Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support

‡Natalie Enright Jerger, ⋆Li-Shiuan Peh, and ‡Mikko Lipasti
‡Electrical and Computer Engineering Department, University of Wisconsin-Madison
⋆Department of Electrical Engineering, Princeton University

Abstract

Current state-of-the-art on-chip networks provide efficiency, high throughput, and low latency for one-to-one (unicast) traffic. The presence of one-to-many (multicast) or one-to-all (broadcast) traffic can significantly degrade the performance of these designs, since they rely on multiple unicasts to provide one-to-many communication. This results in a burst of packets from a single source and is a very inefficient way of performing multicast and broadcast communication. This inefficiency is compounded by the proliferation of architectures and coherence protocols that require multicast and broadcast communication. In this paper, we characterize a wide array of on-chip communication scenarios that benefit from hardware multicast support. We propose Virtual Circuit Tree Multicasting (VCTM) and present a detailed multicast router design that improves network performance by up to 90% while reducing network activity (hence power) by up to 53%. Our VCTM router is flexible enough to improve interconnect performance for a broad spectrum of multicasting scenarios, and achieves these benefits with straightforward and inexpensive extensions to a state-of-the-art packet-switched router.

1. Introduction

Future many-core architectures with dozens to hundreds of nodes will require scalable and efficient on-chip communication solutions [15]. This has motivated substantial research into network-on-chip designs. Recent proposals [12], [14], [20], [21], [30] have successfully driven down interconnect delay to approach that of pure wire delay. However, one of the implicit assumptions in the evaluation of these proposals is that the vast majority of traffic is of a one-to-one (unicast) nature. Unfortunately, current router architectures are extremely inefficient at handling multicast and broadcast traffic.

In this work, we leverage several popular research innovations to demonstrate that the assumption of predominantly unicast traffic is not a valid one for on-chip networks and motivate the design of our multicast router, Virtual Circuit Tree Multicasting (VCTM). The inability of current router architectures to efficiently handle multicast communication can also have performance ramifications for unicast communications. Unicast communications occurring at the same time as a multicast communication are likely to be delayed by the burst of communication.

This research was supported in part by the National Science Foundation under grants CCR-0133437, CCF-0429854, CCF-0702272, CNS-0509402, the MARCO Gigascale Systems Research Center, an IBM PhD Fellowship, as well as grants and equipment donations from IBM and Intel. The authors would like to thank Niket Agarwal and Noel Eisley for their assistance with the traces used in this study. Additionally, we thank the anonymous reviewers for their thoughtful comments and constructive suggestions.

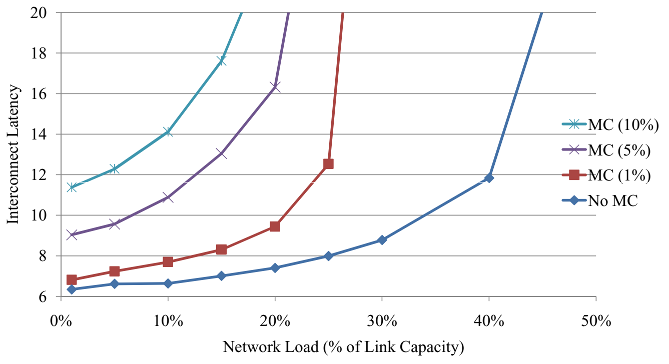


Figure 1. Performance of multicasts on packet-switched interconnect

Figure 1 shows the performance of a state-of-the-art packet-switched router in a 4x4 mesh in the presence of uniform random traffic. This router performs very well when all injected packets are intended for a single destination (No MC). When we start injecting packets in the same cycle, at the same source destined for multiple nodes, we see significant throughput degradation; MC 1% converts 1% of injected packets into a multicast destined for a random number of destinations ($<=15$). If 1% of injected packets are multicasts the saturation point drops from 40% capacity to 25% capacity. Saturation is defined to be when the latency is double the zero-load latency. The network saturates at 20% and 5% for 5% and 10% multicasts respectively. These multicast packets are broken down into multiple unicasts by the network interface controllers as the packet-switched routers are not designed to handle multiple destinations for one packet. More details about the packet-switched router under evaluation are discussed in Section 3.

Figure 2 shows a multicast originating from node X intended for nodes A, B, C and D. The network interface controller creates four identical copies of the message (1A-1D). With deterministic dimension-ordered routing, all 4 messages want to traverse the same link in the same cycle. Messages 1B and 1D successfully arbitrate for the output in subsequent cycles. However, the messages now intended for nodes A and C are blocked waiting for B and D to gain access to a busy channel. Messages intended for B and D will again compete for the same output port.

There are several problems in this scenario. The first issue is stalled messages 1A and 1C; this problem could be addressed with more virtual channels and buffers at each router. The second problem is competition for the same bandwidth; this can be alleviated with wider links. Both of these solutions are costly in terms of area and power and exhibit poor scalability. The

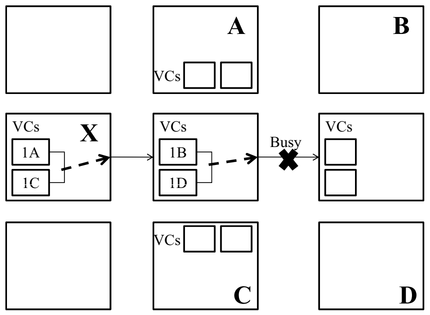


Figure 2. Multiple unicast approach to multicasting

bandwidth bottleneck along the links could also be alleviated through the use of an adaptive routing algorithm. However, this does not address the fundamental inefficiency of the simultaneous presence of multiple unnecessary messages in the network. In a network with moderate load or approaching saturation, these additional messages, even when spread out over disjoint paths, could drive up average network latency significantly. The use of multiple unicasts also causes additional congestion in the network interface controller (NIC); the competition for the injection port into the network can add significant delay to packets.

In addition to the performance inefficiencies, the multiple unicast approach to multicasting consumes additional power due to redundant packets traversing the network. Power has become a first-order design constraint for chip-multiprocessors. On-chip networks consume a significant fraction of total on-chip power [5], [42]; up to $\sim 30\%$ for Intel's 80-core teraflops network [15] and 36% for the RAW [39] on-chip network. Our VCTM router substantially reduces the power consumed by the network by eliminating the majority of these redundant packets (Section 5).

The design choices for the communication medium are tightly coupled with the communication characteristics of the application domain or the coherence protocol. When considering future many-core architectures it would be unwise to design an interconnect without thought to the type of communication that will be most prevalent. Designs requiring multicast communication motivate the need for an efficient on-chip multicast router design. However, the absence of multicast routers for on-chip networks will hinder the deployment of protocols that require this class of communication.

Several recent research proposals and industry trends lend themselves to a multicast routing architecture. In some cases, such as architectures utilizing operand networks (e.g. RAW [39], TRIPS [31], Wavescalar [37]), they perform well without multicasting but could be further enhanced with multicasting. Other designs such as broadcast coherence protocols experience prohibitive latency without on-chip multicast support and are often quickly dismissed as on-chip solutions. We outline these scenarios below and will demonstrate the performance benefits of multicasting in Section 5:

- Broadcast-based coherence protocols
  - Token Coherence: the TokenB protocol requires broadcasting of tokens that maintain ordering amongst requests [26].
  - Intel's next-generation QPI protocol: supports unordered broadcasting between subsets of nodes [17].
  - AMD Opteron Protocol: order is maintained by communicating requests to an ordering point (memory controller) and then broadcasting from the ordering point [10].
  - Uncorq [35]: unordered snoop delivery,
- Multicast-based coherence protocols such as Multicast Snooping [4] and Destination Set Prediction [28]
- Invalidation requests in Directory Protocols
  - Invalidation requests for widely shared data represent the primary need for multicasts.
- Operand delivery in scalar operand networks
  - Operands that are consumed by multiple instructions could be multicast through the network.
- New Innovations
  - New coherence protocols, prefetching, and global cache replacement policies are research areas that could benefit from the presence of on-chip hardware multicast support.

The above-mentioned multicasting scenarios will be explored in greater depth in Section 2.

Multicasting support has been well researched for off-chip networks, particularly for multistage interconnection networks [25], [33], [36], [41]. Communications across off-chip networks are able to benefit from multicasting; the same is true for on-chip network communication. Several of the design trade-offs made to realize off-chip multicasting are re-examined in this work. However, as power and area constraints for off- vs. on-chip routers differ substantially, prior off-chip multicast routers which require huge amounts of resources, such as large central shared buffers [36] and high-port-count switches [41], are not suitable for on-chip usage.

Virtual Circuit Tree Multicasting (VCTM) brings multicast functionality on chip without adding significant overhead to the router pipeline. We employ a *novel* tree construction method to ease routing complexity; tree construction occurs in parallel with message delivery so our design avoids the setup latency associated with circuit-switching. With a modest amount of storage, the Virtual Circuit Table can hold the routing information for a large number of simultaneously active multicast trees. Our low-overhead design realizes substantial network latency savings over current on-chip approaches (e.g. multiple unicasts).

In this work, we

1) Characterize traffic for a variety of architectures and motivate the need for hardware multicast support
2) Explore the deficiencies of packet-switched routers for mimicking multicast functionality
3) Discuss off-chip multicasting mechanisms, their characteristics that can be leveraged on-chip as well as downsides that make them unsuitable for on-chip
4) Design an innovative on-chip multicast router that reduces power and improves performance for a variety of multicasting scenarios.

2. Multicast Motivation

Several recent research proposals could leverage hardware multicast support to further enhance their performance. In this section, we will explore the opportunities within existing proposals for multicasting. Beyond these scenarios, hardware multicast support will enable new research directions that would previously have suffered from poor interconnect performance despite utilizing state-of-the-art on-chip network designs. For all scenarios, we assume a tiled architecture connected via a 4x4 (5x5 for TRIPs) 2D packet-switched mesh with 16-byte wide links (see Table 4).

2.1. Coherence Protocols

In general-purpose chip multiprocessors, the most natural source of multicast traffic will be messages generated by the coherence protocol. A wide variety of implemented and proposed coherence protocols will benefit from hardware multicast support.

2.1.1 Directory-based coherence. Directory-based protocols are often chosen in scalable designs due to the point-to-point nature of communication; however, they are not immune to one-to-many style communications. Directory protocols, such as the SGI-Origin Protocol [22], send out multiple invalidations from a single directory to nodes sharing a block; these invalidations could leverage hardware multicast support. While not necessarily on the critical path, these requests can be frequent and can waste power and hurt the performance of other network requests that are on the critical path.

Characterization of these invalidation messages in a full system simulation infrastructure [7] with a variety of commercial and scientific workloads [34], [40], [44] shows that invalidation messages have an average network latency of up to 2 times the overall average network latency. Table 1 shows the percentage of total requests that are invalidates.

2.1.2 Token Coherence. In broadcast-based protocols, ordering (for correctness) is often implicit through the use of a totally-ordered interconnect. To improve scalability, token coherence removes this implicit assumption and instead uses token counts to ensure proper ordering of coherence requests. A processor must hold at least 1 token to read a cache line and must hold all tokens to write to a cache line.

Broadcasting for these tokens can be a significant bottleneck. Originally intended as a chip-to-chip protocol with off-chip multicast support, the absence of multicast functionality on-chip hurts the performance of token coherence and reduces its attractiveness. Note that token coherence does not require all processors to respond to token requests, leading to fewer messages as compared to the Intel QPI and Opteron-like protocols.

Figure 3 shows the slowdown of the TokenB protocol when the assumption of hardware multicast support is removed. GEMS 2.1 [27] was used to generate this data for a 16-core system. This release of GEMS include's Princeton's Garnet [2], a detailed network simulator which models link contention and router microarchitectures. We used Garnet's Flexible model and modeled the router as a single-cycle pipeline with infinite buffering. Despite the unrealistically aggressive router model, substantial slow down is already observed due to NIC and link bandwidth bottlenecks.

2.1.3 Intel QPI Protocol. Intel's new Quickpath Interconnect [17] will support unordered broadcasting among nodes. As with Token Coherence, ordering in QPI has been decoupled from the interconnect. Broadcasting has the potential to deliver shared data faster than a directory protocol by avoiding the indirection through the directory. However, as the number of coherent nodes in the system grows, the cost of broadcasting in terms of delay, power and area becomes prohibitive. Hardware multicasting will lower the latency and power consumption associated with on-chip broadcasting.

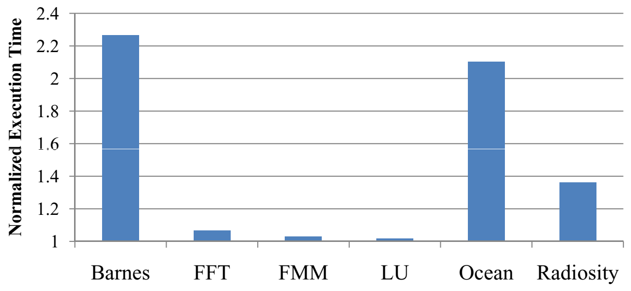


Figure 3. Token coherence slowdown in the absence of hardware multicast support

2.1.4 AMD Opteron (HT) Protocol. AMD's Opteron protocol [10] has been designed for maintaining coherence between chips in a traditional multiprocessor system. Coherence requests are sent to a central ordering point (memory controllers) and then broadcast to all nodes. The feasibility and potential for moving this style of protocol on-chip is directly tied to the performance provided by the interconnect, and can be improved with multicasting. Recent research proposals have compared themselves to an on-chip Opteron-style protocol [35].

2.1.5 Recent Coherence Innovations. Evaluating the feasibility of using the Opteron protocol on-chip also brings to light the issue of new coherence innovations that might be hampered or discarded due to the lack of on-chip multicast support. Virtual Hierarchies [29] highlight a key characteristic of future systems; specifically in many-core CMPs, maintaining global coherence among all nodes will be very rare. The prohibitive cost of global coherence will lead to a common case of maintaining coherence among a limited subset of nodes. Virtual Hierarchies propose two flavors of hierarchical coherence. The first is a two level directory protocol and the second is a first level local directory protocol with a backing broadcast mechanism for global coherence requests. Global broadcasts will likely be rare in this scenario but will experience lower latency and power by utilizing a multicast router.

An alternative protocol, one that broadcasts to a limited subset of nodes with a backing directory for global coherence would benefit substantially from our proposed multicast router. We envision a region-based broadcast mechanism, which leverages Coarse-Grain Coherence Tracking [8], originally designed to avoid unnecessary broadcasts. This structure can be extended to track nodes that are sharing within that region; a multicast can then be used to enforce coherence of lines within that region. On a miss to a shared region, the processor multicasts the coherence request to cores sharing that region. If there is no region information cached at the core, the miss request is sent directly to the second level directory. The directory then multicasts the request to the region sharers and sends the sharing list to the requestor. Broadcasting to this small subset of nodes will provide better performance than a global broadcast or an indirection through a directory.

Maintaining global coherence among all nodes will be prohibitive as many-core architectures scale to 100s of cores. Furthermore, certain classes of applications, such as server consolidation [13] will require only minimal global coherence as virtual machines keep much of the address space private to a subset of cores. With a region-based multicast protocol, only a limited subset of cores will need to be notified of coherence requests within a given memory region.

Without efficient multicasting, these types of otherwise promising coherence protocols become much less attractive. Considering the large number of cores that will be available moving forward, interconnect support for low latency one-to-many communication is critical. The tight coupling of on-chip resources mandates that interconnection network be designed with communication behavior in mind and that coherence protocols be designed with interconnect capabilities in mind. Providing hardware multicasting support will facilitate significant innovations in on-chip coherence.

The following two scenarios represent more specialized architectures; however, operand networks and non-uniform caches represent plausible and interesting solutions to the problems of growing wire delay and scalability. We consider them here due to their amenability to multicasting.

2.2. Operand Network Architectures and NUCA Caches

Architectures such as TRIPS [31], RAW [39] and Wavescalar [37] use operand networks to communicate register values between producer and consumer instructions. The result of an instruction is communicated to consumer tiles which then wake up and fire instructions that are waiting on the new data. If the result of one instruction is consumed by multiple subsequent instructions on different tiles, operand delivery could be expedited by a multicast router. Studies have shown that 35% of dynamic values generated by an application have 2 or more future uses [6]. The cost of on-chip communication can significantly impact compiler decisions in this style of architecture [19].

To mitigate increasing on-chip wire delays, non-uniform cache architectures (NUCA) have been proposed. Dynamic NUCA [18] uses a packet-switched network and is further optimized through the use of a multicast to quickly locate a block within a cache set. Recently, a multicast router has been proposed to speed up this search operation and improve performance [16].

2.3. Characterization

Our evaluation in Section 5 will focus on a subset of the above scenarios, namely directory coherence, TokenB, region-based coherence, the Opteron protocol and the TRIPs operand network.

Table 1 and Figures 4 and 5 highlight some of the different multicast characteristics among these scenarios. In Figure 1, we demonstrate that even a multicast rate of 1% is enough to cause significant throughput degradation. Table 1 shows that all the above scenarios exceed a multicast rate of 1% and will benefit from VCTM, our proposed hardware support for multicasting.

Despite the large potential number of unique multicasting combinations, Figure 4 shows that there is significant reuse of a small percentage of multicasts. Multicast reuse is defined as multicasts from the same source intended for the same destination set. At one extreme, Token Coherence which uses one-to-all communication, has very few distinct multicast combinations. Multicasting for invalidations from a directory shows the least reuse of destination sets for all the scenarios.

TABLE 1
PERCENTAGE OF NETWORK REQUESTS THAT CAN BE MULTICAST

Scenario	Percentage Multicast
Directory Protocols	5.1
Token Coherence	5.5
Region-Based Coherence	8.5
Operand Networks (TRIPS)	12.4
Opteron Protocol	3.1

Figure 5 shows the breakdown of the number of nodes in each multicast destination set. Multicasts in Token Coherence and Opteron include all possible destinations. At the other extreme, the majority of multicasts in TRIPS are to only 2 nodes. Invalidations from the directory in the SGI Origin protocol go to only 2 nodes on average as well.

The bottom line is that architectures and protocols that require multicast support have a variety of characteristics. For example, some protocols perform broadcasts to all nodes while others have relatively small destination sets. A robust multicast router design must be able to perform well under a wide variety of conditions.

3. Network Design

Virtual Circuit Tree Multicasting builds on existing router hardware in state-of-the-art networks, and augments it with a lookup table that performs multicast route calculations. To simplify route construction, multicast trees are built incrementally, by observing the initial set of multiple unicasts and storing the routing information in the lookup table. This approach avoids the overhead of encoding multicast destination sets in the initial setup message, and enables an efficient tree-ID based approach for addressing multicast messages once the tree has been set up. As an additional benefit, conventional unicast traffic is unaffected by these straightforward additions to the network router. Before we explain the details of our proposed VCTM router, we will discuss the highly optimized packet-switched router used as a baseline in all our experiments.

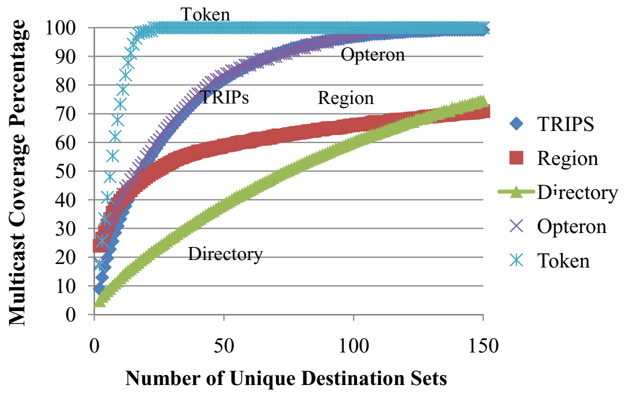


Figure 4. Cumulate distribution of unique multicast destination sets

3.1. Packet-Switched Router

Figure 6a depicts a 4-stage router pipeline. The first stage is the buffer write (BW); the routing computation (RC) occurs in the second stage. In the third stage, virtual channel allocation (VA) and switch allocation (SA) are performed. In the presence of low loads, speculation will be able to eliminate this stage. In stage 4, the flit traverses the switch (ST). Each pipeline stage takes one cycle followed by one cycle to do the link traversal (LT) to the next router.

Recent work [14], [21] uses lookahead signals or advanced bundles to shorten the pipeline to a single stage. Our baseline leverages lookahead signals to reduce the pipeline to 2 stages as depicted in Figure 6b. While the flit is traversing the switch, a lookahead signal is traveling to the next router to perform the routing computation. In the next cycle when the flit arrives, it will proceed directly to switch allocation; resulting in a 2 cycle pipeline (VA/SA + ST).

The vast majority of current network-on-chip proposals ignore the issue of multicast communication. There are a few exceptions which will be discussed further in Section 6. Proposals that might effectively leverage a multicast router either naively assume the existence of an on-chip multicast router or fail to model network contention (once contention is modeled, the need for hardware multicast support becomes abundantly clear).

State-of-the-art packet-switched routers can of course utilize multiple unicast messages to achieve multicast functionality. Decomposing a multicast into several unicasts can consume additional cycles and cause a bottleneck at the injection port as multiple messages try to access the network in the same cycle. In the baseline router, this injection bottleneck can add several cycles to the average network latency.

Many current router optimizations, such as speculative virtual channel allocation, are effective only at low loads. Multiple unicasts drive up the network load, even if only briefly, and easily render these optimizations ineffective. Several redundant messages can be waiting in virtual channels for the same output port (as illustrated in Figure 2).

Even in relatively small systems, on the order of 16 nodes, multiple unicasts can significantly degrade performance, as demonstrated in Section 2. The poor performance of the multiple unicast approach will be exacerbated by the presence of more one-to-many or one-to-all communication as systems grows to encompass dozens or hundreds of nodes.

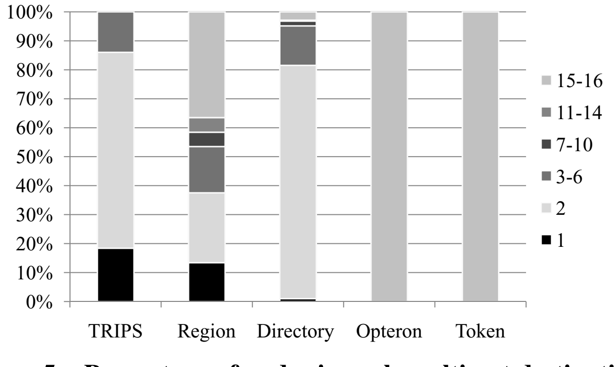
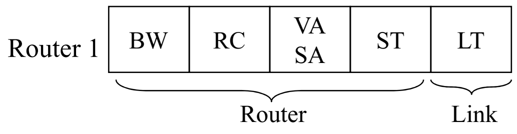
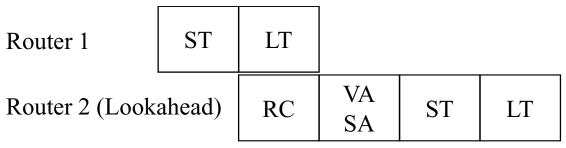


Figure 5. Percentage of nodes in each multicast destination set size



(a) 4-Stage Packet-Switched Router Pipeline (with contention)



(b) Optimized PS Router w/ Lookahead (2 Stage)

Figure 6. Router pipeline

3.2. Virtual Circuit Tree Multicasting

With VCTM, each multicast first forms a virtual circuit connecting the source with the destination set; identified by a VCT number unique to each source and destination set combination. In a tree-based approach, a multicast continues along a common path and branches (replicates) the message when necessary to achieve a minimal route to each destination. An alternative to tree-based routing is a path-based routing, e.g. the network first routes to the closest multicast destination, then from there to the second nearest, and so on, rather than building a multicast tree connecting the destination set (see Section 6 for more details). Once a multicast tree has been set up, packets will be routed based on this VCT number at each router. Multiple VCTs are time-multiplexed on physical links, as in conventional virtual circuit switching [11]. However, unlike virtual circuit switching where intermediate routers do flow control based on virtual circuits, and thus need to support all virtual circuits, we use VCTs *only* for routing. Virtual channels are still used for flow control at the intermediate routers, with virtual channels dynamically allocated to each virtual circuit at each hop.

The virtual circuit table is statically partitioned among source nodes; virtual circuit tree numbers are local to each source. The virtual circuit table is partitioned into n smaller tables each needing a Read/Write port for access from the source assigned that partition. A table with 1024 VCT entries would allocate 64 entries to each source node. In Section 5, we demonstrate that significant performance improvements can be achieved with a much smaller number of virtual trees. Multicast trees can only be evicted at the source; this prevents any multicast packets from missing in a downstream VCT table. Exploring the benefits of dynamically partitioning the VCT table is left to future work.

Restricting the number of currently active VCTs requires that there be reuse of destination sets to see benefit. Data in Figure 4 indicates that there is some amount of reuse across all of our scenarios. The directory and region protocols have less reuse than the other scenarios; however, this figure obscures any temporal component of reuse. Even if a large number of trees are touched across the entire execution, these scenarios see benefit from the temporal reuse of some multicast trees.

VCTM supports three different types of packets: normal

Fields	Head/ Body/Tail	MC/UC	Id	VCT #, Src (Route)	UC Dst	VC #	Payload (Command/Addr)
Width	2 bits	2 bits	1 bit	10 bits	4 bits	3 bits	
Unicast (Normal)	00 - Head	00 Normal	x	Route Encoding	x0001	x001	Invalidate x3000
Unicast (Setup)	00 - Head	01 Setup	1	x003	x0001	x001	Invalidate x3000
Multicast	00 - Head	10 MC	1	x003	xxxx	x001	Invalidate x3000

Figure 7. Header packet encoding format, assuming 1024 virtual circuits, 16 network nodes, and 8 virtual channels.

unicast, unicast+setup, and multicast. Normal unicast packets are equivalent to those found in a traditional packet-switched router, unicast+setup packets are sent to set up a multicast tree, and multicast packets are sent after multicast trees are set up. Figure 7 shows the different fields encoded in each type of packet. If the packet is a normal unicast, the lookahead routing information is encoded in the 4th field instead of the virtual circuit tree number.

3.2.1 Router Microarchitecture. The router microarchitecture is shown in Figure 8. Normal Unicast packets traverse the highly optimized pipeline shown in Figure 6; they do not need to access the Virtual Circuit Tree Table and are routed via existing hardware in the router pipeline (e.g. dimension-ordered routing).

Routing a multicast packet can result in significant complexity; we avoid this complexity through the use of the unicast+setup packets to incrementally construct the trees. Unicast+setup packets deliver a packet payload to a single destination node; however, while delivering this packet, they also add their destination to a multicast tree. The example in Figure 9 walks through the process of constructing a new multicast tree.

When the network interface controller at Node 0 initiates a multicast message, it accesses a Destination Set CAM, containing all of its currently active multicast trees (Step 1). In this example, no matching entry is found, so the node will invalidate its oldest tree (say VCT 1) and begin establishing a new tree (Step 2).

Step 3 decomposes the multicast into one unicast packet per destination. The packet type field is set to be unicast+setup, the Id bit of former VCT 1 was 0 so the new Id bit is 1. Matching Id bits indicate that a new node is being added to an existing tree;

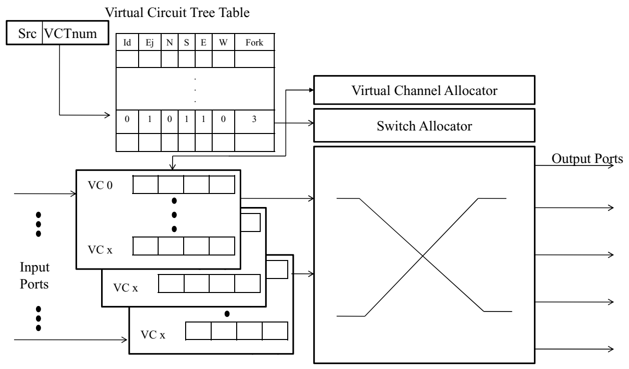


Figure 8. Router microarchitecture

while differing Id bits indicate an old tree is being replaced with a new one. Each packet is given the same VCT number but a different destination. Additionally, all three packets contain the same payload (i.e. memory address plus command).

In Step 4, each packet is injected into the network in consecutive cycles. Unicast packets are dimension-order (X-Y) routed with respect to the source so the resulting multicast tree will also be dimension order routed.

The virtual circuit table updates at Node 1 are shown in Steps 5-8. Step 5 shows the entries prior to the creation of new VCT 1 (highlighted row 2 corresponds to VCT 1). Packet A is routed first. Upon arrival at Node 1, A determines that the VCT entry is stale (due to differing Id bits). Packet A will clear the previous bits in the row and then update the row with a 1 corresponding to its output port based on the routing computation performed by the unicast routing hardware; in this case, the East port. The final column in the row is also updated to reflect the number of output ports this multicast will be routed to.

At Step 7, the unicast destined for Node 4 will update the VCT entry. The Id bits are now the same, so it will not clear the information written by Packet A. A one will be stored into the South column and the output port count is updated to reflect an additional output. Finally, Packet C traverses the link to Node 1; Packet C will also use the East output port. Since Packet A already indicated that this multicast will use the East port, Packet C does not need to make any updates at Node 1.

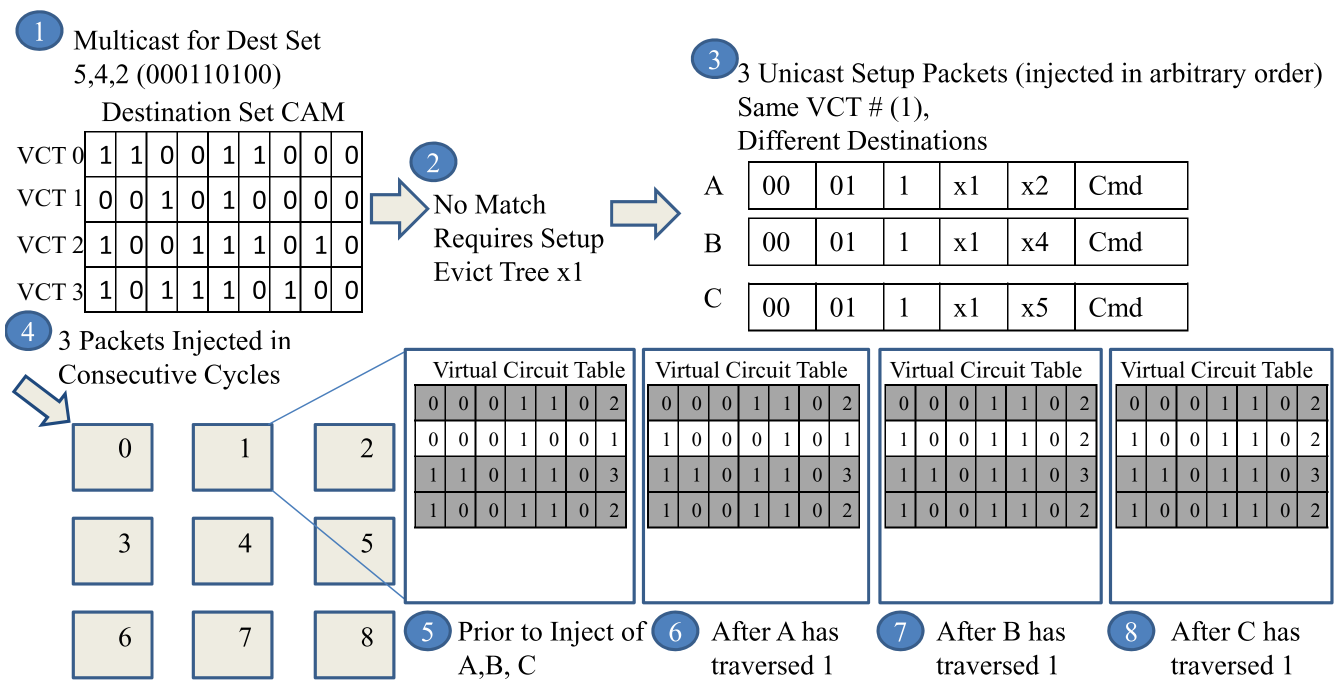
Similar updates will be performed at each Virtual Circuit Table along the route. Updates to the Virtual Circuit Table do not impact the critical path through the router for normal unicast packets as they do not perform any computation needed by the unicast packet to reach its destination.

When a subsequent multicast destined for 5, 4, and 2 arrives at Node 0, it will find its destination set in the CAM. In this case, the node will form a multicast packet with VCT 1. The virtual circuit tree number will index into the Virtual Circuit Table at each router that will output the output ports that this packet needs to be routed to. All three destinations of this packet shared a common route to Node 1 where the first fork occurs. After the first fork, one packet is routed to Node 4 and one packet is routed to the East towards nodes 2 and 5. At Node 2, the packet forks again, with one packet being delivered to the ejection port and another packet continuing on to Node 5.

We encode the destination set by using a virtual circuit tree identifier. The virtual circuit tree identifier can be encoded with $log_2(NumberofVirtualCircuitTrees)$ bits. This is a much more scalable solution than the destination encoding used in prior work while still allowing us the flexibilty to access all destinations and have a variety of multicast tree active concurrently.

3.2.2 Router Pipeline. Figure 10 depicts the changes to the router pipeline originally shown in Figure 6. Unicast packets use the original pipeline. VCTM makes only one change to the router pipeline; for multicast packets the routing computation stage is replaced with the VCT table lookup. Additionally, VA/SA, ST and LT can occur multiple times if the packet is branching at this node; if this is not a multicast branching node, then each of these stages executes once.

Speculative virtual channel allocation is still performed for


1 Multicast for Dest Set 5,4,2 (000110100)
Destination Set CAM
VCT 0 1 1 0 0 1 1 0 0 0
VCT 1 0 0 1 0 1 0 0 0 0
VCT 2 1 0 0 1 1 1 0 1 0
VCT 3 1 0 1 1 1 0 1 0 0
2 No Match Requires Setup Evict Tree x1
3 3 Unicast Setup Packets (injected in arbitrary order) Same VCT # (1), Different Destinations
A 00 01 1 x1 x2 Cmd
B 00 01 1 x1 x4 Cmd
C 00 01 1 x1 x5 Cmd
4 3 Packets Injected in Consecutive Cycles
0
1
2
3
4
5
6
7
8
Virtual Circuit Table
5 Prior to Inject of A,B, C
6 After A has traversed 1
7 After B has traversed 1
8 After C has traversed 1

Note: A traverses 0-1-2, B traverses 0-1-4, C traverses 0-1-2-5

Figure 9. Multicast setup and routing example

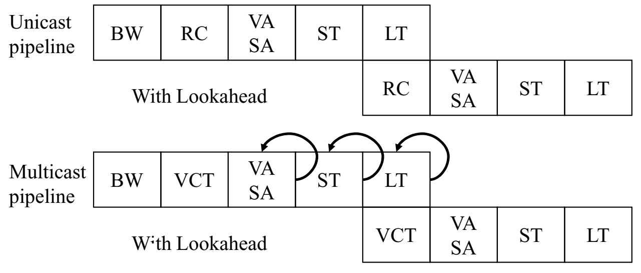


Figure 10. VCTM router pipeline

multicast packets. As in the baseline, we assume that we will get the VCs needed for each output port and speculatively do switch allocation. However, each input may now generate multiple output VC requests. We generate one request each cycle until all multicast outputs are done (the number of output ports in the last column of the table tells us how many times to iterate). Flow control is managed using virtual channels, which are dynamically allocated to virtual circuits. Therefore our design does not require a large number of VCs.

Switch allocation occurs in a similar fashion. Since each input VC may generate multiple switch requests, the router generates one per cycle and queues them up. Our baseline router assumes credit based flow control; credit turnaround is lengthened by the splitting of flits at branch nodes; once the final flit at the branch node has traversed the switch, a credit can be sent to the upstream router.

Our lookahead signal network needs to be wide enough to encode the Virtual Circuit Tree (VCT) number. When the lookahead flit reaches the router, it will access the Virtual Circuit Table to determine the appropriate routing at this stage; this takes the routing computation off the critical path. The Virtual Circuit Table is accessed and that information is used to pre-setup the switch.

Both the baseline router and VCTM use dynamic buffer management to share buffers among the input virtual channels of one port; this reduces the amount of buffering needed to support the potentially longer occupancy of a packet in a buffer. As mentioned in the example, the last column of the VCT entry contains the number of output ports that a multicast is destined for (3 bits wide to accomodate 5 output ports). A flit must remain in the input buffer until the switch allocator has granted the number of requests equal to the port count stored in this column. Once all allocator requests have been granted, the input buffer can be freed and a credit will be sent to the upstream router. Since the multicast trees are dimension order routed, deadlock is avoided. Multicasting in wormhole routed networks can lead to deadlock due to output dependencies between different multicast packets; however, the VCTM router does not reserve resources for trailing flits which eliminates this deadlock scenario.

4. Power and Area Analysis

In the following sections, we explore the overhead associated with virtual circuit tables. While the additional structures in our VCTM router consume additional power, this is offset by power savings shown in Section 5.

4.1. Virtual Circuit Tables

We used Cacti [38] to calculate the area and power overhead associated with adding a virtual circuit tree table to each router shown in Table 2. Four different VCT table sizes are calculated for a 70nm technology; energy reported is dynamic energy per read access. Each entry is 9 bits wide as illustrated in Section 3; these results are estimations as Cacti cannot produce the exact geometry of our table. Assuming a 1 ns clock period, each table can be accessed in less than half a cycle. In Section 5, we will demonstrate that a small number of entries (512) is sufficient to achieve significant performance gains; the table size could be reduced further by using dynamic instead of static partitioning among nodes; this is left to future work. Dynamic partitioning of the VCT tables would allow the approach to scale to a larger number of nodes.

TABLE 2
VCT TABLE OVERHEAD

Number of entries	Area (mm^2)	Energy (nJ)	Time (ns)
512	0.024	0.0018	0.43
1024	0.041	0.0023	0.44
2048	0.078	0.0030	0.46
4096	0.101	0.0037	0.51

We also add a Destination Set CAM to each network interface controller. This *small* CAM is searched for the VCT number matching the given destination set and can be overlapped with the message construction and does not add additional latency to the critical path. The NIC speculates that an active tree will be found and inserts the VCT number returned by the CAM search. In the event of a misspeculation, it is reasonable to assume that decomposing a request into multiple unicast+setup packets will take a couple of cycles, one per destination. Each CAM size in Table 3 corresponds to the number of VCT entries partitioned evenly among 16 nodes. 32 to 64 entry CAMS can be accessed in under a cycle and will give each source a reasonable number of concurrent multicast trees. We do not foresee each core needing more trees as the system scales.

TABLE 3
DESTINATION CAM OVERHEAD

Number of entries	Area (mm^2)	Energy (nJ)	Time (ns)	Total Bytes 16 nodes (25)
32	0.018	0.007	0.87	64 (96)
64	0.021	0.010	0.90	128 (192)
128	0.029	0.017	1.09	256 (384)
256	0.077	0.040	1.53	512 (768)

5. Evaluation

5.1. Workloads Traces

To study a variety of architectures and coherence protocols we leverage traffic traces collected from several simulation environments. Traces for the directory and region-based protocols were generated in PHARMsim [7], a full-system simulator. These traces are collected from end-to-end runs of 8 workloads including 4 commercial workloads, SPECjbb, SPECweb, TPC-H and TPC-W and 4 scientific workloads from the Splash-2 suite. To collect traces for Token Coherence and the Opteron protocol, GEMS 2.1 with Garnet [2] full system simulation environment was used; each Splash-2 workloads was run for the entire parallel phase. Finally, the TRIPs traces use SPEC [34] and MediaBench [1] workloads. They were run on an instantiation of the Grid Processor Architecture containing an ALU execution array and local L1 memory tiles connected via a 5x5 network.

5.2. Synthetic Traffic

In addition to traffic from real workloads, we utilize synthetic traffic to further stress our router design. With a uniform random traffic generator, we can adjust the network load as well as the percentage of multicasts. This traffic generator was used to collect the data in Figure 1.

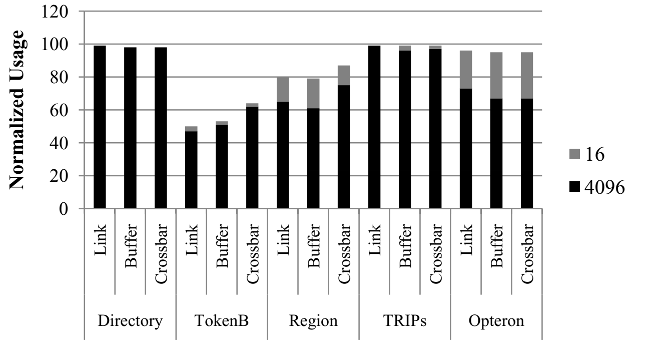


Figure 11. Reduction in buffering, link and crossbar traversals

5.3. Network Configuration

In the next sections, we present the improvement in network latency observed with VCTM relative to our baseline packet-switched network with no multicast support for each outlined scenario. In each graph, 0 Virtual Circuit Trees corresponds to the baseline network where a multicast is decomposed into multiple unicasts. We vary the size of the Virtual Circuit Tree table along the x-axis in Figures 12-16. The network parameters are given in Table 4.

TABLE 4
NETWORK & VIRTUAL CHANNEL ROUTER PARAMETERS

Topology	4-ary 2-mesh 5-ary 2-mesh TRIPS
Routing	X-Y Routing
Channel Width	16 Bytes
Packet Size	1 flit (Coherence req Addr+Cmd) 5 flits (Data) 3 flits (TRIPs)
Virtual Channels	4
Buffers per port	24
Router ports	5
VCTs	Varied from 16 to 4K (1 to 256 VCTs/core)

5.4. Power Savings

The virtual circuit table increases the power consumption of our multicast router over the baseline packet-switched router; however, this increase in power consumption is offset by reducing redundant link switching and buffering.

Figure 11 shows the reduction in buffering, link and crossbar traversals across the different scenarios under evaluation compared to the baseline router. These three component consume nearly 100% of the router power [43]; the reduction in switching shown will translate directly into significant dynamic power savings over the use of multiple unicast messages. This figure shows the power savings for a very small number of multicast trees (16) and a very large number of trees (4096). As performance levels off at or before 4K VCTs, these numbers represent the maximum power savings that can be achieved with our technique.

Buffer accesses are already reduced in our baseline through the use of bypassing in state-of-the-art routers. VCTM is able to

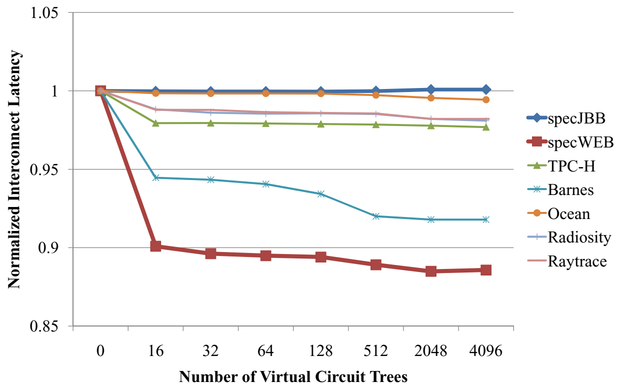


Figure 12. SGI-Origin Directory Protocol network performance

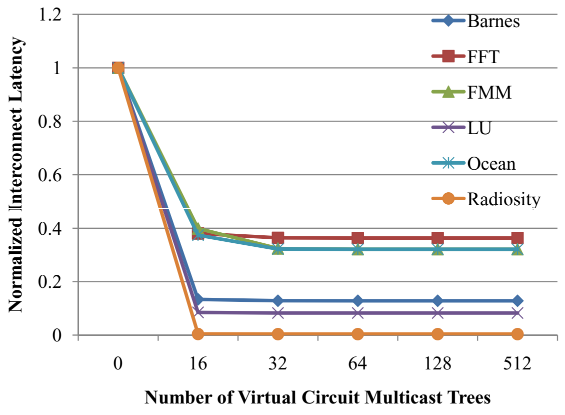


Figure 13. TokenB network performance with VCTM

further reduce overall accessed by removing redundant packets from the network. Removing redundant packets allows more packets to bypass input buffers than in the base case.

Virtual Circuit Trees are constructed along X-Y routing paths. The use of X-Y routing results in deadlock free tree formations; however, X-Y routing does not necessarily produce optimal trees. All destinations are routed to in a minimal fashion but alternative routing algorithms may produce trees that utilize fewer links to reach those destinations. To determine how close to optimal our trees are, we remove the X-Y routing restriction; as a result minimum spanning trees can be found that use up to 60% fewer links; however, the overall savings across all multicasts is only 2%. Despite this large possible reduction, the average reduction in link traversals is less than 1% when compared to the saving achieved with X-Y routing. As a result, the power savings detailed in Figure 11 are close to the maximum possible savings.

5.5. Performance Evaluation

As mentioned earlier, the potential performance improvement come from two main factors, reduction in network load (improved throughput) and reduced contention for network injection ports. The VCTM router reduces the number of messages injected into the network from the size of the destination set to a single message. Injection port contention accounts for up to 35% of packet latency in the baseline. On average, alleviating injection pressure reduces the cycles spent in the network interface by 0.2 (directory), 6 (token), 5 (region), 0.5 (TRIPs) and 3.5 (Opteron) cycles.

5.5.1 Coherence Protocols. For the directory protocol we simulate 32KB L1 caches and private 1MB L2 caches. Addresses are distributed across 16 directories, with one directory located at each processor tile. The reduction in network latency for a directory protocol is shown in Figure 12. Invalidation requests represent approximately 5% of network requests for the directory protocol. However, since the network load for this protocol is low for applications like specJBB, TPC-H, Raytrace and Ocean, VCTM is unable to realize substantial benefits. SPECweb has a slightly higher load which translates into more benefit from VCTM (up to 12%).

Token coherence simulations were configured with 64 KB L1 caches and 1 MB private L2 caches with a MOESI TokenB protocol. Normalized interconnect latency is presented in Figure 13. To request tokens, the source node broadcasts to all other nodes on-chip; as a result, only one virtual circuit multicast tree is needed per node. If the source node sent out a multicast with a variable destination set, a VCT count larger than 16 would be useful. Radiosity has reached network saturation with the baseline configuration; relieving the pressure caused by multiple unicasts results in substantial latency improvement of close to 100%. Barnes and LU are also very close to saturation, leading to close to 90% savings in latency.

We further evaluate VCTM using path-based multicast routing rather than tree-based multicasting. For workloads near or at saturation, a path-based multicast can also effectively relieve network pressure and reduce latency by an average of 70%; however, for workloads not nearing saturation (FFT, FMM, and Ocean), the path-based multicast increases network latency by 48% over the baseline. For more discussion on path-based routing trade-offs see Section 6.

The region-based coherence protocol uses 2KB regions; each region covers 32 cache lines. Region coherence arrays (RCA) are used to store the sharing list for each 2KB region; these RCAs sit alongside the L2 cache. The 16-core configuration used to generate these traces consists of 32 KB L1 caches and a private 1MB L2 cache per core. Figure 14 presents the improvement in network latency for this region-based scheme. This coherence protocol needs more simultaneous multicast trees than the other scenarios to see substantial benefit. The destination sets used by the region protocol vary much more widely; however, the overhead of supporting additional trees is low, 512 or 2048 VCTs would be a feasible design and would reduce network latency by up to 65%. Without multicast support, this type of coherence protocol would see prohibitive network latency for sending out snoop requests.

Figure 15 shows benchmarks with varying degrees of improvement due to multicast support for the TRIPs architecture. Art sees the most benefit due to the network load approaching saturation and a significant reduction in the number of packets by using VCTM. The majority of workloads see up to 20% improvement due to the low number of nodes in the destination set (average 2). Additionally, many of the multicast trees constructed for this workload branch at the source node. If the branch occurs at the source node, no benefit is seen as there will be no reduction in packets over the multiple unicast baseline approach. Path-based multicasting outperforms VCTM for art,

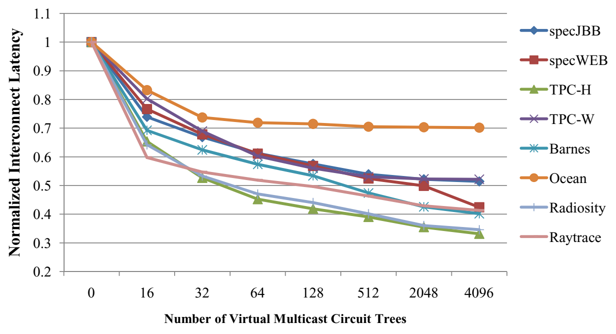


Figure 14. Region Coherence network performance with VCTM

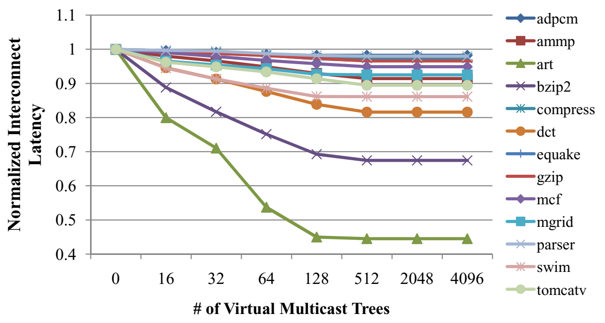


Figure 15. TRIPs network performance with VCTM

bzip2, and swim by 4% here due to more effectively reducing the network load.

Traces were generated for the Opteron protocol with 64 KB L1 caches and a 16 MB fully shared L2 cache. The Opteron protocol sees steady improvement with the addition of Virtual Circuit Trees; once 512 VCTs are available, performance levels off with a savings of 47% in network latency. Some filtering of destinations occurs in this protocol resulting in a larger number of trees than the TokenB protocol. In all cases, path-based multicasting performs worse than the baseline and VCTM. Nearly all multicasts in the Opteron protocol go to 15 destinations so a path-based multicast has to snake through the chip to each node; if a non-optimal path is chosen, latency will be high. An additional downside to path-based multicasting, not reflected in the network latency is that the requestor will have to wait until the last node in the path has received and responded to the snoop. VCTM delivers all snoops in a more timely fashion resulting in faster snoop response.

For VCT tables with 512 entries, we are able to achieve significant reuse across all scenarios. Average VCT table hit rates range from 62% to 99%; with directory coherence seeing the lowest hit rates overall.

5.5.2 Synthetic Traffic. Several aggressive packet-switched networks are evaluated in Figure 17a for their ability to approach the performance of VCTM. The PS baseline represents the same baseline configuration used above. The network interface (NIC) represents a substantial bottleneck for multicasting, so the Wide-NIC configuration allows the NIC to inject as many packets as are waiting in a cycle (in the baseline only one packet can be accepted per cycle). We add to Wide-NIC nearly infinite Virtual Channels and buffers (wide-nic+vcs). To better

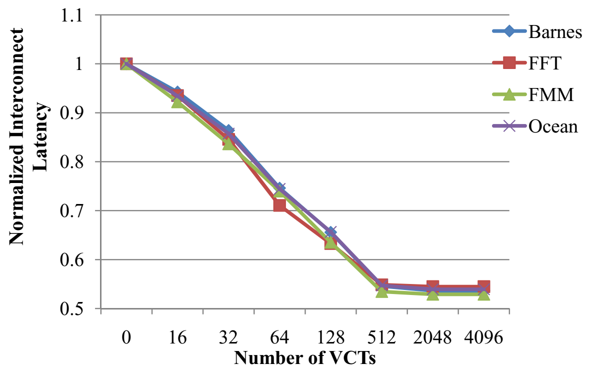


Figure 16. Opteron Protocol

distribute the network load, we utilize adaptive routing (wide-nic+vcs+adapt). Finally, we compare each of these scenarios to VCTM with 2048 VCTs. Here, approximately half of the latency penalty associated with the multiple unicast approach is paid in the NIC; creating a wider issue NIC would likely result in significant overhead and complexity; more so than our proposal. Wide-NIC+VCs+Adapt outperforms VCTM for moderate loads; performance improvements in VCTM are predicated on tree reuse which is very low for uniform random traffic. Building a design with a very large number of VCs would have a prohibitive cost (area and power); we believe our design is much more practical. Figure 17b illustrates that with real workloads, VCTM outperforms a highly aggressive (unrealistic) packet-switched router. We compare one benchmark running with 512 VCTs from each scenario to Wide-NIC+VCs+Adapt.

6. Related Work

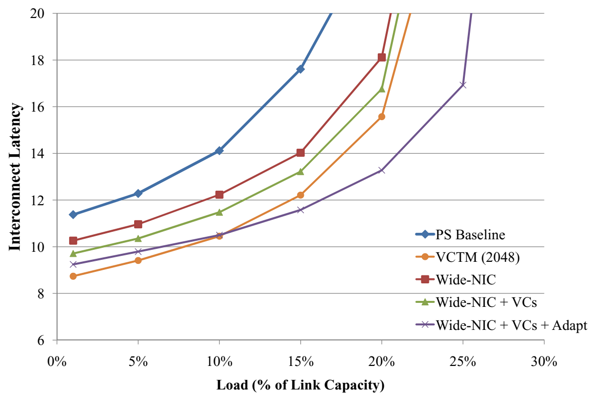
In this section, we differentiate VCTM from prior proposals along three axes: routing, destination encoding and deadlock.

6.1. Routing

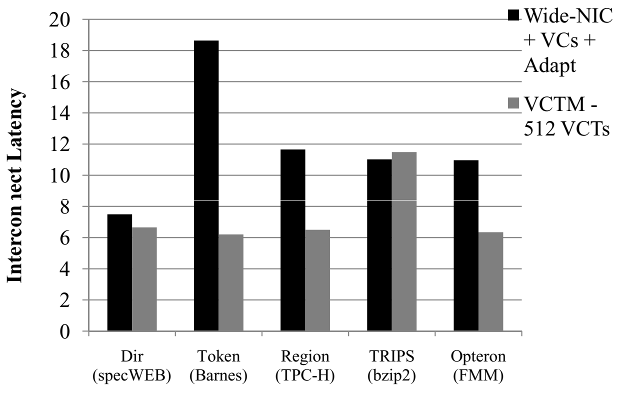
There are two techniques for routing multicast messages: path-based routing and tree-based routing. In path-based routing, each destination is visited sequentially until the last node is reached. Paths must be carefully selected to avoid a deadlock; cycles can occur in the network even in the presence of dimension order routing. Path-based multicasting also has the added complexity of finding the shortest path that visits all nodes in the destination set.

Very little prior work focuses on the design of on-chip multicast routers. In [24], the authors construct circuits for multicasting in a wormhole network. Path-based routing plus the requirement of setup and acknowledgement messages results in a long latency overhead for their approach. Alternatively, we focus on a tree-based multicast routing in this work but also compare against path-based multicasting.

With a path-based multicast, current lookahead routing mechanisms can be used as only one destination is being routed to at a time. Path multicasting is attractive for its simplicity; implementing a path multicast would require only minor modifications to the current packet-switched router. Recent work has shown network latency to be a critical factor for commercial workloads on CMPs [12]; therefore, it is preferable to avoid the sequential latency associated with a path-based routing approach. Our design is able to leverage lookahead techniques



(a) Uniform Random Traffic with 10% Multicasts



(b) VCTM vs. Aggressive PS Network

Figure 17. Comparison against aggressive (unrealistic) network

by using slightly wider bundles to remove the VCT lookup from the critical path.

A multicast router [16] for DNUCA caches is constrained to match very specific characteristics of this design space, i.e. many routes are unused. Additionally, the details of how they realize their multicast router are sparse. Circuit-switching and time-division multiplexing have been used for on-chip networks that provide multicast functionality [23], but suffer from the constrained bandwidth of circuit switching.

Significant work has been done for off-chip multicast routers. Several proposals target multistage interconnection networks [36], [41]. Domain specific requirements of off-chip networks (e.g. ATM switches) can be quite different [41]; this work targets a switch with 1000 input ports. Routers with a large number of ports are prohibitively expensive for on-chip networks making this type of solution unattractive. While a lookup table is also indexed to find the proper output port mappings, VCTM utilizes a much smaller lookup table that is more suitable for on-chip designs. Additionally, their work adds and removes nodes incrementally to a multicast tree while our work creates a new tree at low latency and overhead and hence does not support addition or deletion of nodes; this feature could be easily achieved with VCTM but assessing its usefulness is left to future work. Another design [36] also uses a lookup table to determine the routes but advocates using software to pre-setup this table; this approach would work well for fixed routes that endure for a significant amount of time. For fine-grained on-chip parallelism, software approaches to routing may incur too much overhead.

Recently, Anton [32], specifically designed for molecular dynamics, uses off-chip multicasting; multicasts can be sent to limited sets of nodes.The Piranha architecture's [3] novel technique, *cruise missile invalidates* limits the number of messages injected into the off-chip network from a single request; each invalidate multicasts to a subset of nodes. As with VCTM, Piranha sees improved invalidation latencies by reducing the number of messages. They also reduce the number of acknowledgements; we defer study of similar reduction operations to future work.

6.2. Destination Encoding

Another significant challenge with on-chip multicasting is the destination encoding within the header flit. There are several approaches to destination encoding [9] including all-destination encoding and bit-string encoding. The all-destination approach encodes each destination node number into the header. The header can be of variable size with an end of header character to delineate it from the payload. Bit string encoding uses a single bit for each possible destination. If the node is included in the destination set, the bit will be set to 1. The header size needed to encode the number of destinations grows as the network grows for each of these approaches.

Nodes can also be partitioned into regions and multicasts sent to destinations within each region. The header size can be fixed to limit the number of possible destinations that a multicast can reach; however, our solution is more flexible. The 16 bits necessary to encode all possible destinations in a 4x4 mesh could reference 2^{16} different trees. Virtual Circuit Tree Multicasting is thus a much more scalable solution than destination or bit-string encoding.

In an off-chip network design, multiport encoding [33] determines when to replicate a packet in each stage of a multistage network; this restricts the destination sets that can be reached by a packet. Some multicasts require multiple passes through the network to reach all destinations; due to the latency sensitivity of our workloads, multiple passes are undesirable. Multiple output ports are encoded in the header flits; a different output is used in each pass until all destination are reached. This approach is closely tied to the multistage topology and would be difficult to implement in a two dimensional mesh. VCTM efficiently routes multicasts in a straightforward manner. Their work also acknowledges the inefficiencies of the multiple unicast approach but in an off-chip setting.

6.3. Deadlock

Routing decisions are more complex with tree-based routing. Difficulties arise when a branch occurs in the multicast tree. Routing to multiple destinations simultaneously would require either replication of the routing hardware for all n possible destinations of a multicast packet or would require n iterations through the routing logic. Lookahead routing could encode multiple output ports for the next hop in network, however, the destination set would need to be properly partitioned. At a fork, a subset of destinations would need to be encoded in each branch. Failing to prune destinations from the header flit or improperly pruning destinations would result in deadlock. VCTM uses the VCT number to avoid the partitioning and pruning the destination set.

Research into deadlock-free multicast tree routing [25] uses pruning to prevent deadlock in a wormhole-routed network. Their work targets small messages such as invalidates in a DSM system. In VCTM, for a given tree, all of the leaf destinations are reached via dimension-order routing with respect to the source node; therefore no cycles can occur within a single multicast tree instance. If a tree were allowed to adaptively route with respect to a branch (an intermediate route point) deadlock would be a problem.

We evaluate VCTM with proposals that represent a range of scenarios; however we believe VCTM is widely amenable to other proposals mentioned in Sections 1 and 2. For example, if destination sets exhibit temporal reuse, VCTM will work well with Destination Set Prediction [28].

7. Conclusion

In this paper, we present a case for hardware support for on-chip multicasting. Our characterization of existing network applications (directory coherence, Opteron coherence protocol) as well as proposed future applications (Token Coherence, TRIPS, Region coherence) strongly supports our claim that multicasting is both necessary and useful. Furthermore, the availability of efficient support for on-chip multicasting will most likely enable future techniques that may otherwise appear unattractive or even infeasible.

In support of these existing and proposed applications, we describe a novel on-chip multicast router that fills a significant gap in the design space. We believe VCTM is the first multicast router for a general-purpose CMP design with the flexibility to provide superior performance across a variety of scenarios; future work will explore additional novel scenarios that are made possible through our VCTM router.

Our VCTM design substantially reduces power consumption over state-of-the-art designs that do not directly support multicasting. On average we see a 29%, 22% and 20% reduction in link switching, buffer and crossbar power respectively; with a maximum savings of 53%, 49% and 38%. Virtual Circuit Tree Multicasting is also able to reduce network latency by up to 90% with an average latency reduction of 39%. Even though many of our scenarios are not operating at saturation, we see significant benefits through improved speculation from this reduction. Network latency-throughput is a critical factor for many applications; this significant reduction will not only benefit existing applications and architectures but also pave the way for new coherence protocol innovations.

References

[1] http://www.eecs.umich.edu/mibench.

[2] http://www.princeton.edu/ niketa/garnet.html, 2008.

[3] L. A. Barroso, et al., "Piranha: A scalable architecture based on single-chip multiprocessing," in *ISCA-27*, 2000.

[4] E. Bilir et. al, "Multicast snooping: A new coherence method using a multicast address network," in *ISCA-26*, 1999.

[5] S. Borkar, "Networks for multi-core chips: A contrarian view," Special Session at ISLPED 2007.

[6] J. A. Butts and G. Sohi, "Characterizing and predicting value degree of use," in *MICRO-35*, 2002.

[7] H. Cain, K. Lepak, B. Schwarz, and M. H. Lipasti, "Precise and accurate processor simulation," in *Workshop On Computer Architecture Evaluation using Commercial Workloads*, 2002.

[8] J. F. Cantin, M. H. Lipasti, and J. E. Smith, "Improving multiprocessor performance with coarse-grain coherence tracking," in *ISCA-32*, 2005.

[9] C. Chiang and L. Ni, "Multi-address encoding for multicast," in *Proc. of the Workshop on Parallel Comp. Routing and Communication*, 1994.

[10] P. Conway and B. Hughes, "The AMD Opteron Northbridge architecture, present and future," *IEEE Micro Mag.*, Apr. 2007.

[11] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Pub., 2003.

[12] N. Enright Jerger, , L.-S. Peh, and M. Lipasti, "Circuit-switched coherence," in *IEEE 2nd Network on Chip Symposium*, 2008.

[13] N. Enright Jerger, D. Vantrease, and M. H. Lipasti, "An evaluation of server consolidation workloads for multi-core designs," in *IEEE of Symposium on Workload Characterization*, 2007.

[14] P. Gratz et. al, "Implementation and evaluation of on-chip network architectures," in *ICCD*, 2006.

[15] Intel, "From a few cores to many: A Tera-scale computing research overview," 2006. [Online]. Available: http://download.intel.com/research/platform/terascale/ terascale_overview_paper.pdf

[16] Y. Jin, E. J. Kim, and K. H. Yum, "A domain-specific on-chip network design for large scale cache systems," in *HPCA*, 2007.

[17] D. Kanter, "The common system interface: Intel's future interconnect," http://www.realworldtech.com/page.cfm? ArticleID=RWT082807020032, 2007.

[18] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non- uniform cache structure for wire- delay dominated on-chip caches," in *Proceedings of ASPLOS*, 2002.

[19] M. M. Kim, S. Swanson, A. Peterson, A. Putnam, A. Schwerin, M. Oskin, and S. Eggers, "Instruction scheduling for a tiled dataflow architecture," in *Proceedings of ASPLOS*, 2006.

[20] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *ISCA-34*, 2007.

[21] A. Kumar et al, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *ICCD*, 2007.

[22] J. Laudon and D. Lenoski, "The SGI Origin: a ccNUMA highly scalable server," in *ISCA-24*, 1997.

[23] J. Liu, L.-R. Zeng, and J. Tenhunen, "Interconnect intellectual property for network on chip," *Journal of Sys. Arch.*, 2004.

[24] Z. Lu, B. Yin, and A. Jantsch, "Connection oriented multicasting in wormhole-switched networks on chip," in *Proceedings of Emerging VLSI Technologies and Architectures*, 2006.

[25] M. P. Malumbres, J. Duato, and J. Torrellas, "An efficient implementation of tree-based multicast routing for distributed shared memory multiprocessors," in *IEEE IPDPS*, 1996.

[26] M. M. K. Martin, M. D. Hill, and D. A. Wood, "Token coherence: Decoupling performance and correctness," in *ISCA-30*, 2003.

[27] M. Martin, et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *CAN*, Sept 2005.

[28] M. M. K. Martin et. al, "Using destination-set prediction to improve the latency/bandwidth tradeoff in shared-memory multiprocessors," in *ISCA-30*, 2003.

[29] M. Marty and M. Hill, "Virtual hierarchies to support server consolidation," in *ISCA-34*, 2007.

[30] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *ISCA-31*, 2004.

[31] K. Sankaralingam et. al, "Exploiting ILP, TLP, and DLP using polymorphism in the TRIPS architecture," in *ISCA-30*, 2003.

[32] D. E. Shaw et. al, "Anton, a special-purpose machine for molecular dynamics simulation," in *ISCA-34*, 2007.

[33] R. Sivaram, D. K. Panda, and C. B. Stunkel, "Efficient broadcast and multicast on multistage interconnection networks using multiport encoding," *IEEE TPDS*, vol. 9, no. 10, October 1998.

[34] SPEC, "SPEC benchmarks," http://www.spec.org.

[35] K. Strauss, X. Shen, and J. Torrellas, "Uncorq: Unconstrained snoop request delivery in embedded-ring multiprocessors," in *MICRO-40*, 2007.

[36] C. B. Stunkel, J. Herring, B. Abali, and R. Sivaram, "A new switch chip for IBM RS/6000 SP systems," in *Proceedings of Supercomputing*, 1999.

[37] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin, "Wavescalar," in *MICRO-36*, 2003.

[38] D. Tarjan, S. Thoziyoor, and N. P. Jouppi, "Cacti 4.0," HP Technical Report, Hewlett Packard, Tech. Rep., 2006.

[39] M. B. Taylor et. al, "Scalar operand networks: On-chip interconnect for ILP in partitioned architectures," in *HPCA*, 2003.

[40] TPC, "TPC benchmarks," http://www.tpc.org.

[41] J. Turner, "An optimal nonblocking multicast virtual circuit switch," in *Proceedings of Infocom*, 1994.

[42] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *MICRO-35*, 2002.

[43] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitecture in on-chip networks," in *MICRO-36*, 2003.

[44] S. Woo et. al, "The SPLASH-2 programs: Characterization and methodological considerations," in *ISCA-22*, 1995.