To be presented in International Conference on Parallel Processing, Aug. 1996

Reducing Cache Invalidation Overheads in Wormhole Routed DSMs Using Multidestination Message Passing [*]

Donglai Dai and Dhabaleswar K. Panda
Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210-1277
Email: {dai,panda}@cis.ohio-state.edu

Abstract: *Current generation DSM systems use point-to-point (unicast) messages for cache invalidations. This incurs a large number of control messages, heavy network traffic, and high occupancy at home nodes. This paper introduces a new approach to reduce these overheads by using multidestination-based reservation and gather worms for distributing invalidation requests and collecting acknowledgments. Different grouping schemes to generate multidestination worms on networks supporting deterministic (e-cube) or adaptive (turn-model) routing are investigated to implement the fully-mapped cache-coherence protocol. For different applications on a 2D mesh system, our simulation results indicate that up to 15% reduction in overall execution time can be achieved by using multidestination messages.*

1 Introduction

To design scalable parallel systems with distributed shared memory (DSM) paradigm, architects typically use directory-based cache-coherence protocols atop message-passing hardware [2, 4]. Most recent DSM related research has emphasized on designing efficient directory organizations and cache coherence protocols, deriving optimal cache sizes, analyzing cache invalidation patterns, and reducing synchronization overheads. However, very little research is done towards reducing overheads of cache coherence protocols on message-passing systems by taking advantage of underlying interconnection networks and routing schemes.

Under closer examination, it can be observed that cache coherence protocols use two fundamental message-passing operations for a cache invalidation: sending *one-to-many* messages from one node to a set of other nodes and collecting *many-to-one* messages from a set of nodes to one node. Both patterns belong to the class of *collective communication* [7]. Currently the wormhole-routing switching technique [8] is the trend in building scalable parallel systems. In recent years, many new adaptive routing schemes and multicasting schemes [10] have been proposed for wormhole routed systems. These developments lead to a natural question whether we can take advantage of these schemes to reduce cache coherence overheads in DSM systems.

Consider current generation DSM systems, which use directory-based protocols to enforce conventional sequential consistency. In such systems, when an invalidation transaction occurs, a home node sends multiple *unicast* messages to all sharing nodes and receives unicast acknowledgments from them. Such unicast message passing incurs high traffic and contention in the network. It also makes the home nodes as *hot-spots* in the system. This has considerable impact on the *occupancy* of messages at directories [3]. Such overheads get reflected as high-latency for write operations, leading to degradation on the overall system performance. In WWT [5], the invalidation requests are broadcasted using a dedicated broadcast network. This leads to a costly design. Recently, Bhuyan et al. [6] have proposed an embedded hierarchical ring broadcast mechanism for implementing fast invalidations. However, this approach leads to high latency for invalidations.

Recently *multidestination* message-passing mechanisms have been introduced for wormhole networks to achieve low-latency multicast [10] and gather [9] operations on distributed memory systems. In this paper, we propose a set of multidestination-based *reservation* and *gather* worms to implement cache-coherence with reduced overheads. A small set of *invalidation acknowledgment (i-ack)* buffers are proposed to be used at the router interfaces to facilitate fast collection of acknowledgments. Different grouping schemes are proposed to send cache-invalidation requests and collect acknowledgments for deterministic (e-cube) and adaptive (turn-model) routing schemes. The effect of these grouping schemes to reduce number of invalidation request/ack messages, total network messages and overall execution time is studied through simulation experiments. Depending on the invalidation characteristics in these applications, a wide range of improvement (2-15% reduction in overall execution time) is observed. It is shown that turn-model routing with a *density-dependent column grouping* leads to the best reduction on overall execution time for these applications. For simplicity, in this paper, we consider a fully-mapped directory system with sequential consistency. However, the methodology is quite general and can be applied to other partial-directory schemes and consistency models.

2 Cache Invalidation Overheads in DSMs

In this section, we briefly overview DSM architecture and directory-based protocols. Then we analyze communication characteristics of cache invalidation transactions and the associated overheads.

[*]This research is supported by NSF Career Award MIP-9502294.

2.1 Architecture and Directory-based Protocols

Let us consider a typical DSM architecture as shown in Fig. 1. The system consists of a set of identical processing nodes connected by a low latency interconnection network. Each node consists of processor (PE), cache, cache controller (CC), a module of the global shared-memory (MM), directory controller (DC), incoming message controller (IC), and outgoing message controller (OC). The IC and OC modules connect the node to the interconnection network through the router interface. Every node is the *home node* to a set of memory blocks it contains. Each memory block has an associated directory entry consisting of current *state* information and a *pointer array*. A simple implementation of the pointer array uses a presence bit vector with one bit for each node, popularly known as *fully-mapped* directory.

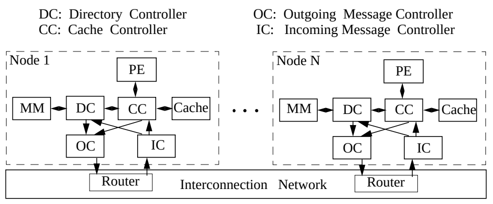


Figure 1: Typical architecture of a DSM system using directory-based protocols.

2.2 Characteristics of Invalidations

In a system using a directory-based write-invalidation protocol, when a request to obtain exclusive-write access comes from a writer node to the home node of a cache block, the home node sends invalidation request messages to all sharers. These sharers, after receiving the invalidation requests, invalidate their respective cache blocks and send an acknowledgment each to the home node. The home node receives all these acknowledgments and then provides exclusive-write access to the writer node requesting the cache block. The entire sequence can be defined as an *invalidation transaction*, which consists of mainly two phases: *request* and *acknowledgment*.

Figure 2 illustrates the two phases for a sample sharing distribution in an e-cube routed 8×8 mesh DSM supporting point-to-point (unicast) message passing. In this example, the degree of sharing for the memory block being invalidated is assumed to be 23. Thus, the home node sends 23 invalidation requests and receives 23 acknowledgments. Let us denote such a framework as *Unicast-based Invalidation and Unicast-based Acknowledgment (UI-UA)*. Under this framework, the invalidation takes considerable time due to sending and receiving a large number of messages. The hot-spot effect occurs at the home node in both the request phase as well as the acknowledgment phase.

It is to be noted that a strong dependency exists among the request and reply messages for achieving cache coherence, leading to deadlocks in such systems. As a common practice, a pair of separate networks (at least logically separated) are used. Any request and reply messages that are related to each other are forced to travel in different networks to break the hold-and-wait condition that is necessary for a deadlock to occur. This means that the above two phases of an invalidation transaction usually take place in separate networks. Thus, for a major fraction of the time of an invalidation transaction, these two phases progress in an overlapped manner.

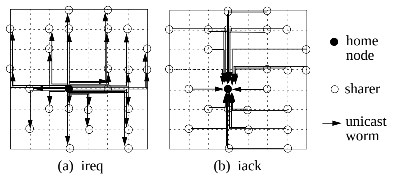


Figure 2: Example of an invalidation in an e-cube mesh DSM supporting unicast communication: (a) the request phase and (b) the acknowledgment phase.

2.3 Latency and Traffic Estimates

In order to analyze the latency of an invalidation transaction quantitatively, let us use the following four simple measures. *Home node occupancy* [3] reflects the amount of processing time required at a home node in order to send the requests and receive the acknowledgments. It is proportional to the number of messages sent from and received by the home node. *Average distance* of a sharer from the home node reflects the network latency component of invalidation latency. Based on the underlying routing scheme, this component is related to the average number of hops traveled by a message. The *number of messages* and *total number of hops* traversed by the messages offer us valuable insight to the volume of network traffic generated by an invalidation transaction. For the example shown in Fig.2, the values of these four measures are 46, 4, 46, and 186, respectively.

Let us now derive an estimate for latency of an invalidation on a $k \times k$ mesh using wormhole routing [8]. Assume the average number of nodes sharing a block is d. Thus, the invalidation will require $2d$ messages. Let us assume the following timing parameters: message startup time equal to α, router delay equal to β, and one hop delay per flit equal to γ. Assume the message length is l flits. For a $k \times k$ system, the average distance traveled by a single message is k. During the invalidation transaction, the home node sends requests to the sharers one after another. If we ignore network contention, the overall time for all the requests to be received by the sharers $T_{ireq} = d\alpha + k\beta + (l-1)\gamma$. Let us assume that cache invalidation at a sharer takes δ time on average. An acknowledgment takes $T_{iack} = \alpha + k\beta + (l-1)\gamma$ time to reach the home node. By assuming the best overlapping, i.e., $(d-1)$ acknowledgments have been received and processed by the home node prior to the arrival of the last acknowledgment from a sharer, the overall invalidation latency $T_{inv} = T_{ireq} + \delta + T_{iack} = (d+1)\alpha + 2k\beta + 2(l-1)\gamma + \delta$.

Typically, network contention and hot-spot effect surrounding the home node increase this latency considerably. For the above analysis, the total number of hops traveled by $2d$ messages are $2dk$. This relates closely to the volume of communication traffic required for an invalidation. As d and k increase, it can be observed that the volume of communication traffic increases, leading to a potential increase in network contention; It also increases the no-contention invalidation latency, T_{inv}. This leads to a question whether less than d invalidation requests and d acknowledgments can be used to accomplish an invalidation with d sharers. We use a multidestination message passing approach to accomplish this.

3 Multidestination Message Passing

The multidestination message passing mechanism allows data to be delivered to or picked up from multiple nodes with a single message. A new Base-Routing-Confirmed-Path (BRCP) model has been recently proposed in [10] to implement multidestination mechanism on wormhole networks with different routing schemes.

3.1 The BRCP Model

Figure 3 shows feasible paths under the BRCP model for a 2D mesh supporting two kinds of routing. In an e-cube system (assuming messages are routed first along the row and then along the column), a multidestination worm can cover a set of destinations in row/column/row-column order as shown in Fig. 3. On a turn-model system, a multidestination worm can cover destinations along any west-first non-minimal path, in addition to e-cube paths, as shown in Fig. 3. The significant benefit of this BRCP model comes from the fact that a message can be delivered to multiple destinations with the same overhead as that of sending it to a single destination. Similarly, information can be gathered from multiple destinations with a single message.

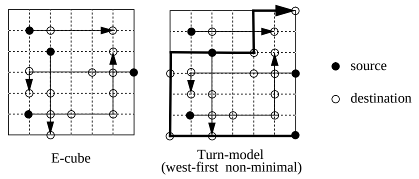


Figure 3: Feasible paths for multidestination worms.

3.2 Multicast and Gather Worms

A multidestination multicast worm consists of a set of destinations along a feasible path in its header. The worm uses *forward-and-absorb* capability at the router interface of each intermediate destination. Using such worms, sophisticated multicasting schemes are feasible to implement a multicast with reduced latency [10].

A multidestination gather worm collects information from multiple nodes at their respective router interfaces [9]. Each router interface can have a set of buffer entries containing special flags. An 'on' state of such a flag indicates that the associated processing node has arrived at the gather execution point. A typical gather worm, while passing through the router interface of intended destinations, checks for this flag. If the flag is 'on', the worm collects the information and proceeds ahead. If the flag is not 'on', the worm waits for this flag to be 'on'. Such a mechanism allows a gather worm to collect data/signal in a cumulative manner from all its intermediate destinations and deliver it to the final destination. In the following section, we propose augmentation to these multidestination worms for effective cache invalidations.

4 Frameworks to Reduce Overheads

In this section, we introduce new frameworks and mechanisms to implement efficient invalidations with multidestination messages.

4.1 MI-UA Framework

By using a multidestination multicast worm, a home node can send an invalidation request to a set of sharers along a single path. Let us name such a worm as *multidestination-based invalidation request (m-ireq)* worm. After receiving the request, a sharer invalidates its local cache and sends a unicast-based acknowledgment message back to the home node. The home node collects all the individual invalidation acknowledgments. We identify such a framework as *Multidestination-based Invalidation request and Unicast-based Acknowledgment (MI-UA)*.

Let us compare the MI-UA and the UI-UA frameworks through an example. Figure 4 highlights the distinction between these two frameworks. We only illustrate a portion, relating to the sharing nodes along column 6, of the example invalidation transaction previously shown in Fig. 2. The rest of the invalidation transaction can be implemented similarly and is not illustrated here for clarity. During the request phase of the invalidation transaction in the UI-UA framework, as shown in Fig. 4(a), home node sends five unicast-based worms to the sharers. While in the MI-UA framework, as shown in Fig. 4(b), home node sends only two (up-turn and down-turn) m-ireq worms. During the acknowledgment phase, each of these sharers sends a unicast-based invalidation acknowledgment back to home node in both frameworks. It can be observed that for the example invalidation transaction in Fig. 2, the home node needs to send only 11 invalidation worms in the MI-UA framework as compared to 23 worms in the UI-UA framework. This demonstrates considerable potential to reduce occupancy at the home node as well as the invalidation latency.

Besides supporting *forward-and-absorb* mechanism at each router interface [10], the MI-UA framework does not

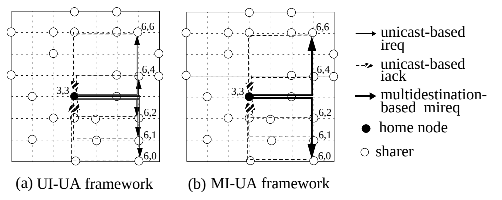


Figure 4: Comparing the communication traffic for an invalidation transaction in two frameworks: (a) UI-UA and (b) MI-UA.

require any additional modifications to the current generation DSM systems. The main thrust of this framework is to reduce occupancy of the home node and the volume of network traffic incurred in the request phase of an invalidation transaction. In the acknowledgment phase, however, this framework reduces neither occupancy of the home node nor occupancy of sharers. It also does not reduce the volume of network traffic. The question is whether a better framework is feasible which can achieve the above three kinds of reduction in the acknowledgment phase. In order to have such a framework, we first introduce three novel mechanisms.

4.2 Three Novel Mechanisms

4.2.1 I-gather Worm

Let us consider using gather worms to collect the acknowledgments. Assume that after receiving the request and invalidating the local cache block, a sharer can set a buffered signal at its associated router interface instead of being forced to send a unicast-based acknowledgment. A multidestination-based gather worm can pass through each router interface associated with the sharers, collect the signal, and deliver the information to the home node. We name such a gather worm as a *multidestination-based invalidation gather (i-gather)* worm. Such i-gather worms significantly reduce: 1) the occupancy of cache invalidation for most sharers, 2) the volume of network traffic, and 3) the occupancy of the home node.

4.2.2 I-ack Buffer

A special buffer at the router interface is used to store the signal associated with an invalidation acknowledgment of a memory block for a short period of time. We denote such buffer as *i-ack buffer*, an entry in the i-ack buffer as *i-ack entry*, and the signal as *i-ack signal*. Ideally, every router interface could have a dedicated i-ack entry for each block of the global memory. However, it is not feasible to do so because of cost consideration. Fortunately, the average number of invalidation transactions, which a node participates during a given time interval, is quite small. Let us consider allowing sharing of the i-ack entries at a router interface among the ongoing invalidation transactions. In order to distinguish which i-ack entry is used by which invalidation transaction of a memory block, the block address of the invalidation transaction is stored in the i-ack entry and used as an identifier. When an i-gather worm arrives, it can do a fully-associative search based on the block address to find its corresponding i-ack entry and collect the i-ack signal. Hence, we propose that the structure of an i-ack entry consists of a free/used flag, an invalidation acknowledgment signal (i.e.,*i-ack signal*), a block address field, and a field to hold a copy of a message. We discuss the usage of the message field in next section. Figure 5 shows the enhanced node organization and the associated router interface.

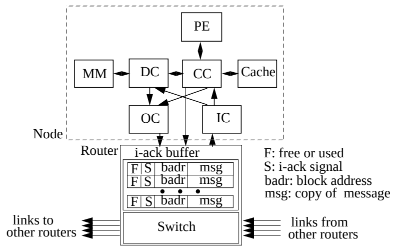


Figure 5: Enhanced node organization and the associated router interface with the i-ack buffer.

4.2.3 I-reserve Worm

With the above enhanced node organization and router interface, we introduce a *multidestination-based invalidation reservation (i-reserve)* worm. The i-reserve worm is a variation of the general multidestination-based multicast worm. Basically, an i-reserve worm is an augmented m-ireq worm. In addition to delivering the invalidation request message, it also *reserves* an i-ack entry at the router interface of each destination. When a sharer receives an invalidation request, it invalidates its cache block and sets the i-ack signal in the reserved i-ack entry. Later, when the associated i-gather worm arrives, it collects the i-ack signal and releases the i-ack entry.

4.3 MR-MA Framework

We glue the above three mechanisms together to make a *Multidestination-based invalidation Reservation and Multidestination-based gather Acknowledgment (MR-MA)* framework. For this framework to function smoothly, we need to investigate several important issues closely.

4.3.1 I-ack Hits and I-ack Misses

In our proposed design, there are only a few i-ack entries at the router interface. When an i-reserve worm reaches the router interface, it is possible that there is no available i-ack entry. What should the MR-MA framework do on such scenarios? When such a scenario occurs (denoted as *i-ack miss*), we allow the i-reserve worm to deliver the invalidation request and keep moving. Under such circumstances, after receiving and processing the invalidation request, the sharer sends a unicast-based invalidation acknowledgment message. Later when the corresponding i-gather worm reaches this sharer, it can find out that there is no i-ack entry reserved for the block and proceeds to the next sharer/home node. We define an *i-ack hit* as a success in reserving an i-ack entry by an i-reserve worm. In section 6, we will show simulation results corresponding to i-ack hits/misses with varying number of i-ack entries. It is to be observed that 2-4 entries are sufficient to have a hit ratio higher than 98.7%.

4.3.2 I-ack Signal Counts

Once i-ack misses occur, it becomes impossible for the home node to decide when an invalidation transaction completes, based on the number of acknowledgment messages it has received. Hence, we incorporate an i-ack counter field in the i-gather worm. This field can be used to remember how many i-ack signals that an i-gather worm actually collects. For a unicast-based acknowledgment, this count is assumed to be 1. At the beginning of an invalidation, from the directory entry for a memory block, the home node knows the total number of sharers (denoted as x). On receiving the invalidation acknowledgments for the memory block, the home node decrements x by the i-ack count in the acknowledgments. When x becomes 0, the invalidation transaction is completed.

Let us apply this MR-MA framework to the example invalidation transaction shown in Fig. 2. Again, only the portion relating to sharers along column 6 is illustrated in Fig. 6 for clarity. Home node (3,3) sends two i-reserve worms. As the i-reserve worms propagate, let us assume that only one i-ack miss occurs at sharer (6,1). The i-gather worms later collects two i-ack signals each. (Special unicast-based acknowledgment messages are needed in an e-cube system because the i-gather worms can not reach the home node under X-Y routing.) Independently, sharer (6,1) sends a unicast-based acknowledgment back to the home node. Overall, assuming the best-case where only i-ack hits occur, it can be observed that for the example invalidation transaction in Fig. 2, the home node needs to send and receive only 22 invalidation worms in the MR-MA framework as compared to 46 worms in the UI-UA framework or 34 worms in the MI-UA framework. This demonstrates further reduction in volume of network traffic, occupancy at the home node, and most importantly the invalidation latency.

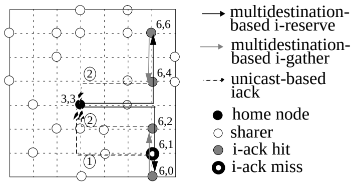


Figure 6: An example of a portion of invalidation using the MR-MA framework.

Other issues in this MR-MA framework, like deadlock-freedom, vitual cut-through, etc., have also been investigated. Interested readers are encouraged to refer to [1] for details.

5 Grouping Schemes

It can be seen that i-reserve and i-gather worms must be generated on-the-fly by the directory controller because of the dynamic property of sharing. Let us define *grouping* as the procedure of selecting a set of i-reserve and i-gather worms to cover all sharers of a memory block. In this section, we propose and analyze some heuristic grouping schemes for e-cube routing and turn-model routing. All grouping schemes are illustrated with the example invalidation pattern introduced in Fig. 2.

5.1 Associated Issues

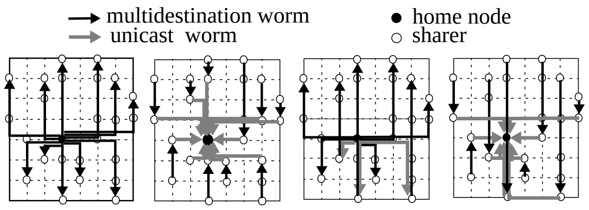
Directory Organization and Grouping for I-reserve Worms: It can be very helpful if the pointer array is so organized that when an invalidation transaction occurs, the home node can send the i-reserve worms with little overhead. Assume that the fully-mapped protocol is used to enforce coherence. Let us consider organizing the presence bits in the column major order and use the bit string address encoding for multidestination worms [9]. Hence, different portions of the presence bits can be directly taken as the routing headers of the i-reserve worms.

Grouping for I-gather Worms: In all the schemes we discuss, the final destination node of an i-reserve worm initiates an i-gather worm to collect the i-ack signals. This node needs the identifiers of the sharers that are covered by the associated i-reserve worm. To satisfy such a requirement, a copy of the routing header of the i-reserve worm can be duplicated and carried as an additional data item in the worm. This may increase the length of an i-reserve worm slightly. However, the delay induced by such an increase in worm length is very small under wormhole routing. After the final destination of an i-reserve worm receives the message, its directory controller does a grouping with itself as the source node and the home node as the final destination in order to generate appropriate routing header for the corresponding i-gather worm. Readers are encouraged to refer to [1] for further details.

5.2 E-cube Routing

5.2.1 Up-and-Down Column Grouping (UD) Scheme

In an *up-and-down column grouping* (UD) scheme, a home node needs maximum two i-reserve worms in each column for an invalidation transaction. One such worm, the *up i-reserve* worm, moves along the +Y direction after a possible turn from ±X. The complementary *down i-reserve* worm moves along the –Y direction. A sharer in a column locating on the row containing the home node can be covered by either worm. An associated i-gather worm visits the sharers in the reverse order of the i-reserve worm along the ±Y direction to collect the i-ack signals. The i-ack signal count is finally carried back by a unicast-based worm to the home node. Figures 7(a) and 7(b) illustrate the request and acknowledgment phases using the UD scheme. As discussed in section 2.2, we consider two e-cube routed virtual networks, i-reserve worms moving in one and i-gather worms in the other.



(a) ud_ireq (b) ud_iack (c) sc_ireq (d) sc_iack

Figure 7: E-cube routing grouping schemes

To do a performance analysis similar to that for the UI-UA framework, we use similar assumptions, as described in section 2.3. In addition, let us assume the startup time for multidestination-based messages as α_m. Assume d_{ud} i-reserve worms are initiated at the home node on average. Thus, the best-case invalidation latency, $T_{inv,ud}$, can be derived as $\alpha+(d_{ud}+1)\alpha_m+2h_{ud}\beta+2(l_{ud}-1)\gamma+2\delta$ and the total number of hops traversed by the messages, $H_{inv,ud}$, as $3d_{ud}h_{ud}$.

5.2.2 Selective Column Leader Grouping (SC) Scheme

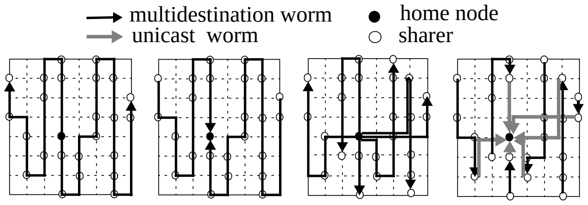
In a *selective column leader grouping* (SC) scheme, a home node initiates an up or down i-reserve worm as in the UD scheme if it can cover all the sharers in a column. Otherwise, the home node initiates a unicast-based worm to a selected column leader. The leader is the highest or lowest sharer in a column whichever is closer to the home node. Figures 7(c) and 7(d) illustrate the request phase and the acknowledgment phase using the SC scheme. In the SC scheme, for each column, the home node sends maximum one worm. Compared to the UD scheme, this scheme achieves a better balance between the occupancy of the home node and the propagation delay of the invalidation request than the UD scheme. Assume $d_{sc,u}$ unicast-based worms and $d_{sc,m}$ multidestination-based worms are initiated at a home node on average for an invalidation. This leads to: $T_{inv,sc}=(d_{sc,u}+1)\alpha+(d_{sc,m}+1)\alpha_m+2h_{sc}\beta+2(l_{sc}-1)\gamma+2\delta$ and $H_{inv,sc}=3(d_{sc,u}+d_{sc,m})h_{sc}$.

5.3 Turn Model Adaptive Routing

It is to be noted that any e-cube grouping scheme can be used in networks supporting turn-model routing [8]. However, we discuss two new grouping schemes to exploit the adaptivity better. Similar to e-cube routing, we consider two virtual networks, one supporting west-first routing and the other east-first.

5.3.1 Dual-Path Grouping (DP) Scheme

In this scheme a home node needs maximum two i-reserve worms for an entire invalidation transaction. One such worm, the *left i-reserve* worm, covers all the sharers on the left side of the home node and travels in the network using east-first routing. The complementary *right i-reserve* worm covers all the sharers on the right side and travels in the network using west-first routing. The sharers along the column containing the home node can be divided into two half columns: upper and lower with respect to the home node. Each of the half column, but not both, can be covered by either of the i-reserve worms. This scheme achieves the greatest reduction in volume of network traffic among all the proposed grouping schemes. Assume d_{dp} i-reserve worms are initiated at a home node on average for an invalidation. This leads to: $T_{inv,dp}=\alpha+(d_{dp}+1)\alpha_m+2h_{dp}\beta+2(l_{dp}-1)\gamma+2\delta$ and $H_{inv,dp}=2d_{dp}h_{dp}$. Figures 8(a) and 8(b) illustrate the request phase and the acknowledgment phase using the DP scheme.



(a) dp_ireq (b) dp_iack (c) dcg_ireq (d) dcg_iack

Figure 8: Turn model routing grouping schemes

5.3.2 Density-dependent Column Grouping (DCG) Scheme

Two major drawbacks of the DP scheme are: 1) high latency and 2) complicated routing headers for the multidestination worms. A *density-dependent column grouping* (DCG) scheme tries to balance the number of destinations covered by each worm. In addition to statically partitioning the network into c adjacent columns each, this scheme restricts maximum destinations a multidestination-based worm can cover in the c columns to a certain threshold value (d). When the threshold is reached, more than one i-reserve worms are used to cover destinations in the c columns. Figures 8(c) and 8(d) illustrate the request phase and the acknowledgment phase of the invalidation transaction in the DCG scheme with $c = 2$ and $d = 6$. Assume d_{dcg} i-reserve worms are initiated at a home node on average for an invalidation. This leads to: $T_{inv,dcg}=\alpha+(d_{dcg}+1)\alpha_m+2h_{dcg}\beta+2(l_{dcg}-1)\gamma+2\delta$ and $H_{inv,dcg}=3d_{dcg}h_{dcg}$.

Table 1: Comparison of different grouping schemes.

grouping scheme	home occup.	avg dist.	# of mesg	total hops
`unicast`	46	4.0	46	186
`d_ud`	22	4.5	31	99
`d_sc`	16	6.6	32	106
`t_dp`	4	22.0	4	88
`t_dcg`	12	8.5	18	102

5.4 Comparison of Grouping Schemes

We compared these schemes by using the four measurements proposed earlier in section 2. For a fair comparison across different schemes, if worms traveled in a chained fashion, they are counted as one message with a distance equivalent to the sum of distance traveled by component worms; while in the calculation of the number of total messages they are counted individually. Table 1 summarizes the results for the sample invalidation transaction, introduced in Fig. 2. A simple observation from this table is that all the proposed schemes cut down the volume of network traffic for the invalidation effectively with respect to unicast scheme. Furthermore, it can be observed from Table 1 and derived from earlier latency expressions that the SC scheme is more efficient than the UD scheme while the DCG scheme is more efficient than the DP scheme.

Table 2: System parameters used in simulation.

Parameter	Values	Time
Processor	1 cycle	5 ns
Cache access	1 cycle	5 ns
Cache block size	16 bytes	
Cache block fill time	8 cycles	40 ns
Cache set associativity	1	
Cache size per node	64 Kbytes	
Memory word width	4 bytes	
Memory block access time	8 cycles	40 ns
Directory check	2 cycles	10 ns
Directory check and update	4 cycles	20 ns
mesg startup (unicast)	5 cycles	25 ns
mesg startup (multidest)	10 cycles	50 ns
mesg dispatch	2 cycles	10 ns
Control mesg size (unicast)	4 bytes	
Control mesg size (multidest)	6 bytes	
Data message size	20 bytes	
Injection channels per node	2	
Consumption channels per node	4	
I-ack entries per router	4	
Channel width (Flit size)	2 bytes	
Link propagation delay	1 cycle	5 ns
Router switch delay	1 cycle	5 ns
Router delay (header)	4 cycles	20 ns

6 Simulation Experiments and Results

We evaluated our schemes for a set of different applications using a CSIM-based execution-driven simulation environment. The environment accurately models intranodal and internodal communication, memory access contention, network contention, and port contention.

6.1 Simulation Setup

System Parameters: The following parameters were assumed: 100 MHz processor, 200 Mbytes/sec communication link and 20.0 nanosec router delay. For unicast-based message-passing we assumed 5 processor clock cycles as the startup time. For multidestination messages this time was assumed to be 10 clock cycles. We considered an 8×8 system. Table 2 lists the relevant system parameters used.

Derived Memory Latencies: Assuming no contention, Table 3 lists the derived latencies of common memory operations from our simulation. Table 4 shows the break down of the derived latencies of a clean read-miss to neighboring node. These results are very comparable with the results reported on the DASH, Alewife, and FLASH.

Table 3: Memory miss latencies on our system (in cycles).

Access type	Home location	# Inv CpyBk	Lat.	Dir State
Load	local	0	13	
	remote	0	65	
	rmt(2-party)	1	74	home excl
	rmt(3-party)	1	121	rmt excl
Store	local	0	13	
	local	1	79	rmt shrd
	rmt	0	65	rmt shrd
	rmt(2-party)	1	70	home shrd
	rmt(3-party)	1	102	rmt shrd

Application Characteristics: We considered three different applications. The first application, Barnes-Hut, was ported to our simulator from Stanford SPLASH2 benchmark suite. We simulated 128 bodies for 4 time steps. We also ported the blocked LU-decomposition kernel from Stanford SPLASH2 benchmark suite. We simulated 128×128 matrices with 8×8 blocks. The third application we chose was All Pairs Shortest Path (APS). The APS kernel was developed locally by our research group. It uses the Floyd-Warshall algorithm to compute all pairs shortest paths problem on an input matrix. We used the conventional in-place variation. We simulated 128×128 matrices with 50% of links not connected (infinite link cost).

Table 4: Break down of clean read-miss latency (in cycles).

Suboperation	Lat.
Cache-miss to request in the network	6
Request transmit time (4 bytes)	9
Request arrives home to response transmit	14
Data return in network (20 bytes)	25
Response arrival to cache fill	3
Cache fill	8
Total	65

6.2 Results and Discussions

We observed three important parameters: number of messages used for invalidation, total network messages, and overall execution time. In order to perform comparative evaluation, we used dimension order system with unicast-based message passing (d-uni) as the base case (100%). All other results are compared and reported against this base case.

Figure 9 shows the number of invalidation messages used. It can be seen that all multidestination schemes reduce the number of messages required for invalidation compared to the unicast-based (uni) message-passing. Turn-model with dual-path grouping (DP) always leads to maximum reduction (up to 95%).

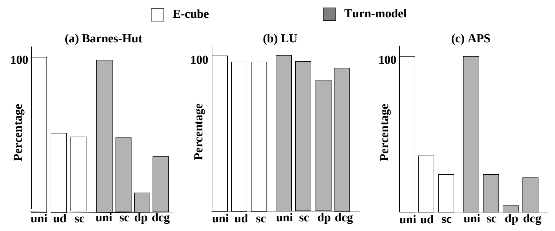


Figure 9: Comparing number of invalidation messages required under unicast-based and multidestination message passing.

Figure 10 shows the total number of network messages used. It can be observed that reduction in the total network messages varies significantly depending on the ratio of invalidation messages and other messages. For application like LU-decomposition which exhibits very low number of invalidations, the gain is very minimal (0.6% at the best). For other two applications, different grouping schemes lead to 20-40% reduction.

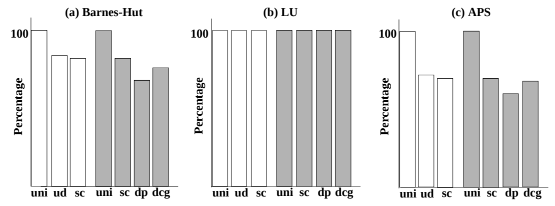


Figure 10: Comparing total number of messages.

Figure 11 shows the overall execution time. It can be observed that substantial reduction in network messages do not necessarily lead to reduction in overall execution time. This depends on the critical path of execution. However, some grouping schemes like SC, DCG are able to reduce overall execution time up to 15%.

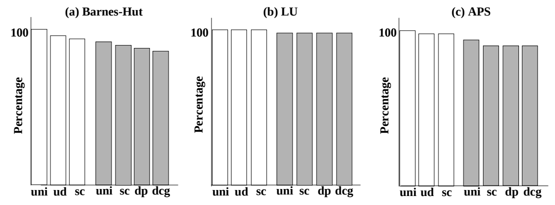


Figure 11: Comparing overall execution time.

To understand the impact of i-ack buffer size on system performance, we ran Barnes-Hut with different number of i-ack entries at each router interface. Figure 12(a) shows the normalized of i-ack misses over the total number of reservation attempts. Figure 12(b) shows the normalized overall execution time. It can be observed that with only few i-ack entries (2-4) i-ack misses can be dramatically reduced without apparent execution-time loss.

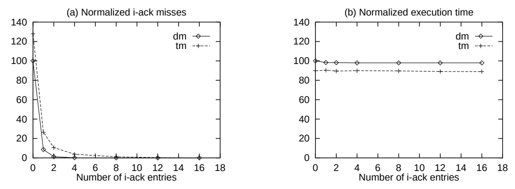


Figure 12: The impact of i-ack buffer size in Barnes-Hut.

7 Conclusions

In this paper we have introduced a new multidestination message-passing approach to implement directory-based cache coherency in wormhole distributed shared memory (DSM) systems. By applying the BRCP model, reservation and gather worms were used to distribute invalidation messages and to collect acknowledgments. Compared to the conventional approach, this new method produces less number of messages, less network traffic, and reduced occupancy at home nodes. New grouping schemes were proposed to generate these multidestination worms to implement the fully-mapped protocol. Simulation results demonstrate considerable potential for these schemes to be applied to current generation DSM systems. Currently we are investigating limited directory organizations with this new approach.

Readers are encouraged to refer to home page *http://www.cis.ohio-state.edu/˜panda/pac.html* for related papers.

References

[1] D. Dai and D. K. Panda. Reducing Cache Invalidation Overheads in Wormhole Routed DSMs Using Multidestination Message Passing. OSU-CISRC-4/96-TR24, 1996.

[2] A. Agarwal et al. The MIT Alewife Machine: Architecture and Performance. *ISCA'95*, pp.2–13.

[3] C. Holt et al. The Effects of Latency, Occupancy, and Bandwidth in Distributed Shared Memory Multiprocessors. CSL-TR-95-660, Stanford University, 1995.

[4] J. Kuskin et al. The Stanford FLASH Multiprocessor. *ISCA'94*, pp.302–313, 1994.

[5] S. K. Reinhardt et al. The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers. *SIGMETRICS'93*, pp.48–60.

[6] P. Mannava, A. Kumar, and L. N. Bhuyan. Cache Coherence Architecture for Large Scale Multiprocessors. *5th SSMM Workshop, ISCA'95*.

[7] P. K. McKinley and D. F. Robinson. Collective Communication in Wormhole-Routed Massively Parallel Computers. *Computer*, pp.39–50, 1995.

[8] L. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *Computer*, pp.62–76, Feb. 1993.

[9] D. K. Panda. Fast Barrier Synchronization in Wormhole k-ary n-cube Networks with Multidestination Worms. *Future Generation Computer Systems*, 11:585–602, Nov 1995.

[10] D. K. Panda, S. Singal, and P. Prabhakaran. Multidestination Message Passing Mechanism Conforming to Base Wormhole Routing Scheme. *PCRCW'94*, pp.131–145.