Practical and Efficient Incremental Adaptive Routing for HyperX Networks*

Nic McDonald
Google
nicm@google.com

Mikhail Isaev
Georgia Institute of Technology
michael.v.isaev@gatech.edu

Adriana Flores
Nvidia
adrianaf@nvidia.com

Al Davis
Hewlett Packard Enterprise
ald@hpe.com

John Kim
Korea Advanced Institute of Science and Technology
jjk12@kaist.edu

ABSTRACT

In efforts to increase performance and reduce cost, modern low-diameter networks are designed for average case traffic and rely on non-minimal adaptive routing for network load-balancing when adversarial traffic patterns are encountered. Source adaptive routing is the predominant method for adaptive routing even though it presents many deficiencies related to making global decisions based solely on local information. In contrast, incremental adaptive routing, which performs an adaptive decision at every hop, is able to increase throughput and reduce latency by overcoming the deficiencies of source adaptive routing. We present two incremental adaptive routing algorithms for HyperX which are the first to be fully implementable in modern high-radix router architectures and interconnection network protocols. Using cycle accurate simulations of a 4,096 node network, our evaluation shows these algorithms are able to exceed the performance of prior work by as much as 4x with synthetic traffic and 25% with 27-point stencil traffic.

ACM Reference Format:
Nic McDonald, Mikhail Isaev, Adriana Flores, Al Davis, and John Kim. 2019. Practical and Efficient Incremental Adaptive Routing for HyperX Networks. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3295500.3356151

1 INTRODUCTION

For high-performance computing systems the interconnection network is the critical subsystem that makes the system a "supercomputer". In other realms, cloud and enterprise data centers and personal computers for example, one can find the same processors, memory, storage devices, and accelerators. It is the network that tightly couples these devices in such a way that it can be viewed and used as a single high-performance system. The network enables the programming runtimes (e.g., MPI, SHMEM, UPC, etc.) to be highly productive to programmers by supplying large amounts of bandwidth and low message latencies. Supercomputer architectures are seeing a large increase in size and complexity in an attempt to achieve exaflop performance levels [1]. As the number and complexity of nodes continues to increase to achieve exascale, the importance and critical role of the network also increases.

It has been shown that with high router I/O bandwidth, the pairing of high-radix routers [2, 3] with low-diameter networks [4–8] most efficiently uses the router's bandwidth, minimizes the cost of the network, and increases application performance. Ultra low-diameter topologies (e.g., Flattened Butterfly [4], Dragonfly [5], HyperX [6], SlimFly [7], Dragonfly+ [9] or Megafly [10]) optimize the bisection bandwidth of the network to support full throughput for uniformly distributed traffic patterns. While they are more cost efficient and have higher performance, these topologies require intelligent non-minimal adaptive routing to realize their full potential when the traffic is not uniformly distributed. Traditionally these topologies have been paired with a global adaptive routing [11] algorithm often with topology specific modifications, for example, Clos-AD [4] for the Flattened Butterfly and progressive adaptive routing [12] for the Dragonfly.

The Dragonfly is the current state-of-the-art topology for large-scale systems and was designed to exploit modern link-level technologies which allows it to reduce cabling costs by 50% when compared to the Fat Tree and 10% when compared to the HyperX. However, due to high signaling speeds, link-level technologies are rapidly changing. The reach of direct attach copper cables is shrinking. The cost delta between direct attached copper and active optical cables is growing even though $/bps cost is shrinking. New link-level technologies based on co-located, co-packaged, or fully integrated photonics allow the use of low-cost passive optical cables [13–17]. Under these conditions, the HyperX can be more cost-efficient compared to the Dragonfly (discussed in Section 3.1).

HyperX is a high-performance network topology designed for efficient packaging and scaling to exascale and beyond. Due to its symmetric all-to-all structure in every dimension, HyperX has significantly more path diversity that routing algorithms can exploit during the entire path of a packet. This structure supports incremental adaptive routing where routing algorithms can make many small adaptive decisions for a packet instead of one large adaptive decision at the source. As shown in this paper and in [6], this can yield significant performance benefits.

*This work was done while all authors were with Hewlett Packard Enterprise.

SC '19, November 17–22, 2019, Denver, CO, USA

ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356151

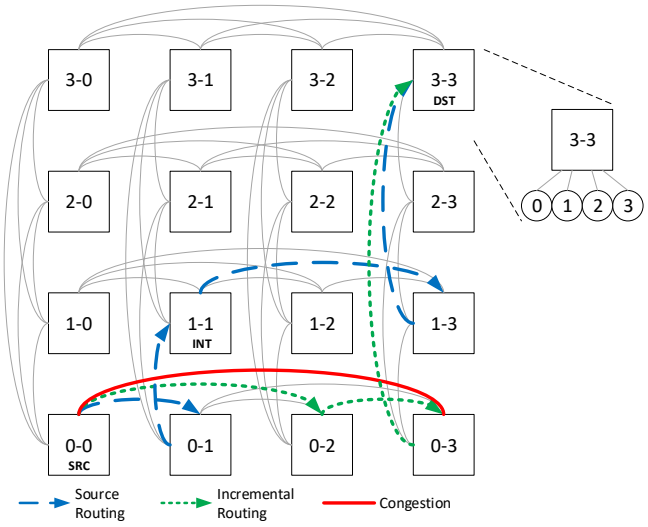


Figure 1: Paths taken by source and incremental adaptive routing algorithms after experiencing a congested channel from the source router.

Source adaptive routing, which determines entire paths at the source router, does not exploit the fully connected dimensions of HyperX as only source router's local information is used to make global routing decisions. As shown in Figure 1, incremental adaptive routing algorithms make best use of the structure of HyperX by dynamically moving around congestion only when needed and choosing minimal paths when they are uncongested. Unfortunately, previously proposed HyperX routing algorithms suffer from either poor performance due to the use of source adaptive routing[1] and/or are impractical to implement due to requirements for specialized router architectural features that do not exist in modern high-radix router architectures (see Sections 4 and 5.4).

In this work, we propose two new incremental adaptive routing algorithms for HyperX that are practical for implementation in modern high-radix large-scale networks that provide highest performance across various traffic patterns. We provide a comprehensive and detailed evaluation using cycle-accurate simulations of realistic router architectures across six synthetic traffic patterns. In addition, we implement a model to analyze performance for real application communication patterns derived from common HPC physics-based 27-point stencil discretization workloads. Our results show that our algorithms are able to exceed the performance of prior work by as much as 4× when remote congestion exists and provide up to 25% reduction in communication time for realistic stencil workloads.

2 BACKGROUND & RELATED WORK

2.1 Deadlock Avoidance

For networks that employ a form of active flow control (e.g., credit-based flow control, on/off flow control, etc.), routing algorithms must carefully utilize the resources of the network to avoid routing deadlock [18]. Deadlock free resource usage comes in two flavors: restricted routes and resource classes. Restricted routes is a methodology that uses only a subset of the total routes available to ensure deadlock freedom by never creating cyclic resource dependencies. Dimension order routing (DOR) is a common routing algorithm for integer lattice networks that employs restricted routes to achieve deadlock freedom. Resource classes employs multiple sets of resources such that moving through the network never creates cyclic dependencies. This is commonly implemented using virtual channel (VC) flow control [19, 20] where physical channels are virtually divided into multiple logical channels. Dateline routing on a ring topology employs resource classes by forcing packets to traverse into a higher ordered VC each time around the ring. This guarantees deadlock freedom by breaking the cyclic dependencies that structurally exist within a ring topology. Similarly, distance classes are a subset of resource classes where the resource class is incremented every hop. This guarantees acyclic resource usage and prevents deadlock. Previously this was considered to be an infeasible method of deadlock avoidance due to the large amount of resources (i.e., VCs) needed for high-diameter networks. With the advent of low-diameter networks, this is again being considered [7]. Many routing algorithms use both restricted routes and resource classes for deadlock avoidance, for example, dimension order routing on a Torus topology [21] and minimal routing on a Dragonfly topology [5].

2.2 Adaptive Routing

Decades ago when Mesh, Torus [21], and HyperCube [22] topologies [18] were prevalent, several key decisions were made that influenced how routing is performed. One implementation decision of adaptive routing is whether it is source based (i.e. adaptive decisions made at the source router) or incremental (e.g., adaptive decisions made at every hop). Minimal adaptive routing (i.e., Min-AD [23, 24]) is an incremental adaptive routing algorithm that attempts to find the best minimal path across the network by selecting the output port along one of the minimal paths that has the least congestion. On most topologies, all minimal routing algorithms, including O1Turn [25] and ROMM [26, 27], have significant throughput deficiencies when traffic is not uniformly distributed. For example, on the topology evaluated in this paper all minimal algorithms achieve 4x less worst case throughput compared to non-minimal algorithms. Thus, non-minimal adaptive routing is necessary to achieve high performance for high-radix topologies. The primary goal of non-minimal adaptive routing is to choose a minimal path when the traffic is load balanced and choose a non-minimal path when the traffic needs additional load balancing.

For high-diameter networks, source adaptive routing has been assumed for non-minimal adaptive routing with the use of Universal Global Adaptive Load-balancing (UGAL) [11] algorithm which is able to achieve 100% throughput for benign traffic and 50% throughput for worst case traffic (assuming the bisection capacity of the network is 50%). This is the highest theoretical performance in these two extreme cases. The routing algorithm switches between using minimal routing and Valiant's randomized routing [28]. UGAL is able to accomplish this using only two or four VCs on flat integer lattice networks (e.g., Mesh(2), Torus(4), HyperCube(2), Flattened Butterfly(2), HyperX(2)) by using two phases of dimension order routing (DOR) which only requires either one or two VCs per phase.

[1] At source 0-0 congestion on the minimal path 0-0 → 0-3 is sensed and the packet is sent non-minimally.

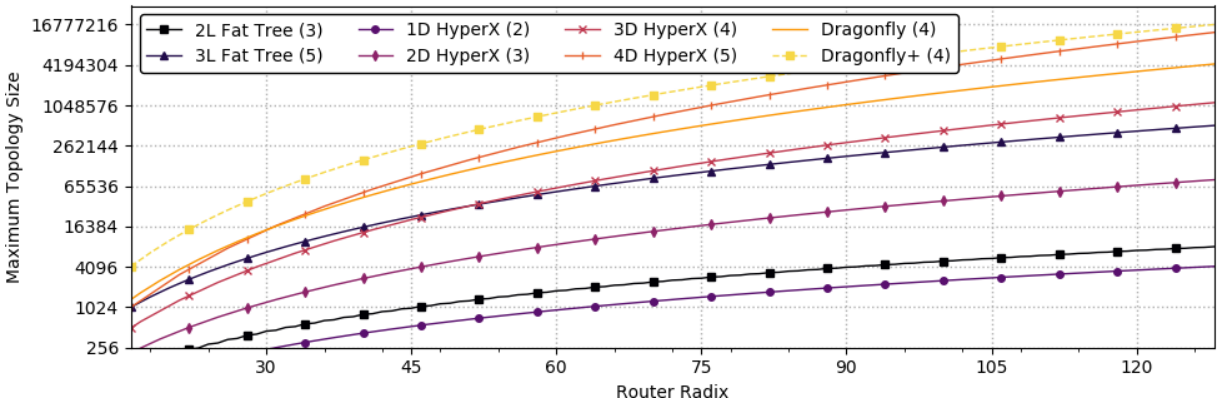


Figure 2: Scalability of various low-diameter networks.

On high-diameter networks, incremental adaptive routing proved hard to employ while also producing a stable network (i.e., a network where the throughput does not decrease significantly after the network becomes saturated [19]). Since incremental adaptive routing was hard to stabilize and resource expensive, source adaptive routing eventually became the prevalent scheme for non-minimal adaptive routing.

However, with low-diameter networks constructed of high-radix routers, source adaptive routing can be problematic. For the Dragonfly topology, attempting to load-balance the global channels from two router hops away produces significantly suboptimal throughput and latency performance [5]. Progressive adaptive routing (PAR) [12] was proposed where UGAL is re-evaluated within the source group. One particular extension [29] of PAR on the Dragonfly allows the routing algorithm to run a localized UGAL algorithm within every group the packet happens to traverse.

Both source and progressive adaptive routing have a significant deficiency as global routing decisions are made based only on local information. In some scenarios, this means they are unable to adapt to congestion not seen at the location where the adaptive decision is made. In other scenarios, a slight amount of congestion near the location where the adaptive decision is made will cause global load-balancing resulting in a 2× increase in both bisection bandwidth usage and packet latency. For these algorithms, once it is decided that load-balancing is needed, they fully load-balance (or route non-minimally) from that point forward, regardless of the remaining network conditions experienced. PAR and its variants bring light back to the original intent of incremental adaptive routing where it was noticed that making a large decision at the source can lead to suboptimal behavior. In an ideal routing algorithm, the adaptive decision could be determined at every router hop along the chosen path. This allows packets to opportunistically attempt to take the minimal path as long as it remains less congested. If a packet encounters a scenario where all minimal paths are congested, it can begin load balancing from that point forward and only as long as it is needed. After moving around congestion a packet should again be able to route minimally to its destination. This minimizes the amount of bandwidth overhead the packet utilizes and the latency overhead it experiences.

3 MOTIVATION

In this section we describe how changes in signaling technologies has resulted in the HyperX topology becoming a more cost-efficient alternative. We also discuss the challenges of irregular workloads and how *incremental* adaptive routing helps address some of these challenges.

3.1 Cost-Efficient HyperX Networks

HyperX [6] was proposed as a generalization to all flat integer lattice networks where dimensions are fully connected (e.g., HyperCube, Flattened Butterfly). Since HyperX is a superset of these topologies, the remainder of this paper will use the name HyperX as the methodologies presented herein apply to all HyperX configurations. The HyperX methodology is designed as a low-diameter network to fit with high-radix routers. One of the primary benefits of HyperX is that it can fit to any physical packaging scheme as each dimension can be individually augmented to fit within a physical packaging domain (e.g., a chassis, a rack, a row of racks, etc.). Figure 2 shows a scalability plot for a few key network topologies. The number next to each topology name represents the network diameter measured in number of router traversals along the longest minimal path. With a 64-port router, the HyperX topology is able to build 10,648 nodes in 2 dimensions, 78,608 nodes in 3 dimensions, and 463,736 nodes in 4 dimensions.

For large-scale computing systems, the cost of the network is often the major metric used for topology selection. As described by Kim et al. in [5], the HyperX (or Flattened Butterfly) yields a 10% higher cabling cost than Dragonfly with standard cabling techniques using direct attached copper cables (DACs) and active optical cables (AOCs). This is the major reason that the Dragonfly became the state-of-the-art HPC topology from 2008 until now. Figure 3 shows the costs of the Dragonfly relative to the HyperX for various system sizes and various cable technologies. For this we calculated the length of every cable in each of these networks based on common physical dimensions and placement. When calculating every cable the reproduction of the 2008 model, as described by Kim et al., shows that the Dragonfly has a 10% lower cost than the HyperX at large scales.

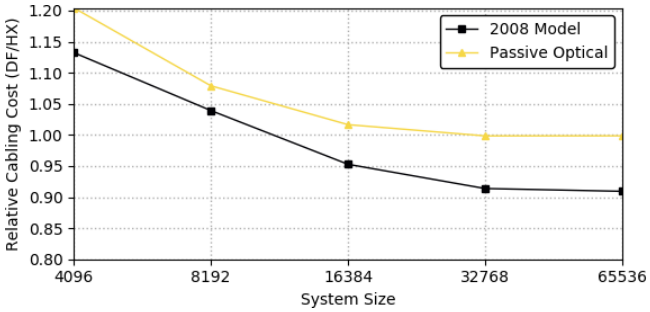


Figure 3: A cabling cost analysis comparing Dragonfly to HyperX with various system sizes and cable technologies.

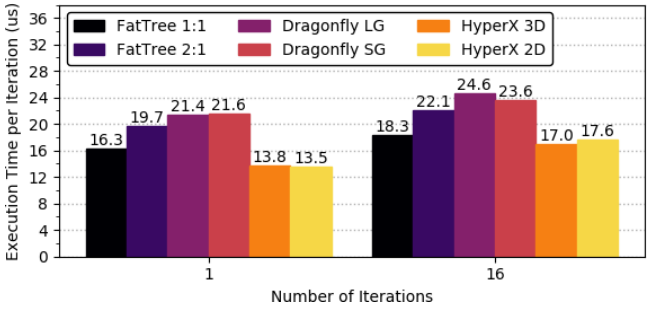


Figure 4: A topology performance (execution time) results using a 27-stencil application traffic.

Link-level technologies are currently on the brink of change. The major contribution to this change is the decreased distance of high-speed SerDes (serializer/deserializer) electrical signaling due to power consumption overheads and signal integrity limitations of high signaling rates. 2.5 GHz signaling reached 8 meters as described by Kim et al. 10 GHz reached 5 meters and 25 GHz now reaches 3 meters. It is expected that 50 GHz will reach 2 meters and 100 GHz will only reach 1 meter. Co-located, co-packaged, and fully integrated photonics are now being developed as a way to fully embrace photonics in the network and overcome the distance and power limitations of electrical signaling [13–17]. These technologies allow for the use of low-cost passive optical cables. Figure 3 also shows the cost comparison when using passive optical cables[2]. With this technology, the HyperX is always lower or equal in cost compared to the Dragonfly.

If the Dragonfly and HyperX are equal cost, then the selection between the two comes completely down to performance and packagibility. We performed a head-to-head analysis of application-level performance comparing the Fat Tree, Dragonfly, and HyperX using a 27-point stencil application model. This model represents many workloads used in physics-based HPC applications and is fully described in Section 6.2. Figure 4 outlines these results which show that HyperX has higher overall performance than the Fat Tree and Dragonfly. In these results the HyperX yields 25-38% reduction in communication time. The performance increase of the HyperX comes from its lower latency during collectives and its higher adaptive routing capability during the halo exchanges which yields higher achievable throughput. With the HyperX's better or equal cost and higher performance, we focus our attention on developing the highest performing implementable adaptive routing algorithms for HyperX.

HyperX is an good candidate topology for implementation of incremental adaptive routing because its symmetric multi-dimensional fully connected structure creates an environment where each dimension can be load-balanced individually. In contrast, the UGAL algorithm running on a HyperX decides whether to perform zero load-balancing or full load-balancing on all dimensions in a bipolar fashion. It does this because it makes only one adaptive decision solely at the source router. An incremental adaptive routing scheme can load-balance the dimensions individually based on the conditions of each particular dimensional plane. While HyperX is not as scalable as some other topologies, its primary benefit is its ability to support *fine-grained* incremental adaptive routing. An incremental adaptive routing algorithm that is considered fine-grained is one that always allows packets to divert from the minimal path while only increasing the total path length by one hop. Unlike source adaptive routing which forces a $2\times$ increase in bandwidth consumption and latency, a HyperX network with N dimensions can allow non-minimal routes with as little as $\frac{1}{N}$ bandwidth and latency overhead in the best case and still only $2\times$ overhead in the worst case.

[2]Based on 64-port routers and confidential quotes gathered from multiple vendors for various passive optical cable technologies.

3.2 Irregular Workloads

Most of the prior routing algorithm work has applied synthetic global traffic patterns where the injection process of all endpoints is acting the same. However, nearly every high performance computing system in use today provides a platform in which multiple applications and tenants run simultaneously. Each job that runs on a system has a different size and duration. This creates many scenarios where localized congestion occurs but is not wide-spread enough to be considered global congestion. For example, a small job might only consume a few 10s of nodes but have very high bandwidth requirements between its nodes. A very large job might be running at the same time and some of its traffic will need to cross the area in which the small job resides.

Source adaptive routing would not be able to detect this distant congestion and would send packets along the minimal path only to run straight into the small job's localized congestion causing unnecessary high packet latencies for both jobs. Because the algorithm only adapts at the source router, it can not do anything about the congestion that it discovers and the packets will incur increased latency. Eventually back pressure will propagate back to the source router. The algorithm would then see the congestion and decide to take a global non-minimal route to avoid the congestion. This non-minimal route will attempt to load balance the entire network which increases the bandwidth consumption of those packets by $2\times$.

This behavior exemplifies the bipolar nature of many modern low-diameter networks that employ non-minimal adaptive routing. They achieve high throughput for load-balanced traffic but slight imbalances will drop throughput below 50%. In contrast, an incremental adaptive routing algorithm would send the packets along the minimal path until they reach the location of the small job's congestion. For each packet the algorithm could then find a non-minimal path around the congestion to get to the destination. If the congestion

exists within only a single dimension of the network, a fine-grained incremental adaptive routing algorithm would take a non-minimal path around the congestion and only increase the path length by one hop.

4 LIMITATIONS OF PRIOR WORK

4.1 Adaptive Clos (Clos-AD)

Flattened Butterfly topology [4] was proposed with the Adaptive Clos (Clos-AD) routing algorithm which is similar to the UGAL algorithm with three optimizations:

(1) The Clos-AD algorithm only chooses intermediate nodes based on the least common ancestor methodology – ensuring that the packet does not route away from a dimension that is already aligned.
(2) At the source router the Clos-AD algorithm observes the congestion state of all output ports according to the first optimization (i.e., unaligned output ports). Each output port is assigned a weight following the UGAL algorithm (i.e., $weight = congestion \times hopcount$) and the route with the least weight is selected. If the chosen output port corresponds to a non-minimal path, an intermediate node is randomly selected that would use that output port and Valiant's randomized routing is used to traverse to the intermediate node and subsequently to the destination node.
(3) The Clos-AD algorithm relies on the use of sequential allocation where each input of a router makes the routing decision sequentially.

Clos-AD's dependence on sequential allocation makes it impractical for high-radix router architectures since it requires excessive routing delay to sequentially traverse all input port routing engines. The result is either enormous underutilization of internal router datapaths or extremely low clock frequencies. Since this work focuses only on routing algorithms that are practically implementable in high-radix routers, we present results for the Clos-AD algorithm without sequential allocator.

4.2 Dimensionally Adaptive Load-balancing

Dimensionally Adaptive Load-balancing (DAL)[6] was proposed for HyperX topology. It uses "derouting" where a deroute is defined as a lateral move in a network that does not bring the packet closer to or further from the destination. In DAL, each packet tracks which dimensions it has derouted and only deroutes once per dimension. If a packet deroutes in all dimensions it is then forced to minimally route the rest of the path.

DAL's deadlock avoidance mechanism uses escape paths [30] which have major limitations for large-scale networks. Escape paths, as originally defined, require the use of atomic queue allocation where no two packets can share a queue at the same time. During queue allocation the allocator must ensure that the downstream queue is empty before allowing any packet to traverse towards the queue. This ensures that a deadlock is always able to be alleviated. For large-scale networks with long channel latencies, atomic queue allocation causes severe link underutilization because the upstream router is not notified that the queue is empty until the last credit propagates upstream.

Further work on escape paths [31] found that some specialized router architectures can be created that do not require atomic queue allocation. However, escape paths without atomic queue allocation can not be supported on modern high-radix router architectures (e.g., the fully buffered crossbar [2] and all hierarchical router architectures such as the hierarchical crossbar [2], the Cray YARC [3], and a network-within-a-network [32]). Escape paths are only feasible if the architecture is a single monolithic datapath design (e.g., a monolithic crossbar) and the architecture is purely input queued (e.g., credits for downstream space can be consumed at path selection time). However, high-radix router architectures all have complex datapaths where packets traverse an internal network to get to the output port chosen during the routing phase. Making a high-radix router architecture support escape paths is not fundamentally impossible, but it would require excessive resources and significant architectural complexity.

The only way to practically implement DAL in a high-radix router is to use atomic queue allocation; however, this reduces channel efficiency to one packet per VC per credit round-trip latency. The DAL evaluations [6] did not show negative impact since the network simulated had single cycle channel latencies. For the topology evaluated in this paper with realistic channel latencies and 8 VCs, the maximum achievable throughput is 8% for single flit packets and 68% for randomly sized packets between 1 and 16 flits[3]. Because of the limited performance of the DAL algorithm with realistic channel latency, we do not present results for DAL in the evaluation.

5 PRACTICAL INCREMENTAL ADAPTIVE ROUTING

In this section we present two new routing algorithms for HyperX. These algorithms implement fine-grained incremental adaptive routing that can be implemented on high-radix router architectures. In particular, incremental routing across the different dimensions are proposed but leverages deadlock avoidance that does not require escape paths, special modifications to packet formats, or router microarchitectural features.

5.1 Dimensionally-ordered Weighted Adaptive Routing (DimWAR)

Dimensionally-ordered Weighted Adaptive Routing (DimWAR) is a light-weight routing algorithm that performs fine-grained incremental adaptive routing by moving through the network in dimension order. It supports one deroute per dimension when needed. It utilizes both restricted routes and resource classes requiring only two resource classes (i.e., VCs) for deadlock avoidance. The DimWAR algorithm follows these steps:

(1) Configure the routing algorithm with 2 resource classes (e.g., 2 VCs). When taking a minimal hop use the first resource class (e.g., VC 0) and use the second resource class (e.g., VC 1) for deroute hops.
(2) At each router, assess all valid outputs with their corresponding remaining hop count and current detected congestion. Only paths of the next ordered dimension to be traversed are

[3]The maximum throughput when using atomic queue allocation is: $PktSize \times NumVcs \div CreditRoundTrip$

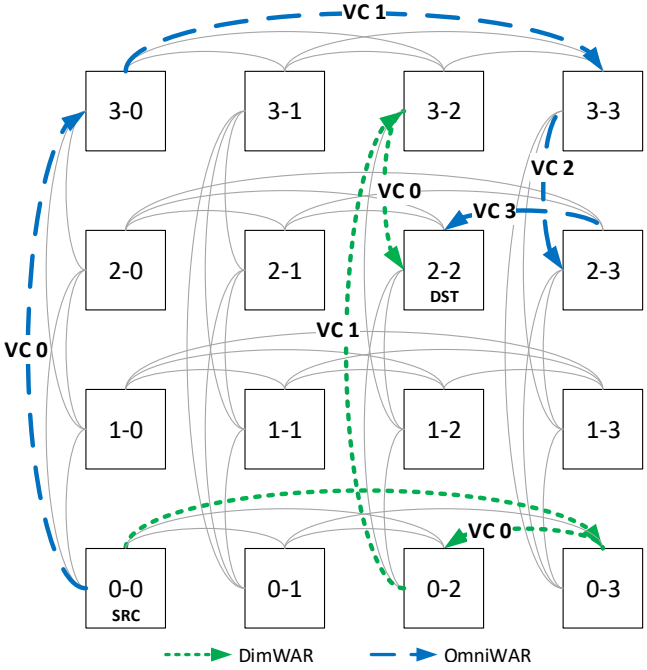


Figure 5: Virtual channel usage of DimWAR and OmniWAR.

valid. Deroute outputs are only valid in the current dimension and if the current resource class is the first class (e.g., VC 0).

(3) For each output, compute a weight (e.g., $congestion \times hopcount$) that estimates latency to the destination. Choose the output with the minimal weight (i.e., least estimated latency).

The deadlock avoidance scheme of DimWAR is shown by the green path in Figure 5 and is similar to dateline routing in a torus network. In a torus, dateline resource classes break the cyclic dependencies in the structure of the network and the result is a virtual mesh topology. A mesh network only needs a single VC when paired with dimension order routing. Because the dimensions are traversed in order, the two dateline resource classes get re-used for each dimension. This is the same scheme that is used in DimWAR except that no natural cyclic dependency exists in HyperX. Instead, DimWAR creates cyclic dependencies by allowing one deroute per dimension. It uses two resource classes to break the dependencies in each dimension, then it is able to reuse those resource classes on every subsequent dimension, which are traversed in order. Because of this DimWAR requires only 2 resource classes regardless of the number of dimensions of the network.

5.2 Omni-dimensional Weighted Adaptive Routing (OmniWAR)

Omni-dimensional Weighted Adaptive Routing (OmniWAR) is a heavy-weight routing algorithm that performs fine-grained incremental adaptive routing by moving through the network using any unaligned (not yet resolved) dimension at any time. Deroutes do not get tied to specific dimensions and dimensions do not need to be resolved completely before traversing another dimension. The algorithm is tunable to allow any number of deroutes across an entire path, thus, the number of VCs given to OmniWAR determines how much routing path diversity it is able to exploit. It utilizes distance classes for deadlock avoidance. The OmniWAR algorithm follows these steps:

(1) Configure the routing algorithm with $N+M$ distance classes, where N is the number of network dimensions, and M is the number of allowed deroutes. On the first internal network hop (i.e., router-to-router), use the first distance class (e.g., VC 0). For each subsequent hop, use the next ordered distance class (e.g., $VC_{out} = VC_{in} + 1$).

(2) At each router, determine if derouting is currently allowed for this packet by comparing the number of remaining minimal hops to the destination and the number of remaining distance classes available. If the difference is zero derouting is not allowed, otherwise any deroute can be taken.

(3) Assess all valid outputs with their corresponding remaining hop count and current detected congestion. Only paths of currently unaligned dimensions are valid. Deroute outputs are only valid when it has been determined that derouting is allowed per the prior step.

(4) For each output, compute a weight (e.g., $congestion \times hopcount$) that estimates latency to the destination. Choose the output with the minimal weight (i.e., least estimated latency).

The blue path in Figure 5 shows the deadlock avoidance scheme of OmniWAR. When configured with $2N$ VCs, OmniWAR can deroute the same number of times as the number of dimensions. However, OmniWAR can be tuned down to save VCs if the expected traffic does not create congestion in all dimensions. OmniWAR does not require that deroutes be taken once per dimension. Instead, there are M deroutes that can be taken at any time regardless of the history of the packet. While the example path shows two non-minimal hops then two minimal hops, many other combinations exist.

An optimization of OmniWAR is to restrict back-to-back deroutes on the same dimension. This is an easy modification to the routing algorithm as it is a simple function of the input port taken and the candidate output ports. This optimization still allows back-to-back deroutes, and more than one deroute per dimension, but it restricts the combination thereof.

5.3 Comparison

DimWAR has less path diversity than OmniWAR because it utilizes the restricted routes methodology of dimensional ordering to provide deadlock avoidance. It does this to reduce the number of VCs required for operation (i.e., 2 VCs regardless of the number of dimensions). For HyperX configurations with more than 2 dimensions there are particular traffic patterns, albeit very maliciously architected, where DimWAR will achieve much less throughput than OmniWAR because of this limitation (described in Section 6.1). For HyperX configurations with 1 or 2 dimensions, both algorithms should achieve 100% throughput for benign traffic (i.e., already load-balanced) and 50% throughput for worst case traffic (assuming the bisection capacity of the network is 50%). When configured with $2N$ VCs, OmniWAR theoretically achieves these throughputs regardless of the number of dimensions.

Figure 5 shows potential paths taken by DimWAR and OmniWAR. Because DimWAR moves in dimension order, after seeing

Table 1: Adaptive routing implementation comparison (RR: restricted routes, RC: resource classes, DC: distance classes, N: number of dimensions).

Algorithm	Dimension Ordered	Routing Style	VCs Required	Deadlock Handling	Architecture Requirements	Packet Contents
UGAL	yes	source	2	R.R. & R.C.	*none*	int. addr.
Clos-AD	yes	source	2	R.R. & R.C.	seq. alloc.	int. addr.
DAL	no	incremental	1+1e	escape paths	escape paths	N-bit field
DimWAR	yes	incremental	2	R.R. & R.C.	*none*	*none*
OmniWAR	no	incremental	N+M	R.R. & D.C.	*none*	*none*

congestion between routers 0-0 and 0-2, it chooses to take a deroute which it must do in the X-dimension. After it gets to router 0-3 it uses minimal routing to get to 0-2. In contrast, OmniWAR is able to traverse the network in any dimension order. Its two first minimal options from 0-0 would be 0-2 and 2-0, however seeing congestion on both of those it decides to deroute to 3-0. After another deroute from 3-0 to 3-3, it has run out of its ability to take further deroutes and may only choose minimal routes to the destination.

5.4 Implementation (Complexity) Analysis

In this section we compare the practicality of implementing adaptive routing algorithms using modern high-radix routers. Large-scale chip-to-chip networks have additional difficulty in implementing adaptive routing algorithms, when compared to networks-on-chip (NOCs) [33], because they often adhere to standards-based protocols which govern the structure and format of packets. These protocols, such as Ethernet [34], Fibre Channel [35], InfiniBand [36], and Gen-Z [37] are designed to be topology and routing algorithm agnostic. As such, the packet formats defined do not contain fields for the routing algorithm to use. As shown in Table 1, UGAL, Clos-AD, and DAL all require special fields to be placed and tracked inside each packet. This is one explanation for why large-scale network protocols have been so slow to adopt non-minimal adaptive routing and have heavily adopted deterministic routing algorithms where the route is a simple table lookup based on the destination address.

The architectural infeasibility of sequential allocator in Clos-AD (Section 4.1) and limitation of DAL from escape paths (Section 4.2) were discussed earlier. In comparison to the above, both DimWAR and OmniWAR do not require additional information stored within the packet as all routing information needed by these algorithms is encoded into the VC identifier. These algorithms do not require any special router architecture or architectural features. These algorithms are implementable in common table-based routing architectures that support virtual channel flow control.

For large-scale systems the routing algorithm does not significantly affect network power consumption as the power is dominated by constant power SerDes-based I/Os. Energy can be reduced by achieving higher link utilization (and thus, higher performance) and allowing applications to complete sooner. When using routing tables to implement routing algorithms, the silicon area overhead is proportional to the routing table size (both in depth and width). Non-deterministic (e.g., random oblivious and adaptive) routing algorithms require wider tables based on the number of options given to each entry. Advanced routing architectures (e.g., Cray Aries [38], Gen-Z [37]) have size optimized tables where the area and power overhead of the tables is negligible because the depth of the tables is greatly reduced. DimWAR and OmniWAR have no significant difference in power and area overhead when compared to other adaptive routing algorithms.

6 EXPERIMENTATION RESULTS

In this section we present experimentation results of cycle-accurate simulations using the SuperSim interconnection network simulator [39]. We simulate a 4,096 node 3D HyperX where each dimension is 8 wide (i.e., 8x8x8) and each router connects to 8 terminals. All routing algorithms are given 8 VCs and algorithms that require less than 8 VCs for deadlock avoidance use the spare VCs for head-of-line blocking reduction [18][4]. We simulate a combined input and output queued [40] router architecture with sufficient speedup to ensure the internal router datapath is not a bottleneck. The router crossbar latency is 50ns and we use age-based arbitration [20] for both virtual channel and crossbar scheduling. The network uses packet buffer flow control [18]. Router-to-router channels are 10 meters (50ns) and router-to-terminal channels are 1 meter (5 ns). We supply enough buffering in the routers to cover more than the credit round trip but not too much to prohibit stiff congestion back-pressure. The routing algorithms evaluated are described in Table 2.[5]

Table 2: Adaptive routing algorithms evaluated.

Name	Description
DOR	Dimension Order Routing [21]
VAL	Valiant's Randomized Routing [28]
UGAL	Universal Global Adaptive Load-balancing [11]
Clos-AD	Universal Global Adaptive Load-balancing optimized for HyperX [4]
DimWAR	Dimensionally-ordered Weighted Adaptive Routing (Section 5.1)
OmniWAR	Omni-dimensional Weighted Adaptive Routing (Section 5.2)

[4] We view this as the proper methodology for performance comparison since giving each routing algorithm its exact number of VCs yields an unfair advantage to the algorithms with higher VC requirements.

[5] Sequential allocation [4] can be used in any adaptive routing algorithm to improve transient response. Due to its infeasible implementation complexity and to provide fair comparison, sequential allocation was not used by any routing algorithm in the evaluation.

6.1 Synthetic Traffic

In this section we simulate steady state synthetic traffic patterns with packets randomly sized from 1 to 16 flits. Before any measurements are taken, the network is warmed up with traffic until packet latency stabilizes. Packet injection continues until all measurements have completed. If the network never reaches a state where latency stabilizes (i.e., stops growing), the network is declared saturated and measurements are not taken. This methodology is useful for understanding the raw performance of routing algorithms. Each traffic pattern pinpoints a particular strength or weakness for the topology and routing algorithm. The traffic patterns evaluated in this section are described in Table 3.

Table 3: Synthetic traffic patterns used for steady state performance analysis.

Name	Description
UR	Uniform Random
BC	Bit Complement
URB	Uniform Random Bisection - destination selected using BC in the targeted dimension and UR in all other dimensions. This traffic results in one dimension being non-load-balanced and all other dimensions are load-balanced. For example, URBy has UR in the X and Z dimensions and BC in the Y dimension.
S2	Swap 2 - destination selected like adversarial BC-like way but only along one dimension. Even numbered terminals use the X dimension and odd numbered terminals use the Y dimension. This presents non-load-balanced traffic but there is a lot of unused bandwidth in total.
DCR	Dimension Complement Reverse - destination selected across the furthest dimensional instance of the network. All terminals in each X dimension instance distribute their traffic across a complement Z dimension instance. Worst-case admissible traffic for 3D.

Figure 6 shows performance results for the synthetic traffic. Figure 6a shows UR traffic where all adaptive routing algorithms do a good job choosing the minimal routes. OmniWAR does slightly better than the rest because its underlying minimal algorithm is like minimal adaptive (MIN-AD) routing that traverses dimensions in any order increasing the minimal path diversity. The bit complement traffic shown in Figure 6b behaves as expected. The adaptive routing algorithms all take the minimal path until the bisections of each dimension become saturated at 12.5% injection at the point the adaptive algorithms sense the congestion and take non-minimal routes. The adaptive algorithms experience higher intermediate latency when choosing non-minimal routes due to longer traversed paths and overcoming queuing delays associated with the backpressure of the minimal routes. All adaptive algorithms achieve near 50% throughput, however, DimWAR and OmniWAR have lower latency and higher throughput than UGAL and Clos-AD.

Figure 6c shows the results for when the first dimension is unbalanced and the other two dimensions are balanced. Since the congestion is experienced at the source router, all the adaptive algorithms do well and achieve close to 50% injection. However, Figure 6d shows the case where the second dimension is unbalanced and the first and third dimensions are balanced. In this case the source adaptive algorithms (i.e., UGAL and Clos-AD) are unable to detect the congestion at the source router and do no better than DOR achieving only 12.5% throughput. In contrast, DimWAR and OmniWAR, being incremental algorithms, are able to move around the congestion in the second dimension and continue on to achieve full 50% throughput.

Figure 6e shows the results for the swap2 traffic pattern which leaves a lot of unused bandwidth in the network. UGAL, being topology agnostic, sees a little bit of congestion and decides to fully behave like VAL achieving only 50%. UGAL+, being tailored for HyperX, is able to use much of the unused bandwidth and achieve mostly full throughput. DimWAR and OmniWAR, being fully tailored to HyperX and being incremental algorithms, achieve 100% throughput.

Figure 6f shows the results for the worst admissible traffic pattern for a 3D HyperX. DOR does not even show up because it creates a 64:1 oversubscription of a single link causing it to only achieve the theoretical 1.56% ($\frac{1}{64}$) throughput. DimWAR does poorly because it is forced to traverse the network in dimension order. UGAL and UGAL+ do slightly better than DimWAR but still fail to achieve the theoretical 50% performance. This is because the bottleneck in this traffic pattern is not always witnessed at the source router. OmniWAR being able to exploit all the path diversity of HyperX is able to achieve the full theoretical 50% throughput.

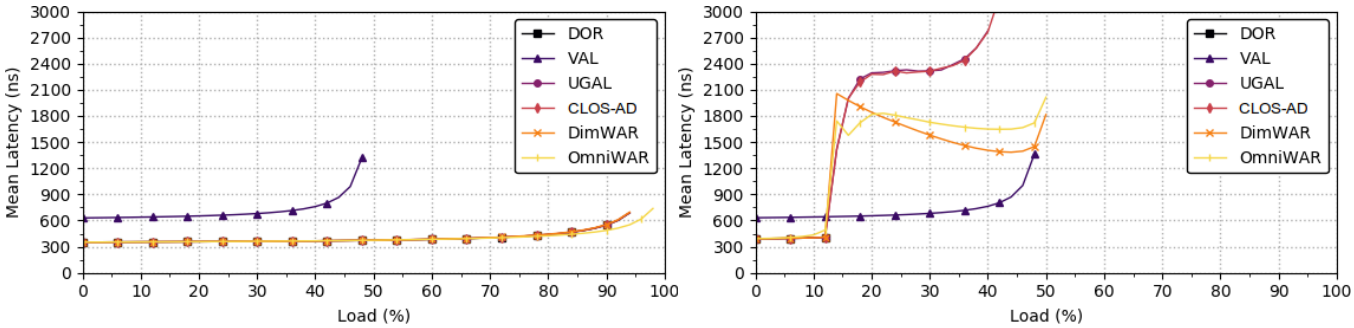
Figure 6g shows the throughput side-by-side for all traffic patterns. As shown, OmniWAR is always the top performer and DimWAR is nearly always a close second place. The only time where this is not the case is the DCR traffic pattern.

6.2 27-Point Stencil Traffic

While synthetic traffic patterns provide excellent stress tests for steady state performance, they have two downfalls. First, they do not stress a routing algorithm's ability to adapt to changing network conditions. Second, they are not representative of real application workloads. To overcome these two limitations, we use an application model of a stencil discretization in SuperSim. In a stencil discretization, a simulated 3D physical space is split into sub-cubes as shown in Figure 7a. Each sub-cube performs a portion of the simulation via coordinating with the others. The following pseudo-code shows the basic operation of each sub-cube:

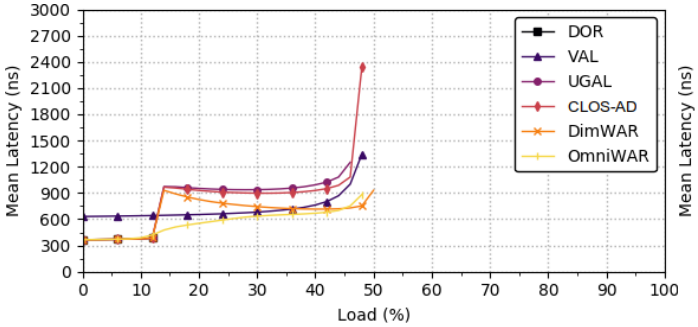
```
for (int i = 0; i < iters; i++) {
    compute();
    exchange();
    collective();
}
```

Since computation is not modeled in SuperSim, the "compute()" is a delay injected into the simulator's execution that skips over modeled time. The "exchange()" is a representation of the stencil discretization. It is driven by a user specified traffic matrix that specifies each node's exchange with other nodes. In our simulations,

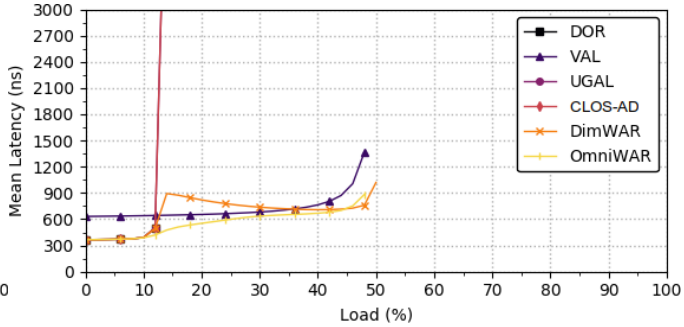


(a) Uniform random (UR)

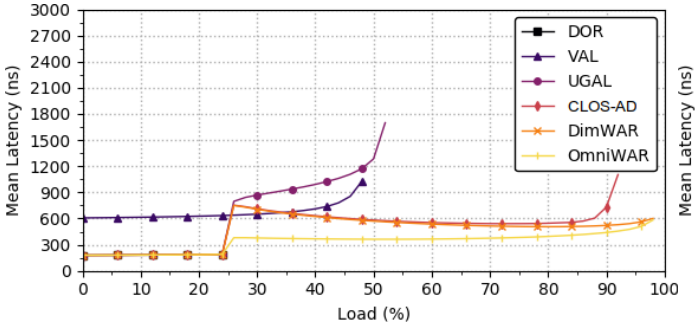
(b) Bit complement (BC)



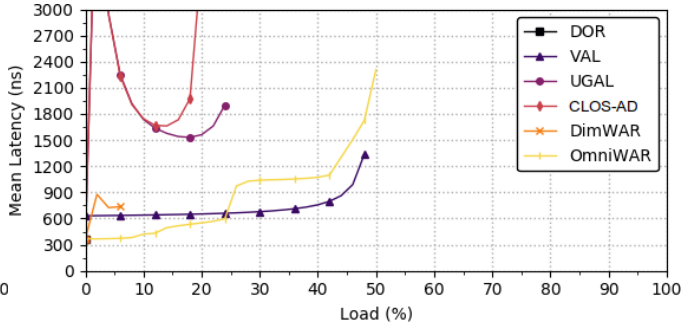
(c) Uniform random bisection: X-dimension (URBx)



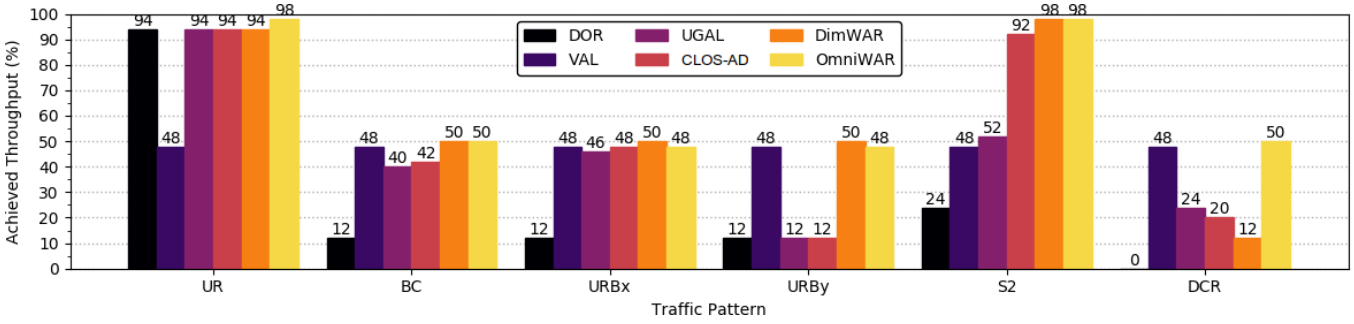
(d) Uniform random bisection: Y-dimension (URBy)



(e) Swap 2 (S2)

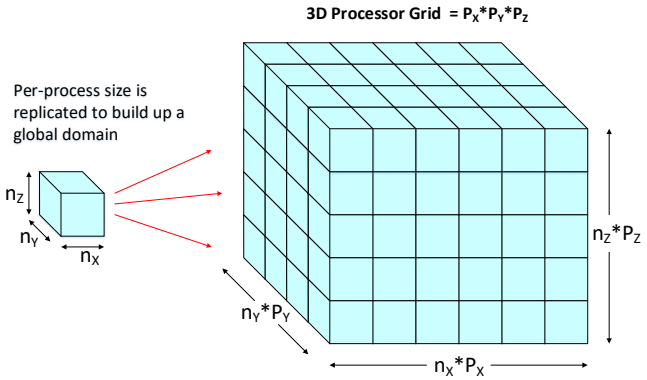


(f) Dimension complement reverse (DCR)

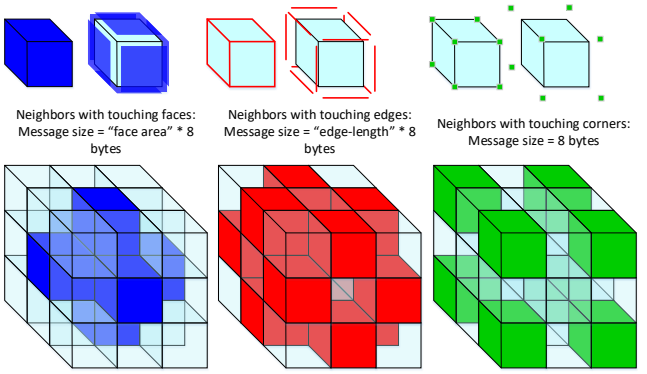


(g) Throughput comparison (higher is better)

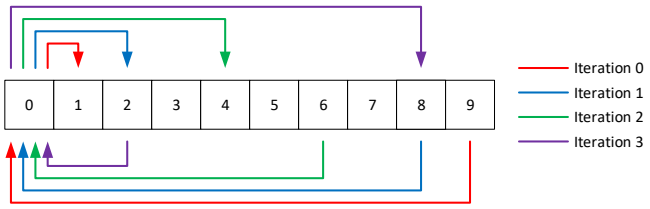
Figure 6: Simulated performance results for synthetic traffic patterns on a 4,096 node 3D HyperX comparing various routing algorithms. Simulations were performed with a 2% injection rate granularity. Plots (a) through (f) are load versus latency plots in which each line stops where the network becomes saturated. Plot (g) is a comparison chart comparing all traffic patterns and each routing algorithm's total achieved throughput.



(a) Discretization into sub-cubes.



(b) Stencil neighbors: 6 faces (left), 12 edges (middle), and 8 corners (right).



(c) Dissemination collective algorithm.

Figure 7: 27-point stencil application model.

we use a 27-point stencil discretization in which nodes have 26 total neighbors as shown in Figure 7b.

The "collective()" is a representation of a global synchronizing communication in which all nodes interchange information. The most common use of this is *MPI_AllReduce* in which each node contributes a value and all nodes complete the collective operation by understanding the reduced value of all values (e.g., minimum, maximum, etc.). The collective algorithm implemented in SuperSim uses a dissemination algorithm [41] as shown in Figure 7c. The dissemination algorithm is an iterative process where each node sends and receives $log_2 N$ messages. This algorithm is very similar to recursive doubling [42] except that it is topology agnostic. In the dissemination algorithm, each node sends and receives with ID+1 and ID-1, then ID+2 and ID-2, then ID+4 and ID-4, and so on.

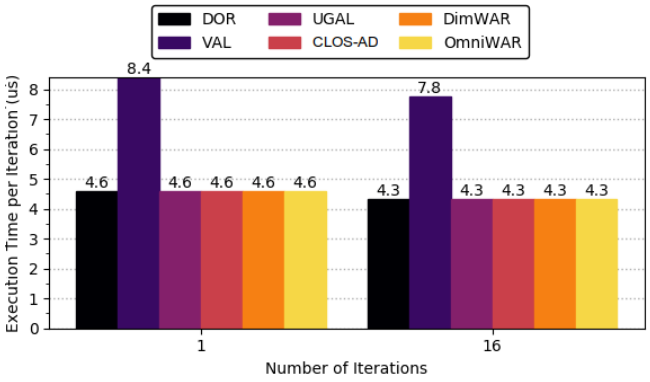
Our purpose for choosing this model of the 27-point stencil discretization is to cover the two previously identified flaws of steady state synthetic workloads. First, this stencil model is realistic for many physics-based HPC applications. Second, the traffic of this stencil application creates an extremely stressful workload for cost-optimized low-diameter networks (e.g., HyperX, Dragonfly, etc.) that rely on non-minimal adaptive routing. The exchange phase creates hot spots in the network that benefit from non-minimal routing. The collective phase on the other hand is highly latency sensitive which ideally uses only minimal routes. Because the stencil workload frequently switches between exchange and collective, adaptive routing algorithms need to quickly adapt to changing network conditions.

In our simulations we turn the compute time to zero, then test a single iteration as well as 16 iterations. With a single iteration, the communication is unencumbered by past traffic. This is representative of applications where the communication phases are significantly spread out by computation phases. With 16 iterations the communication phases are blended together as much as the collective's synchronization will allow. This traffic is representative of applications that overlap communication and computation to the point where the communication phases execute back-to-back. Processes are randomly assigned to nodes.

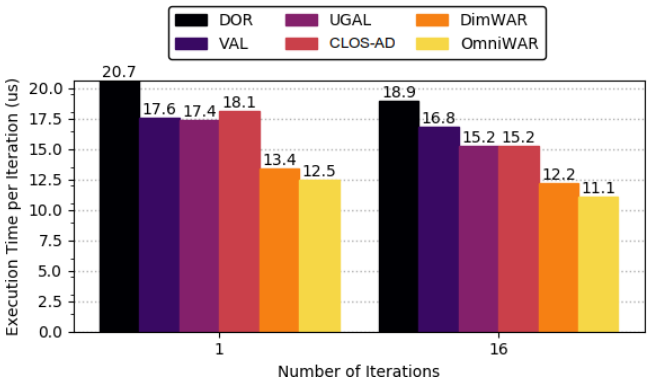
Figure 8 shows the results where each node sends an aggregate of 100kB across its 26 neighbors for each halo exchange. This simulation uses a random placement policy to assign stencil sub-cubes to network endpoints. Bars measure execution time, thus, smaller bars indicate higher performance. Figure 8a shows the results for only collectives. As shown, all routing algorithms except VAL have good performance. All adaptive algorithms consistently choose the minimal paths because the network remains unloaded.

In comparison, halo exchanges are highly bandwidth bound requiring frequent non-minimal routing, the various adaptive algorithms perform differently (Figure 8b). The incremental nature of both DimWAR and OmniWAR allows them to achieve the highest performance. Notice that DOR does the worst but VAL is second worst. From this we can deduce that neither minimal nor non-minimal routing is the ideal strategy. The adaptive algorithms are able to outperform these oblivious algorithms by using non-minimal routes only when required. Adaptive algorithms can hurt performance by being too eager for minimal routing or non-minimal routing. DimWAR and OmniWAR are able maintain excellent performance by striking a better balance.

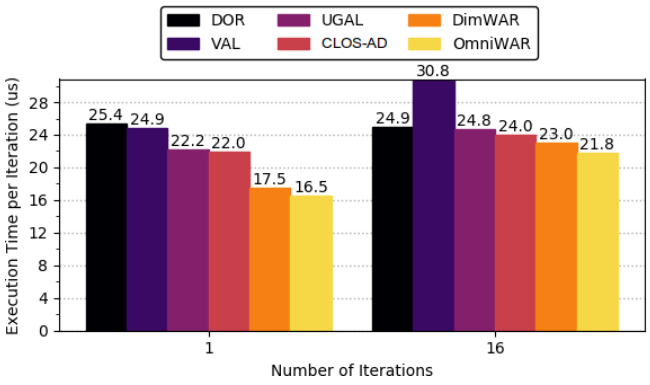
Figure 8c shows the results for the full application where halo exchanges and collectives are both used iteratively. Because of their excellent halo exchange performance, DimWAR and OmniWAR still perform the best with OmniWAR slightly beating DimWAR's performance. An adaptive routing algorithm that is slow to react to the change between halo exchange and collective will cause poor performance. All 4 adaptive routing algorithms have been tuned to react quickly to change. The performance increase in DimWAR and OmniWAR is caused by their better balancing of minimal and non-minimal routing strategies. 100kB is a small halo exchange size for many applications. We use this small size to create a highly stressful and dynamic workload. Applications that use halo exchange sizes much larger than this should expect the relative performance



(a) Collective performance



(b) Halo exchange performance



(c) Halo exchange and collectives performance

Figure 8: Simulation results for the 27-point stencil discretization model using 100kB per node per halo exchange. Lower is better.

of Figure 8b because larger halo exchanges make the collectives become less significant to total execution time of each iteration.

7 CONCLUSION

In this work we have shown that incremental adaptive routing is essential to exploit the potential performance of the HyperX topology. We have presented two new incremental adaptive routing algorithms (DimWAR and OmniWAR) which achieve up to 4× the throughput of state-of-the-art routing algorithms on synthetic traffic and reduce communication time by up to 25% for 27-point stencil discretization workloads. The light-weight DimWAR algorithm exploits most of the path diversity of HyperX and the heavy-weight OmniWAR algorithm exploits all of it allowing it to consistently achieve the highest performance across all workloads. Both DimWAR and OmniWAR are implementable in any network protocol that supports virtual channel flow control. Unlike all prior adaptive routing algorithms for HyperX, DimWAR and OmniWAR require no modification of the packet format and require no special router architectural features.

ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their comments. The authors were supported in part by the U.S. Department of Energy (DOE) under LLNS subcontract B621301. John Kim was supported in part by National Research Foundation of Korea (NRF) funded by MSIP (2015M3C4A7065647) and (2017R1A2B4011457).

REFERENCES

[1] U.S. Department of Enery (DOE), "Exascale computing project."
[2] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a High Radix Router," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 2005.
[3] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The Blackwidow High-Radix Clos Network," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 2006.
[4] J. Kim, W. J. Dally, and D. Abts, "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 2007.
[5] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 2008.
[6] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, ACM/IEEE, 2009.
[7] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, ACM/IEEE, 2014.
[8] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-Effective Diameter-Two Topologies: Analysis and Evaluation," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)*, ACM/IEEE, 2015.
[9] A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, "Dragonfly+: Low Cost Topology for Scaling Datacenters," in *International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, IEEE, 2017.
[10] M. Flajslik, E. Borch, and M. A. Parker, "Megafly: A topology for exascale systems," in *International Conference on High Performance Computing*, Springer, 2018.
[11] A. Singh, *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Stanford University, 2005.
[12] N. Jiang, J. Kim, and W. J. Dally, "Indirect Adaptive Routing on Large Scale Interconnection Networks," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 2009.
[13] Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, "Recent Advances in Optical Technologies for Data Centers: A Review," *Optica*, 2018.
[14] M. R. Tan, P. Rosenberg, W. V. Sorin, B. Wang, S. Mathai, G. Panotopoulos, and G. Rankin, "Universal Photonic Interconnect for Data Centers," *Journal of Lightwave Technology*, vol. 36, no. 2, pp. 175–180, 2018.
[15] D. A. Miller, "Attojoule Optoelectronics for Low-Energy Information Processing and Communications," *Journal of Lightwave Technology*, vol. 35, no. 3, pp. 346–396, 2017.
[16] L. Schares, J. A. Kash, F. E. Doany, C. L. Schow, C. Schuster, D. M. Kuchta, P. K. Pepeljugoski, J. M. Trewhella, C. W. Baks, R. A. John, *et al.*, "Terabus: Terabit/Second-Class Card-Level Optical Interconnect Technologies," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 12, no. 5, pp. 1032–1044, 2006.

[17] D. A. Miller, "Rationale and challenges for optical interconnects to electronic chips," *Proceedings of the IEEE*, vol. 88, no. 6, pp. 728–749, 2000.

[18] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[19] W. J. Dally and C. L. Seitz, "Interconnection Networks," *Transactions on Computers (TC)*, 1987.

[20] W. J. Dally, "Virtual-Channel Flow Control," *Transactions on Parallel and Distributed Systems (TPDS)*, 1992.

[21] W. J. Dally and C. L. Seitz, "The Torus Routing Chip," *Distributed computing*, vol. 1, no. 4, pp. 187–196, 1986.

[22] C. L. Seitz, "The Cosmic Cube," *Communications of the ACM*, vol. 28, no. 1, pp. 22–33, 1985.

[23] D. D. Chinn, T. Leighton, and M. Tompa, "Minimal Adaptive Routing on the Mesh with Bounded Queue Size," *Journal of Parallel and Distributed Computing*, vol. 34, no. 2, pp. 154–170, 1996.

[24] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," in *International Symposium on Computer Architecture (ISCA)*, IEEE, 1992.

[25] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," in *ACM SIGARCH Computer Architecture News*, IEEE Computer Society, 2005.

[26] T. Nesson and L. Johnsson, "ROMM routing: A Class of Efficient Minimal Routing Algorithms," in *International Workshop on Parallel Computer Routing and Communication*, Springer, 1994.

[27] T. Nesson and S. L. Johnsson, "ROMM Routing on Mesh and Torus Networks," in *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, ACM, 1995.

[28] L. G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM journal on computing*, vol. 11, no. 2, pp. 350–361, 1982.

[29] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, J. Labarta, C. Minkenberg, *et al.*, "On-The-Fly Adaptive Routing in High-Radix Hierarchical Networks," in *International Conference on Parallel Processing (ICPP)*, pp. 279–288, IEEE, 2012.

[30] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *Transactions on Parallel and Distributed Systems (TPDS)*, 1993.

[31] J. Duato and T. M. Pinkston, "A General Theory for Deadlock-Free Adaptive Routing Using a Mixed Set of Resources," *Transactions on Parallel and Distributed Systems (TPDS)*, 2001.

[32] J. H. Ahn, S. Choo, and J. Kim, "Network Within a Network Approach to Create a Scalable High-Radix Router Microarchitecture," in *International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2012.

[33] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Design Automation Conference (DAC)*, ACM/IEEE, 2001.

[34] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM*, vol. 19, no. 7, pp. 395–404, 1976.

[35] A. F. Benner, *Fibre Channel: Gigabit Communications and I/O for Computer Networks*. McGraw-Hill, Inc., 1995.

[36] G. F. Pfister, "An introduction to the infiniband architecture," *High Performance Mass Storage and Parallel I/O*, vol. 42, pp. 617–632, 2001.

[37] Gen-Z Consortium, "Gen-Z Specification 1.0."

[38] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC Series Network," *Cray Inc., White Paper WP-Aries01-1112*, 2012.

[39] N. McDonald, A. Flores, A. Davis, M. Isaev, J. Kim, and D. Gibson, "SuperSim: Extensible Flit-Level Simulation of Large-Scale Interconnection Networks," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2018.

[40] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input/Output-Queued Switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039, 1999.

[41] D. Hensgen, R. Finkel, and U. Manber, "Two algorithms for barrier synchronization," *International Journal of Parallel Programming*, 1988.

[42] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE transactions on computers*, 1973.