, we first use a hotspot traffic pattern, following the methodology of Grot et al. [9]. The terminal port of node 0 acts as a hotspot to which all injectors (including the row inputs at node 0) stream traffic. Prior work showed that without QOS support, sources closer to the hotspot get a disproportionately large share of the bandwidth, while distant nodes are essentially starved [15, 9].

Table 2 shows the results of the experiment. In general, all topologies provide good fairness on this workload, and the results are in line with the original PVC work. The maximum deviation from the mean across the topologies is 1.9%, corresponding to the DPS network. MECS has the strongest fairness with a maximum deviation of just 0.3% and standard deviation of 0.1%. Unlike performance results in the previous section, fairness seems to correlate with network buffer capacity, as topologies with more buffering provide better fairness.

Preemption rate is very low, as preemption-throttling mechanisms built into PVC are quite effective here. Key

Table 2. *Relative throughput* **of different QOS schemes, in flits (%).**

	mean	min (% of mean)	max (% of mean)	std dev (% of mean)
mesh_x1	4,184	4,134 (98.8%)	4259 (101.8%)	39.1 (0.9%)
mesh_x2	4,197	4,148 (98.8%)	4256 (101.4%)	27.6 (0.7%)
mesh_x4	4,221	4,167 (98.7%)	4278 (101.4%)	30.1 (0.7%)
MECS	4,193	4,180 (99.7%)	4203 (100.2%)	4.9 (0.1%)
DPS	4,188	4,125 (98.5%)	4266 (101.9%)	44.4 (1.1%)

among these is the reserved flit quota that each source is allocated. In each frame interval, the first N flits from each source are non-preemptable, where N is a function of the rate assigned to the source and the frame duration. With all sources transmitting, virtually all packets fall under the reserved cap, throttling preemptions.

To measure the impact of preemptions on fairness and performance, we crafted two adversarial workloads. Both are based on the hotspot traffic pattern, but with only a subset of sources communicating, ensuring that the reserved quota is exhausted early in the frame, triggering preemptions thereafter.

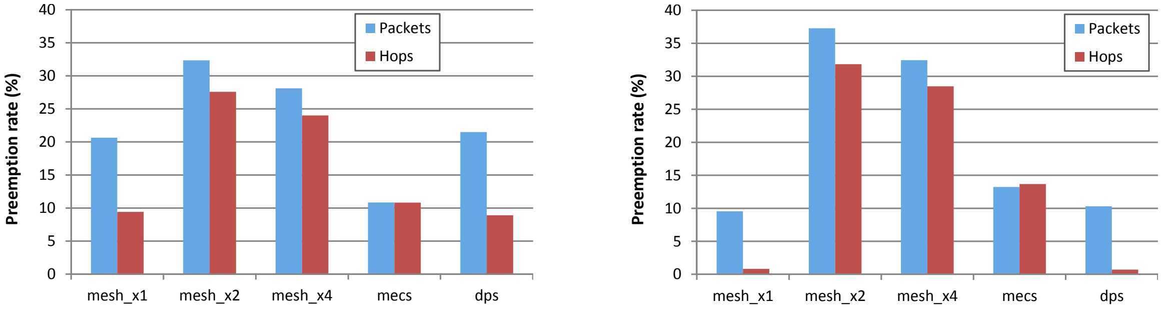
In Workload 1, only the terminal port at each of the eight nodes sends traffic toward the hotspot. With eight sources, the average injection rate must not exceed 12.5% to prevent saturation. We provision the sources with equal priorities, but widely different rates, ranging from 5% to 20%; the average is around 14%, guaranteeing high network contention. Under *max-min fairness*, a standard definition for fairness [3], sources with average injection rate under 12.5% should get their full requested share of network bandwidth; the rest of the bandwidth must be iteratively partitioned among the remaining communicating sources. In a PVC-enabled network operating in saturation, the arrival of a new packet at a source with a low injection rate will often trigger a sequence of preemptions as the packet travels toward the destination. Preemptions occur because the new packet has a higher priority relative to others, and when buffers are scarce, PVC discards lower priority packets to ensure forward progress by higher priority ones.

Figure 5(a) shows the percentage of all packets that experience a preemption and the total number of hop traversals that are wasted and need to be replayed. To normalize the comparison, we convert the hop count in a MECS network to the corresponding number of hops in a mesh based on the communication distance. Note that a single packet may be preempted multiple times; each such occurrence is counted as a separate event. In general, we see that the fraction of preempted packets is greater than the fraction of replayed hops, which occurs because most preemptions occur close to or right at the source node, before all of the victim's flits have been transferred. The sole exception to this trend is the MECS topology, whose fraction of discarded hops is equal to that of discarded packets. We believe this occurs due to the topology's rich buffer resources, which greatly diminish the likelihood of a packet being preempted in the middle of a transfer. Since replayed hops reduce network throughput and increase energy consumption, reducing hop-level replay is more important than lowering the packet discard metric. The mesh_x1 and DPS topologies incur the fewest number of replayed hops (9%), closely followed by MECS (10%). In DPS, all of the traffic is concentrated on a single subnet, mimicking the behavior and performance of the baseline mesh topology. The mesh_x2 and mesh_x4 topologies show the worst preemption rates, with over 28% of all messages (24% hops) replayed. The reason behind such poor performance is the thrashing that results as flows traveling on parallel networks converge at the destination node.

Figure 6(a) shows the impact of preemptions on the performance of different topologies by measuring the slowdown compared to preemption-free execution in the same topology with per-flow queuing. The slowdown is less than 5%, demonstrating that preemptions have small impact on the completion time of the workload. The figure also shows the deviation from the expected throughput based on max-min fairness; the thick blue bar shows the average across all nodes while the error bars plot the range of deviation for individual sources. All topologies show comparable behavior with the average deviation across all nodes under 1%. DPS enjoys the smallest range of deviations among individual sources, indicating good throughput fairness.

In constructing Workload 2, we attempted to stress the MECS topology, as it has significantly larger buffer resources compared to meshes and DPS. The general approach is identical to that of Workload 1; the only difference is in the set of injectors. For this workload, we activated all eight injectors at node 7 (the farthest from the hotspot) to pressure one downstream MECS port and one additional injector at node 6 to ensure contention at the destination output port.

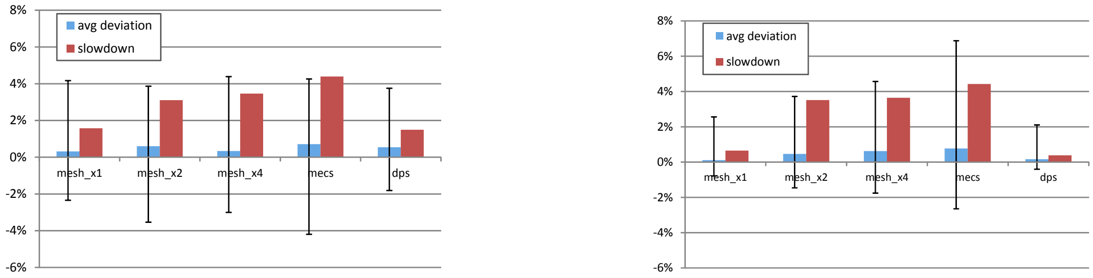
Figures 5(b) and 6(b) summarize the results on Workload 2. Compared to Workload 1, MECS sees only a slight increase in its preemption rate and unfairness, as measured by throughput of individual nodes versus expected throughput, demonstrating good resilience to potential attack patterns. Both mesh_x1 and DPS see their preemption rates drop sig-



(a) Workload 1

(b) Workload 2

Figure 5. Fraction of all packets that experience preemption events and hop traversals that are wasted as a result of preemptions.



(a) Workload 1

(b) Workload 2

Figure 6. Slowdown due to preemptions and deviation from the expected throughput.

nificantly, with few preemptions past the source nodes. The replicated mesh topologies, however, continue to experience high incidence of preemption, as packets that diverge at the source and travel on different networks can trigger preemptions once they reach the destination and compete for the same output port.

5.4 Energy efficiency

We evaluate the energy efficiency of different topologies by deriving the energy expended accessing the input buffers, traversing the crossbar, as well as querying and updating flow state at each network hop. We break down router energy overhead of different topologies based on the type of network hop – source, intermediate, or destination – since the cost of each varies. We also show the energy expended for a 3-hop packet traversal, roughly equivalent to the average communication distance on random traffic.

Figure 7 summarizes our findings. Although mesh topologies have modest per-hop energy overhead, they are least efficient on a 3-hop route requiring four router traversals. In contrast, MECS has energy-hungry routers that are undesirable with intra-node or nearest-neighbor traffic. Despite a small crossbar footprint, MECS has the most energy-hungry switch stage among the evaluated topologies due to the long input lines feeding the crossbar (see Figure 2(a)). However, MECS achieves good efficiency on 3-hop traffic by avoiding intermediate hops. DPS combines mesh-like efficiency at source and destination nodes with low energy expense at intermediate hops due to elimination of crossbar traversals, resulting in 17% energy savings over mesh_x1 and 33% over mesh_x4. On the 3-hop pattern, MECS and DPS have nearly identical router energy consumption. Longer communication distances improve the efficiency of the MECS topology, while near-neighbor patterns favor mesh and DPS configurations.

6 Related Work

A number of researchers have studied cache-level quality-of-service with the objective of reducing the impact of inter-application interference on performance [25, 10, 20]. We take a different view of QOS in this work, as our goal is providing cost-effective support for service-level agreement (SLA) guarantees and improved security through isolation of cache resources. Previous work in cache QOS also ignores the shared on-chip interconnect that is used to

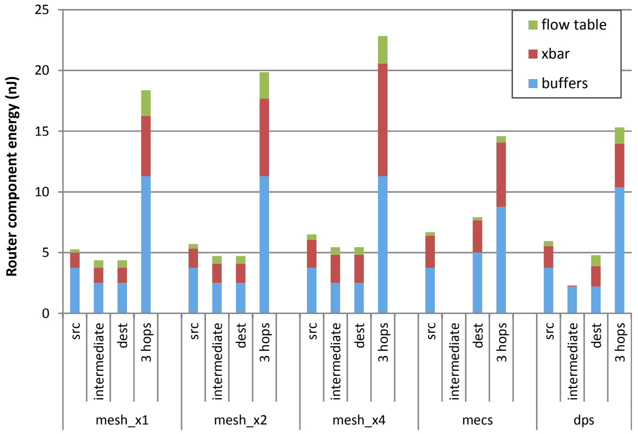


Figure 7. Router energy expended per flit for different topologies.

access the cache.

Recent work examining fairness and interference reduction in memory controllers [18, 19] is complementary to ours, since a comprehensive on-chip QOS solution requires quality-of-service support at shared end-points, including memory controllers. Other researchers proposed coordinated policies to partition cache space and memory bandwidth [6, 2]. None of these efforts consider the on-chip interconnect and its impact on end-to-end quality-of-service.

At the on-chip interconnect level, application-aware prioritization mechanisms [4] can improve performance metrics of multiple applications sharing a substrate, but do not provide hard guarantees. Rijpkema et al. proposed a router that combines guaranteed and best-effort service levels through a combination of circuit and wormhole switching [22]. This approach requires explicit segregation of memory traffic based on the expected service level, provides no guarantees to best effort traffic, and fails to take topology into account.

Finally, Marty and Hill advocate mapping VMs to dedicated regions on a chip to facilitate data sharing while reducing interference across VMs [16]. Our work relies on a similar approach, but goes farther by providing low-cost QOS support that could be used to provide service-level guarantees at the chip level. Coherence optimizations described by Marty and Hill are orthogonal to our work and may potentially benefit from efficient broadcast and multicast enabled by MECS.

7 Conclusion

The industry is on a trend of increasing the degree of integration in chip multiprocessors, with parts containing hundreds of cores, cache tiles, and other resources likely to appear in the near future. Meanwhile, increasing reliance on server consolidation and cloud-based services raises the possibility that multiple workloads, users, or even competing businesses will share resources on a common execution substrate. To enable performance isolation, security, and SLA guarantees on a die, CMPs must incorporate hardware QOS mechanisms. Unfortunately, quality-of-service support at each node of a highly-integrated CMP may be expensive due to area, energy, and performance overheads associated with today's QOS schemes.

In this work, we propose reducing the various costs of chip-wide QOS support via a topology-aware approach. Our scheme isolates shared resources in dedicated, QOS-enabled regions of the chip, allowing designers to forego QOS hardware in the larger part of the die containing cores and caches. We leverage the richly-connected MECS topology to provide single-hop access from any source node to the QOS-protected shared region with physical isolation from memory traffic of other nodes.

We evaluated several topologies as potential interconnect candidates inside the shared region with PVC-based QOS support. All topologies show good fairness, but differ widely in their preemptive behavior, performance, area overhead, and energy efficiency. The most promising configurations are based on MECS and Destination Partitioned Subnets (DPS), a new topology explored in this work. DPS uses a light-weight dedicated network for each destination node, combining low router complexity of mesh topologies with MECS-like energy and delay efficiency on multihop transfers. On synthetic traffic, DPS matches or outperforms mesh-based topologies in terms of performance, energy efficiency and preemption resilience. Compared to MECS, DPS has better energy efficiency and lower latency on shorter transfers, while MECS is superior on longer routes. These initial results are promising and motivate further research into the interaction between topology and onchip quality-of-service.

8 Acknowledgement

We wish to thank Naveen Muralimanohar for his help with CACTI, and Emmett Witchel for clarifying the mechanics of inter-process communication.

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

References

[1] J. D. Balfour and W. J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *International Conference on Supercomputing*, pages 187–198, June 2006.

[2] R. Bitirgen, E. Ipek, and J. F. Martinez. Coordinated management of multiple interacting resources in chip multipro-

cessors: A machine learning approach. In *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture*, pages 318–329, 2008.

[3] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[4] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das. Application-aware prioritization mechanisms for on-chip networks. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 280–291, 2009.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM)*, pages 1–12, 1989.

[6] E. Ebrahimi, C. J. Lee, O. Mutlu, and Y. N. Patt. Fairness via source throttling: a configurable and high-performance fairness substrate for multi-core memory systems. In *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 335–346, 2010.

[7] S. Golestani. Congestion-free communication in high-speed packet networks. *IEEE Transactions on Communications*, 39(12):1802–1812, Dec 1991.

[8] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In *Proceedings of the 15th International Symposium on High-Performance Computer Architecture*, pages 163–174, 2009.

[9] B. Grot, S. W. Keckler, and O. Mutlu. Preemptive virtual clock: a flexible, efficient, and cost-effective qos scheme for networks-on-chip. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 268–279, 2009.

[10] R. Iyer. CQoS: a framework for enabling QoS in shared caches of CMP platforms. In *ICS '04: Proceedings of the 18th Annual International Conference on Supercomputing*, pages 257–266, 2004.

[11] 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 *Proceedings of the Conference on Design, Automation, and Test in Europe*, pages 423–428, 2009.

[12] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[13] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 172–182, 2007.

[14] J. H. Kim and A. A. Chien. Rotating combined queueing (RCQ): bandwidth and latency guarantees in low-cost, high-performance networks. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 226–236, 1996.

[15] J. W. Lee, M. C. Ng, and K. Asanovic. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 89–100, 2008.

[16] M. R. Marty and M. D. Hill. Virtual hierarchies to support server consolidation. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 46–56, 2007.

[17] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 3–14, 2007.

[18] O. Mutlu and T. Moscibroda. Stall-time fair memory access scheduling for chip multiprocessors. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 146–160, 2007.

[19] O. Mutlu and T. Moscibroda. Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared dram systems. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 63 –74, 2008.

[20] K. J. Nesbit, J. Laudon, and J. E. Smith. Virtual private caches. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, pages 57–68, 2007.

[21] NVIDIA Fermi architecture. http://www.nvidia.com/object/fermi_architecture.html.

[22] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, 2003.

[23] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009.

[24] J. Shin, K. Tam, D. Huang, B. Petrick, H. Pham, C. Hwang, H. Li, A. Smith, T. Johnson, F. Schumacher, D. Greenhill, A. Leon, and A. Strong. A 40nm 16-core 128-thread CMT SPARC SoC processor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 98–99, 2010.

[25] G. E. Suh, S. Devadas, and L. Rudolph. A new memory monitoring scheme for memory-aware scheduling and partitioning. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, pages 117–128, 2002.

[26] Tilera TILE-Gx100. http://www.tilera.com/products/TILE-Gx.php.

[27] D. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, S. Chu, S. Islam, and V. Zyuban. The implementation of POWER7: A highly parallel and scalable multi-core high-end server processor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 102 – 103, 2010.

[28] L. Zhang. Virtual clock: a new traffic control algorithm for packet switching networks. *SIGCOMM Computer Communication Review*, 20(4):19–29, 1990.