Low-Overhead Network-on-Chip Support for Location-Oblivious Task Placement

Gwangsun Kim, Michael Mihn-Jong Lee, John Kim, *Member*, *IEEE*, Jae W. Lee, Dennis Abts, and Michael Marty

Abstract—Many-core processors will have many processing cores with a network-on-chip (NoC) that provides access to shared resources such as main memory and on-chip caches. However, locally-fair arbitration in multi-stage NoC can lead to globally unfair access to shared resources and impact system-level performance depending on where each task is physically placed. In this work, we propose an arbitration to provide equality-of-service (EoS) in the network and provide support for location-oblivious task placement. We propose using *probabilistic* arbitration combined with distance-based weights to achieve EoS and overcome the limitation of round-robin arbiter. However, the complexity of probabilistic arbitration results in high area and long latency which negatively impacts performance. In order to reduce the hardware complexity, we propose an *hybrid* arbiter that switches between a simple arbiter at low load and a complex arbiter at high load. The hybrid arbiter is enabled by the observation that arbitration only impacts the overall performance and global fairness at a high load. We evaluate our arbitration scheme with synthetic traffic patterns and GPGPU benchmarks. Our results shows that hybrid arbiter that combines round-robin arbiter with probabilistic distance-based arbitration reduces performance variation as task placement is varied and also improves average IPC.

Index Terms—Network-on-chip (NoC), arbitration, equality-of-service (EoS)

1 Introduction

Emerging many-core accelerators will integrate dozens of small processing cores with an on-chip interconnect consisting of point-to-point links. As more components, including processors, caches, and memory controllers, are interconnected, the Network-on-Chip (NoC) [1] becomes an important shared resource that needs to be properly managed. One concern is that, depending on the location of the component, the amount of bandwidth that can be received by the component can vary significantly. This has significant impact on task placement in future multiprocessors as location needs to be considered. As a result, it becomes critical for the on-chip network to provide global fairness and equality-of-service (EoS) [2]—i.e., provide equal share of bandwidth regardless of the location of the components in the on-chip network.

Recently, cost-efficient quality of service (QoS) for on-chip networks have been proposed [3], [4]. Unlike QoS, which strives to provide differentiated service and hard (or soft) guarantees for end-to-end latency or bandwidth profile, EoS does not provide guarantees yet provides equal access to shared on-chip resources.[1] In this work, we propose and evaluate an arbitration mechanism to achieve equality of service and predictable performance. By tackling arbitration in the interconnect, we ensure the packets delivered to a shared resource are not unfairly biased by source location. However, for *on*-chip networks, arbitration must be fast and simple to reduce overheads.

In this paper, we introduce *distance-based* arbitration by taking into account the distance or the hop count which a packet travels en route to its destination—allowing nodes located many hops from the edge to get equal service compared to a node close to the edge [2]. We propose using *probabilistic* arbitration with a distance-based selection algorithm to achieve EoS while providing a livelock-free arbitration allowing for consistent latency and bandwidth characteristics for all cores. Since nodes that are farther away are serviced at a ratio that is geometrically proportional to the hop count, we propose using *nonlinear* weights in probabilistic arbitration to provide fairness to nodes that are farther away. Three different arbitration weight metrics are proposed which all provide EoS but have varying trade-offs in terms of complexity and performance degradation on different traffic patterns. By combining distance-based weight metric with probabilistic arbitration, we obtain probabilistic distance-based arbitration (PDBA).

However, the long arbitration latency of PDBA can impact the router cycle time and overall performance. Thus, we propose a hybrid arbiter that combines a simple arbiter with a complex arbiter—and enables the arbitration latency to be effectively hidden.

- *G. Kim and J. Kim are with the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Republic of Korea. E-mail: {gskim, jjk12}@kaist.ac.kr.*
- *M.M.-J. Lee is with the Department of Computer Science and Engineering, University of California, San Diego (UCSD), La Jolla, CA 92093-0404. E-mail: mmlee@cs.ucsd.edu.*
- *J.W. Lee is with the Department of Semiconductor Systems Engineering, Sungkyunkwan University (SKKU), Suwon, Gyeonggi-do, Republic of Korea. E-mail: jaewlee@skku.edu.*
- *D. Abts and M. Marty are with the Google Inc., Mountain View, CA 94043. E-mail: {dabts, mikemarty}@google.com.*

Manuscript received 03 May 2012; revised 14 Sep. 2012; accepted 24 Sep. 2012. Date of publication 01 Oct. 2012; date of current version 09 June 2014. Recommended for acceptance by E. Rotenberg.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2012.241

1. EoS can be considered a subset of QoS.

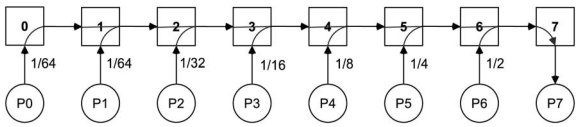


Fig. 1. 8-ary 1-mesh example where all nodes are sending to P7 and merging traffic at each hop.

Specifically, the contributions of this work include the following:

- We propose how nonlinear weights based on distance are required to achieve equality-of-service (EoS) and describe three different arbitration weight metrics based on the hop count and the degree of contention.
- To provide fairness using priority-based arbitration [5] with distance as a metric, we propose a distributed *probabilistic* arbitration where arbitration decisions are made probabilistically at each router based on the weights of input requests.
- We show how probabilistic distance-based arbitration can provide additional benefits which include providing QoS-like characteristics and provide stronger fairness than conventional round-robin arbitration to enable a more stable network.
- We propose a hybrid arbiter that combines a simple arbiter with a complex arbiter to effectively hide the latency of complex arbitration.
- We show that the hybrid arbiter combining PDBA with RR arbiter provides low variance in IPC as spatial core scheduling is varied, hence practical support for location-oblivious task placement.

2 Motivation

2.1 Location-Oblivious Task Placement

As the number of cores integrated in a chip increases, efficiently scheduling the task on to the many cores becomes increasingly difficult since the scheduling complexity increases super-linearly with the number of cores [6]. One of several aspects of many-core processors that makes optimal scheduling difficult is network-on-chip (NoC) topology. In a multi-hop NoC, the access latency and bandwidth to shared resource such as memory controllers can vary depending on the location of the core.

The system-level performance can be impacted by the placement the tasks in a multi- and many-core system. The OS can provide improved scheduling or placement policies to minimize variance across different placement. However, the OS is often oblivious to the on-chip network organization. Thus, support in the NoC for location-oblivious task placement can not only improve system performance but also be used in conjunction with an OS task placement policy.

2.2 Equality-of-Service Problem in NoC

In networks-on-chip, as traffic flows through the network, it merges with newly injected packets and traffic from other directions in the network. This merging of traffic from different sources causes packets that have further to travel (more hops) to receive geometrically less bandwidth. For example, consider the 8-ary 1-mesh in Fig. 1 where processors P0 thru P6 are sending to P7. The switch allocates the output port by granting packets fairly among the input ports. With a round-robin arbitration policy, the processor closest to the destination (P6 is only one hop away) will get the most bandwidth—$1/2$ of the available bandwidth. The processor two hops away, P5, will get half of the bandwidth into router R6, for a total of $1/2 \times 1/2 = 1/4$ of the available bandwidth. That is, every two arbitration cycles P7 will deliver a packet from source P6, and every four arbitration cycles it will deliver a packet from source P5. As a result, P0 and P1 each receive only $1/64$ of the available bandwidth into P7, a factor of 32 times less than that of P6. To summarize, although round-robin arbitration provides local fairness at each router, it does not provide any global fairness across all routers. This is an important problem, because reducing the variation in bandwidth is critical for application performance, particularly as applications are scaled to higher processor counts.

Age-based arbitration [8] is known to provide global fairness as when two or more packets arbitrate for a shared resource, the packet with the oldest age wins the arbitration. However, it becomes complex to implement within the constraints of an on-chip network. In this work, we avoid the complexity with age-based arbitration by proposing to approximate the age of a packet with *distance* or hop count. By using information already present in the packet, such as source node, current node, or destination node and using distance as a proxy for the packet's age, age-based arbitration is greatly simplified. To help understand how hop count can approximate the age of a packet, which corresponds to the packet latency (T) from the source to its destination, below equation shows how they are related:

$$\begin{aligned} T &= T_h + T_s + T_w + T_c \\ &= Ht_r + T_s + Ht_w + Ht_q \\ &= H(t_r + t_w + t_q) + T_s, \end{aligned}$$

where T_h is the header latency, T_s is the serialization latency, T_w is the wire delay, and T_c is the contention and queuing latency. For all packets, T_s is constant, regardless of the total latency, and is only dependent on the channel bandwidth and packet size. All the other terms are directly proportional to the total hop count (H) from source to destination and other parameters such as per-hop router latency (t_r), per-hop wire delay (t_w), and per-hop queuing delay (t_q). Since we assume a 2D mesh topology, all t_w are identical. We approximate T_c with Ht_q since we assume per-hop queuing latency dominates the contention latency. Thus, the age of a packet is directly proportional to the hop count (H) and can be used to approximate the packet's age. In this paper, we show how the hop count can be used with probabilistic arbitration and nonlinear weight priorities to provide equality of service and support for location-oblivious task placement.

3 Arbitration Design

In order to use hop count as an arbitration metric, we need to guarantee fairness since, by providing preference based on weights, there is potential for livelock and starvation. In this section, we propose probabilistic arbitration which can provide fairness while using hop count to determine the weight of packets. We also present why *nonlinear* weights based on hop

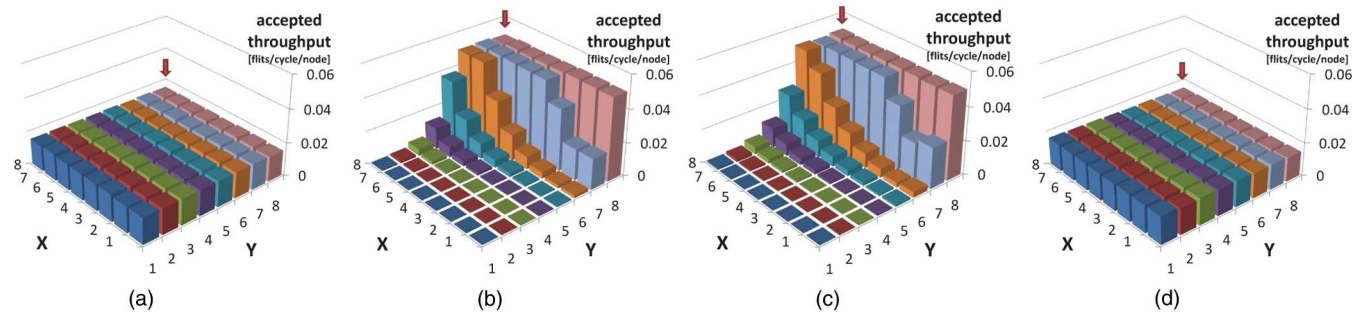


Fig. 2. Accepted throughput per source node by a hotspot resource (indicated by arrow) using (a) age-based arbitration, (b) round-robin arbitration, (c) probabilistic arbitration with linear weights, and (d) probabilistic arbitration with non-linear weights.

count are required and how different weight metrics provide different degrees of EoS.

3.1 Probabilistic Arbitration

Previous arbitration architectures are *deterministic*—that is, given a set of input requests and the switch's current state (such as a state of the arbitration pointer or priorities), the output grants are always deterministically assigned. For sorted priority-based arbitration [5] such as age-based arbitration, arbitration is done deterministically based on the relative age of the requests. Starvation is inherently prevented by age-based arbitration. However, if priority based on hop count is used for a deterministic arbiter, livelock and fairness issues are problematic because packets with a lower priority (i.e., a lower hop count) can continually lose arbitration due to a constant stream of newly injected traffic with higher priority. To overcome this problem while still using hop count as the weight, we propose *probabilistic* arbiter which probabilistically determines its output based on the weights of input requests.

Assume an arbiter shown in Fig. 3 with two requests r_1 and r_2, each with a corresponding weight w_1 and w_2. The probability of each grant g_1 and g_2 being asserted with probabilistic arbitration is proportional to its weight as follows:

$$P(g_1) = \frac{w_1}{w_1 + w_2},$$
$$P(g_2) = \frac{w_2}{w_1 + w_2}.$$

Since both grants cannot be asserted in the same cycle, the arbiter needs to probabilistically select only one of the two requests based on this probability. If the incoming weights are identical (i.e. $w_1 = w_2$), the arbiter output is identical to a random arbiter—randomly selecting one of the two requests. In general, for a request r_i to an arbiter with m requests, the probability of r_i being granted is

$$P(g_i) = \frac{w_i}{\sum_{j=1}^{j=m} w_j}.$$

Fig. 3. High-level block diagram of a probabilistic arbiter.

A request in probabilistic arbitration will not starve indefinitely, since the probability that a request will not be granted converges to 0 as the number of arbitrations tried increases. However, a request can incur significant wait time if it continuously loses arbitration.

3.2 Linear Weight

With *linear* weight, the weight will be linearly proportional to the hop count of a packet in probabilistic arbiter—i.e., $w = h_x$ or $w = h_x + h_y$ where h_x and h_y represent the hop count from source to destination in each dimension. However, probabilistic arbitration using linear weight hop count cannot provide EoS since farther nodes will be serviced linearly instead of geometrically. The weight inputs to the probabilistic arbiter will only differ linearly and not be able to provide EoS to farther nodes. For example, for two packets that are separated by x hop count, the linear weights for the two packets will be w and w–x—assuming both packets have the same destination. The probability of each packet winning an arbitration is $\frac{w}{2w-x}$ and $\frac{w-x}{2w-x}$, respectively. For large values of w ($w \gg x$) or for small values of x, the probability of each packet winning the arbitration is approximately $1/2$. Thus, the result of probabilistic arbitration with linear weight is very similar to round-robin arbitration. This is shown with the results in Fig. 2(c) for hotspot traffic where all traffic is sent to a single node.[2] The resulting acceptance rate of each node is very similar to the round-robin arbitration shown in Fig. 2(b) and does not provide the equality of service as shown in Fig. 2(a) with an ideal age-based arbitration.

3.3 Nonlinear Weight

As shown earlier in Fig. 1, with round-robin arbiter, nodes that are farther away are serviced at a rate that is exponentially proportional to the hop count—for example, packets that are h hops away are serviced at a rate of $(1/2)^h$ and the service rate is not linearly proportional to the hop count. To account for this difference, we introduce *nonlinear* weights based on the distance. Instead of using a weight which is equal to the hop count (i.e., $w = h$), we use nonlinear weights in probabilistic arbitration—i.e., $w = C^h$ where C is the *contention degree* or the number of packets contending for the same output port. By using nonlinear weights, better fairness is provided for nodes whose packets travel longer distance. In Fig. 4, three different possible hotspots are shown in an 8×8 2D mesh network and the details of the merging traffic is shown in Fig. 5.

2. Simulation setup is described in Section 5.

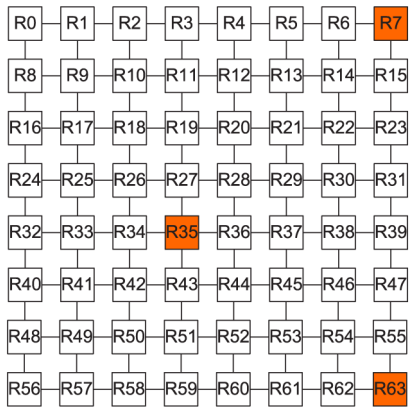


Fig. 4. 8×8 2D-mesh block diagram with several hotspots highlighted. The details of hotspot traffic to these nodes are shown in Fig. 5.

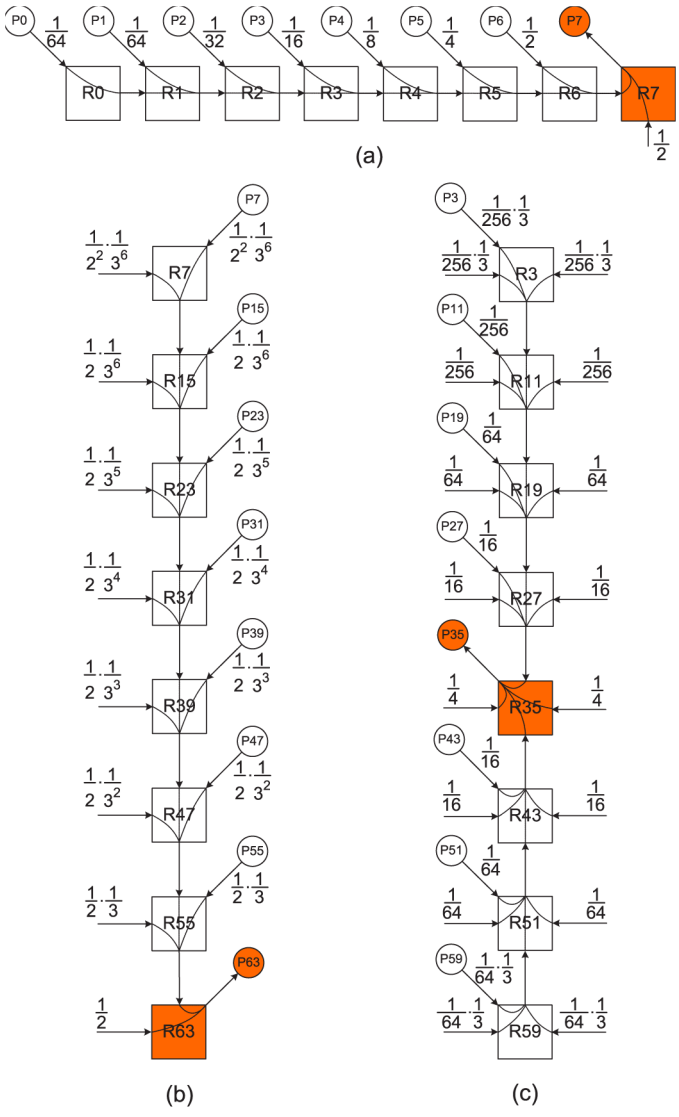


Fig. 5. Merging traffic in 8×8 2D-mesh. The fraction numbers represent the amount of bandwidth that the corresponding nodes would be received if a locally fair, round-robin arbitration is implemented. The highlighted node represents hotspot traffic in a 2D mesh network shown in Fig. 4.

TABLE 1
Arbitration Metrics to Determine Weight of Probabilistic Arbitration Where h Is the Hop Count and C Is the Contention Degree

h	C	description
static	static	fixed weight (FW)
dynamic	static	constantly increasing weight (CW)
static	dynamic	N/A
dynamic	dynamic	variably increasing weight (VW)

In Fig. 5(a), if nodes are serviced at a rate of $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots$, in order to provide EoS, each node needs to be prioritized with a weight of $2, 4, 8, 16, \ldots$, respectively. Thus, for the traffic pattern shown in Fig. 5(a), $w = 2^h$ can be used with probabilistic arbitration to achieve EoS. $C = 2$ value is used since for each output, there are two flows contending for a router output. With XY routing, packets traveling in the x-dimension will merge similar to the traffic shown in Fig. 5(a).

For hotspot traffic shown in Fig. 5(b), when traversing the y-dimension, there are 3 traffic flows merging at each router, resulting in each flow being serviced at a rate of $1/3$. Thus, weight used for y-dimension is $w = 3^h$. For traffic shown in Fig. 5(c) where the destination node is located in the non-edge location of a 2D mesh network, the number of flows merging is 4, thus $w = 4^h$ needs to be used while traversing this path to provide fairness across all nodes.

3.4 Arbitration Weight Metrics

To better understand some of the design tradeoffs, we first define several metrics that can be used as an input to probabilistic arbitration. The arbitration metric can be categorized as either *static* or *dynamic*. With a static metric, the priority of a packet is determined before the packet is injected into the network. On the other hand, dynamic metrics will cause the priority of a packet to change en route. Leveraging the nonlinear weight (C^h), the different metric can be categorized based on whether C and h are either static or dynamic as summarized in Table 1. In describing the different metrics, we assume that a packet is sent from a source node located at (s_x, s_y) to a destination at (d_x, d_y) and the current location is (c_x, c_y). Throughout this work, we assume dimension-ordered routing (DOR) with XY routing.

3.4.1 Fixed Weight (FW)

The total number of hops a packet must travel from its source to its destination is a static value in a mesh network with minimal routing (e.g., dimension-ordered routing). This value is known when the packet is injected into the network. Using this distance, packets which have a longer distance to travel are biased by giving them higher priority at each hop along the way. The static value of the hop count is used based on the source and destination node.

$$h_x = |s_x - d_x|,$$
$$h_y = |s_y - d_y|.$$

Using these hop count, the weight is calculated according to the dimension being traversed with a contention degree C. While traversing in the x-dimension, $w = 2^{h_x}$ is used and when traversing in the y-dimension, $w = 2^{h_x} \times C^{h_y}$ is used. When traveling in the y-dimension, the weight from the

x-dimension is included as well to properly prioritize packets that have traversed longer overall distance. However, the y-dimension weight C^{h_y} is not included while traversing the x-dimension since when a packet only needs to traverse the x-dimension, a packet that needs to traverse both the x and the y-dimension will be unfairly biased. With this metric, the weight of each packet remains constant or *fixed* throughout the network.

The value of C is dependent on the location of the destination. For a radix-k 2D mesh topology (i.e., $k \times k$ mesh),

$$C = \begin{cases} 3, & if\ d_x = 0 \| d_x = k-1, \\ 4, & otherwise. \end{cases}$$

Since 2D mesh is a non-edge symmetric topology, for destination located on the edge of the 2D mesh network, $C = 3$ while $C = 4$ is used for all other destination.

3.4.2 Constantly Increasing Weight (CW)

Instead of relying on static values, *dynamic* values based on how long a packet has traversed can be used. The distance *traveled* is defined as the number of hops from the packet source to the current position. A packet's weight increases as it gets closer to its destination. The dynamic value of the hop count is determined as follows:

$$\begin{aligned} h_x &= |c_x - s_x|, \\ h_y &= |c_y - s_y|. \end{aligned} \quad (1)$$

Similar to the fixed weight (FW) metric, when traveling in the x-dimension, the weight is calculated as 2^{h_x} based on the distance traveled (Equation (1)) and when traveling in the y-dimension, $2^{h_x} \times C^{h_y}$ is used where C is determined based on the destination location as described earlier for FW. When the packet reaches the destination, the weight will be identical to the weight using FW.

To describe concisely, when a packet is injected, it is assigned a weight of 1. As a packet traverses the network, the weight is continually increased. In the x-direction, the weight is multiplied by a factor of 2 at each hop and when traveling in the y-dimension, the weight is multiplied by a factor of C at each hop.

3.4.3 Variably Increasing Weight (VW)

We also evaluate a metric where the value of C per hop is variably changed, instead of using a constant C value as the weight is increased at each hop. Packets are injected with a priority of 1 and the priority also increases dynamically as the packet traverse the network to its destination, similar to CW. However, the increase in weight is not constant as in CW but is dynamically varied based on the actual contention degree (C) for the output port. The contention degree is defined as the number of packets that are destined for the same router output port. The range of values for C is $1 \leq C \leq (p-1)$ where p is the number of router ports.[3] For example, if there are 3 packets that need to be routed through one output port of a router, each of these packets will have a contention degree of 3. Thus, when these packets are forwarded to the next router, their priorities are increased by $3\times$. However, if there are no other packets contending for the same output in a given cycle, the weight of the packet does not change.

3. Since we assume no U-turn routing, the maximum value of C is $p-1$.

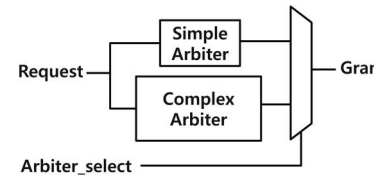


Fig. 6. Block diagram of a hybrid arbiter.

3.4.4 Other Metric Considerations

As shown earlier in Table 1, another possible weight metric is using static hop count and dynamic contention degree. However, this metric is not applicable since if the hop count is static or determined from the source, the entire weight needs to be fixed at the source—otherwise, the weight would increase by C^h per hop but this will not provide EoS as significant more priority is provided to nodes that are farther away. In addition, the dynamic hop count can be obtained from the distance or hop count *remaining*, which is the number of hops to the destination from its current location in the network. This metric would decrease the packet's priority as it approaches the destination. However, decreasing the weight can negate the effect of using probabilistic arbitration. For example, in Fig. 5(a), if each packet begins with a fixed weight at its source and if the packet's weights were decreased by $C = 2$ at each hop, the packets that are merged at each router will have equal weights—resulting in an arbitration very similar to round-robin arbitration. Therefore the arbiter will not be able to provide any EoS.

3.5 Hybrid Arbiter

Although the area occupied by the arbiter is relatively small, the arbitration is often on the router's critical path and determines its cycle time [9], [10]. In Section 4.1, we discuss different heuristics that reduce the complexity of the PDBA but the arbitration is still on the critical path. To hide the impact of complex arbitration latency, we propose a *hybrid* arbiter that combines a simple, low-complexity arbiter with a complex, fair arbiter. A high level block diagram of a hybrid arbiter is shown in Fig. 6 and the router pipeline that uses a hybrid arbiter is shown in Fig. 7.

The observation that enables a hybrid arbiter to work is that a complex arbiter (such as PDBA) only impacts the performance at a high load or when contention occurs in the network. Thus, at a low load or a near zero load, simple arbitration can be used with minimal impact on performance

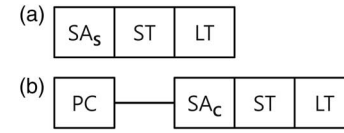


Fig. 7. Router pipeline stages of hybrid arbiter when (a) simple arbiter is used ($\mathrm{SA_s}$) and (b) complex arbiter with earlier pre-calculation (PC) stage is used ($\mathrm{SA_c}$). We assumed the switch allocator uses next-hop routing information stored in the packet header and the next-hop routing computation is done in parallel with switch allocation.

or fairness since there is little or no contention in the network. At a high load, arbitration can impact the performance, but the high load also ensures that there are multiple flits queued in the buffers. As a result, some of the steps required in the complex arbitration can be done in the previous cycles while waiting in the queue and hiding some of the latency in the complex arbitration. An example usage of the complex arbiter is shown in Fig. 7(b). In the figure, the packet is queued up in the input buffer for two cycles so the pre-calculation could be done in the first cycle. At the third cycle, this packet becomes the front packet in the buffer and is granted for output port and goes through the remaining router pipeline in consecutive cycles. Because pre-calculation was already done, the complex arbiter could be used for switch allocation as designated as SA_c stage with reduced latency.[4]

Our proposed hybrid arbiter uses a round-robin (RR) arbiter as the simple arbiter and PDBA as the complex arbiter. To enable the complex arbiter to exploit the hybrid arbiter organization, some of the steps required in the complex arbiter need to be *pre-calculated*. For PDBA, the weights used in probabilistic arbitration can be pre-computed—i.e., the weight calculation that involves generating the random numbers and scaling it appropriately—and the results can be stored in a separate buffer. (Detailed implementation of the hybrid arbiter with PDBA is presented in Section 4.2.)

The select signal used in Fig. 6 is based on whether this pre-calculation is done or not. If there are no pre-computed partial weights, that signifies there is only one request in the input buffer (and that there was no congestion or the network is at low load), and thus, the simple arbiter is used. Since only the remaining part of the weight calculation is done in the actual arbitration stage, the critical path of the arbitration is reduced and thus the router can operate at a higher clock frequency.

4 Implementation

4.1 Implementation Complexity

This section discusses the implementation details of PDBA. We first describe the implementation complexity of the original PDBA, which is called *baseline* PDBA. The hardware complexity of the baseline PDBA is very high. Thus, we introduce heuristics to reduce the complexity via approximation. The final area/latency results reported in Section 6.5 were obtained after applying these heuristics first. We used the VW weight metric for PDBA since it is provides the best EoS according to results in Section 6.

4.1.1 Baseline Implementation of PDBA

The weight in PDBA is represented by C^h where C is the contention degree and h is the hop count. As with age-based arbitration [8], each packet header carries information needed to calculate the packet's weight. There are two ways to calculate it:

1) Calculate the weight at each hop while carrying the hop count information.
2) Incrementally calculate the weight at each hop by carrying the weight information.

4. In this work, we do not assume arbitration is pipelined for a hybrid arbiter. Instead, we are effectively trying *pre-process* some of the weights and metrics needed in the *complex* arbiter—e.g., PDBA.

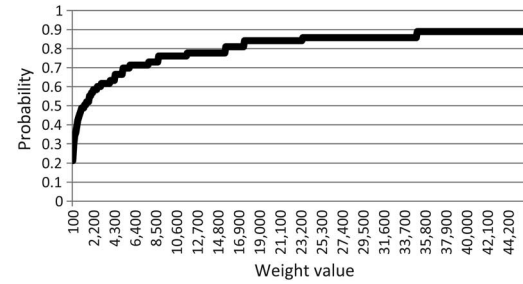


Fig. 8. Cumulative distribution of weight values. Small weight values are dominant.

Method (1) requires fewer bits of information to be carried by each packet header to reduce the bandwidth overhead, but the weight should be recalculated at each hop. In contrast, with method (2), the actual weight is carried by the packet header, which reduces computation at each hop. Thus we focus on (2) in implementing PDBA.

One distinctive aspect of PDBA is that the arbitration is done not deterministically, but probabilistically. PDBA employs a pseudo-random number generator that produces a k-bit random number at each cycle using a Linear Feedback Shift Register (LFSR). For probabilistic arbitration, a random number $RNG(w_T)$ needs to be generated in the range of $[0, w_T - 1]$, where w_T is the sum of all the weights of flits involved in the arbitration. A random number could be generated after calculating w_T, but this would sequentialize the weight calculation and the random number generation. To remove this bottleneck, we pre-generated w_T using a k-bit random number generated by an LFSR. However, since this random number is in the range of $[0, 2^k - 1]$, it has to be scaled to produce a random number in the range of $[0, w_T - 1]$. This can be done as described in the following equation:

$$RNG(w_T) = \frac{w_T \times v}{2^k},$$

where v is a k-bit random number. To scale the random number, the k-bit random number from LFSR is multiplied by the sum of weight values of each request, and then divided by 2^k (right-shifted by k bits). Then, the resulting $RNG(w_T)$ is compared with each weight value to generate the grant, as shown in Fig. 10. The critical path in PDBA arbitration involves the following steps: weight calculation, weight summation, random number generation, and arbitration. Based on our implementation of the baseline PDBA using Method (2), our results show that the area increases by $3.6\times$, while the critical path increases by $5.5\times$. In the following section, we explore different heuristics to reduce the complexity of baseline PDBA.

4.1.2 Reducing Complexity of Baseline PDBA

To reduce the complexity of the weight calculation, the precision of the weights can be reduced with minimal impact on performance. Instead of carrying around the complete, precise weight information, the number of bits can be reduced by limiting the maximum value of the weight. In Fig. 8, the cumulative distribution of flit weight is plotted as a result of the simulation of VW in hotspot traffic in an 8×8 2D mesh network at a high load. As shown in the figure, relatively

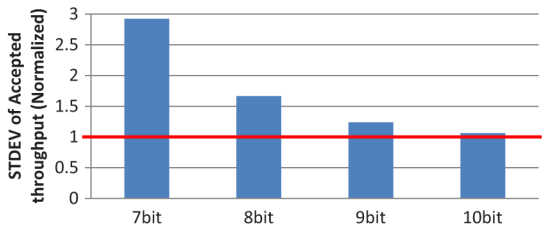


Fig. 9. Performance impact of using approximation, compared with a baseline with no approximation. Smaller value means better fairness is achieved in the hotspot traffic pattern.

small weight values are dominant in PDBA, even in hotspot traffic pattern, where the weight values continually increase because of the contention at each hop. As the weight precision is reduced, Fig. 9 shows the performance comparison on an 8×8 mesh with hotspot traffic, compared with the baseline arbiter with full precision. With 9 or fewer bits, there is an impact on the overall performance, but with 10 bits, there is minimal impact on performance. Furthermore, for multiple hotspot traffic pattern, where each hotspot can be considered as memory controller in a real system, small weight values tend to be more dominant. According to our real application evaluation described in Section 5, 4-bit approximation resulted in a comparable performance while effectively reducing implementation complexity. Therefore, we assume 4-bit approximation of weight values for the rest of this paper.

4.2 Hybrid Arbiter Implementation

We present the implementation details of a hybrid arbiter described in Section 3.5. To use the PDBA as the complex arbiter in our hybrid arbiter, we used the following equation to pre-process packets, while they are in the buffer.

$$\begin{aligned} RNG(w_T) &= w_T \times \frac{v}{2^k}, \text{ where } 0 \le RNG(w_T) < w_T \\ &= \left(\sum w_i\right) \times \frac{v}{2^k} \\ &= \sum \left(w_i \times \frac{v}{2^k}\right) \\ &= \sum \left(\frac{w_i \times v}{2^k}\right) \\ &= \sum RNG(w_i). \end{aligned}$$

According to above equation, the order of the calculation of $RNG(w_T)$ can be changed. Instead of calculating w_T, and then generating a random number $RNG(w_T)$, a random number $RNG(w_i)$ for each weight w_i (or each flit) is generated first, and then they are summed up to obtain $RNG(w_T)$, as shown in Fig. 10. For each incoming packet, its $RNG(w_i)$ is calculated

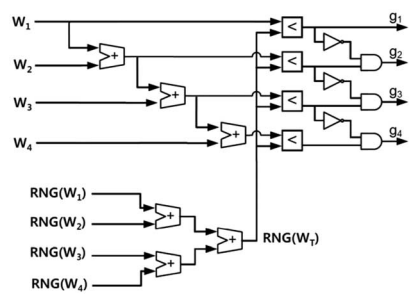


Fig. 10. Block diagram of PDBA with pre-calculation technique. $RNG(w_i)$ are pre-calculated one cycle before arbitration.

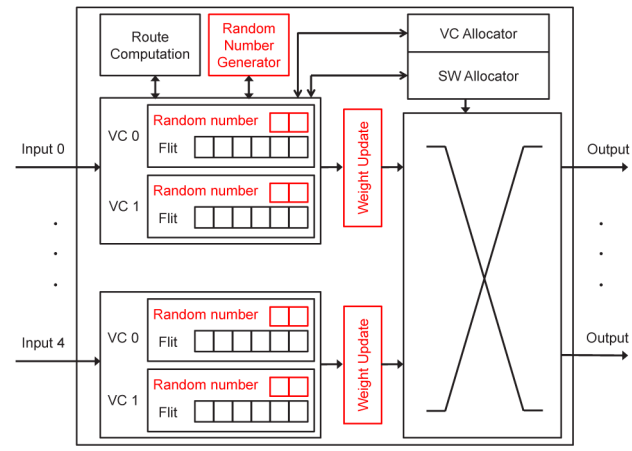


Fig. 11. Microarchitecture of our proposed router with hybrid arbiter.

and stored in a separate buffer. If the packet participates in an arbitration in the next cycle, the saved values are read out and used for calculation of $RNG(w_T)$. This pre-calculation would require another set of buffers whose depth is proportional to the depth of the router's input buffers. However, since only the packets in the head of the buffer participate in the arbitration and only one flit can leave a buffer every cycle, we only need two entries in this buffer without degrading the performance, while reducing the buffering requirement. Note that the width of this additional buffer (e.g., 4 bits) is normally very small compared to the width of the flit buffers at each input port.

With a hybrid arbiter, one challenge is to decide when to switch between a simple and a complex arbiter. Since our work uses PDBA as the complex arbiter, maximizing its usage will give the highest performance. However, since $RNG(w_i)$ calculation takes one cycle, if there is no buffered $RNG(w_i)$ for the request, the complex arbiter cannot be used.

Therefore, we used the simple arbiter whenever any of the requests did not have pre-calculated $RNG(w_i)$. In this method, the availability of $RNG(w_i)$ has to be known to the arbiter. To find out if there is a request whose $RNG(w_i)$ is not calculated, the hybrid arbiter remembers requests from the previous cycle and compares them with current requests. If there is a new request, then it means that its $RNG(w_i)$ is not available yet. Otherwise, each request's $RNG(w_i)$ must have been calculated.

Our proposed router microarchitecture with the hybrid arbiter of PDBA and round-robin arbiter is shown in Fig. 11. A random number generator is needed for each input port and for calculating $RNG(w_i)$ for incoming flits. The calculated random number is stored in the per-VC random number buffer. The weights, $RNG(w_i)$ are sent to VC and SW allocators along with request information for arbitration. When a packet is granted for output port, its weight is multiplied by the contention degree and the packet header is updated accordingly before crossbar switch traversal.

5 Methodology

We evaluated our proposed probabilistic distance-based arbitration (PDBA) with synthetic traffic pattern and GPGPU benchmarks [12]–[15]. For synthetic traffic simulations, we

TABLE 2
Synthetic Traffic Simulation Parameters

Parameters	Values
Network size	64
Topology	2D mesh
Routing	XY routing
Router latency	2 cycle
Buffers	8 flit entry per input port
Virtual channels	1
Packet size	bimodal(50% 1 flit and 50% 4 flits)

implemented probabilistic distance-based arbitration on a cycle-accurate interconnection network simulator [33] and used parameters described in Table 2. PDBA does not require additional VCs so we used a single virtual channel (VC). We assume a FIFO buffer structure—i.e., packet reordering is not allowed at each router input buffer. If additional VCs are required for protocol deadlock, PDBA can support additional VCs for different classes of traffic as long as packets stay within the same VCs from source to its destination. The only change required is that VC allocation needs to implement probabilistic arbitration based on distance as well. For the long packets, the head flit goes through switch arbitration using probabilistic arbitration.

To evaluate the latency-throughput with synthetic traffic pattern, the simulator is warmed up under load without taking measurements until steady-state is reached. Then, a sample of injected packets is labeled during a measurement interval. The simulation is run until all labeled packets exit the system. Different synthetic traffic patterns including hotspot traffic, uniform random, bit complement, bit reverse, shuffle, tornado, random permutation, and transpose were used to evaluate PDBA. Due to page constraints, only selected results are presented in the next section.

TABLE 3
GPGPU-Sim Simulation Parameters

Parameters	Values
Number of compute cores	56
Number of MCs (memory controllers)	8 (diamond placement [11])
MSHRs / core	64
Warp size	32
SIMD pipeline width	32
Number of threads / core	1024
Number of CTAs / core	8
Number of registers / core	16384
Shared memory / core	16 KB
Constant cache size / core	8 KB
Texture cache size / core	8 KB
L1 cache size / core	16 KB
L2 cache size / MC	128 KB
Compute core clock	1540 MHz
Interconnect clock	refer to Table 7
L2 cache clock	700 MHz
Memory clock	800 MHz
Network count	2 (each for memory request and reply message class)
GDDR3 memory timing	$t_{CL} = 10, t_{RP} = 10, t_{RAS} = 25$ $t_{RC} = 35, t_{RCD} = 12, t_{RRD} = 8$
DRAM request queue size	32
Memory scheduling	out-of-order (FR-FCFS)
Branch divergence method	Immediate post denominator
Warp scheduling policy	Round-robin among ready warps

TABLE 4
GPGPU Benchmarks Used for Evaluation

Benchmarks	abbrev.	Injection rate (packets/cycle)
StoreGPU [14]	STO	0.172
MonteCarlo [15]	MC	0.163
Rodinia Hotspot [13]	HSP	2.132
Laplace Solver [14]	LPS	3.478
Separable Convolution [15]	CON	3.619
Speckle Reducing Anisotropic Diffusion [13]	SRAD	3.632
Back Propagation [13]	BP	3.747
MUMmerGPU [14]	MUM	4.528
BlackScholes [14]	BLK	4.908
3D Finite Difference Computation [15]	3DFD	5.113
Fast Walsh Transform [15]	FWT	5.876
BFS Graph Traversal [13]	BFS	6.242
LIBOR Monte Carlo [14]	LIB	6.719

We also evaluate different arbiters in terms of how well they can support location-oblivious task placement. Since many compute cores will be integrated in a future GPU, and scheduling complexity increases super-linearly with the number of cores [6], providing predictable performance regardless of task placement on cores is desirable. We evaluated the performance as task placement is varied on a spatial multitasking-enabled GPGPU-sim [14], [7]. The parameters are described in Table 3. For clock frequencies of different arbiters, we used synthesis result shown in Table 7. For age-based arbiter, we assumed that the ideal implementation is possible and it can run at the clock frequency of round-robin (RR) arbiter. We used the benchmarks listed in Table 4 to simulate GPGPU spatial multitasking. Our configuration for simulation of spatial multitasking of GPGPU-sim is shown in Fig. 12. We run three workload mixes each consisted of eight benchmarks as described in Table 5. For each workload mix, the eight benchmarks are placed randomly on one of clusters of cores shown in Fig. 12 as shaded region and run for 50 million instructions. Each workload mix is simulated over 100 different random placements and we report the distribution of overall IPC (total number of instructions executed divided by runtime in cycle). Benchmarks that finish during simulation are re-spawned for steady-state simulation.

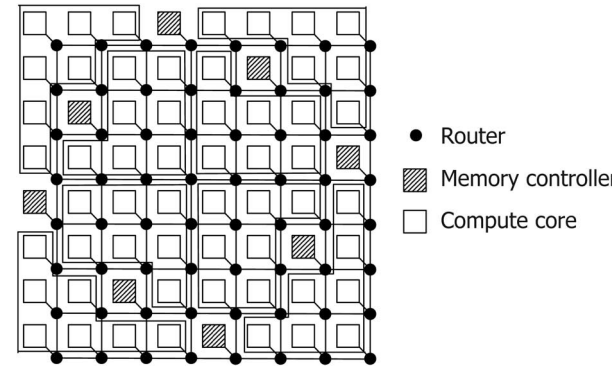


Fig. 12. Configuration for GPGPU-sim with spatial multitasking. Each shaded region represents a cluster of compute cores that runs a single GPGPU application.

TABLE 5
Workload Mixes for GPGPU Multitasking

Workload mix	Benchmakrs
1 (compute-intensive)	STO, MC, HSP, LPS, CON, SRAD, BP, and MUM
2 (memory-intensive)	SRAD, BP, MUM, BLK, 3DFD, FWT, BFS, and LIB
3 (half compute-intensive and half memory-intensive)	STO, MC, HSP, LPS, 3DFD, FWT, BFS and LIB

To measure the hardware area and timing overhead of PDBA, we used Synopsys Design Compiler with 45 nm TSMC technology file for synthesis. We used the Verilog router model from [16] as the baseline and modified it to implement our arbiter.

The following different arbitration algorithms are compared in the evaluation.

- **round-robin arbitration (RR)**: a locally fair, round-robin arbitration at each router node.
- **age-based arbitration (AGE)**: an ideal implementation of age-based arbitration where each packet is timestamped at packet generation time and the age continues to increase every cycle.
- **probabilistic distance-based arbitration**: arbitration determined probabilistically by the following different weights.
  a) **fixed weight (FW)**
  b) **constantly increasing weight (CW)**
  c) **variably increasing weight (VW)**
- **hybrid arbiter (Hybrid 4-bit)**: a hybrid arbiter that combines a PDBA (VW) with a RR arbiter. We use 4-bit weight approximation, as this provides comparable performance at reduced hardware cost.

6 Evaluation

6.1 Hotspot Traffic

We first evaluate probabilistic distance-based arbitration (PDBA) on hotspot traffic where all nodes send traffic to a single destination. As shown in Fig. 2, we verify that equality of service is achieved by measuring the accepted throughput across all nodes. The different metrics (FW, CW, VW) all provide very similar results so only the result for CW is shown in Fig. 2(d). As a result, by approximating age with hop count and using nonlinear weight with probabilistic arbitration, we can match the performance of age-based arbitration in hotspot traffic and achieve equality of service.

Latency variation is another important factor in determining overall performance—thus, minimizing the variance is also critical in providing EoS. In Table 6, we measure the packet latency variation in hotspot traffic. The packet latency variation is calculated using latency difference for consecutive packets within one flow where a flow is defined as the traffic from a source to the hotspot destination. Age-based arbitration provides the tightest distribution with the lowest variance but all three arbitration weight metric also achieve a very tight distribution with slightly higher variance while the average values are nearly identical. However, locally fair round-robin arbitration not only has a higher mean value but also has a significantly higher variation.

TABLE 6
Packet Latency Variation

	mean(cycles)	max(cycles)	std dev
RR	739	3153	1026
AGE	62.93	63	0.088
VW	62.93	66.2	1.20
CW	62.96	68.8	1.96
FW	62.92	65.5	1.25

6.2 Memory Controller Traffic

We also evaluate PDBA with multiple hotspot traffic (such as the traffic to memory controller) that can occur with future many-core accelerators. We evaluate a diamond placement of memory controllers [11] with 16 memory controllers and assume a uniform random distribution to 16 memory controllers. Fig. 13 plots the accepted throughput of all the nodes that are sending traffic to the memory controllers. The 16 nodes with zero accepted throughput are the location of the memory controllers. As shown in Fig. 13(a), although the diamond placement was shown to provide good performance for on-chip network memory traffic, if round-robin arbitration is used, unfairness is created in reaching the distributed number of MCs—nodes in the middle of the chip are able to send more traffic than the nodes in the corner. Age-based arbitration is able to provide a global fairness and achieve the same throughput for all nodes [Fig. 13(b)]. Using PDBA [Fig. 13(c)], we are able to significantly reduce the unfairness compared to round-robin arbitration.

6.3 Support for Location-Oblivious Task Placement

Fig. 14 shows the distribution of overall IPC (total number of instructions executed for all the applications in each workload mix divided by the total number of simulated cycles) with random task placement. For compute-intensive workload mix shown in Fig. 14(a), the IPC distribution is similar for different arbiters except for VW, as it runs at a very low clock frequency. However, with memory-intensive and heterogeneous workload shown in Fig. 14(b) and (c), the distribution differs significantly. As opposed to RR arbiter which resulted in a very large variance in IPC, the age-based arbiter and VW resulted in a smaller variance, but were not able to provide high average IPC. On the contrary, hybrid arbiter provides low IPC variance as well as high average IPC. By providing EoS in the network, the variation in overall IPC is reduced and resulted in higher average IPC.

6.4 Performance on Synthetic Traffic Pattern

In this section, we evaluate the impact of PDBA on the performance of different synthetic traffic patterns and evaluate its impact on performance. The latency vs. throughput curve for different traffic patterns are shown in Fig. 15. Since we only modify the switch arbitration, the zero-load latencies of the different arbitration are all identical for a given traffic pattern. For some traffic patterns such as bitrev (not shown), all of the different arbitration mechanism achieve nearly identical latency vs. throughput curves. However, for other traffic patterns, the different weight metrics with PDBA result in different throughput. For example, with uniform random traffic, CW reduces the saturation throughput by

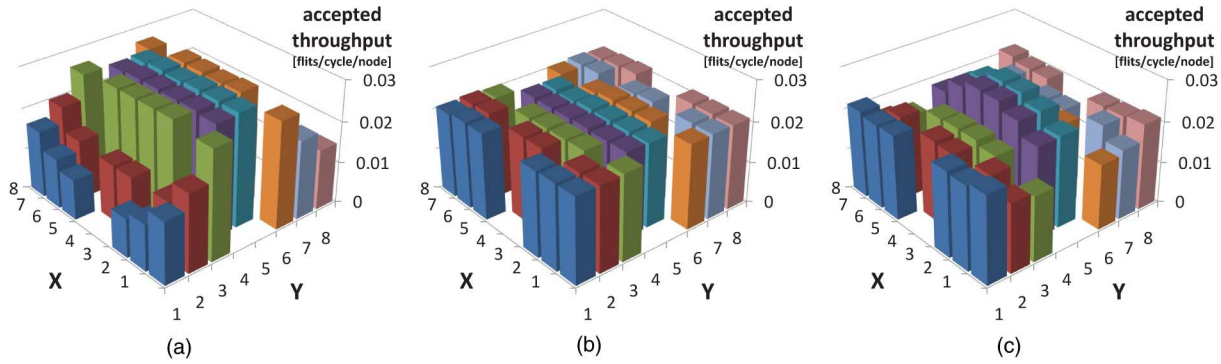


Fig. 13. Performance across multiple hotspots of (a) round-robin, (b) age-based, and (c) PDBA.

approximately 13% compared to RR while VW and FW provide better performance than CW. For tornado traffic pattern, VW approximately matches the throughput of RR—thus, the ability of providing EoS has minimal impact of performance. Across all traffic patterns, VW generally provides the highest performance compared to FW or CW because of its ability to adapt to the contention by calculating the contention degree at each router before increasing the weight.

In addition to latency vs. throughput curve, we also plot the offered load vs. *minimum* accepted throughput for the different traffic patterns in Fig. 16. For traffic patterns such as UR, regardless of the arbitration mechanism, the network continues to accept same amount of traffic past saturation. However, it is known that simple round-robin arbitration can create an *unstable* network for different permutation traffic [9]—i.e., beyond the maximum saturated accepted throughput, as the load continues to increase, the accepted throughput actually *decreases*. By providing globally fairness with age-based arbitration, the maximum accepted throughput can be maintained as offered load continues to increase as shown in Fig. 16. For RR, the throughput drop significantly because of global unfairness. The different weight metrics (FW, CW, VW) provide similar saturation throughput but differ significantly on the accepted throughput as load increases beyond saturation. For example, with transpose traffic pattern in Fig. 16(a), after saturation around 0.14, as load continues to increase, the throughput drop by approximately 67% for FW while CW and VW maintains stability. For bitcomp [Fig. 16(b)], PDBA still provides better stability than RR, with VW again providing the highest stability compared to CW and FW. However, VW cannot achieve high sustained throughput as age-based arbitration and it is noticeable in the tornado traffic [Fig. 16(c)].

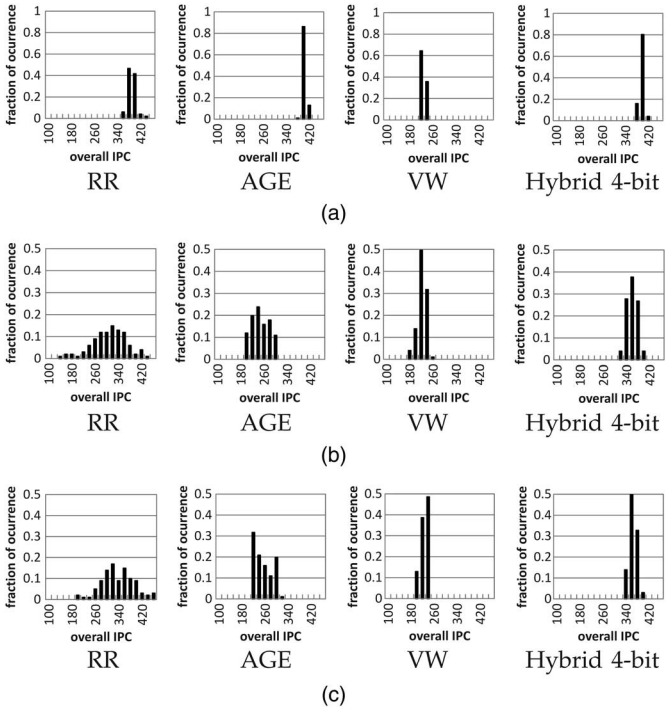


Fig. 14. Distribution of total IPC for (a) compute-intensive, (b) memory-intensive, (c) heterogeneous mix of compute-intensive and memory-intensive workloads.

In order to understand the limitations of CW and FW, we use the traffic patterns shown in Figs. 17 and 18. Fig. 17 highlights the limitation of the FW metric. Assume P1, P2, and P3 sends traffic to P4, P5 and P6, respectively. With this traffic pattern, all of the packets will have a hop count of $h_x = 3, h_y = 0$ and use a weight of $w = 2^3$. As a result, the arbitration at each router (R2, R3, R4, R5) will be round-robin arbitration because of the equal weights. Thus, more bandwidth will be serviced to P3 while the bandwidth used by packets from P1 and P2 will be reduced geometrically—thus, reducing the minimum accepted throughput beyond saturation.

The traffic pattern in Fig. 18 highlights the limitation of CW PDBA. Assume P0 sends traffic to P7 and P5 sends traffic to P6 and assume the other nodes in the row P1 - P4 are sending traffic to another node in the same column and does not require traversing any channel within this row. With a static constant degree metric using CW, the weight of packet injected at P0 continues to increase and once it reaches R5, it has a weight of 32. However, the packet injected from P5 at R5 will have weight of 1. Thus, using PDBA, P0 will receive 32/33 of the bandwidth from the channel between R5 and R6 while P5 will only obtain 1/33 of the bandwidth—unfairly, biasing the packet that have traveled long distance. Ideally, since there are only two flows sharing the channel between R5 and R6, each should access 1/2 the bandwidth. In order to overcome the limitation of CW, *variably* increasing weight metric is needed. Thus, for packet that is injected at P0, it does not encounter any contention until it reaches R5 and maintains a weight of 1. At R5, $w1 = w2 = 1$ and each flow from P0 and P5 will be serviced approximately equally.

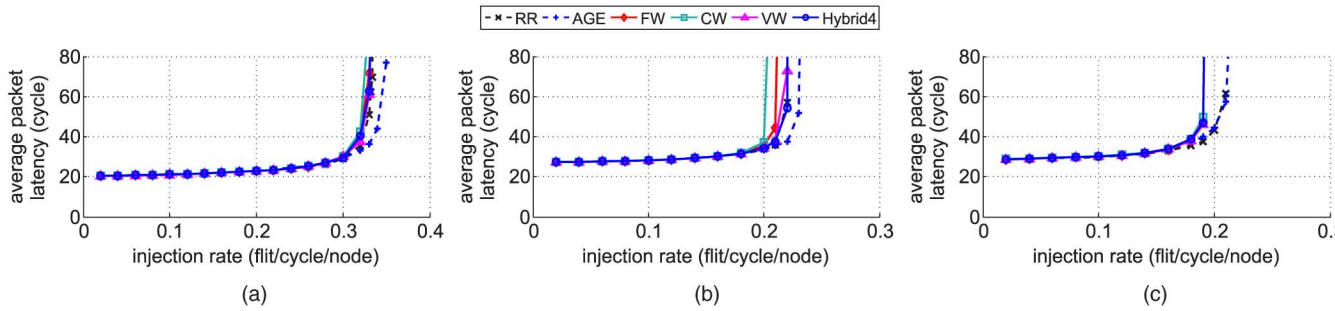


Fig. 15. Latency throughput curve for (a) uniform random, (b) tornado, and (c) bitcomp traffic patterns.

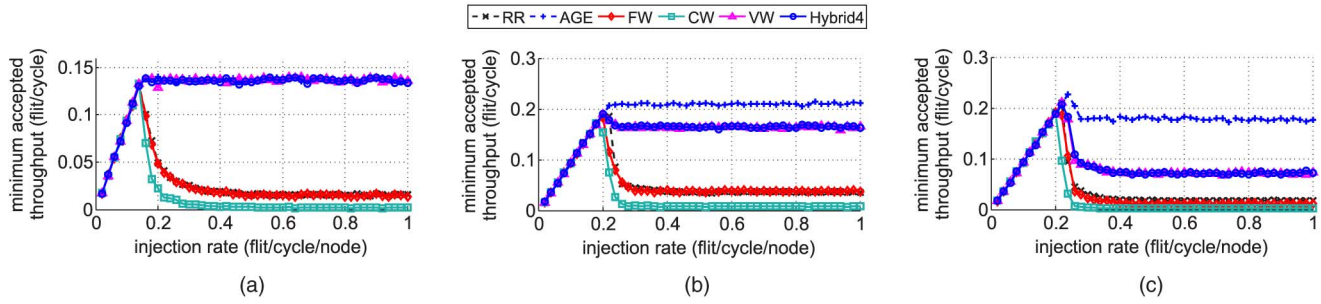


Fig. 16. Offered load vs. minimum accepted throughput for (a) transpose, (b) bitcomp, and (c) tornado traffic patterns. The legend is shown in Fig. 15.

6.5 Hardware Overhead

The overhead of the hybrid arbiter is shown in Table 7. The baseline PDBA (VW) incurs a high overhead compared to RR arbiter as the clock frequency is decreased by $5.5\times$ and the area is increased by $3\times$. By using 4-bit approximation for packet's weight and hybridization with RR arbiter, the frequency overhead is reduced to 56%, achieving 1050 MHz, and the area overhead is reduced to 17% compared to RR. Although the clock frequency is slower than RR (1640 MHz), as shown in Fig. 14, the system-level performance is improved over RR by supporting EoS in the network.

We compared the *additional* area overhead of different QoS mechanisms [3], [4], [17] to that of the hybrid arbiter in Fig. 19. The overhead of baseline PDBA is taken from Table 7. For GSF [3], we only estimated the overhead of the source buffer with 1000 flits. For LOFT [17], the total area is estimated by using CACTI [18] for SRAM-based structures such as reservation table, input table and flow status table, and synthesizing its lookahead network. Likewise, for PVC [4], we used CACTI to estimate the area of the SRAM-based source buffer with 30 flits and flip-flop based flow state registers with 10 flows, and synthesized its ACK network. Compared with the four state-of-the-art QoS mechanisms for NoCs, our proposed hybrid arbiter achieved a significantly lower area overhead. Although our work does not support strict QoS guarantees, such as packet delay bound or minimum throughput, our hybrid arbiter does provide practical EoS with much lower cost.

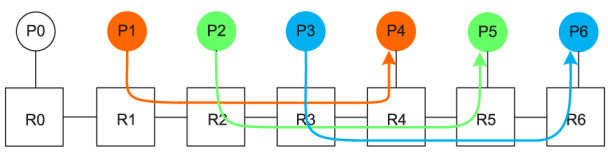


Fig. 17. Traffic pattern that highlights the limitation of the fixed weight PDBA.

7 Related Work

Probabilistic techniques for centralized arbitration/scheduling have been proposed in OS scheduling and system-on-chip shared bus system. Lottery scheduling [19] chooses a thread to run using random numbers and Lotterybus [20] uses probabilistic arbitrations to choose the owner of a shared bus. Probabilistic arbitration has also been proposed within memory schedulers [21]. These works have a single centralized arbiter/scheduler, but our work uses multiple distributed probabilistic arbiters in on-chip networks and we present novel weight metrics to achieve fairness with probabilistic arbiter. Distance or hop count was also used in the arbitration within Aérgia architecture [22] where they used hop count to determine *slack* calculation and provide application-level fairness.

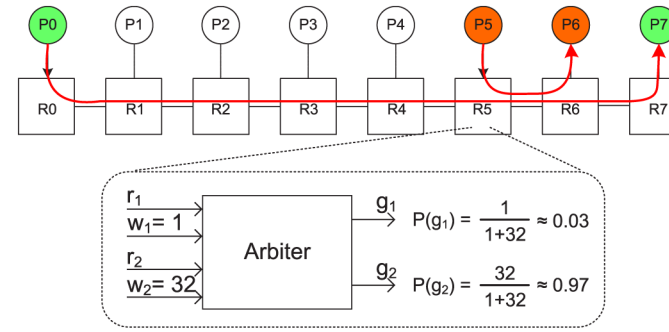


Fig. 18. Traffic pattern that highlights the limitation of the constantly increasing weight PDBA. r_1, w_1 represents a packet from P0 and r_2, w_2 represents a packet from P5.

TABLE 7
Clock Frequency and Area of Routers with Different Arbiters Obtained from Synthesis of RTL

	RR	VW	Hybrid (4-bit)
Freq. (MHz)	1640	300	1050
Area (mm^2)	0.0441	0.1599	0.0519

Although EoS has been well investigated in other fields such as computer networking and real-time system, their solutions cannot be easily applied to on-chip environment because of different constraints, compared to off-chip. We divide the solution space into two classes. The first class of approaches is based on injection rate control. Injection rate control can be placed at either the injection point of each source or the input channel of each intermediate node to limit the maximum number of flits a network or an individual node can service for each flow over a period of time. This time period of bandwidth accounting is called *frame* in some proposals [23], [3], [5]. Æthereal [24] uses pipelined time-division-multiplexed (TDM) circuit switching to implement guaranteed performance services. Each 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 fair bandwidth share even if the network is underutilized. SonicsMX [25] can support EoS without explicit channel setup. However, each node has to maintain per-thread queues, which make it only suitable for a small number of threads. QNoC [26] takes a source regulation approach and 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, QNoC requires a sophisticated secondary network (either logical or physical) for credit token request/reply not to slow down the source injection process and potentially penalizes short-lived flows because of credit token fetch.

The second class of approaches proposes sophisticated arbitration techniques to provide EoS. (Weighted) Fair Queueing [27] and Virtual Clock [28] are best-known queueing and scheduling algorithms in this class. They are developed for EoS 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. The MediaWorm router [29] evaluates the performance of both Fair Queueing and Virtual Clock in a multiprocessor environment using multimedia traffic. Weighted round-robin arbiter [9] is a degenerated form of Fair Queueing, which does not take packet size into account.

QoS support in multi-hop on-chip networks has been an active research area in recent years. GSF [3] takes a frame-based approach to provide guaranteed QoS for all the flows in terms of minimum bandwidth and maximum delay without maintaining per-flow data structures at each router. However, it still requires significant hardware overhead for source buffers, multiple frame buffers (VCs) and a secondary barrier network, as discussed in Section 6.5. Also, GSF potentially underutilizes the network bandwidth if the bandwidth reservation parameters are poorly selected. There are several proposals to overcome these limitations of GSF, including

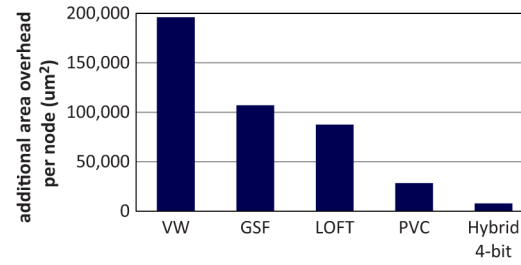


Fig. 19. Comparison of area overhead for different QoS mechanisms.

LOFT [17], and PVC [4]. LOFT is based on another frame-based technique, called Locally Synchronized Frames (LSF), which is augmented with flit-reservation (FRS) to improve the network utilization. However, LOFT still incurs a high area overhead. as shown in Fig. 19, for its three tables at each router (reservation, input and flow status tables) and a lookahead network. PVC [4] customizes Virtual Clock [28], which was originally proposed for IP networks, for efficient implementation in a resource-constrained on-chip environment. To reduce the buffer overhead, routers do not have per-flow buffers. Instead, a router simply drops the lowest priority packet from the network when it is blocking a higher priority packet and the buffer is full. In this case, a 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, compared to Virtual Clock because of retransmission overhead. Although PVC reduces the buffering overhead compared to Virtual Clock or GSF, each PVC router is still required to maintain per-flow structures, such as bandwidth counters and reserved rate registers, whose overhead becomes significant with a large number of nodes.

To mitigate the performance variation problem caused by resource contention and task placement, there are purely software-based approaches. Zhuravlev et al. proposed AKU-LA [30], which provides API and a rapid evaluation model to create custom contention-aware scheduling algorithms. They also proposed an algorithm for classifying an application's execution behavior, which can be exploited by the scheduler to minimize resource contention [31]. Das et al. introduced the notion of *cluster* to reduce contention in the on-chip network, as well as memory controllers [32]. By grouping cores into clusters and having them mostly use the memory controller assigned to the cluster, inter-cluster interference is reduced.

This work extends our prior study [2] with implementation details/complexity analysis, different methods to reduce hardware overhead and latency, and evaluations using real GPGPU workloads regarding location-oblivious task placement.

8 Conclusions

In this work, we presented probabilistic distance-based arbitration to provide equality-of-service (EoS) and support location-oblivious task placement in network-on-chip. By using probabilistic distance-based arbitration (PDBA) with nonlinear weight metric to approximate the age of a packet, we showed PDBA can approach the behavior of ideal

age-based arbitration and provide EoS regardless of the location of the node. However, the baseline PDBA can result in $3\times$ area overhead and $5.5\times$ increase in clock cycle because of the complexity in implementing PDBA. In order to reduce the hardware overhead, we describe how the weights used in PDBA can be approximated and propose a hybrid arbiter that switches between a complex arbiter and a simple arbiter to minimize the impact on critical path. According to our simulation of spatial multitasking with randomly placed GPGPU benchmarks, the standard deviation of IPC is reduced by $3\times$ $(4.24\times)$ for memory intensive (heterogeneous) workload mix and the average IPC is improved by 17% (8%) for memory intensive (heterogeneous) workload mix compared with RR arbiter.

Acknowledgment

This research was supported in part by the MKE, Korea, under the ITRC program supervised by the NIPA (NIPA-2013-H0301-13-1011) and in part by the NRF grant funded by the Korea government (MSIP) (NRF-2013R1A2A2A01069132).

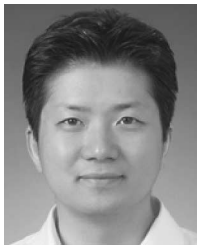
References

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Conf. Des. Autom. (DAC)*, 2001, pp. 684–689.
[2] M. M. Lee et al., "Probabilistic distance-based arbitration: Providing equality of service for many-core CMPs," in *Proc. 43rd Int. Symp. Microarchitecture*, 2010, pp. 509–519.
[3] J. W. Lee et al., "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Proc. Int. Symp. Comput. Architecture (ISCA)*, 2008, pp. 89–100.
[4] B. Grot, S. W. Keckler, and O. Mutlu, "Preemptive virtual clock: A flexible, efficient, and cost-effective QOS scheme for networks-on-a-chip," in *Proc. Int. Symp. Microarchitecture (MICRO)*, New York, NY, 2009.
[5] D. Stiliadis and A. Varma, "Design and analysis of frame-based fair queueing: A new traffic scheduling algorithm for packet-switched networks," in *Proc. ACM SIGMETRICS Conf. (SIGMETRICS'96)*, 1996, pp. 104–115.
[6] A. Fedorova, "Say yes to dumb operating systems and smart applications," 2008, WACI-VI.
[7] J. T. Adriaens, K. Compton, N. S. Kim, and M. J. Schulte, "The case for GPGPU spatial multitasking," in *Proc. Int. Symp. High-Perform. Comput. Architecture (HPCA'10)*, 2010, pp. 1–12.
[8] D. Abts and D. Weisser, "Age-based packet arbitration in large-radix k-ary n-cubes," in *Proc. IEEE Conf. Supercomput. (SC'07)*, Reno, NV, 2007, pp. 1–11.
[9] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.
[10] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N. Jha, "A 4.6Tbits/s 3.6 GHz single-cycle NoC router with a novel switch allocator in 65 nm CMOS," in *Proc. Int. Conf. Comput. Des. (ICCD)*, October 2007, pp. 63–70.
[11] D. Abts et al., "Achieving predictable performance through better memory controller placement in many-core CMPs," in *Proc. Int. Symp. Comput. Architecture (ISCA'09)*, 2009, pp. 451–461.
[12] J. A. Stratton et al., "Parboil: A revised benchmark suite for scientific and commercial throughput computing," Center for Reliable and High-Performance Computing, Univ. Illinois at Urbana-Champaign, Urbana, IL, Tech. Rep. IMPACT-12–01, 2012.
[13] S. Che et al., "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC'09)*, 2009, pp. 45–54.
[14] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS'09)*, April 2009, pp. 163–174.
[15] NVIDIA Corporation. *NVIDIA CUDA SDK Code Samples* [Online]. Available: http://developer.download.nvidia.com/compute/cuda/sdk, accessed on 2012.
[16] D. U. Becker and W. J. Dally, "Allocator implementations for network-on-chip routers," in *Proc. IEEE Conf. Supercomput. (SC'09)*, 2009, pp. 1–12.
[17] J. Ouyang and Y. Xie, "LOFT: A high performance network-on-chip providing quality-of-service support," in *Proc. Int. Symp. Microarchitecture (MICRO'10)*, 2010, pp. 409–420.
[18] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2007, pp. 3–14.
[19] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," in *Proc. Oper. Syst. Des. Implementation (OSDI)*, 1994, article 1.
[20] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "LOTTERYBUS: A new high-performance communication architecture for system-on-chip designs," in *Proc. Conf. Des. Autom. (DAC'01)*, 2001, pp. 15–20.
[21] I. Hur and C. Lin, "Adaptive history-based memory schedulers," in *Proc. Int. Symp. Microarchitecture (MICRO)*, 2004, pp. 343–354.
[22] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aérgia: Exploiting packet latency slack in on-chip networks," in *Proc. Int. Symp. Comput. Architecture (ISCA'10)*, 2010, pp. 106–116.
[23] J. H. Kim and A. A. Chien, "Rotating combined queueing (RCQ): Bandwidth and latency guarantees in low-cost, high-performance networks," in *Proc. Int. Symp. Comput. Architecture (ISCA'96)*, 1996, pp. 226–236.
[24] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test*, vol. 22, no. 5, pp. 414–421, Sep./Oct. 2005.
[25] W.-D. Weber, J. Chou, I. Swarbrick, and D. Wingard, "A quality-of-service mechanism for interconnection networks in system-on-chips," in *Proc. Des. Autom. Test Eur. (DATE)*, Munich, Germany, 2005, pp. 1232–1237.
[26] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, no. 2–3, pp. 105–128, 2004.
[27] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM Conf. SIGCOMM*, Austin, Texas, 1989, pp. 1–12.
[28] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," *SIGCOMM Comput. Commun. Rev.*, vol. 20, no. 4, pp. 19–29, Aug. 1990.
[29] K. H. Yum, E. J. Kim, C. R. Das, and A. S. Vaidya, "MediaWorm: A QoS capable router architecture for clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 12, pp. 1261–1274, Dec. 2002.
[30] S. Zhuravlev et al., "AKULA: A toolset for experimentine and developing thread placement algorithms on multicore systems," in *Proc. Int. Conf. Parallel Architectures Compilation Tech. (PACT'10)*, 2010, pp. 249–260.
[31] S. Zhuravlev et al., "Addressing shared resource contention in multicore processors via scheduling," in *Proc. Int. Conf. Architectural Support Program. Languages Oper. Syst. (ASPLOS'10)*, 2010, pp. 129–142.
[32] R. Das, R. Ausavarungnirun, O. Mutlu, A. Kumar, and M. Azimi, "Application-to-core mapping policies to reduce memory interference in multi-core systems," in *Proc. 21st Int. Conf. Parallel Architect. Compilation Tech.*, 2012, pp. 455–456.
[33] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 21–23, 2013, pp. 86–96.

Gwangsun Kim received the BS degrees in electrical engineering and computer science and engineering from Pohang University of Science and Technology (POSTECH), South Korea, and MS degree in computer science from Korea Advanced Institute of Science and Technology (KAIST). Daejeon, South Korea. He is a first year PhD student at KAIST. His research interests include in on-chip interconnection network and memory system design for many-core processors.

Michael Mihn-Jong Lee received the BS degree from KAIST, South Korea. He is a second year PhD student at the University of California, San Diego. He is working in Systems and Networking Group with Prof. Yuanyuan Zhou. His research interests include distributed system, software reliability, and computer architecture.

John Kim received the BS and MEng degrees from Cornell University, Ithaca, NY, in the Department of Electrical Engineering in 1997 and 1998. He then received the PhD degree from Department of Electrical Engineering, Stanford University, California, in 2008. He is currently an assistant professor with the Department of Computer Science, KAIST with joint appointment in the Web Science & Technology Division at KAIST. He spent several years working on the design of different microprocessors at Motorola and Intel. His research interests include multicore architecture, interconnection networks, and datacenter architecture. He is a member of ACM.



Jae W. Lee received the MS degree in electrical engineering from Stanford University, California, and the PhD degree in computer science from Massachusetts Institute of Technology (MIT), Cambridge. He is an assistant professor with the Department of Semiconductor Systems Engineering, 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.

Dennis Abts is a Member of Technical Staff at Google, where he is involved in the system architecture and design of next-generation large-scale clusters. His research interests include scalable coherence protocols, memory consistency models, interconnection networks, fault tolerant computing, and robust system design.

Michael Marty received the PhD degree in computer science from the University of Wisconsin-Madison. He is an engineer at Google and is currently working on advanced technologies for its computing platform. His interests include computer architecture, technical infrastructure, distributed software infrastructure, and data center networking.