RAIR: Interference Reduction in Regionalized Networks-on-Chip

Lizhong Chen, Kai Hwang and Timothy M. Pinkston
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA, USA
{lizhongc, kaihwang, tpink}@usc.edu

Abstract—**With the advent of many-core systems capable of hosting multiple concurrently running applications, the traffic characteristics of networks-on-chip (NoCs) may exhibit new regional behaviors. By recognizing and exploiting these traffic behaviors, the effectiveness of NoC interference reduction techniques can be greatly improved. However, few works have investigated these regional behaviors and their potential impact on interference, leaving the opportunity largely unexplored. In this paper, we identify and characterize regional behavior in NoC and propose *RAIR*, a region-aware interference reduction technique that not only removes any restrictions on the inter-region traffic patterns, but also captures and exploits regional behavior throughout the design, thus improving the effectiveness of interference reduction. Evaluation using a cycle-accurate simulator shows that RAIR can improve the average packet latency by up to 17% on synthetic traffic patterns and up to 26% on PARSEC benchmarks compared to state-of-the-art interference reduction techniques.**

Keywords: network-on-chip, regional behaviors, interference reduction

I. INTRODUCTION

Continuing advancement in manufacturing technology has enabled multiprocessor systems-on-chip (MPSoCs) and chip-multiprocessors (CMPs) to be implemented with tens to possibly hundreds of cores on a single die. For example, Intel's Single-chip Cloud Computer incorporates 48 cores onto a chip at 45nm [12], and Tilera's TILE-Gx100 chip integrates 100 general purpose cores at 40 nm in its latest offering [21]. To meet the growing communication demand among the escalating number of cores, networks-on-chip (NoCs) have been proposed as the primary communication subsystem due to its superior scalability. Meanwhile, multiple applications can be running concurrently to utilize the abundant computing resources offered by many-core chips. As all applications on a chip share the same NoC substrate, traffic from different applications may interfere with each other, potentially causing considerable performance degradation if not addressed appropriately. It is thus very important to devise effective interference reduction techniques for on-chip networks when multiple and possibly distinct applications run concurrently.

Of paramount importance to improving the effectiveness of interference reduction is to identify and utilize the traffic characteristics exhibited in the NoCs. While the issue of reducing interference has been looming in recent years, very little research has explored traffic behaviors resulting from the mapping of multiple applications onto many-core NoCs. A number of recent optimizations targeting system components other than the NoC place threads belonging to the same application closer to each other [20, 23] or move frequently accessed cache data closer to the requesting cores [13, 14, 15], so as to reduce the communication delay and minimize overall traffic volume. These techniques essentially transform a considerable amount of chip-wide traffic into short-range traffic within physically-close groups or regions, leaving a small amount of traffic needing to traverse among regions – a case we call *regionalized networks-on-chip (RNoCs)*.

However, until recently, most NoC interference reduction techniques (e.g., round-robin, age-based [1], and application-aware prioritization [6]) are oblivious to regional traffic behaviors. These region-oblivious techniques are inherently unable to exploit regional behavior and, thus, have limited effectiveness in the case of RNoCs. Recent works on multiple concurrently running applications have started to consider regional behavior in their interference reduction techniques [11, 16, 22]. However, these techniques either place various strict restrictions on traffic patterns (e.g., inter-region traffic is disallowed in [22]), or reduce merely part of the possible interference in RNoCs, thereby limiting their usefulness to only some particular RNoC scenarios

In this paper, we address the looming issue of traffic interference reduction by proposing *Region-Aware Interference Reduction (RAIR)*, which captures the regional behaviors of RNoC to minimize interference for generic RNoCs without any restrictions on traffic patterns. Specifically, three mechanisms are devised to achieve this. The first mechanism, *VC regionalization*, enables inter-region traffic to traverse freely across the chip while still being treated differently from intra-region traffic. The second mechanism, *multi-stage prioritization*, solves the problem of how to efficiently and effectively enforce traffic prioritization in different stages of the pipelined router microarchitecture to reduce interference, and the third mechanism, *dynamic priority adaptation*, addresses the issue of recognizing and utilizing load heterogeneity among regions and providing starvation avoidance. The main contributions of this paper are the following:

- Formation of RNoCs resulting from recent proposals are analyzed and key regional behaviors are identified;
- A new region-aware interference reduction technique is proposed consisting of VC regionalization, multistage prioritization and dynamic priority adaptation, to take full advantage of the regional behaviors exhibited in RNoCs to improve effectiveness;
- Evaluation using a cycle-accurate simulator shows a reduction of average packet latency by up to 17% on synthetic traffic patterns and up to 26% on PARSEC benchmarks.

The rest of the paper is organized as follows. Section II discusses the formation of RNoCs and identifies the resulting regional behaviors. Section III summarizes related work on reducing interference and investigates their applicability in the RNoC environment. Section IV provides the details of the proposed region-aware interference reduction technique. Section V presents our evaluation methodology and simulation results. Finally, several related issues are discussed in Section VI, and Section VII concludes the paper.

II. Background on Regionalized NoC

A. Formation of Regionalized NoC

A *Regionalized Network-on-Chip (RNoC)* refers to an on-chip network in which traffic exhibits clustered regional patterns, as if the network is divided into multiple regions. A *conventional NoC* can be considered as a special case of RNoC in which the number of regions equals one. To better illustrate the concept of RNoC, three examples are described below, each of which represents a class of techniques having the similar goal of leveraging non-uniformity in many-core chips.

1) Example 1: Application-to-core mapping

The first class of techniques that leads to RNoC is application-to-core mapping. This class of optimizations is based on the observation that, in MPSoCs or CMPs with multiple concurrently running applications, each application may in turn spawn a few parallel threads working simultaneously on multiple cores. With non-uniform core-to-core distance in large many-core chips, significant performance improvement can be achieved when frequently communicating threads belonging to the same application can be placed closer by mapping them to cores[1] with fewer hop distance. For example, by using mapping policies based on the above intuition instead of randomly assigning threads to cores, over 53% bandwidth savings and 23% delay reduction are achieved in [20] and [23], respectively. This is mainly because most of the previously chip-wide long-distance traffic is now converted to short-distance communications, thereby reducing both traffic volume and latency. A direct result of these proximity-based optimizations is the formation of regionalized NoC as, essentially, the multiple collaborating threads of an application are clustered into physically-close groups through the mapping process, and the majority of traffic occurs within each application's region with a small fraction of traffic traversing among regions.

2) Example 2: Cooperative cache structure

At the system level, there are a number of recent proposals that initially target critical problems in system components other than the NoC but in fact, indirectly precipitate the occurrence of RNoC. For instance, several techniques have been proposed to optimize cache structure for many-core chips [13, 14, 15]. The idea of these approaches is that, instead of uniformly distributing data in cache banks across the chip, active data that are needed by an application are adaptively moved closer to the running threads of that application. In this way, cache access time can be greatly reduced. In [13], for example, L2 cache access latency is reduced by 33%~54%, most of which comes from the reduced traffic hops (e.g., request and reply messages). Notice that, as more cached data can be accessed in the local or nearby nodes, a considerable amount of chip-wide traffic is transformed into regional traffic, essentially resulting in an RNoC configuration.

3) Example 3: Coherence protocol optmization

Another example is coherence protocol optimization for on-chip server consolidation, where multiple virtual machines (VMs) run concurrently in a CMP, with each VM having a designated region. In [19], a two-level coherence hierarchy is proposed to accelerate coherence transactions based on dynamic home nodes. The basic idea is to select the home node of cache lines wisely so that the amount of protocol messages that need to traverse outside their designated region are minimized. As a result, the cycle-per-transaction is reduced by 15%~65% depending on the applications, indicating that a large percentage of the protocol transactions have been satisfied within a region. Therefore, although the intention is to optimize coherence protocols, a side-effect is an increase in intra-region traffic and a decrease in inter-region traffic, thus indirectly changing the traffic pattern toward RNoC.

B. Regional Behaviors of RNoC

In the above examples, while the degree of "regionality" may differ, there are some common influences and behaviors affecting regionality that can be identified and summarized by the following regional behaviors (RBs):

- RB-1: Multiple applications run concurrently in a many-core chip, each consisting of one or more nodes;
- RB-2: Nodes belonging to the same application are often clustered into a region;
- RB-3: Majority of traffic becomes intra-region with a small fraction being inter-region, and
- RB-4: Different regions may have heterogeneous traffic characteristics (e.g., different intensity)

These regional behaviors raise new challenges and opportunities for traffic interference reduction in on-chip networks. For instance, not only can an application running in one region benefit from interference reduction, but it may also *require* interference reduction from applications in

[1] We use the terms *core* and *node* interchangeably as a core in a chip is abstracted to a node in the corresponding NoC.

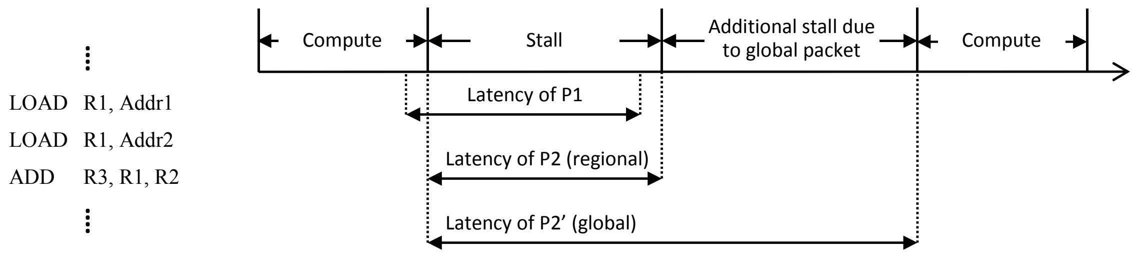


Figure 1: Example of performance criticality for regional and global traffic.

other regions. This is particularly true in the above server consolidation example where if one VM goes awry or is under malicious attack, the remaining VMs should be minimally affected. However, interference reduction in RNoCs is much harder to achieve than before as we can no longer strictly confine packets to flow within a region; otherwise, an application would be unable to access certain shared resources, such as memory controllers located outside its region, to perform normal operations. In addition, by taking into account the regional behaviors exhibited by RNoCs, it is possible to achieve more effective interference reduction and, thus, higher performance improvement for multiple concurrently running applications.

C. *Intra-region Traffic vs. Inter-region Traffic*

To facilitate the following discussion, we define a few terms that are helpful in illustrating the traffic properties in RNoC. The terms *regional traffic* and *global traffic* refer to the intra-region traffic portion and inter-region traffic portion of an application, respectively. Using the previous examples, global traffic can be the traffic to and from memory controllers located outside the region, the requests and replies of cooperative cache data in other regions due to misses in the local region, and the inter-VM data sharing in server consolidation. In addition, for application *A* mapped to region *R* of the RNoC, *native traffic* refers to the traffic in region *R* that belongs to application *A*, and *foreign traffic* refers to the traffic currently traversing region *R* but belonging to applications not mapped to that region. For example, global traffic of application *B* (not mapped to region *R*) is foreign traffic of region *R* when traversing *R*.

In many-core chips, global traffic is usually more critical to performance than regional traffic. Figure 1 depicts a simple example of instruction execution in a core during a program's execution. Suppose the two LOAD instructions on the left miss in the local cache and, therefore, packets *P1* and *P2* are sent to request data on other nodes. ADD cannot be computed until both requested data are returned. Assume *P1* is regional traffic (i.e., the request can be satisfied within the region where the core belongs). Now consider the request in *P2*. As shown in the figure, if *P2* is also regional traffic, its latency should be similar to that of *P1*, and the reply of *P2* is back only slightly after the reply of *P1*. In other words, the latency of *P2* can be largely overlapped with the latency of *P1* (in a better case, the reply of *P2* may be back sooner than that of *P1,* and the latency is completely overlapped). However, if *P2'* is global traffic (i.e., the request can only be satisfied by the node in some other region), then the reply of *P2'* comes back much later than that for *P1*. As a result, a large portion of the latency of *P2'* cannot be overlapped, which incurs additional stall cycles directly on the critical path of the program's execution. Therefore, compared with regional traffic, global traffic is more likely to have higher criticality. Another factor that affects the criticality of regional and global traffic is load intensity. As observed in [6], low intensity traffic is usually more critical than high intensity traffic (details will be discussed in later sections). According to RB-3, global traffic in the RNoC is likely to have lower load than regional traffic, thus making global traffic even more critical.

III. Related Work and Motivation

A few techniques have been proposed to tackle interference reduction in on-chip networks, but they all have either no or very limited region-awareness, as discussed below.

A. *Region-oblivious Techniques*

Region-oblivious techniques are those that do not distinguish the characteristics of different regions in the on-chip network, thus are inherently unable to exploit regional behavior to achieve effective interference reduction. The early proposals in this category, such as round-robin and oldest-first, are both application- and region-oblivious. In these techniques, due to application-unawareness, packets that are critical to an application are treated equally with packets that are not critical to another application such that more performance-critical packets are not accelerated over less critical ones. When applied in RNoCs, these techniques are even more ineffective as they also treat the regional traffic equally with global traffic, which is usually not the case considering their differences in packet latency and traffic intensity.

Recent region-oblivious techniques incorporate application-awareness, which avoids the disadvantages of the application-oblivious ones by taking application characteristics into consideration. They are, however, still subject to the limitations caused by their region-oblivious nature. For example, STC [6] is an application-aware interference reduction technique for conventional NoCs, based on ranking the relative importance of applications. Among concurrently running applications, STC prioritizes network non-intensive applications (in terms of L1 misses per instruction) over network intensive applications, and for

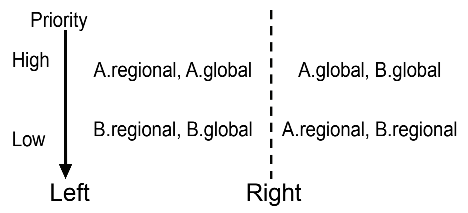


Figure 2: Prioritization on per-application basis is possible in STC (left) but not on per-region basis (right).

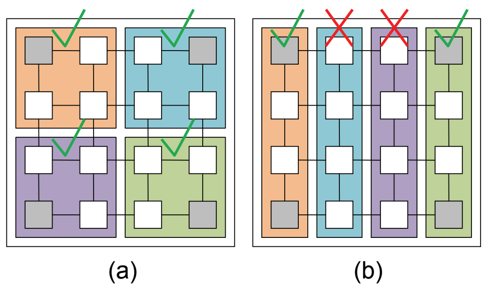


Figure 3: (a) Valid mapping and (b) Invalid mapping in LBDR.

packets belonging to the same application, a round-robin technique is used. The rationale behind STC is that: 1) low intensity applications issue requests relatively infrequently, so prioritizing these requests allows the applications to make faster progress without burdening the network much; 2) low intensity applications are likely to have low MLP and, hence, their requests are likely to be stall-time critical.

While STC has been shown to be effective for conventional NoCs, it does not consider the regional layout (RB-1 and RB-2 regional behaviors) of RNoC and the resulting regional and global traffic classification (RB-3). Consequently, if used in RNoCs, the prioritization of STC is suboptimal both within and among applications. First, within an application, regional traffic and global traffic are always treated equally in STC when they should not be because of the different criticalities of each traffic type as mentioned before. Second, among applications, as Figure 2 shows, STC prioritizes both regional and global traffic of network non-intensive application *A* over network intensive application *B* (left scenario). However, because of the region unawareness, STC cannot recognize and achieve prioritizations that differentiate regional and global traffic, such as the one shown in the right scenario. Also, to avoid starvation, packets in STC are divided into batches (e.g., based on time) and older batches always have higher priority than younger batches. In a regionalized NoC, for example, if a VM assigned to a region becomes faulty and injects a considerable amount of packets, batching may unnecessarily prioritize those packets over other normal but younger packets.

In summary, without further work to incorporate region-awareness, the above techniques proposed for conventional NoCs would have limited effectiveness when applied to regionalized NoCs.

B. *Region-aware Techniques*

Region-aware techniques aim to reduce interference in regionalized NoCs by taking into account regional behaviors in the first place. Techniques in this category can be further classified based on whether they place restrictions on traffic patterns or not. *Restricted interference reduction techniques* assume that certain traffic patterns are disallowed in the network in order to minimize interference. For example, global traffic may not be allowed or can only be entered into certain areas of the NoC. In contrast, *non-restricted interference reduction techniques* do not place any restrictions on traffic patterns to reduce interference, hence have broader application but are also much harder to realize.

Logic-Based Distributed Routing (LBDR) [8, 22] reduces region-aware interference by confining packets to traverse only within a designated region via routing restrictions. Since no global traffic outside a region is allowed, LBDR is a restricted technique. As each application must also access the memory controllers (MCs) for on-chip cache misses, LBDR requires each region to contain at least one MC. For example, the mapping in Figure 3(a) is a viable configuration but the mapping in Figure 3(b) is invalid as the two middle regions cannot access any MC. In fact, with 16 cores, 4 MCs and 4 applications (each having 4 threads), only

$$4!\binom{12}{3}\binom{9}{3}\binom{6}{3}\binom{3}{3}/\binom{16}{4}\binom{12}{4}\binom{8}{4}\binom{4}{4} \approx 14\%$$

of all possible configurations is allowed, which greatly restricts the opportunity to find the optimal application-to-core mapping. Moreover, the number of regions that can be accommodated on the chip is at most the number of MCs. As a reference, Intel's recent 48-core SCC chip [12] has 4 MCs, supporting only a maximum of 4 regions if LBDR is used. Therefore, the restrictions placed on the global traffic in LBDR result in severe limitations.

In [11], Kilo-NoC is proposed as a low overhead quality-of-service (QoS) scheme. Although it is designed for on-chip service guarantees, it can be extended for the purpose of interference reduction[2]. Kilo-NoC proposes an elaborate design by taking advantage of certain topologies. QoS rules are imposed only on a few selected routers in the so-called shared regions (SRs). Nevertheless, the global traffic in Kilo-NoC is restricted. For instance, the global traffic may need to use a single-hop long-distance connection to "bypass" the intermediate region and reach the SR. In addition, global traffic may need to detour through the SR in order to enforce QoS. More importantly, Kilo-NoC greatly depends on MECS-like topologies that can provide rich connectivity. These restrictions that Kilo-NoC places on the flow of global traffic and on the underlying network topologies limit its usefulness to only particular RNoC scenarios.

[2] The differences between the interference issue focused in this paper and the general QoS will be discussed in Section VI.

A non-restricted technique, DBAR, is recently proposed in [16] by Ma et al. As pointed out by the authors, DBAR not only is a mechanism for load-balanced routing but also serves as a technique to reduce inter-region interference. The novelty is that, in the selection function, DBAR discards the redundant information generated by other regions, thus reducing interference among different regions. Figure 4 illustrates this idea. Suppose region *R0* has low load and regions *R1-R3* have high load as given by their shading. A packet destined to *X* is currently in router *S*, so it needs to evaluate the congestion status of the east and south directions. Different from a prior congestion-aware technique [9] that uses congestion information along the entire path from *S-to-C* and *S-to-E*, DBAR only uses the congestion information along the path from *S-to-A* and *S-to-D*. In this way, the high-load status of regions *R1* and *R2* will not interfere with packets in region *R0*. Using the previously defined terms, DBAR successfully avoids interference between native traffic of different regions (e.g., no interference between the native traffic in *R0* and the native traffic in *R1*).

However, DBAR cannot reduce interference between the native and foreign traffic of a given region. Consider a packet that is sourced from *S*, destined to *Y* and currently in *B* (i.e., it becomes foreign traffic with respect to *R1*). Now, even with DBAR, regardless of which direction the packet will take next, it will inevitably be slowed down by the heavy load condition in *R1*. Therefore, although inter-region traffic is not strictly disallowed (i.e., unlike LBDR), DBAR only works best when all packets are intra-regional. As inter-region traffic must still be supported in RNoCs, a more effective way would be to recognize all the four regional behaviors involving both intra-region and inter-region traffic and reduce their interference accordingly.

In summary, on the one hand, priority-based region-oblivious techniques do not place any restrictions on traffic patterns, but their inherent unawareness of regional behaviors greatly limits their usefulness in RNoCs. On the other hand, current region-aware techniques are built based on the regional behaviors of RNoCs, but they either place strict restrictions on traffic patterns, or reduce only part of the possible interference in RNoCs, which limit their effectiveness in generic RNoCs. A more effective technique is proposed in the next section.

IV. Region-aware Interference Reduction

We propose *RAIR (region-aware interference reduction)* that captures the regional behaviors of RNoC to minimize interference for generic RNoCs without any restrictions on traffic patterns. To achieve this, we propose three mechanisms that work cooperatively to meet existing challenges. Our first mechanism *VC regionalization* answers the question of how inter-region traffic can traverse freely across the chip while still being treated differently from intra-region traffic. The second mechanism *Multi-stage prioritization* solves the problem of how to efficiently and effectively reduce traffic interference in different stages of the pipelined router microarchitecture, and the third mechanism *Dynamic priority adaptation* addresses the issue of how to recognize and utilize load heterogeneity among regions and provide starvation avoidance. All three mechanisms take advantage of the regional behaviors exhibited in RNoCs, thus improving the effectiveness of interference reduction.

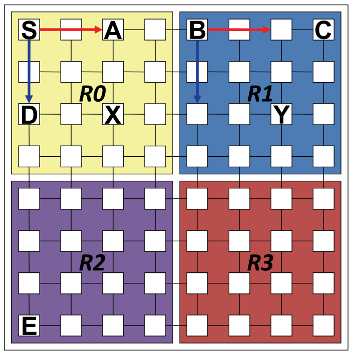


Figure 4: DBAR becomes ineffective when packets traverse outside the originating region (more heavily loaded regions have darker shade).

A. Removing Restrictions

To allow inter-region traffic to use any physical channel freely in the chip but still be differentiated from intra-region traffic of other applications, we need some way of separating the shared physical resources to reduce interference. In the first mechanism, *VC regionalization*, virtual channels (VCs) associated with a physical channel are classified into *regional VCs* and *global VCs*. Both classes of VCs can be used by any of the native or foreign traffic, but different prioritization is imposed in each class so that native traffic and foreign traffic are treated differently. Specifically, a 1-bit field is tagged to each VC to indicate the classification. Figure 5 shows one possible example where VC0 and VC1 are global VCs and VC2 and VC3 are regional VCs. The prioritization policy is that, in the global VCs, foreign traffic always has higher priority than native traffic to reflect that foreign traffic is typically more performance critical because of its global nature. In the regional VCs, however, the priority between native and foreign traffic is configured dynamically to reflect the intensity difference between native and foreign traffics at runtime, in addition to the latency factor. This dynamic priority configuration is set by DPA logic (dynamic priority adaptation) shown in Figure 5, which makes the decision based on the relative criticality among different traffic types (details will be discussed in Section IV.C).

Overall, by classifying virtual channels and adopting dynamic prioritization, VC regionalization can achieve the following advantages. First, inter-region traffic can freely traverse any physical channels allowed by the routing algorithm. Meanwhile, when necessary, it can be minimally affected by the intra-region traffic by traversing global VCs (always have higher priority). Second, VC classification is realized using priority instead of strict partitioning, which allows each type of traffic to access any virtual channels (though with different priority). Hence, no VC resource is

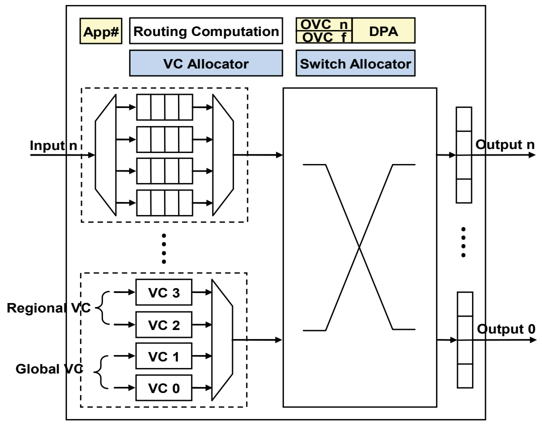


Figure 5: RAIR router microarchitecture. Light-shared blocks are added; dark-shared blocks are modified.

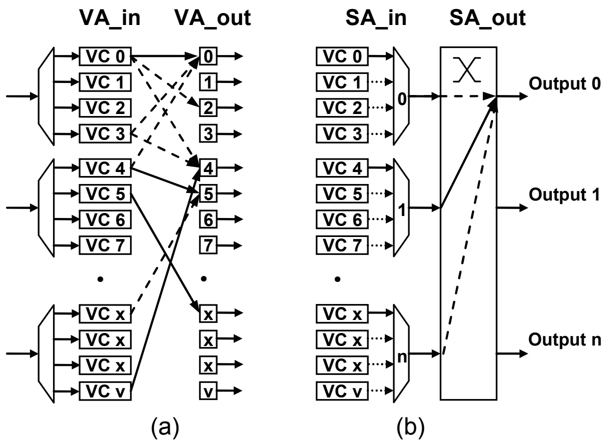


Figure 6: Arbitration steps in (a) VA; (b) SA stage.

wasted even when one type of traffic is absent. Third, each region only needs to maintain information for two flows, namely native and foreign traffic. If the foreign traffic consists of global traffic from multiple applications, round-robin is used within the foreign traffic based on two insights: 1) if multiple contending flows have light loads, priority-based policies can only reap marginal benefit as the contention is minimal in the first place; 2) global traffic indeed has low load according to RB-3, which results from the initial intention of RNoC to minimize chip-wide communication. Therefore, we target at reducing the primary contention between native and foreign traffic, and use simple fair arbitration within the foreign traffic to reduce complexity.

B. *Enforcing Prioritization*

While VC regionalization fulfills the objective of removing restrictions on traffic patterns by making virtual channels "aware" of the existence of regions, it accomplishes only part of the objective of reducing interference as VCs are not the only resource for which traffic flows are sharing and competing. There are multiple arbiters within a router to allocate different shared resources to consumers. In order to reduce interference effectively among traffic flows, we propose our second mechanism, *multi-stage prioritization (MSP)*, which not only enforces prioritization in multiple arbitration steps in the router, but also takes into account the characteristics of each step and the regional behaviors in optimizing the prioritization. In a NoC composed of canonical routers, each hop consists of routing computation (RC), VC allocation (VA), switch allocation (SA), switch traversal (ST) and link traversal (LT). There are four major arbitration steps in these stages, and the design choices for each step are explained below.

VA input arbitration (VA_in). In general, the routing function may return multiple valid output VCs for a given input VC. For example, in Figure 6(a), input VC0 is allowed by the routing function to request output VC0, 2 and 4. The function of VA_in is to return one of these requests for each input VC. For example, for input VC0, the request to output VC0 is granted (solid arrow) and the other two are denied (dotted arrows). Note that each input VC performs arbitration independently and is free to request any desired output VC without contention from other input VCs. Therefore, in the proposed MSP, no change is made to VA_in as different traffic flows do not content with each other yet at this arbitration step. Hence, MSP incurs no additional performance loss for VA_in.

VA output arbitration (VA_out). After VA_in, an output VC may receive requests from multiple input VCs. For example, in Figure 6(a), output VC0 may be requested by input VC0, 3 and 4. The function of VA_out is to arbitrate among these requests and grant at most one winner. This is the arbitration step where we implement the priority policy of VC regionalization. As discussed in the previous subsection, the output VCs tagged as global VCs are allocated with higher priority to the input VC that contains packets of foreign traffic, whereas the output VCs tagged as regional VCs are allocated to the input VCs with priority determined by the DPA logic. These two classes of output VCs with differentiated priorities implicitly act as two non-strictly separated resources for native traffic and foreign traffic. Such separation prior to the SA stage greatly reduces the chances of priority inversion and increase the effectiveness of interference reduction. Also, in this step of VA_out, MSP still maintains high VC utilization as the prioritization will not leave an output VC idle if it is requested by any input VC.

SA input (SA_in) and SA output (SA_out) arbitration. Now that each requesting input VC has been allocated a distinctive output VC, the SA stage will set up the crossbar. Each input port has multiple input VCs, so the function of SA_in is to select one requesting input VC within an input port. For example, in Figure 6(b), input port 0 selects input VC0 among inputs VC0~3. After SA_in, as multiple input ports may request the same output port, SA_out is used to choose one winner from these requests. For example, input port 1 is granted to traverse the crossbar to output port 0. During both SA_in and SA_out arbitrations, MSP chooses either the native or foreign traffic to have higher priority as determined by DPA logic to enforce prioritization. Again, prioritization in this step of MSP neither degrades crossbar utilization nor wastes any bandwidth compared with a round-robin policy, as no resource is left idle if it is requested by any input VC. Note that the same priority produced by DPA logic is used for VA_out, SA_in and SA_out at a given time, so the priority is consistent among the different stages.

C. *Utilizing Load Heterogeneity*

We now illustrate why it is indispensable to have the DPA logic to dynamically determine priority between native and foreign traffic, and how the priority can be configured appropriately. On-chip networks typically exhibit load heterogeneity as long as more than one application is running simultaneously. Recall that, in conventional NoCs, STC utilizes load heterogeneity by prioritizing network non-intensive applications over network intensive applications. In regionalized NoCs, while a similar relationship exists between network intensity and criticality, additional care should be paid to address the new load heterogeneity across regions (RB-4) to avoid any potential abnormalities and performance degradation. To achieve this, our third mechanism, *dynamic priority adaptation (DPA)* prioritizes native and foreign traffic according to their relative criticality. To facilitate illustration, we analyze the prioritization of DPA from the viewpoint of any chosen region R with application A assigned to it. Assume currently R also has inter-region traffic of another application B (i.e., B is assigned to another region). According to our definition, traffic belonging to A is the native traffic to R and traffic belonging to B is the foreign traffic to R. Since the relative performance criticality depends on the nature of the traffic (i.e., regional or global) and on the traffic intensity, there are three different cases:

(1) A and B have similar overall network intensity (i.e., both low load or both high load). Since the foreign traffic is only a small portion of the overall traffic of B, the load of foreign traffic in R would be lower than the load of native traffic. Therefore, to benefit most from the prioritization, DPA gives higher priority to the foreign traffic as it has lower intensity (indicating higher criticality) and is global traffic (also indicating higher criticality).

(2) A is more network intensive than B (i.e., A has high load whereas B has low load). Considering that the foreign traffic is a small portion of the overall traffic of B, the load of foreign traffic would be much lower than the load of native traffic. Thus, DPA gives higher priority to the foreign traffic for the similar reasons as in (1).

(3) A is less network intensive than B (i.e., A has low load whereas B has high load). In this case, it is subtle to determine the relative criticality between native traffic and foreign traffic. On the one hand, the low intensity of A would signify to give higher priority to the native traffic, similar to the motivation of STC. On the other hand, the global nature of the foreign traffic would indicate to give higher priority to foreign traffic. In order to balance between these two factors and at the same time utilize the criticality characteristics of global traffic, DPA gives higher priority to foreign traffic by default, but will reverse the priority as soon as the intensity of native traffic exceeds that of foreign traffic.

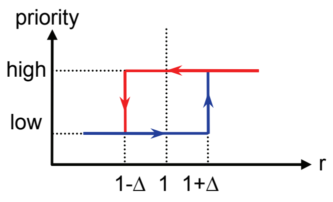


Figure 7: Hysteresis priority transition for native traffic.

Combining (1)~(3), for a given region with DPA, native traffic is given higher priority only when the relative intensity of foreign traffic is larger than that of native traffic; otherwise, foreign traffic is given higher priority. As foreign traffic is the minority component of total traffic according to RB-3, foreign traffic will have a larger chance of being higher priority than native traffic, which is consistent with the observation that global traffic is usually more critical than regional traffic. To estimate the relative intensity, prior work [3] shows that the number of occupied VCs in an input port is a strong indicator of the load status. Therefore, DPA uses similar VC information to assess intensity, but with two additional techniques to tolerate variance. First, the status of all VCs in a router is accounted for in counting the number of occupied VCs for native (*OVC_n*) and foreign traffic (*OVC_f*), instead of only the input port in which the requesting packet resides. This mitigates the inaccuracy caused by the non-uniform VC status among different ports. Second, hysteresis is used for priority transition. As shown in Figure 7, the priority of native traffic does *not* transition from low to high immediately after the ratio r of *OVC_f* over *OVC_n* is greater than 1; instead, there is a hysteresis process in which the priority only transitions after the ratio is greater than (1+Δ). Similarly, the priority transits from high to low only after the ratio becomes smaller than (1-Δ). We observed from simulation that, values of Δ between 0.1~0.3 typically render better performance with the best case achieved at around 0.2, which is assumed for Δ in our evaluation. This hysteresis transition helps to tolerate the temporal variation of VC utilization in a router.

D. *Avoiding Starvation and Deadlock*

The above implementation of DPA already avoids the starvation induced by prioritization. This is because the relative priority and the ratio consist of a negative feedback loop. For example, if native traffic occupies too many resources as indicated by a very low ratio, it will be switched into low priority. A similar negative feedback loop exists for foreign traffic as well. Thus, no starvation can occur in DPA due to this self-throttling attribute.

Regarding deadlock, unlike [8, 11, 16], none of the three proposed mechanisms constituting RAIR places any restrictions on traffic patterns or routing, so virtually any deadlock avoidance or recovery routing algorithms can be incorporated in RAIR to achieve load balance. We will evaluate the effectiveness of RAIR with two different adaptive routing algorithms. In case of using deadlock-avoidance routing algorithms based on Duato's theory [7], if a coherence protocol has multiple message classes such as MOESI, each message class is provided with additional one set of escape VCs. However, all message classes can share the same set of regional VCs and global VCs.

E. *Putting It All Together & Router Microarchitecture*

The three mechanisms with any deadlock-free routing algorithm compose our proposed RAIR technique to reduce interference for RNoCs. Figure 5 shows the modifications needed to implement RAIR. Each router is tagged with the application number that is assigned to it, and each packet carries the application number to which it belongs. When traversing a router, a packet is identified as either native traffic if the above two application numbers match or foreign traffic if not. The DPA logic keeps track of *OVC_n* and *OVC_f* in two registers and determines the relative priority between native and foreign traffic by comparing the two register values in the hysteresis manner. The calculated priority is used in VA_in, SA_in and SA_out arbitration steps. Packets then go through these steps similar to the pipeline of canonical routers but using the region-aware prioritization rules of VC regionalization and MSP. As RAIR introduces additional control dependence between DPA and VA/SA, to remove the delay of DPA logic from the critical path, we use the priority calculated from the previous cycle. This is based on the fact that the intensity difference between two consecutive cycles is insignificant and can largely be filtered by the hysteresis transition. Overall, RAIR does not impose any particular restriction on the traffic patterns and improves the effectiveness of interference reduction in RNoCs by recognizing and utilizing the regional behaviors throughout the prioritization process.

V. Evaluation

A. *Simulation Methodology*

We use a cycle-accurate interconnection network simulator, GARNET [2], to model the microarchitecture and router pipeline discussed in Section IV. A 64-node mesh network configured with 2, 4 and 6 regions is evaluated. To provide fair comparison, all schemes under evaluation are augmented with adaptive routing algorithms based on Duato's theory [7].

Both synthetic traffic patterns and application traces from multi-threaded PARSEC benchmarks [4] are used. For synthetic traffic patterns, the simulator is warmed up for 10K cycles and then the average performance is measured over another 100K cycles. We simulate uniform random (UR), transpose (TP), bit complement (BC) and hotspot (HS) traffic [5]. Packets are uniformly assigned two lengths: short packets are 16B single-flit while long packets carrying 64B data plus a head flit have 5 flits. For application traces, traffic is obtained from full-system simulation (SIMICS [17] plus GEMS [18] with sufficient warmup) configured as shown in Table 1. The simulation infrastructure supports all 13 applications in PARSEC 2.0; of these, we present results for *blackscholes*, *swaptions*, *fluidanimate* and *raytrace* as a representative subset containing both low and high intensity traffic.

Table 1: Full-system simulator configuration

Cores	64 Sun UltraSPARC III+, 1GHz
Private I/D L1$	32KB, 2-way, LRU, 1-cycle latency
Shared L2$/bank	256KB, 16-way, LRU, 6-cycle latency
Memory latency	128 cycles
Block size	64 Bytes
Virtual Channel	4 per protocol class, atomic, 5-flit/VC
Link bandwidth	128 bits/cycle

In the following subsections, the various mechanisms and techniques composing RAIR are first evaluated individually. We then combine them together as the complete RAIR and compare with other region-oblivious and region-aware techniques.

B. *Impact of Multi-stage Prioritization*

The contention among multiple concurrently running applications is the combined effects of a series of interferences: the inter-region traffic of an application interferes with the native traffic of several regions and in the meantime its own region has interference from the global traffic of multiple other applications. In order to study the separate impact of MSP on contention, we start with a simpler scenario consisting of two applications. As shown in Figure 8, App 0 and App 1 are running on the left half and right half of the chip, respectively. App 0 is configured with low load uniform traffic (10% of its saturation load) and a certain percentage p of its traffic is inter-region traffic. App 1 is configured with high load (90% of its saturation load) but all of its traffic is intra-regional. In this way, the only contention that can occur is between the inter-region traffic of App 0 and the intra-region traffic of App 1. We sweep the inter-region percentage p from 0% to 100% to assess the impact of MSP under varying degree of contention.

Figure 9 plots the average packet latency (APL) of both applications for different schemes. RO_RR is a region-oblivious technique based on round-robin. RAIR_VA is the RAIR technique with the region-aware rules of MSP performed only at the VA stage, whereas in RAIR_VA+SA, the rules of MSP are enforced at both VA and SA stages. It can be seen that as p increases, all APLs increase due to two reasons: 1) more traffic of App 0 becomes inter-regional, thereby increasing the average hop count and the packet propagation delay of App 0; 2) more contention takes place between App 0 and App 1, increasing the contention delay of both applications.

Compared with RO_RR, RAIR techniques with MSP can reduce the APL of App 0 significantly while incurring little increase in the APL of App 1. This is because MSP prioritizes the low intensity inter-region traffic of App 0 over the high intensity intra-region traffic of App 1, so that the contention that App 0 experiences can be greatly reduced while the contention that App 1 experiences is not

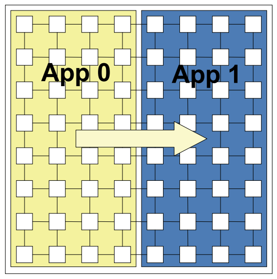


Figure 8: Two applications with varying percentage of inter-region traffic.

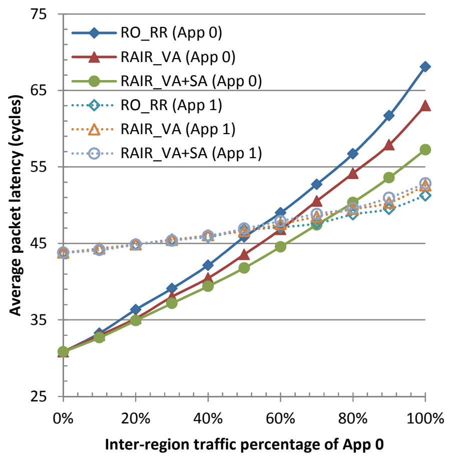


Figure 9: Impact of multi-stage prioritization.

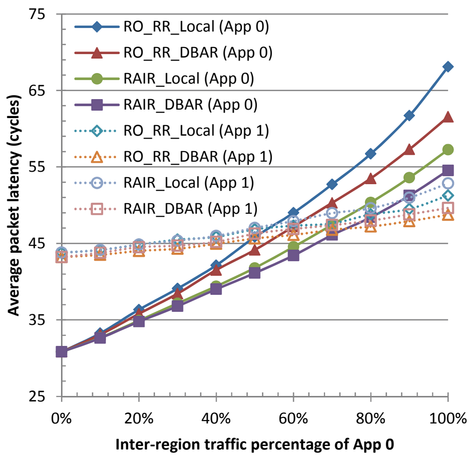


Figure 10: Impact of routing algorithm.

affected much. This effect becomes more evident as *p* increases. For example, when p is 100%, RAIR_VA+SA reduces APL by 18.9% for App 0 with less than 3% increase in APL for App 1. In addition, RAIR_VA+SA is more effective than RAIR_VA across the range of *p*, illustrating the necessity of enforcing prioritization in multiple arbitration steps.

C. *Impact of Routing Algorithm*

The above evaluation adopts a typical adaptive routing algorithm that uses the information available at the local router (e.g., # of free VCs). To demonstrate the ability of RAIR being compatible with other routing algorithms, we evaluate RAIR with an enhanced adaptive routing algorithm, DBAR [16], that leverages both local and non-local information to improve load balance.

Figure 10 presents the average packet latency of RO_RR and RAIR with local adaptive routing and with DBAR under the same two-application scenario. As can be seen, by using DBAR, RAIR_DBAR can reduce the APL of both App 0 and App 1 compared to RAIR_Local because of the better load balance. Note that the APL of App 1 in RAIR_DBAR is even lower than that of RO_RR_Local, indicating that RAIR can well utilize advanced adaptive routing algorithms to restore its slowdown in intra-region traffic of App 1. For example, when *p* is 100%, RAIR_DBAR avoids any latency degradation compared to RO_RR_Local and reduces APL by 24.8% and 3.3% for App 0 and App 1, respectively. Figure 10 also plots the APL of using DBAR alone on top of RO_RR. Compared with RO_RR_DBAR, RAIR_DBAR improves the APL of App 0 by 12.8% with only 1.8% degradation in the APL of App 1. This illustrates that, while an adaptive routing algorithm can reap additional benefits from better route selection, the largest performance improvement comes from the contention reduction offered by the RAIR technique.

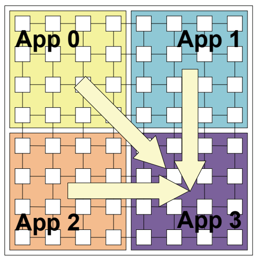
D. *Impact of Dynamic Priority Adaptation*

To validate the need for DPA and examine its effectiveness in utilizing load heterogeneity among regions, we consider two contrasting scenarios. As depicted in Figure 11(a) and (b), both scenarios consist of four applications, in which App 0 ~ App 2 have low loads and App 3 has high load. In (a), 30% of the traffic of App 0 ~ App 2 are inter-regional and towards App 3, whereas all traffic of App 3 is intra-regional. In (b), all traffic of App 0 ~ App 2 are intra-regional, whereas 30% of App 3's traffic is inter-regional and randomly towards other applications.

Figure 12(a) and (b) shows the reduction of average packet latency of each application for different schemes, corresponding to Figure 11(a) and (b). RAIR_NativeH (alternatively, RAIR_ForeignH) denotes the RAIR technique without DPA setting native traffic (alternatively, foreign traffic) to higher priority for all regions at all times. It can be seen that, for scenario (a), RAIR_ForeignH has lower average packet latency than RAIR_NativeH. This is because, by prioritizing the low intensity foreign traffic from App 0 ~ App 2 over high intensity native traffic of App 3 in region 3, the APLs of App 0 ~ App 2 can be reduced substantially with little performance degradation of App 3. However, for scenario (b), RAIR_NativeH is actually better as most benefit comes from prioritizing the low-intensity native traffic of App 0 ~ App 2 over high-intensity foreign traffic from App 3. Thus, neither RAIR_NativeH nor RAIR_ForeignH performs well for both cases, so dynamic priority adaptation is indispensable. The fifth bar of each series shows the reduction in APL for RAIR with DPA. Overall, RAIR_DPA reduces APL by 12.8% and 12.2% for case (a) and (b), respectively. Note that there could be a slight improvement of RAIR_DPA over the better of RAIR_NativeH and RAIR_ForeignH, as RAIR_DPA also dynamically adjusts the relative priority between native and foreign traffic among the three low load applications (i.e., App 0 ~ App 2). However, this difference is small due to the light contention among those applications.

E. *Effects of RAIR on Synthetic RNoC*

In this subsection, we evaluate the proposed RAIR technique as a whole and compare it against other schemes in a generic RNoC environment consisting of six concur-


(a)

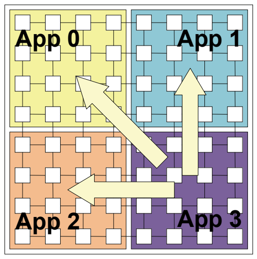
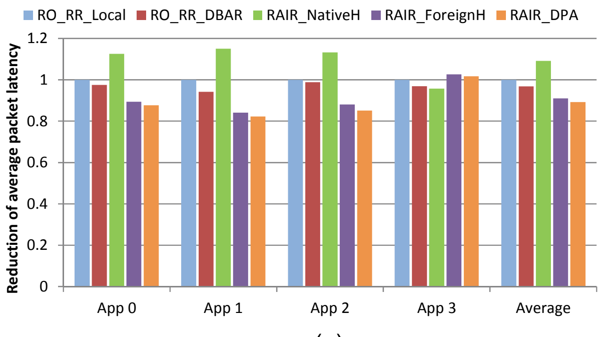

(b)

Figure 11: Two contrasting scenarios to evaluate DPA.


(a)

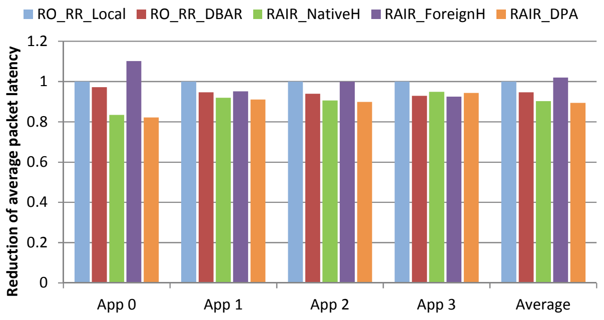

(b)

Figure 12: Impact of dynamic priority adaptation.

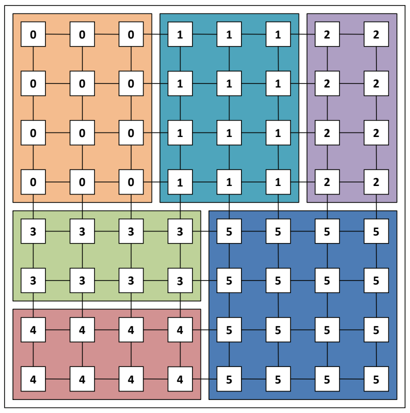


Figure 13: Six-application scenario with various global traffic patterns.

rently running applications with differentiated load rates. As shown in Figure 13, App 0, 2, 3 and 4 have low to medium loads (10% to 30% of their corresponding saturation loads), and App 1 and 5 have high load (90% of the saturation loads). Each application generates three types of synthetic traffic: 75% intra-region uniform random traffic, 20% inter-region global traffic with various traffic patterns (explained shortly), and 5% traffic to and from the 4 corner nodes to mimic memory controller traffic.

Four interference reduction schemes are compared. RO_RR is a region-oblivious technique based on round-robin. RO_Rank is an optimized version of STC – a region-oblivious but application-aware prioritization technique. This optimized STC is assumed to be able to always find the optimal application rankings during a given interval based on load intensity. RA_DBAR is a region-aware technique built on DBAR (we choose DBAR, as LBDR cannot allow any global traffic and Kilo-NoC relies on MECS in addition to other restrictions, which makes DBAR the least restrictive region-aware technique available so far). Finally, RA_RAIR is the proposed region-aware interference reduction technique.

Figure 14 shows the reduction of average packet latency compared to RO_RR for different techniques. The synthetic pattern for the global traffic component is uniform random for the moment. On average, RA_DBAR reduces average packet latency by 3.4%. This is mainly because, although RA_DBAR recognizes the regional layout, it reduces interference only when packets are in their originating regions. Therefore, in this generic RNoC setting, it cannot reduce interference effectively for global traffic that traverses unrestrictedly. In contrast, although being region-oblivious, RO_Rank actually performs better than RA_DBAR. It makes a trade-off by prioritizing low to medium load applications over high load applications and reduces average packet latency by 5.8%. However, it does not distinguish the regional and global traffic across regions and is also subjected to batching to avoid starvation, which limits the maximum achievable latency reduction.

The best performance is achieved by RA_RAIR because of its awareness of regional behaviors. In RA_RAIR, the foreign traffic of App 1 and App 5 can be prioritized over the native traffic of other applications when DPA determines that the global traffic has higher relative criticality for that region. As a result, the improvement of APL for App 1 and App 5 is only 1.3% less than RA_DBAR while the improvement in APL for App 0, 2, 3, 4 is 12.4% beyond that of RA_DBAR. Compared with RO_RR, RA_RAIR reduces APL by 10.1% when averaged over all applications.

F. Effects on Different Traffic Patterns

To demonstrate that RAIR removes the restrictions on traffic patterns that occur in other restricted techniques, Figure 15 shows the average reduction in APL for different synthetic global traffic patterns[3], with other configurations the same as the previous subsection. As can be seen, RA_RAIR achieves an average APL reduction of 13.4% over all traffic patterns compared to RO_RR, indicating that RAIR does not place any implicit restrictions on the global

[3] Due to space limitations, only the average value is shown. The relative trend of individual applications is similar to Figure 14.

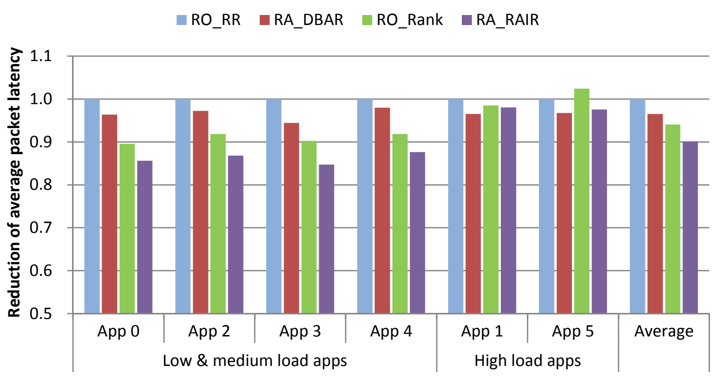


Figure 14: Average packet latency comparison of different techniques under uniform random global traffic.

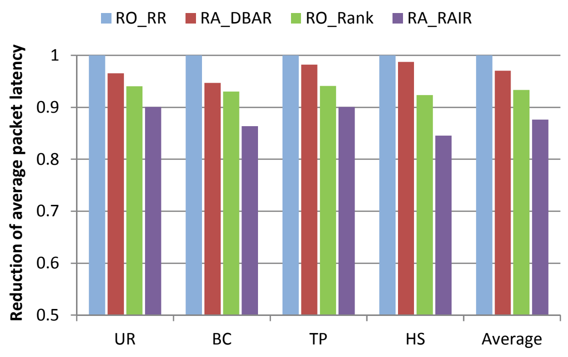


Figure 15: Reduction of average packet latency under different global traffic patterns.

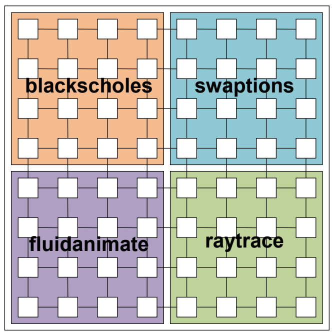


Figure 16: PARSEC simulation setup.

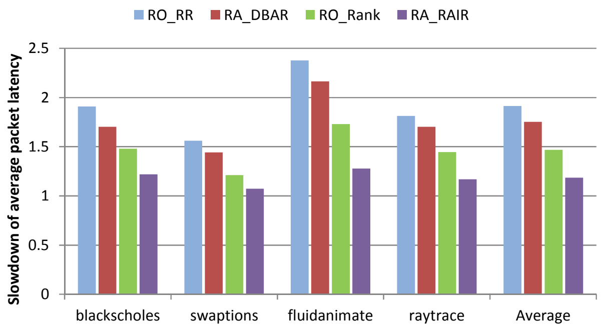


Figure 17: Average packet latency slowdown on PARSEC workloads with adversarial traffic.

traffic and can reduce interference effectively for different traffic patterns.

G. *Effects on Applications*

We next examine an important capability of RAIR to protect normal applications from adversarial traffic. As depicted in Figure 16, four PARSEC applications are running concurrently on the many-core chip. We model malicious traffic (e.g., an elaborated attack, or simply an OS bug) by adding uniform chip-wide global traffic with a load rate of 0.4 flits/cycle/node. Figure 17 shows the slowdown of average packet latency that is experienced by each application when different techniques are used. As can be seen, RO_RR performs worst with an average slowdown of 1.92 relative to the no adversarial traffic case. RA_DBAR reduces the slowdown to 1.75 through limited region -awareness. For RO_Rank, we assume that this STC-based RO_Rank can optimally rank applications, so all packets from the adversarial traffic have the lowest priority. However, as all packets are still subject to batching and RO_Rank does not allow the global traffic of normal low ranking applications to be prioritized over the regional traffic of high ranking applications, RO_Rank only reduces the slowdown to 1.47, on average. Finally, RA_RAIR can identify the adversarial traffic as foreign traffic to every region and assign it a lower priority than the native traffic through dynamic priority adaptation, thereby accelerating packets of the native traffic. The average slowdown is reduced to 1.18 for RA_RAIR, which is 38%, 32% and 19% better than RO_RR, RA_DBAR and RO_Rank, respectively.

VI. Discussion

Number of Regional and Global VCs. The impact of the relative quantity between regional and global VCs mainly depends on the traffic patterns. When there are more regional VCs than global VCs, native traffic will have a larger chance of getting high priority so that if foreign traffic has lower load, it cannot be effectively accelerated over native traffic. Similarly, when there are more global VCs than regional VCs, in the case of foreign traffic having much higher load, the native traffic needs to wait for a long delay before successfully acquiring high priority. Therefore, to support generic traffic patterns and simplify the implementation in practice, the number of regional VCs and global VCs are assumed to be configured roughly the same in RAIR.

Scalability and Overhead. We examine scalability in two dimensions: number of cores and number of regions. First, the DPA logic can be implemented with a couple of registers and comparators such that overhead is small and remains constant per router regardless of the NoC size. Also, unlike

STC, RAIR does not require any central control logic for batching or determining application ranking. Therefore, the number of cores can be easily scaled up without incurring much overhead in RAIR. Second, each router in RAIR maintains only two-flow status instead of per-region or per-application status, so there is no additional overhead with increased number of regions. Therefore, the number of regions can be as much as the number of cores on a chip. From the above two aspects, we conclude that RAIR has good scalability.

Relation to Quality-of-Service. Interference reduction is an important component of QoS, but QoS typically requires more than that. As the name implies, QoS provides applications with equal or differentiated service guarantees in addition to interference reduction. For example, it is able to enforce the pre-determined bandwidth allocation set by the OS, or provide end-to-end delay guarantees. Therefore, QoS has broader objectives and is also more complicated to implement than interference reduction alone (e.g., QoS polices typically need to record and update per-flow information). Due to these reasons, this paper compares RAIR with other interference reduction techniques rather than QoS mechanisms (such as [10, 11]). It is possible, however, to integrate RAIR with prior QoS mechanisms to further improve service quality, which can be investigated in the future.

VII. Conclusion

Many-core systems have enabled multiple applications to run concurrently in a chip. In the meantime, interference among traffic from different applications arises due to the shared nature of on-chip networks. To reduce interference effectively, traffic characteristics exhibited in the NoCs need to be exploited. In this paper, we present a case for interference reduction in regionalized NoCs, which results from a series of recent optimizations that leverage non-uniformity in many-core chips. We analyze the formation of RNoC using three representative examples and also identify four common regional behaviors. To address the limited region awareness in existing techniques, a new region-aware interference reduction technique (RAIR) is proposed. RAIR cleverly removes restrictions on inter-region traffic patterns and, therefore, is applicable to generic RNoCs. More importantly, RAIR can dynamically determine the relative criticality between native and foreign traffic, and take into account the regional traffic characteristics in multi-stage prioritization and starvation avoidance, thereby improving the effectiveness of interference reduction. Simulation results show considerable improvement in both synthetic traffic patterns and PARSEC benchmarks, as compared to other interference reduction techniques.

Acknowledgments

We sincerely thank the anonymous reviewers for their helpful comments and suggestions. This research was supported, in part, by the National Science Foundation (NSF), grant CCF-0946388.

References

[1] D. Abts and D. Weisser, "Age-based packet arbitration in large-radix k-ary n-cubes," in *ACM/IEEE Conference on Supercomputing*, 2007.

[2] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 33-42, 2009.

[3] E. Baydal, P. Lopez, and J. Duato, "A family of mechanisms for congestion control in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 16, pp. 772-784, 2005.

[4] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72-81, 2008.

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

[6] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 280-291, 2009.

[7] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol. 4, pp. 1320-31, 1993.

[8] J. Flich, S. Rodrigo, and J. Duato, "An efficient implementation of distributed routing algorithms for NoCs," in *2nd IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 87-96, 2008.

[9] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *14th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 203-214, 2008.

[10] B. Grot, S. W. Keckler, and O. Mutlu, "Preemptive virtual clock: A flexible, efficient, and cost-effective QOS scheme for networks-on-chip," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 268-279, 2009.

[11] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Kilo-NOC: a heterogeneous network-on-chip architecture for scalability and service guarantees," in *38th annual international symposium on Computer architecture (ISCA)*, pp. 401-412, 2011.

[12] J. Howard, S. Dighe, *et al.*, "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 108-109, Feb. 2010.

[13] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," *IEEE Transactions on Parallel and Distributed Systems (TPDS),* pp. 1028-1040, 2007.

[14] C. Jichuan and G. S. Sohi, "Cooperative caching for chip multiprocessors," in *33rd International Symposium on Computer Architecture (ISCA)*, pp. 264-275, 2006.

[15] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 211-222, 2002.

[16] S. Ma, N. E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *38th Annual International Symposium On Computer Architecture (ISCA)*, pp. 413-424, 2011.

[17] P. S. Magnusson, *et al.*, "Simics: A full system simulation platform," *IEEE Computer,* vol. 35, pp. 50-58+12, 2002.

[18] M. K. Martin, *et al.*, "Multifacet's general execution-driven multiprocessor simulator toolset," *ACM SIGARCH Computer Architecture News,* vol. 33, pp. 92-99, 2005.

[19] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *34th Annual International Symposium on Computer Architecture (ISCA)*, pp. 46-56, 2007.

[20] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 896-901, 2004.

[21] Tilera Corporation. http://www.tilera.com/products/processors

[22] F. Trivino, J. L. Sanchez, F. J. Alfaro, and J. Flich, "Virtualizing network-on-chip resources in chip-multiprocessors," *Microprocessors and Microsystems,* vol. 35, pp. 230-245, 2011.

[23] Z. Wenbiao, Z. Yan, and M. Zhigang, "An application specific NoC mapping for optimized delay," in *Intl. Conf. on Design and Test of Integrated Systems in Nanoscale Technology*, pp. 184-8, 2006.