J. Parallel Distrib. Comput. 72 (2012) 1401–1411

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

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

Globally Synchronized Frames for guaranteed quality-of-service in on-chip networks

Jae W. Lee [a,*], Man Cheuk Ng [b], Krste Asanović [c]

[a] Department of Semiconductor Systems Engineering, Sungkyunkwan University, Suwon, Gyunggi-do 440-746, Republic of Korea
[b] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[c] Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Received 23 July 2011
Received in revised form
1 January 2012
Accepted 29 January 2012
Available online 4 February 2012

Keywords:
Chip multiprocessors (CMP)
On-chip networks
QoS
Router

ABSTRACT

Future chip multiprocessors (CMPs) may have hundreds to thousands of threads competing to access shared resources, and will require quality-of-service (QoS) support to improve system utilization. This paper introduces Globally-Synchronized Frames (GSF), a framework for providing guaranteed QoS in on-chip networks in terms of minimum bandwidth and maximum delay bound. The GSF framework can be easily integrated in a conventional virtual channel (VC) router without significantly increasing the hardware complexity. We exploit a fast on-chip barrier network to efficiently implement GSF. Performance guarantees are verified by analysis and simulation. According to our simulations, all concurrent flows receive their guaranteed minimum share of bandwidth in compliance with a given bandwidth allocation. The average throughput degradation of GSF on an 8×8 mesh network is within 10% compared to the conventional best-effort VC router.

1. Introduction

Advances in fabrication technology allow the integration of many processors on a chip to form a chip multiprocessor (CMP), possibly in the form of a complex system-on-a-chip (SoC) with custom application accelerators. These platforms will be required to support a variety of complex application workloads, with possibly hundreds to thousands of concurrent activities competing for shared platform resources. Without effective quality-of-service (QoS) support, the gap between best-case and worst-case throughput will continue to grow, requiring overprovisioning and hence poor utilization of platform resources [13–15,18].

We believe that future integrated platforms must implement robust QoS support providing both *performance isolation* and *differentiated services*. Performance isolation is the property that a minimum level of performance is guaranteed regardless of how other concurrent activities access the same shared resources (e.g., preventing denial-of-service on a chip [23]). Differentiated services is the ability to allocate each resource flexibly among competing tasks.

Robust QoS support is only possible if all shared resources are managed together, as guaranteed service level for an application is determined by the weakest guarantee from any of its shared resources. For example, allocating a portion of off-chip memory bandwidth at a memory controller is ineffective if the on-chip network does not guarantee adequate bandwidth to transport memory requests and responses. Even in a case where the on-chip network is not a bandwidth bottleneck, tree saturation [28] can produce a tree of waiting packets that fan out from a hotspot resource, thereby penalizing remote nodes in delivering requests to the arbitration point for the hotspot resource.

In this paper, we present a new scheme, *Globally-Synchronized Frames (GSF)*, to implement QoS for multi-hop on-chip networks. GSF provides minimum bandwidth guarantees as well as bounded network delay without significantly increasing the complexity of the on-chip router. In a GSF system, time is coarsely quantized into "frames" and the system only tracks a few frames into the future to reduce time management costs. Each QoS packet from a source is tagged with a frame number indicating the desired time of future delivery to the destination. At any point of time, packets in the earliest extant frame are routed with highest priority but sources are prevented from inserting new packets into this frame. GSF exploits fast on-chip communication by using a global barrier network to determine when all packets in the earliest frame have been delivered, and then advances all sources and routers to the next frame. The next oldest frame now attains highest priority and does not admit any new packets, while resources from the previously oldest frame are recycled to form the new futuremost frame.

The system can switch frames at a rate that sustains any desired set of differentiated bandwidth flows with a bounded maximum

* Corresponding author.
E-mail address: leejw@csail.mit.edu (J.W. Lee).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.01.013

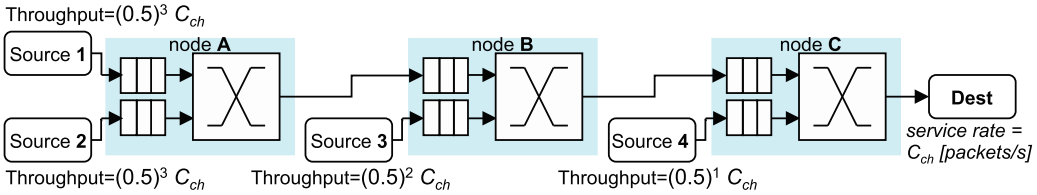
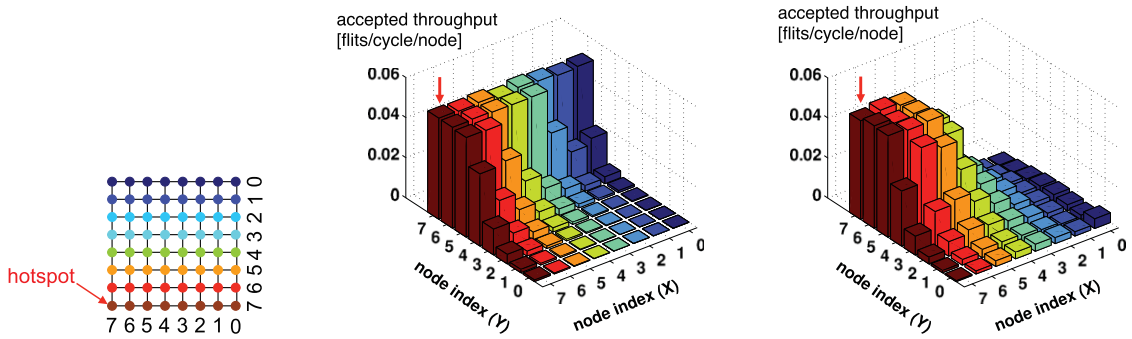


Fig. 1. Fairness problem in a simple three-node network with round-robin arbitration.



(a) Network configuration: 8 × 8 2D mesh.

(b) Round-robin scheduling with dimension-ordered routing.

(c) Round-robin scheduling with minimal-adaptive routing.

Fig. 2. Fairness problem in an 8 × 8 mesh network.

latency, provided that the pattern of injected packets in each frame does not oversubscribe the capacity of any network link. Note that bandwidth and latency are decoupled in this system, as multiple frames can be pipelined through the system giving a maximum latency of several frame switching times. The scheme does not maintain any per-flow information in the routers, which reduces router complexity and also avoids penalizing short-lived flows with a long route configuration step. The scheme supports bursty traffic, and allows best-effort traffic to be simultaneously supported with little loss of network utilization.

2. Background

2.1. Fairness problems in best-effort on-chip networks

Best-effort on-chip routers do not differentiate flows. (A flow is loosely defined as a distinct sequence of packets between a single source and a single destination.) Instead, most of them implement some variants of locally-fair round-robin arbitration [6,16] to provide fair service for input ports in allocating buffers and switching bandwidth.

However, local fairness does not imply global fairness. Fig. 1 illustrates such an example with locally-fair round-robin arbitration. This is a simple three-node network with four flows (whose sources are labeled 1 through 4) all having a shared destination (labeled "Dest"). Assuming the channel rate of C_{ch} [packets/s], the throughput of Flow 4 is half of the channel rate because it wins the congested output link with a probability of one half. The throughput of Flow 3 is a quarter of C_{ch} because it has to go through arbitration twice and has a probability of one half to win each arbitration, and so on. With round-robin arbitration, the degree of uneven bandwidth distribution increases exponentially with the network diameter.

Detailed network simulation with a realistic setup shows this phenomenon as well. In Fig. 2, all nodes generate traffic toward a hotspot shared resource located at (7, 7) (indicated by arrow), and the bar graph shows accepted throughput per node by the hotspot resource. In Fig. 2(b), locally-fair round-robin scheduling leads to globally-unfair bandwidth usage, penalizing remote nodes. The minimal-adaptive routing [6] shown in Fig. 2(c) does not resolve this problem, either. Hence, QoS support in on-chip networks is highly desirable for fair service across competing nodes.

2.2. Solution space for QoS support in on-chip networks

We divide the solution space for QoS into four quadrants according to two orthogonal criteria: scheduling granularity and extent of QoS. The first criterion is scheduling granularity. Each scheduling point could take either a *(sorted) priority-based* approach or a *frame-based* approach [27]. Priority-based schedulers [1,5–8,12,17,26,33] typically have per-flow queues. Competing requests' priorities are computed based on their classes, arrival time and the requesting flow's bandwidth share (e.g., virtual finish time in Fair Queueing [7]), and then compared to determine which request to service next. In contrast, frame-based schedulers [2,10,11,22,25,30,31] group a fixed number of time slots into a frame and control bandwidth allocation by giving a certain number of time slots per frame to each requester. The frame-based scheduler has per-frame queues instead of per-flow queues and services requests with the lowest frame number (i.e., earliest time) first.

The second criterion to classify possible approaches to QoS is the extent of QoS. The extent of a QoS mechanism determines both how much of the system is covered by the QoS mechanism and what type of QoS guarantees are made between end-points. In *component-wise* QoS approaches [2,7,8,11,12,17,22,25,30,31,33], a QoS mechanism is contained within each scheduling node, and QoS guarantees are made at a node level. Then a flow-level end-to-end QoS is guaranteed by construction. Alternatively, the operations of a QoS mechanism can be defined between end-points across multiple components, and its QoS properties described as such—we call this class of approaches *end-to-end* QoS [1,3,5,6,26,28]. One good example is age-based arbitration [1,5,6,26], where a router relies on a source-tagged global timestamp to calculate the priority of each packet.

Fig. 3 illustrates the four quadrants of solution space for QoS according to the two criteria. Generally, frame-based schedulers use much less hardware than priority-based schedulers while increasing delay bound. Likewise, end-to-end approaches can simplify

	Component-wise (better modularity & isolation)	End-to-end (simpler hardware & better composability)
Priority-based (lower delay bound)	• e.g., Fair Queueing	• e.g., Age-based Arbitration
Frame-based (simpler hardware)	• e.g., Stop-and-Go	• e.g., Globally Synchronized Frames (GSF) – our proposal

Fig. 3. Four quadrants of the solution space for QoS in the multi-hop on-chip networks.

the hardware compared to component-wise approaches while potentially sacrificing modularity. Among the four quadrants, *frame-based*, *end-to-end* approaches are a promising yet less explored direction toward efficient and robust QoS in an on-chip environment.

3. Globally-Synchronized Frames (GSF)

This section presents the design of GSF starting from an idealized deadline-based arbitration scheme. We then transform this scheme step-by-step into an implementable GSF queueing and scheduling algorithm.

3.1. Global deadline-based arbitration for bandwidth guarantees

GSF was originally inspired by deadline-based arbitration, which is a generalization of age-based arbitration [1,5,6,26]. In age-based arbitration, each packet carries a global timestamp, issued when the packet enters the network, and each arbiter (router) forwards the packet with the earliest timestamp first. Instead of using the timestamp, we allow each source to assign a deadline other than the current time. Our premise is that we can achieve a desired flow property, including guaranteed minimum bandwidth, by controlling deadline assignment, at least in an idealized setup.

One way to provide bandwidth guarantees is to assign the deadline for the n-th packet of Flow i (d_i^n) as follows:

$$d_i^n(\rho_i) = \text{MAX}[current\_time, d_i^{n-1}] + L_i^n/(\rho_i C)$$

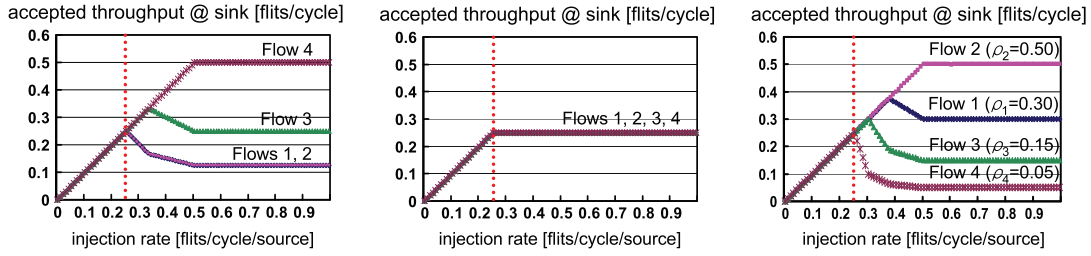
where ρ_i is the guaranteed minimum bandwidth of Flow i represented as a fraction of channel bandwidth C ($0 \leq \rho_i \leq 1$) and L_i^n is the length of the n-th packet of Flow i. This formula directly follows from what is known as the *virtual finish time* in Fair Queueing-variant algorithms [7,33]. The deadline specifies the time when a packet's last bit would arrive at the destination if the channel were infinitely divisible and shared by multiple packets simultaneously transmitting according to their guaranteed shares (ρ's), provided we ignore the network traversal delay (or zero-load latency).

Fig. 4 shows per-flow throughput in the simple three-node network in Fig. 1 to compare three arbitration schemes: (a) round-robin, (b) age-based and (c) deadline-based with the deadline assignment for bandwidth guarantees presented above. For simulation, we assume each input port has a perfect priority (sorting) queue with infinite capacity. Dotted vertical lines indicate minimum injection rate causing congestion. In locally-fair round-robin arbitration in (a), the throughput of a flow decreases exponentially as the number of hops increases. Age-based arbitration in (b), where deadline is assigned as network injection time, gives fair bandwidth allocation among all flows. With deadline-based arbitration in (c), we achieve bandwidth distribution proportional to the ratio of ρ's in face of congestion.

Although deadline-based arbitration provides minimum bandwidth guarantees to flows, there are several issues that make this scheme infeasible to implement. First, the scheme is based on perfect priority queues with infinite capacity. Second, there is a large overhead for sending and storing the deadline along with the payload data. *Baseline GSF* addresses these issues.

3.2. Baseline GSF

To make deadline-based arbitration practical, Baseline GSF adopts a frame-based approach [34] to approximate the ideal deadline-based arbitration. Fig. 5 shows a step-by-step transformation toward such an approximate implementation. Fig. 5(a) shows an ideal implementation of deadline-based arbitration introduced in Section 3.1. We first group all data entries having a range of deadlines (whose interval is F) to form a *frame*. Frame k is associated with packets whose deadline is in the range of $[kF + 1, (k + 1)F]$. The frame number k is used as a coarse-grain deadline and assigned to a frame buffer as shown in Fig. 5(b). By introducing frames, we enforce an ordering *across* frames but not *within* a frame because the service order within a frame is simply FIFO. The Baseline GSF arbitration is shown in Fig. 5(c), where we



(a) Round-robin (RR) arbitration. (b) Age-based arbitration. (c) Deadline-based arbitration.

Fig. 4. Per-flow accepted throughput with three arbitration schemes.

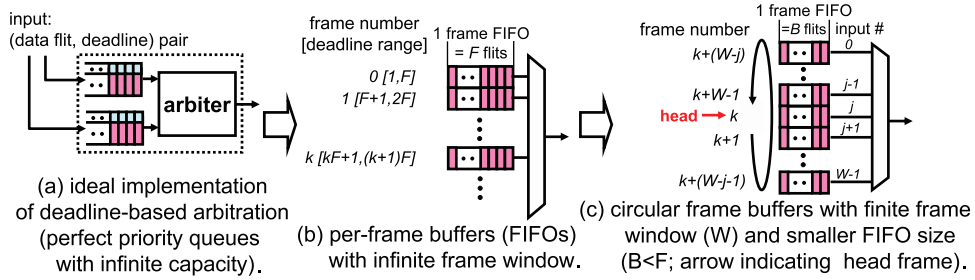


Fig. 5. Step-by-step transformation toward Baseline GSF.

Table 1
Variables and parameters used in GSF.

Variable	Range	Description
Global parameters and variables*		
W	$2 \cdots \infty$	Active frame window size
HF	$0 \cdots (W-1)$	Current head frame
F	$1 \cdots \infty$	Frame size in flits
B	$1 \cdots \infty$	Frame buffer depth in flits
C	$(0, 1]$	Channel bandwidth in flits/cycle
L^{MAX}	$1 \cdots \infty$	Maximum packet length in flits
$epoch$**	$0 \cdots \infty$	Current epoch number
e_k	$1 \cdots e^{\mathrm{MAX}}$	Interval of k-th epoch
e^{MAX}	$1 \cdots \infty$	Maximum epoch interval
T^{epoch}	$0 \cdots e^{\mathrm{MAX}}$	Epoch timer
Per-flow variables		
ρ	$0 \cdots 1$	Fraction of bandwidth allocated to Flow i (normalized to C)
R_i	$0 \cdots F$	Flit slots reserved for Flow i in a single frame
IF_i	$0 \cdots (W-1)$	Current injection frame of Flow i
C_i	$-L^{\mathrm{MAX}} \cdots R_i$	Available credit tokens for Flow i to inject flits to IF_i

* Global variable with a subscript i denotes a local copy of the variable maintained by Flow i.
** Not implemented in hardware.

have a finite active frame window having W frames (i.e., Frame k through $(k + W - 1)$) and each active frame has a dedicated frame buffer (FIFO) whose depth is B flits. The head pointer indicates which frame buffer is currently bound to the earliest frame (frame buffer j in the figure), which we call the *head frame*. Note that Baseline GSF in Fig. 5(c) asymptotically reduces to the ideal deadline-based arbitration in Fig. 5(a) as $W \to \infty$ and $F \to 1$.

Here is a brief sketch of how the GSF network operates. For each active frame, every flow is allowed to inject a certain number of flits, denoted by R_i for Flow i. As a flow consumes its flit allocation for each frame, the flow's frame number is incremented and it begins injecting flits into the next future frame, up to the maximum number of active frames (W) in the system. Unlike simple frame-based bandwidth allocation schemes (e.g., Stop-and-Go [10]), we allow W frames to overlap at any given time to service simultaneously up to WR_i flits from Flow i arriving in burst. The frame window size (W) is chosen to accommodate bursty traffic while preventing an aggressive flow from injecting too much traffic into the network.

Once a packet is injected into the network, it traverses the network using only the frame buffers of the frame it belongs to. Therefore, there is no possibility of a lower-priority packet blocking a higher-priority packet. Combined with earliest-frame-first scheduling for bandwidth allocation, the head frame is guaranteed to drain in a finite amount of time because only a finite sum of packets can be injected into a single frame by all flows. The drained head frame buffers across the entire network are reclaimed and allocated to a newer frame synchronously, which is called an *(active) frame window shift*.

We define an *epoch* as the period of time between adjacent frame window shifts, and the interval of the k-th epoch (i.e., the period of time when frame k is the head frame) is denoted by e_k. We also define $e^{\mathrm{MAX}} \equiv \max_{\forall k} e_k$. Table 1 summarizes variables and parameters used in the GSF algorithm. A more detailed description of each network component's operation is given as follows.

Packet injection process: Algorithm 1 describes a packet injection algorithm used by the Baseline GSF network. Flow i can inject packets into the active frame pointed to by IF_i as long as it has a positive credit balance for the frame ($C_i > 0$). The flow can go overdrawn, which allows it to send a packet whose size is larger than R_i. This is our design decision to maximize the network throughput at the cost of an increase in the fairness observation window size [20]. If the flow has used up all reserved slots in Frame IF_i, it can use reserved slots further in the future by incrementing IF_i by one (mod W) until it hits the tail of the active frame window (Lines 5–13). Once the flow uses up all reserved slots in the active frame window, it must stall waiting for a frame window shift to open a new future frame.

Switching bandwidth and buffer allocation: Frame buffer allocation is simple because every packet is assigned a frame at the source, which determines the frame buffer to be used by the packet at each node. In allocating switching bandwidth, we give the highest priority to the earliest frame.

Frame window shifting algorithm: Algorithm 2 shows an algorithm used to shift the active frame window. Source injection control combined with earliest-frame first scheduling yields a finite drain time for the head frame, bounded by e^{MAX}. Therefore, we shift the active frame window at every e^{MAX} cycles by default. The frame window shifting algorithm does not allow a flow to inject a new packet into the head frame (Lines 4–7) to keep e^{MAX} tight. Every flow maintains a local copy (T_i^{epoch}) of the global epoch timer (T^{epoch}) and decrements it at every clock tick (Lines 9–10). Once T_i^{epoch} reaches zero, all the flows synchronously increment the head frame pointer HF_i (mod W) to reclaim the frame buffer associated with the earliest frame.

The Baseline GSF network provides the following guaranteed bandwidth to Flow i if none of the physical channels along the path are overbooked (i.e., the sum of requested bandwidth for a channel does not exceed its bandwidth):

$$\text{Guaranteed bandwidth}_i = R_i / e^{\mathrm{MAX}}.$$

Although Baseline GSF provides guaranteed services in terms of bandwidth and bounded network delay (See Appendix A for a sketch of proof), there are several drawbacks to the scheme. First, frame buffers are underutilized, which degrades overall throughput. Second, it is difficult to bound e^{MAX} tightly, which directly impacts the guaranteed bandwidth. Even with a tight bound, it is too conservative to wait for e^{MAX} cycles every epoch because the head frame usually drains much faster. To address these issues without breaking QoS guarantees, we propose two optimization techniques: *carpool lane sharing* and *early reclamation of empty head frames*.

3.3. Carpool lane sharing: improving buffer utilization

Guaranteed bandwidth in Baseline GSF does not depend on the active frame window size (W). The multiple overlapping frames only help claim unused bandwidth to improve network utilization by supporting more bursty traffic. Therefore, as long as we provide a dedicated frame buffer for the head frame at each router, we do not compromise the bandwidth guarantees.

Algorithm 1 GSF packet injection algorithm into source queue for Flow i

Initialize: $epoch = 0$, $HF_i = HF = 0$
Initialize: $R_i = C_i = \lfloor \rho_i F \rfloor$
Initialize: $IF_i = 1$
1: AT EVERY PACKET GENERATION EVENT:
2: **if** $C_i > 0$ **then**
3: $SourceQueue_i.enq(packet, IF_i)$
4: $C_i = C_i - packet.size()$
5: **else** {used up all reserved slots in Frame IF_i}
6: **while** $(IF_i \oplus_W 1) \neq HF_i$ and $C_i < 0$ **do**
7: $C_i = C_i + R_i$
8: $IF_i = IF_i \oplus_W 1$
9: **end while**
10: **if** $C_i > 0$ **then**
11: $SourceQueue_i.enq(packet, IF_i)$
12: $C_i = C_i - packet.size()$
13: **end if**
14: **end if**

Algorithm 2 GSF frame window shifting algorithm

Initialize: $T_i^{epoch} = e^{MAX}$
Initialize: $HF_i = HF = 0$
1: FOR ALL FLOWS, AT EVERY CLOCK TICK:
2: **if** $T_i^{epoch} == 0$ **then**
3: $HF_i = HF_i \oplus_W 1$
4: **if** $HF_i == IF_i$ **then**
5: $IF_i = IF_i \oplus_W 1$
6: $C_i = MIN(R_i, C_i + R_i)$
7: **end if**
8: $T_i^{epoch} = e^{MAX}$
9: **else**
10: $T_i^{epoch} = T_i^{epoch} - 1$
11: **end if**

* $\oplus_W$: modulo W add

We propose carpool lane sharing to relax the overly restrictive mapping between frames and frame buffers. Now we reserve only one frame buffer to service the head frame (like a carpool lane), called the head frame buffer, but allow all active frames, including the head frame, to use all the other frame buffers. That is, any packet can occupy any frame buffer, except that the head frame buffers are reserved only for packets in the head frame. Each packet carries a $\lceil \log_2 W \rceil$-bit frame number (mod W) in its head flit, and the router services the earliest frame first in bandwidth and buffer allocation. Note that, with carpool lane sharing, a router no longer needs to have per-frame buffers since all frame buffers except the head frame buffer are shared by all frames opportunistically. In a minimal setup, a router may have only two frame buffers—one reserved for the head frame and the other freely used by all frames. According to our evaluation, carpool lane sharing significantly improves buffer utilization, hence increasing the overall throughput.

3.4. Early frame reclamation: increasing frame reclamation rate

One important factor affecting the overall throughput in the GSF network is the frame window shift rate. According to our analysis, only a small fraction of e_k's ever come close to e^{MAX}. This implies that the head frame buffer is often lying idle waiting for the timer to expire in each epoch.

Therefore, we propose to use a global barrier network to reclaim the empty head frame as quickly as possible. Instead of waiting for e^{MAX} cycles every epoch, we check whether there is any packet in the source or network buffers that belongs to the head frame. If not, we retire the head frame immediately and allocate its associated buffers to the new futuremost frame. Note early reclamation does not break the original bandwidth guarantees, because we always see a net increase, or at worst no change, in available flit injection slots. The barrier network is only a small fraction of the cost of the primary data network, as it uses a single wire communication tree and minimal logic.

4. Hardware implementation

Implementing GSF requires relatively minor modifications in a conventional virtual channel (VC) router. Fig. 6 shows a proposed GSF router architecture. Newly added blocks are highlighted while existing blocks are shown in gray. Various aspects and design issues in the GSF router follow.

Baseline VC router. We assume a three-stage pipelined VC router with lookahead routing [9] and credit-based flow control as our baseline. The three stages are next-hop routing computation (NRC) in parallel with virtual channel allocation (VA), switch allocation (SA) and switch traversal (ST).

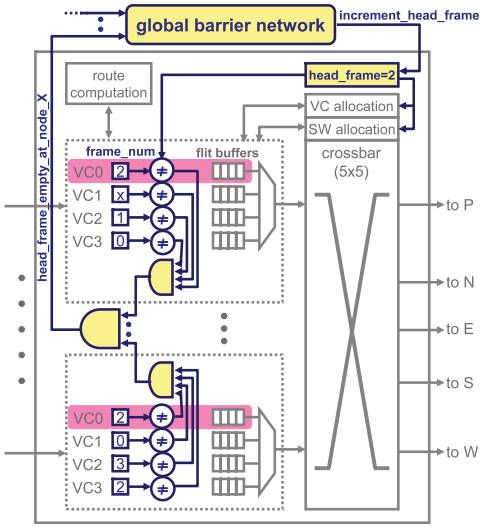


Fig. 6. GSF router architecture for 2D mesh network.

Added blocks. Each router node keeps a local copy of the global head frame (*HF*) variable. This variable increments (mod W) at every frame window shift triggered by the global barrier network. Each VC has a storage to maintain the frame number (mod W) of the packet it is servicing. The frame number at each VC is compared against *HF* to detect any packet belonging to the head frame. Then the global barrier network gather this information to determine when to shift the frame window.

Next-hop routing computation (NRC). To reduce the burden of the VA stage, which is likely to be the critical path of the router pipeline, we precalculate the packet priority at this stage. The packet priority can be obtained by (*frame_num* − *HF*) (mod W). The lowest number has the highest priority in VC and SW allocation. When calculating the routing request matrix, NRC logic is responsible for masking requests to VC0 from non-head frames, because VC0 is reserved for the head frame only. Then VC and SW allocators service a request with the highest priority.

Global barrier network. One way to achieve barrier synchronization is to use a fully-pipelined dimension-wise aggregation network [28]. In this network, assuming a 2D mesh, the center node of each column first collects the status of its peers in the same column. Then, it forwards the information to the center node of its row where the global decision is made. A broadcast network, which operates in reverse of the aggregation network, informs all nodes to rotate their head frame pointers (*HF*). For k-ary n-cube (or mesh) network, the latency of the synchronization will be $2n\lceil \frac{k-1}{2} \rceil$ cycles assuming one-cycle per-hop latency. Alternatively, we can implement a barrier network using combinational logic which might take multiple fast clock cycles to settle.

Parameter	Value
Baseline router parameters	
Routing	Dimension-ordered
Router pipeline (per-hop latency)	VA/NRC – SA – ST (3 cycles)
Credit pipeline delay (including credit traversal)	2 cycles
Number of VCs/channel	6
Buffer depth (B)	5 flits / VC
Channel capacity (C)	1 flit / cycle
GSF parameters	
Frame window size (W)	Same as number of VCs
Frame size (F)	2048 flits
Global barrier latency (S)	16 cycles in 8x8 2D mesh

Fig. 7. Default network parameters.

5. Credit token allocation

To specify requested bandwidth, one can use either a relative measure (e.g., 10% of link bandwidth) as in [24] or an absolute measure (e.g., 100 MB/s). If a relative measure ρ_i is given, R_i can be set to be $\rho_i F$. If an absolute measure BW (in flits/cycle) is used, R_i can be set to be $(BW * e^{\text{MAX}})$. e^{MAX} is a function of traffic pattern, bandwidth reservation, frame size, flow control overhead, global synchronization latency, and so on, and it is non-trivial to obtain a tight bound analytically. Hence, we rely on simulation to estimate a tight bound. That is, we keep the largest e^{MAX} observed for a given traffic pattern and bandwidth reservation, and add a safety margin to estimate the true e^{MAX}. Note that a tight estimate of e^{MAX} is not required to achieve high network utilization since GSF employs early frame reclamation.

Fair allocation of channel bandwidth is one important special case. If a channel is shared by n flows, the degree of congestion for this channel is n and each flow passing this channel receives $1/n$ of the service bandwidth. The reserved bandwidth for a flow is determined by the most congested channel (i.e., with the highest degree of congestion) on its path. Fig. 8 illustrates such bandwidth allocation for transpose traffic, where each source located at Node (x_i, y_i) generates traffic toward Node (y_i, x_i). Assuming x–y dimension-ordered routing, Fig. 8 shows a congestion map with each channel annotated with its degree of congestion. For example, for Flow 3 (F_3) which generates traffic from Node 3 ($s_3 = 3$) to Node 12 ($d_3 = 12$), the bottleneck channel is the one from Node 1 to Node 0 with the degree of congestion of three. Therefore, the flow's bandwidth reservation equals to one-third of the channel bandwidth ($\rho_3 = 0.333$). Note that different flows may have different bandwidth reservations. Algorithm 3 formalizes this to calculate R_i for each flow when bandwidth is fairly allocated.

6. Evaluation

We implemented a cycle-accurate simulator for 8×8 2D mesh network based on the *booksim* simulator [29]. Each run executes 0.5 million cycles unless the simulation output saturates early, with 50 thousand cycles spent in warming up. The parameters in Fig. 7 are used by default.

Seven traffic patterns are used where the destination of each source at Node (i, j) is determined as follows: hotspot (7, 7), transpose (j, i), nearest neighbor $((i + 1, j + 1) \pmod 8)$, uniform random (random(), random()), bit complement $(\bar{i}, \bar{j})$, shuffle $((2i+j/4, 2j+i/4) \pmod 8)$, and tornado $((i+3, j+3) \pmod 8)$. The injection slots per frame is calculated using Algorithm 3 except for uniform random traffic where $\lfloor F/64 \rfloor = 32$ flits per frame are allocated to each source (not to each source–destination pair).

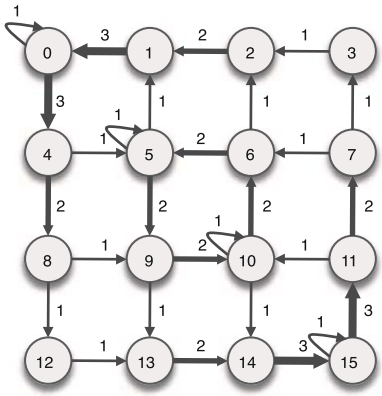


Fig. 8. Channel congestion map for transpose traffic.

6.1. Fair and differentiated services

Algorithm 3 GSF token allocation algorithm

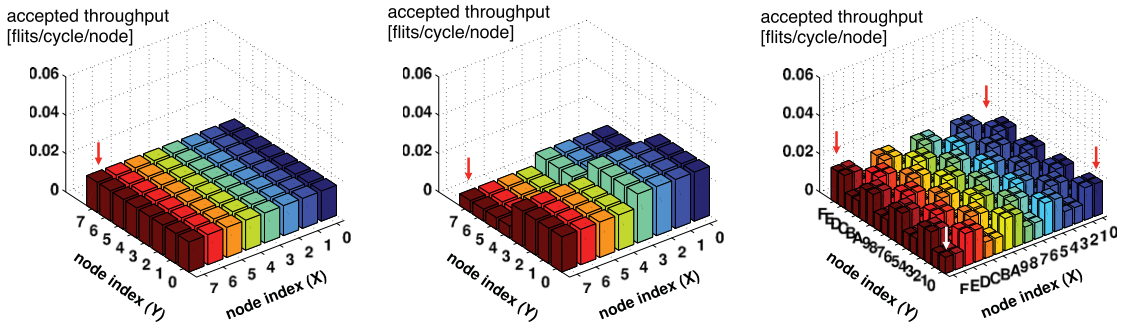
Initialize: F =(Frame size), N =(Number of nodes)
Initialize: $link\_load[N][N]$ =(array elements initialized to 0)
/* Calculate load for each link */
for $src = 0$ to $(N\text{-}1)$ **do**
  $dst = getDst(src)$
  $increaseLinkLoadByOneAlongThePath(link\_load, src, dst)$
end for
/* Calculate token allocation */
for $src = 0$ to $(N\text{-}1)$ **do**
  $dst = getDst(src)$
  $(in, out) = getMostCongestedLink(link\_load, src, dst)$
  $M_{src} = link\_load[in][out]$ /* M_{src}: degree of congestion */
  $R_{src} = F/M_{src}$ /* R_i for Flow i sourced at Node i */
end for

We first evaluate the quality of guaranteed services in terms of bandwidth distribution. Fig. 9 shows examples of fair and differentiated bandwidth allocation in accessing hotspot nodes using two different topologies. Fig. 9(a) and (b) illustrate QoS guarantees on an 8×8 mesh network and Fig. 9(c) on a 16×16 torus network. In both cases, GSF provides guaranteed QoS for each flow. Topology does not affect QoS guarantees as long as no link is oversubscribed.

Fig. 10 shows how bandwidth is distributed across concurrent flows for other traffic patterns and we confirm QoS guarantees are enforced in all cases. Note that the throughput of two flows may be different (e.g. transpose) because they may have different degrees of congestion on their paths.

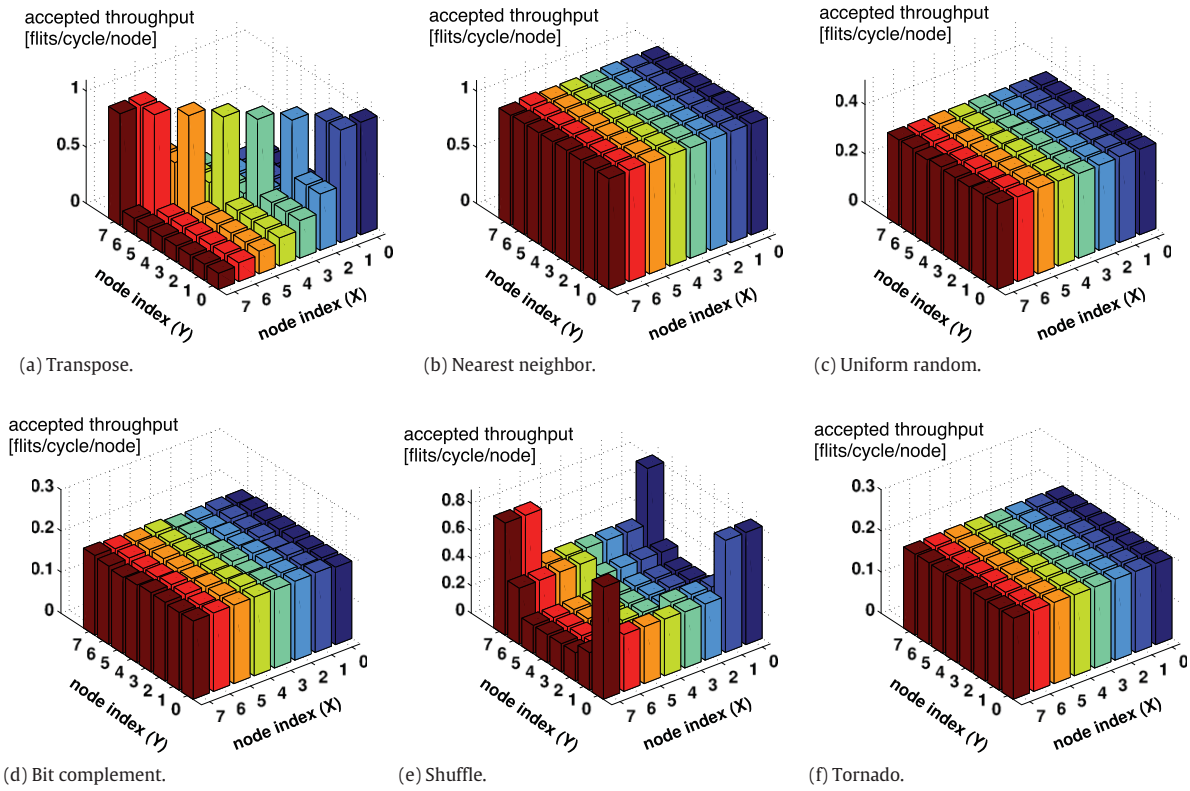
6.2. Cost of guaranteed QoS and tradeoffs in parameter choice

The cost of guaranteed QoS with GSF is additional hardware including an on-chip barrier network and potential degradation of average latency and/or throughput. Unless a router has *a priori* knowledge of future packet arrivals, it must reserve a certain number of buffers for future high-priority packets even if there are waiting packets with lower priorities. This resource reservation is essential for guaranteed QoS but potentially causes resource underutilization, degrading average-case performance. Therefore, it is our primary design goal to provide robust average-case



(a) Fair allocation: 8 × 8. (b) Differentiated allocation: 8 × 8. (c) Differentiated allocation: 16 × 16 torus.

Fig. 9. Fair and differentiated bandwidth allocation for hotspot traffic.



(a) Transpose. (b) Nearest neighbor. (c) Uniform random.

(d) Bit complement. (e) Shuffle. (f) Tornado.

Fig. 10. Fair bandwidth allocation for various traffic patterns.

performance over a wide range of network configurations and workloads.

Fig. 11 shows the average latency versus offered load over six traffic patterns. For each traffic pattern, we consider a best-effort allocation scheme (iSlip [21]) and GSF with various synchronization costs: 1 (GSF/1), 8 (GSF/8) and 16 cycles (GSF/16). We first observe that the GSF network does not increase the average latency in the uncongested region. The network saturation throughput is degraded negligibly except for bit complement traffic with 9.5% degradation. This is caused by underutilization of the head frame VC (VC0) and source throttling effect of finite frame window.

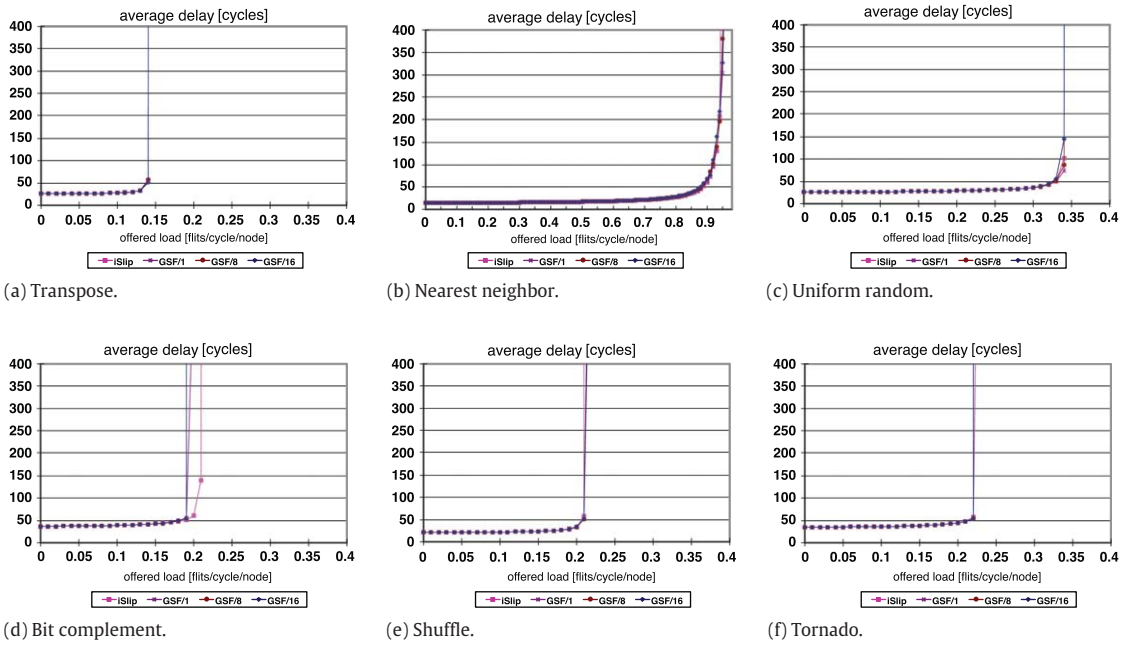
Fig. 12 explains the impact of these two factors on average accepted throughput. With a small number of VCs (e.g., $V = 2$), reserving VC0 only for the head frame severely degrades throughput in GSF. As the number of VCs increases, the gap narrows. In this network configuration, 4 or more VCs are desirable to achieve comparable throughput to the baseline VC router. Increasing W to be larger than $2V$ gives only marginal throughput gain.

7. Conclusion

This paper introduces Globally-Synchronized Frames (GSF) to provide guaranteed QoS from on-chip networks in terms of minimum bandwidth and maximum delay bound. The GSF algorithm can be easily implemented in a conventional VC router without significantly increasing its complexity. This is possible because the complex task of assigning priorities to packets is pushed out to the source injection process at the end-points. The global orchestration of the end-points and intermediate nodes is made possible by a fast on-chip barrier network. Our evaluation of the GSF network shows promising results. We believe that this kind of flexible, low-cost QoS support will be a necessary addition to highly-distributed future multicore processors.

Appendix A. Sketch of proof for minimum bandwidth guarantees and maximum delay bound

The proof sketch for the guaranteed bandwidth in Section 3.2 is simple. Flow i can inject R_i flits into each frame, and the network



(a) Transpose. (b) Nearest neighbor. (c) Uniform random.

(d) Bit complement. (e) Shuffle. (f) Tornado.

Fig. 11. Average packet latency versus offered load with various traffic patterns.

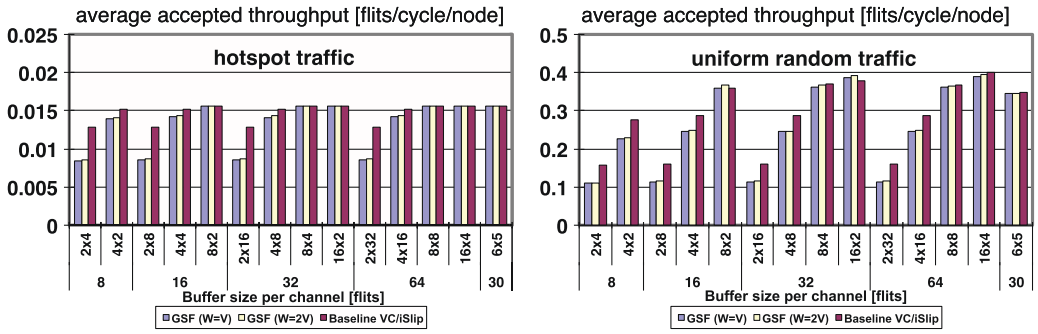


Fig. 12. Tradeoff in buffer organization with hotspot and uniform random traffic.

opens a new frame every e^{MAX} cycles. Because the network does not drop any packets and has a finite buffer size, the guaranteed bandwidth holds. In addition, the worst-case network delay is bounded by We^{MAX} because a packet injected in k-th epoch must be ejected from the network by the beginning of $(k+W)$-th epoch.

Appendix B. Admission control

Admission control is a software process that should guarantee that no channel in the network is oversubscribed. That is, suppose $S_c = \{i_1, i_2, \ldots, i_n\}$ is a set of flows that pass through Channel c. Then $\forall$ Channel c in the network, $\sum_{i \in S_c} R_i \leq F$ must hold in GSF. This inequality condition states that the sum of requested bandwidth from all the flows sharing a channel must not exceed the service capacity of the channel.

A request for a new flow is accepted only if all the channels along the path of the flow have enough extra bandwidth to accommodate the request. If a new flow enters into a previously reserved channel, the OS may need to redistribute the excess injection slots according to its excess bandwidth sharing policy. Tessellation OS describes a similar admission control mechanism in [4]. Alternatively, greedy allocation to maximize network utilization is also possible, where each flow can be granted more than the minimum number of slots required whenever possible. Since the maximum number of flits in flight from Flow i at any given time is upper bounded by WR_i, reserving more slots generally leads to higher sustainable throughput for the flow. In an extreme case, the first flow can greedily reserve the maximum number of available injection slots on its path. When a second flow arrives, the OS renegotiate bandwidth distribution among the two flows according to its resource management policy.

Note that this kind of bandwidth redistribution/renegotiation can be easily done in a GSF network since GSF does not require any explicit channel setup, and so only the R_i control register at each source must be changed. If there are multiple clock domains, possibly with dynamic voltage–frequency scaling (DVFS), any channel c should provide at least the sum of guaranteed bandwidths on the channel to preserve QoS guarantees.

Appendix C. Sensitivity analysis on frame size

In Fig. C.13, we explore the choice of frame size (F). A long frame (whose size is $\geq$1000 in this configuration) amortizes the overhead of barrier synchronization and effectively increases the size of injection window to support more bursty traffic, which is likely to improve the network throughput. The downside is

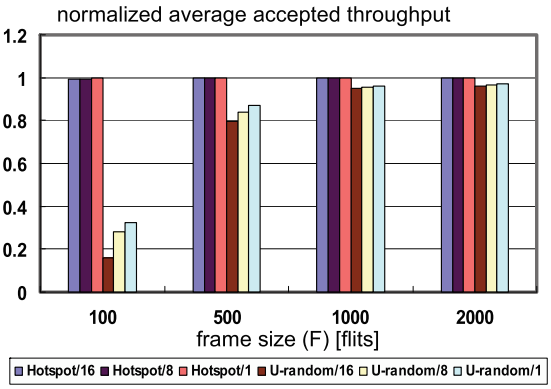


Fig. C.13. Throughput of GSF network normalized to that of the baseline VC router with variable *F* (frame size). Two traffic patterns (hotspot and uniform random) and three synchronization costs (1, 8 and 16 cycles) are considered.

larger source buffers and potential discrimination of remote nodes within a frame. The choice depends on workloads, synchronization overhead and system size.

Appendix D. Related work

We briefly survey proposals for QoS in the context of on-chip and off-chip networks. We classify each proposal to one of the four quadrants according to the two criteria discussed in Section 2.2.

(Weighted) fair queueing [7]/*Virtual Clock* [33] (priority-based, component-wise). They are developed for QoS in long-haul IP networks where large buffers are available. These achieve fairness and high network utilization, but each router is required to maintain per-flow state and queues which would be impractical in an on-chip network.

Preemptive Virtual Clock (PVC) [12] (priority-based, component-wise). It is similar to Virtual Clock but is implemented on an on-chip network. To reduce the buffer overhead, routers do not have per-flow queues. Instead, a router simply removes the lowest priority packet from the network when it is blocking a higher priority packet because of lack of buffer. In this case, an NACK will be sent to the source through a special network so that it can resend the killed packet in the future. PVC achieves comparable fairness but lower network utilization comparing to Virtual Clock. It is because some bandwidth is used to resend killed packets. While PVC requires less buffer overhead than Virtual Clock, each PVC router is still required to maintain per-flow state such as bandwidth counters and reserved rate registers.

Multi-rate channel switching [30] (frame-based, component-wise). The source rate of a flow is fixed at an integer multiple of a basic rate before the source starts transmission and remains unchanged for the duration of the flow. Because of the fixed rate, it cannot claim unused bandwidth efficiently, which leads to low network utilization.

Source throttling [28] (N/A, end-to-end). It dynamically adjusts the traffic injection rate at a source node primarily for congestion control. This keeps the network from suffering overall throughput degradation beyond the saturation point, but does not provide QoS to individual flows.

Age-based arbitration [1,5,6,26] (priority-based, end-to-end). It is known to provide strong global fairness in bandwidth usage in steady state and reduce standard deviation of network delay. Each packet (or flit) carries information to indicate its age, either a counter updated at every hop [26], or a timestamp issued when the packet first enters the network from which age can be calculated by subtracting from the current time [6]. The oldest packet wins in any arbitration step. This approach lacks flexibility in bandwidth allocation because it does not allow for asymmetric resource allocation, and requires sophisticated logic to handle aging, arbitration and counter rollover.

Rotating Combined Queueing (RCQ) [17] (priority-based, component-wise). It is designed for a multiprocessor and provides predictable delay bounds and bandwidth guarantees without per-flow queues at intermediate nodes (though it still maintains per-flow statistics).

Each packet in a RCQ network is assigned a local frame number using per-flow statistics upon arrival at every node on the path. The idea of rotating priorities in a set of queues is similar to GSF, but GSF further simplifies the router using global information, which is only feasible in an on-chip environment. Unlike RCQ, the frame number in the GSF is global, which eliminates expensive book-keeping logic and storage at each node.

MetaNet [25] (frame-based, component-wise). MetaNet implements a frame-based end-to-end QoS mechanism in an ATM network. Two independent networks servicing constant bit rate (CBR) and variable bit rate (VBR) traffic respectively, share physical links to improve the link utilization. Each source is allowed to inject up to a certain number of packets per frame into the CBR network and the rest of packets are injected into the VBR network. Packets carry a frame number, which is used at a destination to reorder packets taking different paths. The CBR network provides QoS guarantees in terms of minimum bandwidth and maximum delay.

The core idea of MetaNet is to create a distributed global clock (DGC) across the entire network using a distributed clocking protocol. They claim that a DGC running at 1 MHz is feasible for a 100-node system with realistic network parameters, which gives enough resolution for their frame-based QoS. However, the proposal has drawbacks. First, each router should detect malicious changes of the global clock values to police network traffic. Second, the DGCs are only frequency locked, and not phase locked and the link propagation delay on the link is not an integer number of time frames. Therefore, partitioning of the incoming stream of packets into frames should be done carefully to achieve deterministic QoS guarantees. Third, the destination requires a large reorder buffer because traffic is decomposed into the two networks and merged at the destination.

MediaWorm Router [32] (priority-based, component-wise). The MediaWorm router aims to support multimedia traffic in a multiprocessor cluster, including constant bitrate (CBR), variable bitrate (VBR), as well as best-effort traffic. It uses Fair Queueing [7] and Virtual Clock [33] for packet scheduling, which require to maintain expensive per-flow state and queues.

Æthereal [11] (frame-based, component-wise). It uses pipelined time-division-multiplexed (TDM) circuit switching to implement guaranteed performance services. Each QoS flow is required to explicitly set up a channel on the routing path before transmitting the first payload packet, and a flow cannot use more than its guaranteed bandwidth share even if the network is underutilized. To mitigate this problem, Æthereal adds a best-effort network using separate queues, but this introduces ordering issues between the QoS and best-effort flows.

SonicsMX [31] (frame-based, component-wise). It supports guaranteed bandwidth QoS without explicit channel setup. However, each node has to maintain per-thread queues, which make it only suitable for a small number of threads (or having multiple sources share a single queue).

Nostrum NoC [22] (frame-based, component-wise). It employs a variant of TDM using virtual circuits for allocating bandwidth. The virtual circuits are set up semi-statically across routes fixed at design time, and only the bandwidth is variable at runtime, which is only suitable for application-specific SoCs.

MANGO clockless NoC [2] (frame-based, component-wise). It partitions virtual channels (VCs) into two classes: Guaranteed

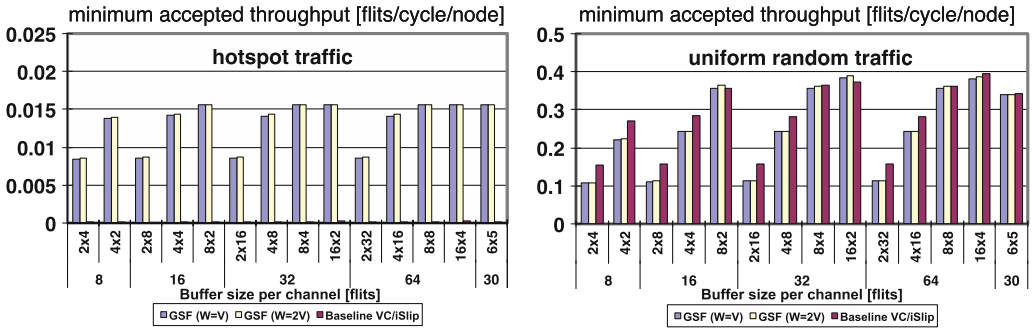


Fig. E.14. Minimum accepted throughput with various buffer organizations and frame window sizes. VC buffer configuration is given by $V \times B$. Frame window size (W) is assumed to be V or $2V$.

Service (GS) and Best-Effort (BE). A flow reserves a sequence of GS VCs along its path for its lifetime. Therefore, the number of concurrent GS flows sharing a physical channel is limited by the number of GS VCs (e.g., 8 in [2]).

Clockless NoC by Felicijan and Furber [8] (priority-based, component-wise). It provides differentiated services by prioritizing VCs. Though this approach delivers improved latency for certain flows, no hard guarantees are provided.

QNoC [3] (N/A, end-to-end). Its goal is to provide "fair" services in accessing a hotspot resource, and it takes a source regulation approach. The QNoC approach requires each source to fetch credit tokens from the destination (hotspot) node before sending out payload packets. It requires only minimal modifications to network routers because most of the intelligence is at end nodes.

However, there are several issues with their approach. First, it does not provide guaranteed QoS. In their approach, the credit recharge is basically asynchronous across all the sources and it is not clear what type of QoS is guaranteed. Second, it requires a more sophisticated secondary network (either logical or physical) for credit token request/reply not to slow down the source injection process. Finally, it is more suitable for application-specific SoCs than general-purpose platforms. To set up (and tear down) a QoS flow, the source needs to explicitly request it to the hotspot destination before sending out the first payload packet. It is acceptable in application-specific SoCs where communication patterns are known a priori and do not change over time, but not in general-purpose platforms because it penalizes short-lived bursty flows.

Probabilistic Distance-Based Arbitration (PDBA) [19] (priority-based, component-wise). Recently, PDBA has been proposed to provide equality-of-service (EoS) while reducing the complexity of managing packet age by approximating age-based arbitration. PDBA does not require many VCs to achieve high throughput and simplifies hardware. However, PDBA does not provide guaranteed QoS; it only provides fairness among flows in the network.

Appendix E. Impact of buffer organization on QoS

Fig. E.14 shows minimum accepted throughput from the least served flow with various buffer organizations and frame window sizes. This figure is in parallel with Fig. 12 but measures *minimum* accepted throughput, instead of average accepted throughput, to demonstrate performance isolation provided by GSF. In the best-effort network (labeled "Baseline VC/iSlip"), starvation is observed for hotspot traffic where accepted throughput is close to zero. However, in the GSF network, the difference between minimum and average throughput is $<0.4\%$ for the same traffic pattern. The difference between the best-effort and GSF networks for uniform random traffic is not as much pronounced because the traffic pattern is well-balanced across the network, hence resulting in balanced network utilization even without QoS support.

References

[1] D. Abts, D. Weisser, Age-based packet arbitration in large-radix k-ary n-cubes, in: SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, ACM, New York, NY, USA, 2007, pp. 1–11. http://doi.acm.org/10.1145/1362622.1362630.

[2] T. Bjerregaard, J. Sparso, A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip, in: DATE '05: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE Computer Society, Washington, DC, USA, 2005, pp. 1226–1231. http://dx.doi.org/10.1109/DATE.2005.36.

[3] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, QNoC: QoS architecture and design process for network on chip, J. Syst. Archit. 50 (2–3) (2004) 105–128. http://dx.doi.org/10.1016/j.sysarc.2003.07.004.

[4] J.A. Colmenares, S. Bird, H. Cook, P. Pearce, D. Zhu, J. Shalf, S. Hofmeyr, K. Asanović, J. Kubiatowicz, Resource management in the Tessellation manycore OS, in: Proceedings of the second Workshop on Hot Topics in Parallelism (HotPar), Berkeley, CA, 2010.

[5] W.J. Dally, Virtual-channel flow control, IEEE Trans. Parallel Distrib. Syst. 3 (2) (1992) 194–205.

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

[7] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, in: SIGCOMM '89: Symposium Proceedings on Communications Architectures & Protocols, ACM, New York, NY, USA, 1989, pp. 1–12. http://doi.acm.org/10.1145/75246.75248.

[8] T. Felicijan, S.B. Furber, An asynchronous on-chip network router with quality-of-service (QoS) support, in: Proceedings IEEE International SOC Conference, 2004.

[9] M. Galles, Spider: a high-speed network interconnect, IEEE Micro 17 (1) (1997) 34–39.

[10] S.J. Golestani, A stop-and-go queueing framework for congestion management, SIGCOMM Comput. Commun. Rev. 20 (4) (1990) 8–18. http://doi.acm.org/10.1145/99517.99523.

[11] K. Goossens, J. Dielissen, A. Radulescu, Æthereal network on chip: concepts, architectures, and implementations, IEEE Des. Test Comput. 22 (5) (2005) 414–421.

[12] S.W.K.B. Grot, O. Mutlu, Preemptive Virtual Clock: a flexible, efficient, and cost-effective QoS scheme for networks-on-a-chip, in: MICRO'09: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, ACM, New York, NY, USA, 2009, pp. 268–279.

[13] F. Guo, H. Kannan, L. Zhao, R. Illikkal, R. Iyer, D. Newell, Y. Solihin, C. Kozyrakis, From chaos to QoS: case studies in CMP resource management, SIGARCH Comput. Archit. News 35 (1) (2007) 21–30.

[14] F. Guo, Y. Solihin, L. Zhao, R. Iyer, A framework for providing quality of service in chip multi-processors, in: MICRO '07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society, Washington, DC, USA, 2007, pp. 343–355. http://dx.doi.org/10.1109/MICRO.2007.6.

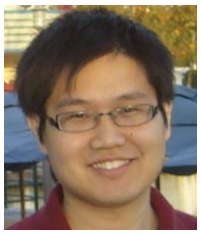
[15] L.R. Hsu, S.K. Reinhardt, R. Iyer, S. Makineni, Communist, utilitarian, and capitalist cache policies on CMPs: caches as a shared resource, in: PACT '06: Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques, ACM, New York, NY, USA, 2006, pp. 13–22. http://doi.acm.org/10.1145/1152154.1152161.

[16] J. Kim, Low-cost router microarchitecture for on-chip networks, in: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42, ACM, NewYork, NY, USA, 2009, pp. 255–266. http://doi.acm.org/10.1145/1669112.1669145.

[17] J.H. Kim, A.A. Chien, Rotating Combined Queueing (RCQ): bandwidth and latency guarantees in low-cost, high-performance networks, in: ISCA '96: Proceedings of the 23rd Annual International Symposium on Computer Architecture, ACM, New York, NY, USA, 1996, pp. 226–236. http://doi.acm.org/10.1145/232973.232996.

[18] J.W. Lee, K. Asanović, METERG: measurement-based end-to-end performance estimation technique in QoS-capable multiprocessors, in: RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE Computer Society, Washington, DC, USA, 2006, 135–147 http://dx.doi.org/10.1109/RTAS.2006.29.
[19] M.M. Lee, J. Kim, D. Abts, M. Marty, J.W. Lee, Probabilistic distance-based arbitration: providing equality of service for many-core cmps, in: MICRO 43: Proceedings of IEEE/ACM International Symposium on Microarchitecture, 2010.
[20] J.W. Lee, M.C. Ng, K. Asanovic, Globally-synchronized frames for guaranteed quality-of-service in on-chip networks, in: ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture, IEEE Computer Society, Washington, DC, USA, 2008, pp. 89–100. http://dx.doi.org/10.1109/ISCA.2008.31.
[21] N. McKeown, The iSLIP scheduling algorithm for input-queued switches, IEEE/ACM Trans. Netw. 7 (2) (1999) 188–201. http://dx.doi.org/10.1109/90.769767.
[22] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip, in: DATE '04: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE Computer Society, Washington, DC, USA, 2004, p. 20890.
[23] T. Moscibroda, O. Mutlu, Memory performance attacks: denial of memory service in multi-core systems, in: USENIX Security, 2007.
[24] K.J. Nesbit, J. Laudon, J.E. Smith, Virtual private machines: a resource abstraction for multicore computer systems, in: University of Wisconsin-Madison, ECE TR 07–08, 2007.
[25] Y. Ofek, M. Yung, The integrated Metanet architecture: a switch-based multimedia LAN for parallel computing and real-time traffic, in: INFOCOM '94, 13th Annual Joint Conference of the IEEE Computer and Communication Societies, Proceedings, 2, IEEE, 1994, pp. 802–811. doi:10.1109/INFCOM.1994.337658.
[26] R.S. Passint, G.M. Thorson, T. Stremcha, United States Patent 6674720: age-based network arbitration system and method, January 2004.
[27] D. Stiliadis, A. Varma, Design and analysis of frame-based fair queueing: a new traffic scheduling algorithm for packet-switched networks, in: SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, ACM, New York, NY, USA, 1996, pp. 104–115. http://doi.acm.org/10.1145/233013.233030.
[28] M. Thottethodi, A.R. Lebeck, S.S. Mukherjee, Self-tuned congestion control for multiprocessor networks, in: HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture, IEEE Computer Society, Washington, DC, USA, 2001, p. 107.
[29] B. Towles, W.J. Dally, Booksim, 1.0, http://cva.stanford.edu/books/ppin/.
[30] J.S. Turner, New directions in communications or which way to the information age, IEEE Commun. 24 (10) (1986) 8–15.
[31] W.-D. Weber, J. Chou, I. Swarbrick, D. Wingard, A quality-of-service mechanism for interconnection networks in system-on-chips, in: DATE '05: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE Computer Society, Washington, DC, USA, 2005, pp. 1232–1237. http://dx.doi.org/10.1109/DATE.2005.33.
[32] K.H. Yum, E.J. Kim, C.R. Das, A.S. Vaidya, MediaWorm: a QoS capable router architecture for clusters, IEEE Trans. Parallel Distrib. Syst. 13 (12) (2002) 1261–1274.
[33] L. Zhang, Virtual Clock: a new traffic control algorithm for packet switching networks, in: SIGCOMM '90: Proceedings of the ACM Symposium on Communications Architectures & Protocols, ACM, NewYork, NY, USA, 1990, pp. 19–29. http://doi.acm.org/10.1145/99508.99525.
[34] H. Zhang, S. Keshav, Comparison of rate-based service disciplines, in: SIGCOMM '91: Proceedings of the Conference on Communications Architecture & Protocols, ACM, New York, NY, USA, 1991, pp. 113–121. http://doi.acm.org/10.1145/115992.116004.

Jae W. Lee is an assistant professor in the Department of Semiconductor Systems Engineering at Sungkyunkwan University, Korea. His research areas include computer architecture, VLSI design, compilers, parallel programming, and computer security, and he has co-authored over a dozen papers in these areas. He led the first ASIC implementation of physical uncloneable function (PUF) at MIT and has held various engineering positions at Nvidia, Nokia, and Parakinetics. He received his M.S. degree in Electrical Engineering from Stanford University and Ph.D. degree in Computer Science from MIT.



Man Cheuk Ng has received his Ph.D. degree in Computer Science from MIT and will join Qualcomm as a senior engineer. He was a member of the Computation Structures Group (CSG) which is led by Professor Arvind. His main research interests are in computer architecture, high-level synthesis and cross-layer wireless protocols.

Krste Asanovic is currently an Associate Professor in the Computer Science Division at UC Berkeley. He also leads the Architecture Group at the International Computer Science Institute, and holds a joint appointment with the Lawrence Berkeley National Laboratory. He received a B.A. degree in Electrical and Information Sciences from the University of Cambridge in 1987 and a Ph.D. degree in Computer Science from UC Berkeley in 1998.