Multiple Multicast with Minimized Node Contention on Wormhole k-ary n-cube Networks

Ram Kesavan and Dhabaleswar K. Panda

Technical Report
OSU-CISRC-4/96-TR26

A preliminary version of this paper has been presented in International Conference on Parallel Processing, Aug. 1996. Manuscript is under review by *IEEE Transactions on Parallel and Distributed Systems.*

Multiple Multicast with Minimized Node Contention on Wormhole k-ary n-cube Networks [1]

Ram Kesavan and Dhabaleswar K. Panda

Dept. of Computer and Information Science
The Ohio State University, Columbus, OH 43210-1277
Tel: (614)-292-5199, Fax: (614)-292-2911
E-mail: {kesavan,panda}@cis.ohio-state.edu

Contact Author: Dhabaleswar K. Panda

Abstract: This paper presents a new approach to minimize node contention while performing multiple multicast/broadcast on wormhole k-ary n-cube networks with overlapped destination sets. The existing multicast algorithms in the literature deliver poor performance under multiple multicast because these algorithms have been designed with only *single multicast* in mind. The new algorithms introduced in this paper do not use any global knowledge about the respective destination sets of the concurrent multicasts. Instead, only local information and source-specific partitioning approach are used. For systems supporting unicast message-passing a new SPUmesh (*Source-Partitioned Umesh*) algorithm is proposed and shown to be superior than the conventional Umesh algorithm [14] for multiple multicast. Two different algorithms, SQHL (*Source-Quadrant Hierarchical Leader*) and SCHL (*Source-Centered Hierarchical Leader*), are proposed for systems with multidestination message-passing and shown to be superior than the HL scheme [18]. All these algorithms perform 1) 5-10 times faster than the existing algorithms under multiple multicast and 2) as fast as existing algorithms under single multicast. Furthermore, the SCHL scheme demonstrates that the latency of multiple multicast can, in fact, be *reduced* as the degree of multicast *increases* beyond a certain number. Such results related to multiple multicast are the first of their kind in the wormhole literature. Thus, these algorithms demonstrate significant potential to be used for designing fast and scalable collective communication libraries on current and future generation wormhole systems.

Keywords: Collective communication, multicast, broadcast, wormhole routing, meshes, k-ary n-cube, unicast and multidestination message passing, and path-based routing.

[1]A preliminary version of the paper [10] has been presented in *International Conference on Parallel Processing*, Aug 1996. This research is supported in part by NSF Grant MIP-9309627 and NSF Career Award MIP-9502294.

Contents

1 Introduction

The wormhole-routing switching technique is becoming the trend in building future parallel systems due to its inherent advantages like low-latency communication and reduced communication hardware overhead [7, 16]. IBM SP1/SP2, Intel Paragon, Cray T3D, Ncube, J-Machine, and Stanford DASH are representative systems falling into this category. Gradually variations of this technique, knows as cut-through routing, are being used in networks of workstations, like Myrinet [3] as well. Distributed-memory or distributed-shared memory programming paradigms are supported on these systems. For efficient support of the above paradigms, these systems require fast implementation of collective communication operations (*broadcast*, *multicast*, *global combine* and *barrier synchronization*) [15].

Multicast/broadcast is a common collective communication operation in parallel systems [13, 16, 17]. Many multicast/broadcast algorithms have been proposed in the literature in recent years [2, 4, 6, 8, 9, 12, 14, 19]. These algorithms have been primarily developed for *single* multicast. However, *multiple multicast* operations (i.e. two or more multicasts executing simultaneously) occur frequently in parallel systems. Examples include cache-invalidation in distributed-memory systems, multiple broadcast in numerical and scientific applications (LU decomposition for example), multiple broadcast/multicast operations during concurrent barrier and reduction operations, etc. In these operations, destination sets of different multicasts often overlap, leading to nodes participating concurrently in multiple multicasts.

Efficient multicast algorithms are typically hierarchical in nature, i.e. some destinations serve as intermediate sources and forward copies of the message to other destinations, when they receive it. Typically, tree-based algorithms are used to minimize the number of communication start-ups (steps) required for multicast [1, 5, 11, 14]. These algorithms are also generally contention-free for single multicast.

In order to implement multiple multicast in current parallel systems, each source node uses the same algorithm designed for single multicast and constructs its multicast tree independently. With overlapped destination sets, such construction of trees may result in *node contention*. This may significantly affect the overall latency of the multiple multicast operation. Providing global knowledge of different multicasts (the source and destinations) to every source is a possible solution. Thus, every source can construct its tree optimally, in order to minimize node-contention. However, exchange of such global knowledge is not realistic for dynamic multicast operations like cache-coherency. Therefore, the challenge in solving the problem of multiple multicast is to design a multicast algorithm while *keeping multiple multicast in mind.* Such an algorithm must 1) provide an optimal solution for single multicast and 2) minimize node contention under multiple multicast, when every source uses this algorithm to construct its tree in the absence of any global knowledge.

In this paper we take on these challenges. We develop efficient multicast algorithms on k-ary n-cube wormhole networks, which satisfy the above objectives. We consider systems with unicast as well as multidestination message-passing [16, 18]. First we identify and analyze the severity of node contention when algorithms designed for single multicast are applied to multiple multicast. We develop analytical models for estimating such node contention for completely and randomly overlapped destination sets. Through these analyses we conclude that node contention along with link contention can be reduced if some source-specific information can be used to construct multicast trees for different sources. Next, we develop algorithms for multiple multicast. For unicast-based systems, we propose a new SPUmesh (*Source-Partitioned Umesh*) algorithm, which uses source-position information to construct the multicast tree. This new algorithm is contention free and is as efficient as the Umesh algorithm [14] for single multicast, but clearly superior for multiple multicast. For systems supporting multidestination message-passing, we propose two different algorithms, SQHL (*Source-Quadrant-based Hierarchical Leader*) and SCHL (*Source-Centered Hierarchical Leader*). The SCHL algorithm uses source-position information as well as load-balancing techniques on network links along different directions to minimize multiple multicast latency. Both these algorithms are significantly better than the HL scheme [18] for multiple multicast. We also compare the best of both kinds of algorithms (SPUmesh and SCHL) to study the benefits of multi-destination message-passing over unicast message-passing for multiple multicast. It is demonstrated that, similar to the HL scheme [18] for single multicast, the SCHL scheme is also capable of *reducing* multiple multicast latency as the degree of multicast *increases.*

The rest of the paper is organized as follows. Section 2 discusses how contention leads to poor performance when existing multicast algorithms designed for single multicast are used for multiple multicast. Section 3 presents the SPUmesh algorithm and evaluates its node-contention property. Section 4 proposes the SQHL and the SCHL schemes for systems supporting multidestination message-passing. Simulation experiments and results, comparing the relative merits of the existing and new algorithms, are presented in Section 5. Concluding remarks are made in Section 6.

2 Contention in Existing Multicast Algorithms

In this section we describe how node and link contention arises when existing multicast algorithms are used to implement multiple multicast. The Umesh algorithm [14], using unicast message passing and the HL scheme [18], using multidestination message passing, are examined. Analytical models to estimate node contention under multiple multicast are developed. An approach to reduce contention using source-based information is outlined.

2.1 Node Contention

Let us consider multiple multicasts occuring simultaneously in a system, with each multicast having a unique source and a set of destinations. The destination sets of various concurrent multicasts may overlap. Efficient multicast algorithms are typically hierarchical in nature [14, 18], where some intermediate destinations are used to forward the message to other destinations in the set. Therefore, a node participating in multiple multicasts may need to forward multiple messages during a given period of time. This leads to *serialization* of the associated communication start-ups, which is defined as *node contention.* Depending on the number of overlapped nodes and the degree of the overlapping, one or more nodes may fall into the *critical path.* The overall latency of the multiple multicast pattern will depend on these nodes and the degree of node contention they experience. Let us analyze how node contention affects the latency of multiple multicasts while using the Umesh algorithm [14] and the HL scheme [18].

2.2 Contention in Existing Algorithms

2.2.1 The Umesh Algorithm

The Umesh algorithm [14] uses the concept of a dimension-ordered chain (Φ) on meshes that use e-cube routing [7]. The chain is a total ordering of the elements of the set which consists of the source and all the destination nodes of the multicast. The ordering ensures zero contention for links between messages of a multicast. However, this does not prevent messages belonging to different concurrent multicasts from contending for some nodes or links. Figure 1(a) shows how a multicast message propagates within the dimension-ordered chain Φ. The node to receive the first message is positioned half-way in the chain, and is called the *half-node*. The algorithm recursively identifies *quarter-nodes*, *one-eighth-nodes* and so on.

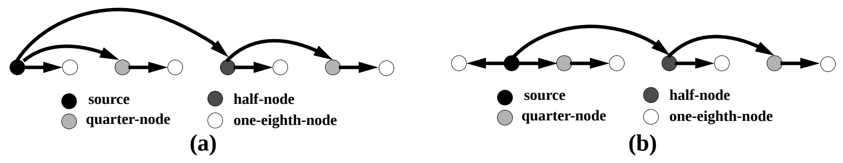

Figure 1: Multicast message pattern using the Umesh algorithm for (a) multicast A and (b) multicast B. The sources, half-nodes and quarter-nodes are highlighted.

Let us consider two multicasts A and B with identical source-destination sets and the same dimension-ordered chain Φ. Also, let us assume that both their sources lie in the lower half of Φ. Figures 1(a) and 1(b) show that both multicasts, to a large extent, share the same half-node, quarter-nodes, one-eighth-nodes and so on. The common half-node for A and B has to sequentialize the four message start-ups that it undergoes. Similarly, if several multicasts have (nearly) identical Φ's, these multicasts need to share the same nodes at the key positions along the chain, leading to hot spots. In the worst case of multiple multicast: *many-to-all broadcast*, each broadcast has the same Φ. Therefore, all the sources choose the node half-way in the chain to send their first messages to, the node quarter-way in the chain to send their second messages to, and so on. This leads to severe node contention and high latency for the multiple multicast pattern.

Figure 2(a) shows two multicasts, A and B, with identical Φ. Their sources are (3,3) and (4,1), respectively. The dimension-ordered chain Φ is shown in Fig. 2(b). Figures 2(c) and 2(d) show the multicast trees of A and B, respectively. It can be observed that nodes (2,2), (3,2) and the nodes in the subtrees rooted at these nodes experience contention from both multicasts. These nodes sequentialize the multicasts, which results in larger multiple multicast latency. For a large number of such concurrent multicasts, such sequentialization can significantly affect their latencies. A detailed analysis of the node contention while using the Umesh algorithm follows in Section 2.3.1.

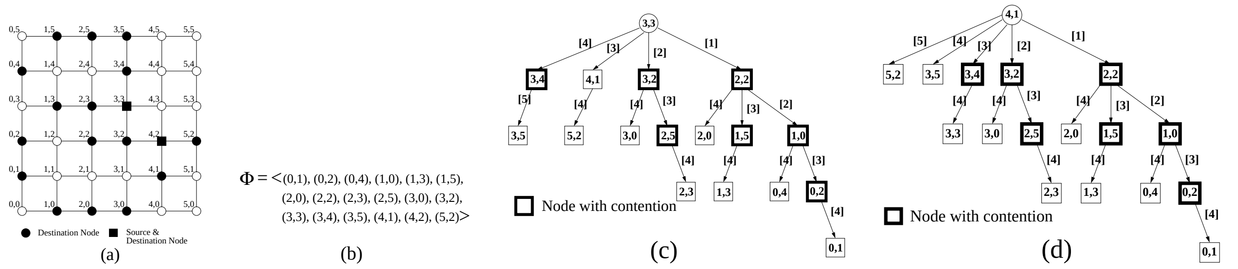
● Destination Node
■ Source & Destination Node
(a)
Φ = <(0,1), (0,2), (0,4), (1,0), (1,3), (1,5), (2,0), (2,2), (2,3), (2,5), (3,0), (3,2), (3,3), (3,4), (3,5), (4,1), (4,2), (5,2)>
(b)
Node with contention
(c)
Node with contention
(d)

Figure 2: Node contention between two concurrent multicasts using the Umesh algorithm: (a) two multicasts A and B on a 6x6 mesh, (b) both multicasts sharing the same dimension-ordered chain Φ, (c) the multicast tree for A with source at (3,3), and (d) the multicast tree for B with source at (4,1). The nodes with contention are highlighted.

2.2.2 The HL Scheme

Multicasting with multidestination message passing while using the HL scheme [18] also has node contention problem for multiple multicast. Let us consider two concurrent multicasts, A and B, with identical source-destination sets on a 2D mesh. Let their respective sources be at (2,3) and (4,4), as shown in Fig. 3. It can be observed that with multidestination message passing A and B have the same level 2 and level 1 leader sets (L_2 and L_1) [18]. Sequentialization of worms (unicast as well as multidestination) corresponding to multicasts A and B at the L_2 and L_1 nodes leads to high latency for the overall multiple multicast operation. With this HL scheme, a large number of concurrent multicasts with overlapping leader sets leads to higher node contention. Consider the worst case of S concurrent broadcasts. The L_2 sets for all the broadcasts is the same singleton set consisting of a corner node. This node undergoes two start-ups per broadcast. This, this will result in the sequentialization of $2S$ message start-ups at the corner node, leading to a significant increases in the multiple multicast latency. A detailed analysis of the node contention while using the HL scheme follows in Section 2.3.2.

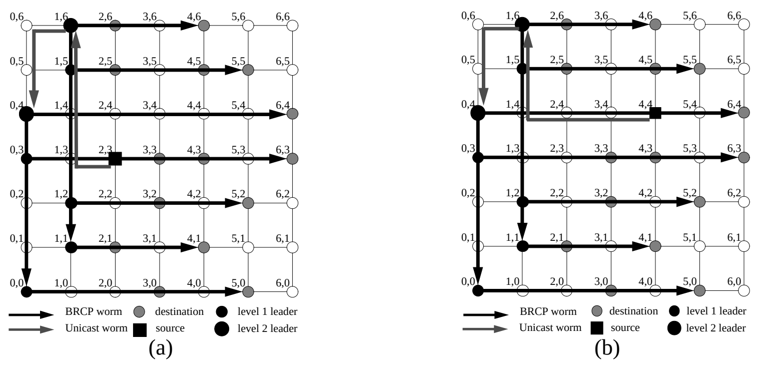

Figure 3: Node contention between two concurrent multicasts, with identical source-destination sets, using the HL scheme: (a) communication pattern for multicast A with source at (2,3) and (b) communication pattern for multicast B with source at (4,4).

2.3 Estimating Contention in Existing Algorithms

We develop analytical models to estimate the contention experienced by nodes while using the existing algorithms in a multiple multicast pattern. The contention directly affects the latency of the multiple multicast pattern.

2.3.1 The Umesh Algorithm

Let us analyze an estimate of the node contention when using the Umesh algorithm for multiple multicast. We consider two cases: (a) worst case scenario of total overlap of source-destination sets and (b) random overlap of source-destination sets. In reality, concurrent multiple multicasts of an application could lie anywhere in the range of random to complete overlap of source-destination sets. In current generation wormhole systems a communication step is dominated by message start-up cost. Hence, in this section, and in analytical estimates done is later sections, we evaluate node contention and multipe multicast latency in terms of steps (message start-ups). However, in Section 5 we consider both, message start-up and propagation costs when presenting simulation results.

Complete Overlap of Source-Destination Sets: Consider a k-ary n-mesh with S multicasts occuring simultaneously. Let all the multicasts have the same source-destination set, Φ, and unique sources. Let $|\Phi| = D$. It can be observed that such complete overlap of source-destination sets is possible when $S \leq D$. Under such overlapping, the node undergoing the most number of start-ups decides the lower bound on the latency of a multiple multicast pattern. This leads to:

Theorem 1 *The latency of a multiple multicast pattern with complete overlap of source-destination sets (S sources and $(D-1)$ destinations for each source with identical Φ, $S \leq D$) while using the Umesh algorithm is given by $(S\lceil log_2(\frac{D+1}{2})\rceil+1)$ steps for odd D, and is bounded below by $(\frac{S}{2}.a_m+b_m)$ steps for even D, where*

$$a_i = \frac{2^i-1}{2^{i-1}}\lceil log_2\frac{D}{2^i}\rceil \text{ and}$$

$$b_i = \begin{cases} \lceil log_2 D\rceil & \text{if } S = D \\ i & \text{if } S < D \end{cases} \quad \text{and}$$

$$m \text{ is such that } \forall i \; a_m \geq a_i$$

Proof: Let $\Phi = < d_0, d_1, \ldots, d_{D-1} >$. Let d_s be a source node in one of the multicasts. For odd D, d_s chooses $d_{\frac{D}{2}}$ as the half-node independent of its relative position in Φ, except when $s = \frac{D}{2}$. If $d_{\frac{D}{2}}$ is one of the sources, it undergoes $\lceil log_2\frac{D}{2}\rceil$ start-ups as the half-node for each of the other $S-1$ multicasts, and an additional $\lceil log_2\frac{D}{2}\rceil$ start-ups as a source. If $d_{\frac{D}{2}}$ is not one of the sources, it undergoes $\lceil log_2\frac{D}{2}\rceil$ start-ups for each of the S multicasts, and an additional start-up is needed for the first message to arrive at $d_{\frac{D}{2}}$. Therefore, for odd D, the latency of a multiple multicast pattern is given by $(S\lceil log_2(\frac{D+1}{2})\rceil + 1)$ steps (start-ups).

For even D, let us assume the best case of uniform distribution of sources in Φ. The node $d_{\lfloor\frac{D-1}{2}\rfloor}$ is chosen as the half-node in $\frac{S}{2}$ of the multicasts, $d_{\lfloor\frac{D-1}{4}\rfloor}$ is chosen as a quarter-node in $\frac{3S}{4}$ of the multicasts, and so on. Depending on the value of D, one of these nodes, say d_x, undergoes the maximum number of start-ups in the multiple multicast pattern. If $S < D$, for estimating the lower bound, let us assume the best case of d_x not being one of the sources. Therefore, if $d_x = d_{\lfloor\frac{D-1}{2}\rfloor}$, there is an extra start-up before the first message arrives at d_x, following which d_x undergoes $\frac{S}{2}\lceil\log_2\frac{D}{2}\rceil$ start-ups. Similarily, if $d_x = d_{\lfloor\frac{D-1}{2^i}\rfloor}$, there are i extra start-ups before the first message reaches d_x, following which d_x undergoes $\frac{(2^i-1)S}{2^i}\lceil\log_2\frac{D}{2^i}\rceil$ start-ups. The lower bound is given by that expression in which d_x undergoes the maximum number of start-ups. If $S = D$, then d_x undergoes an additional $\lceil log_2 D\rceil$ start-ups as a source. In this case, the lower bound is given by $\lceil log_2 D \rceil$ plus the maximum number of start-ups that d_x undergoes. Therefore, for even D, the latency of a multiple multicast pattern is bounded below by $(\frac{S}{2}.a_m + b_m)$ steps, where m is such that $\forall i \; a_m \geq a_i$. ■

A simple experiment was conducted to estimate the latency of multiple multicast (in terms of steps) using the Umesh algorithm with complete overlap of source-destination sets on an 8×8 mesh. In this experiment, all system parameters (propagation time, router delay, switching time etc) were ignored and only a high start-up time was considered. The experimental values were averaged over 30 runs and rounded off to the nearest integer. The third and seventh rows of Table 1 show the results of this experiment for odd and even D, respectively. The analytical values calculated by using Theorem 1 are shown in the fourth and eighth rows for odd and even D, respectively. It can be observed that the analytical values match the experimental values very closely, validating the results of Theorem 1. These results indicate that node contention becomes severe for a multiple multicast pattern as the number of sources and destinations increase.

Table 1: Experimental and analytical latency of multiple multicast (in number of steps) using Umesh on an 8×8 mesh for different values of S and (odd and even) D. Complete overlap of source-destination sets is considered ($S \leq D$).

S	1	1	1	15	15	15	31	31	47	63
odd $D = \lvert\Phi\rvert$	15	31	63	15	31	63	31	63	63	63
Experimental	4	5	6	46	61	76	125	156	236	316
Analytical	4	5	6	46	61	76	125	156	236	316
S	1	1	1	16	16	16	32	32	48	64
even $D = \lvert\Phi\rvert$	16	32	64	16	32	64	32	64	64	64
Experimental	4	5	6	28	42	57	77	105	154	198
Analytical	4	5	6	28	38	50	77	98	146	198

Random Overlap of Source-Destination Sets: Let us now consider the case of random overlap of source-destination sets. Since it is difficult to identify the node which undergoes the maximum node contention, we construct a lower bound on the node contention by calculating the expected number of startups sequentialized at a node using probabilistic analysis. Let us assume each node has equal probability of being in any position in the dimension-ordered chain (Φ) of a multicast. In reality, the nodes which lie in the middle of the mesh have a greater probability of being the half-node for a multicast with large D. Similarily, the nodes in other specific parts of the mesh have a higher probability of being specific nodes in a multicast with large D. Therefore, the values generated by using the equal probability assumption can serve as a lower bound to the multiple multicast latency.

Again, let us consider S multicasts occuring simultaneously, each with $|\Phi| = D$. An arbitrary node s which is a source in one of the multicasts (and hence undergoes $\lceil \log_2 D \rceil$ startups) can be a half-node, quarter-node, and so on, in the other $S-1$ multicasts. Let the terms E_2, E_4, E_8 and so on, denote the events of s being the half-node, quarter-node, one-eighth-node, respectively, in a multicast. Let E_0 denote the event of s not participating in a multicast, and let $A.B$ denote the events A and B occuring simultaneously. Now, each event has a fixed number of start-up cost associated with it. For example, E_2 has $\lceil \log_2 \frac{D}{2} \rceil$ steps associated with it, E_4 has $\lceil \log_2 \frac{D}{2} \rceil$, and so on. Under the equal probability assumption as stated above, we have:

Theorem 2 *The latency of a multiple multicast pattern with random ovelap of source-destination sets (S sources and $(D-1)$ destinations for each source with random Φ) while using the Umesh algorithm is bounded below by $\lceil \log_2 D \rceil$ steps plus the sum of the products of the probabilities of each term in the expression $(E_0 + E_2 + E_4 + E_8 + \ldots + E_D)^{S-1}$ occuring and the number of steps associated with each term.*

Proof: The terms in the expression $(E_0 + E_2 + E_4 + E_8 + \ldots + E_D)^{S-1}$ define all possible events for a source s. For example, the term $E_0^2.E_2^3.E_4^2.E_8$ represents the following event: s does not participate in 2 multicasts, is the half-node of 3 multicasts, the quarter-node of 2 multicasts and the one-eighth-node of 1 multicast. If $\wp(A)$ represents the probability of the event A occuring, then the expected number of start-ups incurred by s due to the event $E_0^2.E_2^3.E_4^2.E_8$ occuring is given by the expression $(2 \times 0 + 3\lceil \log_2 \frac{D}{2} \rceil + 2\lceil \log_2 \frac{D}{4} \rceil + \lceil \log_2 \frac{D}{8} \rceil)\wp(E_0^2.E_2^3.E_4^2.E_8)$. Therefore, the expected number of total start-ups incurred by s is $\lceil \log_2 D \rceil$ (as a source) plus the sum of the products of the probabilities of each term (in the above polynomial) occuring and the number of start-ups s incurs for the term. This number acts as a lower bound because of the assumption of each node having equal probability of being in any position in the Φ of a multicast. ■

For example, for values of $S = 4$ and $D = 8$, the expression is $(E_0 + E_2 + E_4 + E_8)^3$. The node s incurs 3 start-ups as a source, 2 as a half-node (E_2), 1 as a quarter-node (E_4), and none as a one-eighth-node (E_8). If s does not participate in a multicast (E_0) it incurs no start-ups. Let us consider the term $E_2^2.E_4$. The number of start-ups associated with the term is 5. The number of times the term appears in the above expression is 3. The product of the probability of the term and the associated start-ups is $3.5.\wp(E_2^2.E_4)$. Similarily, the products of the probabilities of other terms and their respective startups are $0\wp(E_0^3)$, $6\wp(E_2^3)$, $3\wp(E_4^3)$, $0\wp(E_8^3)$, $3.2\wp(E_0^2.E_2)$, and so on.

A small experiment was conducted to estimate the number of sequentialized steps (start-ups) that multiple multicast incurs for the Umesh algorithm with random overlap of source-destination sets on a 4×4 mesh. The smaller size was chosen to make the calculation of all the terms of the multinomial expression feasible. Like the previous experiment, all system parameters were ignored and only a high startup time was considered. The experiment values were averaged over 30 runs and rounded off to the nearest integer. Table 2 shows that the analytical values calculated by using the above theorem serve as good lower bounds to the experiment values.

Table 2: Experimental and analytical latency of multiple multicast (in number of steps) using Umesh on an 4×4 mesh for different values of S and D. Random overlap of source-destination sets is considered ($S \leq D$).

S	1	1	1	4	4	4	8	8	8	12	12	12	16	16	16
$D = \|\Phi\|$	4	8	16	4	8	16	4	8	16	4	8	16	4	8	16
Experimental	2	4	5	3	6	12	5	8	21	5	11	32	6	13	42
Analytical	2	3	4	2	4	6	2	4	7	2	4	11	2	7	16

2.3.2 The HL Scheme

Let us analyze an estimate of the node contention when using the HL scheme for multiple multicast on a k-ary n-mesh. We only consider the case of total overlap with identical source-destination sets in this section. The random overlap case is difficult to model and analyze. However, we study the impact of both random as well as total overlap on multiple multicast latency using simulations in Section 5.

Let us consider S multicasts with all multicasts having identical source-destination sets of size D. Such identical sets lead to identical leader sets. This leads to:

Theorem 3 *The latency of a multiple multicast pattern with complete overlap of source-destination sets (S multicasts and $(D-1)$ destinations for each source with identical source-destination sets, $S \leq D$) on a k-ary n-mesh using the HL scheme is $\leq (1 + S\lceil log_2 \frac{k}{2} \rceil + Sn)$ steps and is $\geq Sn$ steps.*

Proof: Based on a given source-destination set, let us consider a node x from the highest level leader set, L_g, $1 \leq g \leq n$ [18]. This node undergoes some start-ups when every source uses the Umesh algorithm to multicast to nodes of L_g. In the worst case, if x is the half-node for the Umesh phase of each multicast, x undergoes $\lceil log_2 \frac{|L_g|}{2} \rceil$ start-ups for each multicast. In the best case, x incurs no start-ups in the Umesh phase of each multicast. This occurs when $|L_g| = 1$ for high values of D. Following the Umesh phase, x undergoes a maximum of g start-ups for each multicast. This is to send out g multidestination worms to cover each set of which it is a leader. Therefore, the number of start-ups sequentialized at x lies between $(S\lceil log_2 \frac{|L_g|}{2} \rceil + Sg)$ and Sg. As D increases, the value of $|L_g|$ reduces from k to 1 and the value of g increases from 0 to n. The maximum possible value of $|L_g|$ is k, and that of g is n. Since it is very complex to analytically predict the values of $|L_g|$ and g for different values of S and D, we consider upper and lower bounds for this expression. For the upper bound, occuring for low values of D, we can take $|L_g| = k$ and $g = n$. The lower bound case occurs for high values of D where $|L_g| \approx 1$ and $g = n$.

Let us now estimate the latency of the multiple multicast pattern. If the node x is not one of the sources in the pattern, an extra start-up is added to account for the startup of the first message to arrive from a source to x. This gives us an upper bound of $(1 + S\lceil log_2 \frac{k}{2} \rceil + Sn)$ and a lower bound of $(1 + Sn)$ for the multiple multicast latency. On the other hand, if x is a source for one of the multicasts, and a half-node for the remaining $S - 1$ multicasts, we get a upper bound of $(\lceil log_2 k \rceil + (S-1)\lceil log_2 \frac{k}{2} \rceil + Sn)$, which is equal to $(1 + S\lceil log_2 \frac{k}{2} \rceil + Sn)$. If x is a source and also the sole member of all the L_g sets, it sends out Sg messages to give us a lower bound of Sn. Thefore, the upper bound for the the multiple multicast pattern is $(1 + S\lceil log_2 \frac{k}{2} \rceil + Sn)$ and the lower bound is Sn. ■

A small experiment was conducted to estimate the number of sequentialized steps that multiple multicast incurs for the HL scheme with complete overlap of source-destination sets on a 8×8 mesh.

Like previous experiments, all system parameters were ignored and only a high startup time was considered. The experimental values were averaged over 30 runs and rounded off to the nearest integer. The fourth row of Table 3 shows the results of this experiment. In this 2D mesh $n = 2$ and $k = 8$ giving an upper bound of $1 + 4S$ and a lower bound of $2S$. The upper bounds in Table 3 represent the worst case condition of $|L_g| = k$ and $g = n$, and hence act like a loose upper bound. Nevertheless, the experimental values fall between the two bounds and reflect the severity of node contention in multiple multicast.

Table 3: Experimental and analytical latency of multiple multicast (in number of steps) using the HL scheme on an 8×8 mesh for different values of S and D. Complete overlap of source-destination sets is considered ($S \leq D$).

S	1	1	1	16	16	16	32	32	48	64
D	16	32	64	16	32	64	32	64	64	64
Upper Bound	5	5	5	65	65	65	129	129	193	257
Experimental	5	5	3	39	43	33	84	64	96	128
Lower Bound	2	2	2	32	32	32	64	64	96	128

2.4 Link Contention

As observed above, when each source node uses the same single multicast algorithm to implement multiple multicast, the intermediate nodes of the multicasts might be identical leading to node contention. Even if they are not identical, they may lie in close proximity with each other. When these closely bunched intermediate nodes send out messages as part of their respective multicasts, it leads to *link contention.* This may also affect the overall latency of multiple multicast operation. The effect of link contention obviously increases as link propagation time becomes more of a dominant factor compared to start-up time. It also becomes dominant for long messages.

Thus, a good multicast algorithm should make sure that the intermediate nodes for various concurrent multicasts are different and well spread out over the network. The former condition reduces the node contention experienced, while the latter reduces link contention. In this paper we concentrate on reducing the node contention by making sure that the intermediate nodes for different multicasts are as different as possible. A side effect of the intermediate nodes being well spread out over the mesh is that it provides reduced link contention, as discussed in Section 4.

2.5 Improvement using Source Based Information

A method to reduce node contention is to make each multicast choose unique intermediate nodes, as different as possible from the rest. With dynamic multicast patterns, all concurrent multicasts are unaware of one another. This means that a multicast has no information whatsoever about the source and destinations of the other multicasts. Then, how can a multicast choose unique intermediate nodes different from those of other multicasts?

A good multicast algorithm should use some local information to make its tree as unique as possible. The local information that our new algorithms use is the position of the source in the system. This is unique for each multicast. If each multicast constructs its tree based on the position of its corresponding source, then all the multicasts will end up with trees as unique as possible. In current (Umesh and HL) algorithms the multicast trees are generated using primarily the destination (distribution) information. They depend, to a very small degree, on the position of the source. In our new unicast-based algorithm we use the same dimension-ordered chain concept as the Umesh algorithm, but use the position of the source in the chain to decide the multicast tree. In our multidestination-based algorithms we choose the leader sets for a multicast depending on the position of its source. We present these new algorithms in the following sections.

3 Source-Partitioned-Umesh (SPUmesh) Algorithm using Unicast Messages

In this section we propose a new *Source Partitioned Umesh* (SPUmesh) algorithm, to implement efficient multiple multicast using unicast messages. For single multicast, the algorithm performs as well as the Umesh algorithm by preserving link contention-free property. For multiple multicast, the new algorithm performs far better than the Umesh algorithm by reducing node contention. First, the algorithm is presented formally and its link contention-free property is proved for single multicast. Then its reduced node contention property for multiple multicast is demonstrated.

3.1 The Algorithm

As the name suggests, the Source Partitioned Umesh algorithm partitions the dimension ordered chain according to the position of the source in the chain. Let the source and destination addresses be sorted into a dimension ordered chain, Φ. A new chain Φ_1 is obtained by a *rotate-left* operation on Φ till the source s shifts to the beginning of Φ_1. Now the Umesh algorithm is performed on Φ_1. The algorithm is formally presented in Fig. 4. Figure 7(a) shows an example of a multicast tree generated when the SPUmesh algorithm is used for multicast A from Fig. 2(a).

```
SPUmesh Algorithm

  Input: $\Phi = < D_0, D_1, \ldots, D_n >$, the dimension ordered chain; $D_{source}$, the source node
  Output: Send the message to all destinations
  Procedure:
      Let $D_i = D_{source}$ where $D_i \in \Phi$
      Create $\Phi_1 = < D_i, D_{i+1}, \ldots, D_n, D_0, D_1, \ldots, D_{i-1} >$
      Apply the Umesh algorithm on $\Phi_1$ with $D_i$ as source
```

Figure 4: Outline of the SPUmesh algorithm.

The SPUmesh algorithm differs from the Umesh algorithm only with respect to the dimension ordered chain. Changing Φ to Φ_1 causes the multicast pattern to be dependent on the position of the source. This effect lowers node contention and latency for multiple multicast as compared to the Umesh algorithm. It is to be noted that a similar reordering of the dimension-order chain has been used in [20] to develop multicast algorithms for the torus network and in [6] for multipacket multicast. However, here we use such reordering on meshes to reduce node contention for multiple multicast. Since the multicast tree is generated in exactly the same way as in the Umesh algorithm, the SPUmesh algorithm takes no more than $\lceil \log_2 D \rceil$ start-ups for a single multicast. We prove below that the SPUmesh algorithm, like the Umesh algorithm, guarantees zero link contention among the messages for a single multicast.

3.2 Contention Free Property for Single Multicast

The Umesh algorithm guarantees zero link contention among messages of a *single* multicast [14] for wormhole systems with dimension ordering. Let us use the binary relation "dimension order", denoted as $<_d$, as defined in [14]. According to the definition, if $u <_d v <_d x <_d y$, then the path taken by a message from u to v and the path taken by a message from x to y are arc disjoint. Link contention occurs when two or more messages need to use the same physical link at the same time. We assume meshes with dimension-order X-Y routing for this proof.

Let us consider s to be the source and m to be the half-node in the SPUmesh algorithm. Without loss of generality, let us assume that $s <_d m$. Figure 5(a) shows the positions of s and m in the new dimension-ordered chain, Φ_1, and their positional relationaships with the original dimension-ordered chain, Φ. Figure 5(b) shows how the positions of s and m can be used to divide the mesh into three partitions. According to the new algorithm s sends the first message to m. Subsequently, s covers destinations in partition 1 and m covers destinations in partitions 2a and 2b. Let us define the messages for a single multicast on this system as follows:

Definition 1 *A message is defined as an intra-partition or inter-partition message depending on whether the message traverses within or across a partition.*

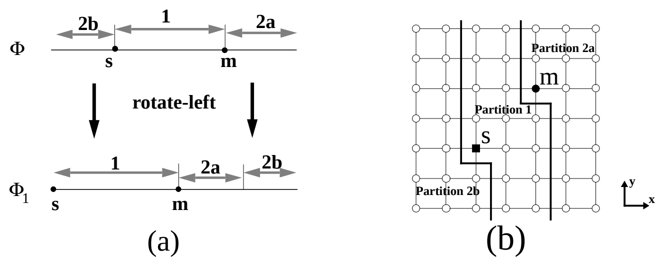

Figure 5: Positional relationships between a typical source node and the half-node: (a) transformation from the original dimension-ordered chain Φ to the new dimension-ordered chain Φ_1; and (b) division of the mesh into three partitions based on their positions.

It can be easily observed from the algorithm that after s sends the first message to m, all further messages are intra-partition messages except for those inter-partition messages sent from nodes in 2a to nodes in 2b. With respect to intra-partition messages, we have:

Lemma 1 *After s sends the first message to m, there is no link contention between any two intra-partition messages.*

Proof: Consider any of the three partitions. All intra-partition messages are sent strictly using the Umesh algorithm according to the dimension ordering. Therefore, intra-partition messages of a partition do not contend for links with each other. Obviously, intra-partition messages of two non-adjacent partitions cannot contend for links with each other. Let us consider intra-partition messages of two adjacent partitions. An intra-partition message can take no link in the x dimension which is completely contained in its adjacent partition. It can only take one x dimension link which straddles the inter-partition boundary, and this link cannot be taken by an intra-partition message of the adjacent partition. Also, an intra-partition message can take the y dimension links (of the common column) in the adjacent partition only in the negative direction, while intra-partition messages of the adjacent partition can take those y dimension links only in the positive direction. This proves that there exists no link contention between any two intra-partition messages. ■

This leaves us to concentrate only on the inter-partition messages which are sent from nodes in 2a to nodes in 2b. Let us define these messages as follows:

Definition 2 *An inter-partition message from partition 2a to partition 2b is called a back message.*

This leads to:

Lemma 2 *At any point of time, there is atmost only one back message in the system.*

Proof: Let there be two or more concurrent back messages. Let us take any two of these messages, m_1 and m_2. Let m_1 be from node x to node y and m_2 be from node u to node v. Therefore nodes x and u are in partition 2a, and nodes y and v are in partition 2b. Without loss of generality, let us assume $x <_d u$. This leads to only two possible situations as shown in Figure 6: (a) $x <_d u <_d v <_d y$ and (b) $x <_d u <_d y <_d v$. It can be proved that both cases never occur. If x is sending a message to y in a given step, then the set of nodes $\{i | (x <_d i) \vee (i <_d y)\}$ will receive the message only during subsequent steps. The node u lies in this set and thus cannot be sending a message to v in the same step. Therefore, there can be atmost only one back message being sent at any point of time. ■

Lemma 3 *There is no link contention between a back message and an intra-partition message belonging to partition 2a, 1, or 2b.*

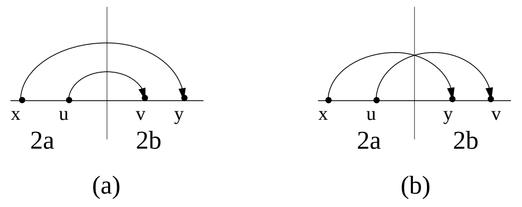

Figure 6: Possible cases for two simultaneous messages going from partition 2a to partition 2b.

Proof: Let us first consider an intra-partition message belonging to partition 2a. For dimensional X-Y routing, a back message takes X dimension links only in the negative direction. An intra-partition message going from node a to node b always has $a <_d b$ and hence, it takes X dimension links only in the positive direction. There is no contention in the Y dimension because Y dimension links are taken by the back message only after it reaches partition 2b.

Using the same reasoning as above, there is no contention in the X dimension betwen a back message and an intra-partition message belonging to partion 1. A back message takes Y dimension links only along a column that has atleast one node in partition 2a. An intra-partition message of partition 1 takes Y dimension links only along the column that has atleast one node in partition 1. The only column that satisfies both these conditions is one which has the source node s. In this column, an intra-partition message of partition 1 can only take those Y dimension links which lie in partition 1 and are in the positive direction. On the other hand, a back message can only take those Y dimension links in the positive direction which lie in partition 2b. Therefore, there cannot be contention between an intra-partition message of partition 1 and a back message.

Using the same reasoning mentioned in the beginning of this proof, there is no contention in the X dimension between a back message and an intra-partition message belonging to partition 2b. Let us consider contention in the Y dimension for a back message from node u to node v such that $v <_d u$. There are three cases: the back message takes (a) atleast one Y dimension link in the positive direction, or (b) atleast one Y dimension link in the negative direction, or (c) no Y dimension links. In case (c) there is no possible contention with an intra-partition message of partition 2b. Let us take cases (a) and (b). Consider an intra-partition message of partition 2b from node a to node b such that $a <_d b <_d u$. If this message shares even one Y dimension link with the back message, then $a <_d v$. According to the Umesh algorithm, if u sends a message to

v, the set of nodes $\{i|(i <_d v)\}$ will only receive the message in subsequent steps. Since the node a lies in this set, it is yet to receive the message and thus cannot be sending a message to b in the same step. Therefore, there cannot be contention between an intra-partition message belonging to partition 2b and a back message. ■

Lemmas 1-3 directly lead to:

Theorem 4 *The SPUmesh algorithm is link contention free for a single multicast.*

The above theorem has been proved assuming that $s <_d m$. The case of $m <_d s$ can be easily proved with similar arguments. Although this theorem has been proved for a 2D mesh, it is quite general and can be easily extended for higher dimensional meshes.

3.3 Reduced Contention for Multiple Multicast

Using the SPUmesh algorithm, each multicast chooses a different half-node depending on the position of its corresponding source node. This reduces the contention on the centrally positioned nodes of Φ. When Φ is divided recursively at each stage of the algorithm, the above effect carries over. Let us take the example from Fig. 2 to see the effect of the SPUmesh algorithm. Figures 7(a) and 7(b) illustrate the results for multicasts A and B, respectively. Now, multicasts A and B choose (1,0) and (2,0) as their respective center nodes and experience zero node contention. In general, the SPUmesh algorithm affects the choice of the center node at each of the successive steps of the algorithm, resulting in reduced node contention.

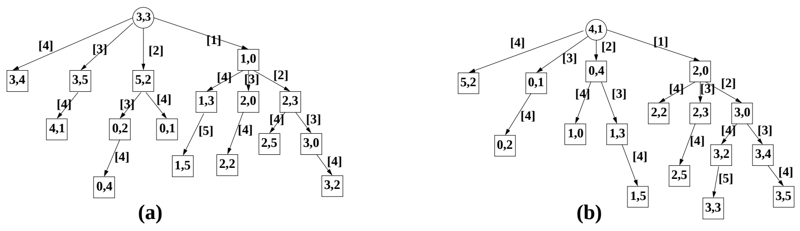

Figure 7: Zero node contention for the example multiple multicasts in Fig. 2 with the SPUmesh algorithm: (a) the new multicast tree for A and (b) the new multicast tree for B.

Analytical Estimate of Node Contention: Let us analyze an estimate of the node contention when using the SPUmesh algorithm for multiple multicast. We only consider the worst case of total overlap of source-destination sets here.

Theorem 5 *The latency of a multiple multicast pattern with complete overlap of source-destination sets (S sources and $(D-1)$ destinations for each source with identical Φ, $S \leq D$) while using the SPUmesh algorithm is bounded above by ($\lceil \log_2 D \rceil + \sum_{i=0}^{n} 2^i \lceil \log_2 \frac{D}{2^{i+1}} \rceil + p \lceil \log_2 \frac{D}{2^{n+2}} \rceil$) steps, where $S = 1 + \sum_{i=0}^{n} 2^i + p$ and $p < 2^{n+1}$.*

Proof: Let s be one of the sources. In the worst case, s can be the half-node for atmost one other multicast, the quarter-node for atmost two other multicasts, the one-eighth node for atmost four other multicasts, and so on. The latency of the multiple multicast pattern is bounded above by the maximum number of steps (start-ups) that s can undergo. Let S be written as $1 + \sum_{i=0}^{n} 2^i + p$ where $p < 2^{n+1}$. The node s undergoes atmost $\lceil \log_2 D \rceil$ start-ups as a source, $\lceil \log_2 \frac{D}{2} \rceil$ start-ups as a half-node, $2\lceil \log_2 \frac{D}{4} \rceil$ start-ups as a quarter-node. Therefore, the multiple multicast pattern latency is bounded above by ($\lceil \log_2 D \rceil + \sum_{i=0}^{n} 2^i \lceil \log_2 \frac{D}{2^{i+1}} \rceil + p \lceil \log_2 \frac{D}{2^{n+2}} \rceil$) steps. ■

A small experiment was conducted to estimate the number of steps incurred by multiple multicast pattern using the SPUmesh algorithm with complete overlap of source-destination sets on an 8×8 mesh. Like previous experiments, all system parameters were ignored and only a high start-up time was considered. The experimental values were averaged over 30 runs and rounded off to the nearest integer. The third and seventh rows of Table 4 show the results of this experiment for odd and even D, respectively. The analytical values calculated by using Theorem 5 are shown in the fourth and eighth rows for odd and even D, repectively. It can be observed that the analytical values act as an upper bound for the experimental values, validating Theorem 5. It can also be seen that, unlike the Umesh case, D being odd or even does not affect the number of steps. The experimental results in Table 4 are significantly less when compared to those in Table 1. The analytical results of Table 4, when compared with those of Table 1 lead to the following observation:

Observation 1 *The upper bound on the latency of a multiple multicast pattern while using the SPUmesh algorithm is less than the lower bound on the latency of the same pattern while using the Umesh algorithm.*

Therefore, the SPUmesh algorithm promises much better performance than the Umesh algorithm for multiple multicast.

Table 4: Experimental and analytical latency of multiple multicast (in number of steps) using SPUmesh on an 8×8 mesh for different values of S and odd and even D. Complete overlap of source-destination sets is considered ($S \leq D$).

S	1	1	1	15	15	15	31	31	47	63
odd $D = \|\Phi\|$	15	31	63	15	31	63	31	63	63	63
Experimental	4	5	6	14	21	26	30	41	55	62
Analytical	4	5	6	15	30	45	31	62	63	63
S	1	1	1	16	16	16	32	32	48	64
even $D = \|\Phi\|$	16	32	64	16	32	64	32	64	64	64
Experimental	4	5	6	15	21	24	31	40	54	63
Analytical	4	5	6	15	31	47	31	63	63	63

4 Improved Algorithms using Multidestination Messages

In this section two multicast algorithms are proposed for systems supporting multidestination message-passing. They are shown to be better than the HL scheme [18] for multiple multicast.

4.1 The Source Quadrant-based Hierarchical Leader (SQHL) Scheme

4.1.1 The Algorithm

Let us consider a multicast pattern from source s to a destination set D. Let $L_0 = D \cup \{s\}$. In the ith step, $1 \leq i \leq n$, the algorithm partitions set L_i into disjoint subsets, each with a leader node. The leader node can forward a message to the members of its set using a single multidestination worm under the dimension-order routing model. The groupings are done exactly like the HL scheme. However, the leader nodes are chosen according to the algorithm in Fig. 8. In effect, when dealing with the rth dimension, $1 \leq r \leq n$ an $(n-1)$-dimension plane, P_r, is drawn perpendicular to the rth dimension such that it divides the nD mesh into halves. The leaders are chosen as close to that end of the grouping as the half in which the source lies. The remaining steps are the same as the original HL scheme. The number of startups for a single multicast using the SQHL scheme is the same as when using the HL scheme i.e., $(\lceil \log_2(|L_m| + 1) \rceil + m)$.

The SQHL Scheme

Input: Source $s = (s_1, s_2, \ldots, s_n)$, s_i is the coordinate of s in the ith dimension; $\forall i\ 1 \le s_i \le k_i$, k_i is the radix in the ith dimension.
A given grouping of w destinations in the rth dimension $g = \{d_1, d_2, \ldots, d_w\}$, where $t_{i,j}$ is the coordinate of d_i in the jth dimension ($\forall j\ 1 \le t_{i,j} \le k_j$; k_j is the radix in the jth dimension)

Output: Leader node $d_l \in g$ such that d_l can cover all the members of g with a single multidestination worm

Procedure:

```
if $s_r \le \lfloor \frac{k_r}{2} \rfloor$ then
    Choose $d_l$ ($d_l \in g$) as the leader such that $d_i \in g \Rightarrow t_{l,r} \le t_{i,r}$
else
    Choose $d_l$ ($d_l \in g$) as the leader such that $d_i \in g \Rightarrow t_{l,r} \ge t_{i,r}$
endif
```

Figure 8: Leader node choice in the SQHL Scheme.

Figure 9 shows the message pattern for multicast B (from Fig. 3(b)) using the SQHL scheme on a 2D mesh. Since the source for the multicast, (4,4), is in the right upper quadrant, the L_1 leaders are chosen on the rows to the right end of the mesh, and the L_2 leaders are chosen on the columns to the upper end of the mesh. Figure 10 shows the patterns of multidestination worms that get generated for different positions of the source in a 2D mesh with large destination sets. It is assumed that grouping is done first along the x dimension. Depending on the position of the source the L_1 set tends towards one of the edges, and the L_2 set towards one of the corners of the mesh. The first set of multidestination worms (arrows marked as 1 in Fig. 10) go from the L_2 nodes to the L_1 nodes and the next set of worms (arrows marked as 2) go from the L_1 nodes to the remaining destinations. The Umesh stage of the multicasts is not shown for clarity. This scheme has potential to reduce node contention compared to the HL scheme.

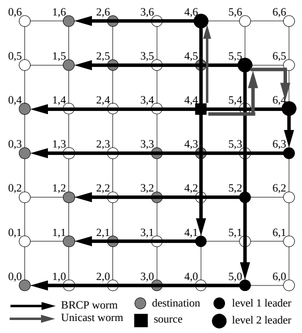

Figure 9: Message pattern for multicast B (from Fig. 3(b))using the SQHL scheme.

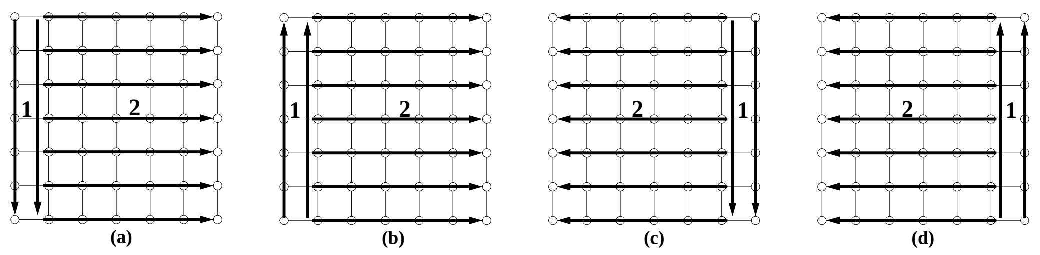

Figure 10: Multidestination worm patterns generated by the SQHL scheme when the source is in the (a) left upper quadrant, (b) left lower quadrant, (c) right upper quadrant, and (d) right lower quadrant of the mesh.

4.1.2 Reduced Contention for Multiple Multicast

Let us take the example from Fig. 3. Multicasts A and B have identical destination sets with sources at (2,3) and (4,4), respectively. With the SQHL scheme, the message pattern for multicast A remains the same as that of Fig. 3(a). However, for multicast B, the new pattern is shown in Fig. 9. Since A and B have their sources in different quadrants of the mesh, their L_2 and L_1 sets are different. This leads to zero node contention. Thus, in general, the SQHL scheme is found to be better than the HL scheme for multiple multicast.

Analytical Estimate of Node Contention: Let us analyze an estimate of the node contention when using the SQHL scheme for multiple multicast on a k-ary n-cube. Again, let us assume that there are S multicasts and all multicasts have identical source-destination sets of size D ($S \leq D$) which leads to identical leader sets.

Theorem 6 *The latency of a multiple multicast pattern with complete overlap of source-destination sets (S sources and $(D-1)$ destinations for each source with identical source-destination sets, $S \leq D$) on a k-ary n-cube using the SQHL Scheme is $\leq (1+min(S, \frac{k^n}{2^n})\lceil log_2 \frac{k}{2} \rceil + \sum_{i=0}^{n} min(\frac{k^n}{2^{n-i}}, S))$ and is $\geq (max(1, S(1-\frac{1}{2^n})))$ steps.*

Proof: Based on the grouping characteristics of the SQHL scheme, it can be easily observed that the lower bound occurs when there is uniform distribution of sources in the mesh. On the other hand, the upper bound occurs when maximum number of multicasts have their sources in the same quadrant. Similar to the reasoning in the proof for Theorem 3, let us consider a node x from the one of the highest leader sets, L_g. The number of sources in the same quadrant is $\frac{S}{2^n}$ for the uniform distribution case. For high values of D, $|L_g| \approx 1$ and $g = n$. Therefore, the number of start-ups that x undergoes is given by $\frac{S}{2^n} + \frac{S}{2^{n-1}} + \ldots + \frac{S}{2}$. This gives us the lower bound of $(max(1, S(1 - \frac{1}{2^n})))$ steps.

Consider maximum number of multicasts having their sources in the same quadrant. This number will be atmost $min(S, \frac{k^n}{2^n})$. All these multicasts have identical $|L_g|$ sets. Therefore, x undergoes atmost $\lceil log_2 \frac{|L_g|}{2} \rceil$ start-ups for the Umesh phase of each of the multicasts. Subsequently, x undergoes $min(S, \frac{k^n}{2^n})$ start-ups as an L_n leader, $min(S, \frac{k^n}{2^{n-1}})$ start-ups as an L_{n-1} leader, and so on. Like in the proof for Theorem 3, there is an extra start-up added to the upper bound, independent of whether x is one of the sources. This gives us the upper bound of $(1 + min(S, \frac{k^n}{2^n}) \lceil log_2 \frac{k}{2} \rceil + \sum_{i=0}^{n} min(\frac{k^n}{2^{n-i}}, S))$ steps. ■

A small experiment was conducted to estimate the number of sequentialized steps that multiple multicast incurs for the SQHL Scheme with complete overlap of source-destination sets on a 8×8 mesh. Like previous experiments, all system parameters were ignored and only a high startup time was considered. The experimental values were averaged over 30 runs and rounded off to the nearest integer. The fourth row of Table 5 shows the results of this experiment. The analytical values in the third row of Table 5 represent the worst case, and therefore act as loose upper bounds. Nevertheless, the experimental values fall between the two bounds. Based on the lower and upper bounds derived in Theorems 3 and 6, it can be easily observed:

Observation 2 *The upper bound on the latency of a multiple multicast pattern while using the SQHL scheme is less than the upper bound on the latency of the same pattern while using the HL scheme. Also, the lower bound for the SQHL scheme is consistently below the lower bound for the HL scheme.*

Therefore, the SQHL scheme promises much better performance than the HL scheme for multiple multicast.

4.1.3 Potential for Improvement

Though the SQHL scheme promises better performance, there are some cases where the *SQHL* scheme performs poorly. For example, if two (or more) multicasts have their sources in the same

Table 5: Experimental and analytical latency of multiple multicast (in number of steps) using the SQHL scheme on an 8×8 mesh for different values of S and D. Complete overlap of source-destination sets is considered ($S \leq D$).

S	1	1	1	16	16	16	32	32	48	64
D	16	32	64	16	32	64	32	64	64	64
Max	5	5	5	65	65	65	81	81	81	81
Exp	5	5	3	20	20	16	35	28	40	48
Min	2	2	2	12	12	12	24	24	36	48

quadrant of the mesh, the groupings and the choice of leader sets for both (or all) the multicasts are identical. This leads to high node contention. Let us consider two multicasts, C and D, with identical destination sets and (3,4) and (5,3) as their respective sources on a 2D mesh. The destination distribution and resultant multicast patterns are shown in Fig. 11. Since both sources lie in the same quadrant, the L_1 and L_2 sets are identical. This leads to node contention and results in higher latency for both multicasts. An ideal algorithm should use source-dependent information to minimize node contention further. We achieve this in our next algorithm.

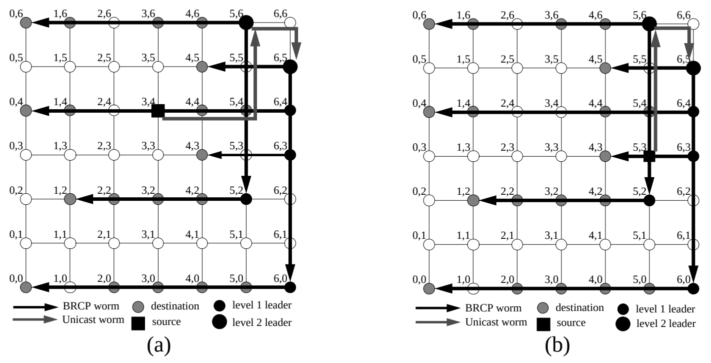

Figure 11: Message pattern indicating node contention between two multicasts with identical destination sets using the SQHL scheme: (a) multicast C with source at (3,4) and (b) multicast D with source at (5,4).

4.2 The Source Centered Hierarchical Leader (SCHL) Scheme

4.2.1 The Algorithm

Let us consider a multicast with source s and destination set D. Let $L_0 = D \cup \{s\}$. Like the HL and SQHL schemes, the groupings are done along the dimensions for systems supporting dimension-order routing. Each grouping is partitioned into atmost two sets, and each of them is given a leader node according to the algorithm in Fig. 12. Once the groupings have been done, the remaining steps are the same as the HL scheme.

The SCHL Scheme

```
Input: Source $s = (s_1, s_2, \ldots, s_n)$, $s_i$ is the coordinate of $s$ in the $i$th dimension; $\forall i\ 1 \le s_i \le k_i$, $k_i$ is
       the radix in the $i$th dimension.
       A given grouping of $w$ destinations in the $r$th dimension $g = \{d_1, d_2, \ldots, d_w\}$, where $t_{i,j}$ is the
       coordinate of $d_i$ in the $j$th dimension ($\forall j\ 1 \le t_{i,j} \le k_j$; $k_j$ is the radix in the $j$th dimension)
Output: Leader nodes for $g$ such that they can cover all the members of $g$ in a single startup
Procedure:
    Partition $g$ into $g_1$ and $g_2$ such that $d_i \in g_1 \Rightarrow t_{i,r} \le s_r$ and $d_i \in g_2 \Rightarrow t_{i,r} > s_r$
    if $g_1$ is non-empty then
        Choose $d_l$ ($d_l \in g_1$) as the leader for $g_1$such that $d_i \in g_1 \Rightarrow t_{i,r} \le t_{l,r}$
    endif
    if $g_2$ is non-empty then
        Choose $d_l$ ($d_l \in g_2$) as the leader for $g_2$such that $d_i \in g_2 \Rightarrow t_{i,r} \ge t_{l,r}$
    endif
```

Figure 12: Leader node choice in the SCHL Scheme.

In effect, while dealing with the rth dimension, each group g from the HL scheme is partitioned into atmost two groups g_1 and g_2 as follows. An $(n-1)$-dimension plane, P_r, is drawn through the source perpendicular to the rth dimension. All the nodes in g lying to one side of the plane (including nodes on the plane) are added to g_1 and the remaining to g_2. Each of these groups has a leader node as close to P_r as possible.

Let us consider a 2D mesh with dimension-order routing and an example multicast with source at (2,2) as shown in Fig 13. It is assumed that grouping is done first along the x ($0th$) dimension. The L_1 nodes are on both sides of P_0 (1-dimensional plane corresponding to the $0th$ dimension), which is the vertical line V drawn through the source. The L_1 leader nodes are now grouped column-wise. The L_2 nodes lie on both sides of P_1, the horizontal line H drawn through the source. Thus, the L_2 set will be located close to, and around the source node.

In the SCHL scheme, it can be observed that $|L_i|$ can potentially be atmost 2^i times the $|L_i|$ of the HL scheme. Assuming m levels of groupings, and L_m as the leader set in the original HL scheme, $\lceil \log_2(2^m|L_m| + 1) \rceil$ startups are required for the Umesh stage of the SCHL scheme. The remaining m multidestination worm phases take m startups. Therefore, the maximum number of

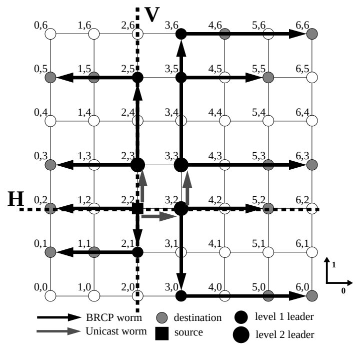

Figure 13: Message pattern for an example multicast using the SCHL scheme.

startups required for single multicast is $(\lceil \log_2(2^m|L_m|+1)\rceil + m) \simeq (\lceil \log_2(|L_m|)\rceil + 2m)$.

4.2.2 Reduced Contention for Multiple Multicast

Let us take multicasts C and D from Fig. 11, with sources at (3,4) and (5,3), respectively. The multicast message patterns for C and D using the SCHL scheme are shown in Fig. 14. It can be observed that the intersection of the leader sets of C and D is almost negligible. This results in reduced note contention as compared to the SQHL scheme. Therefore, the SCHL scheme shows significant potential for performance improvement for multiple multicast.

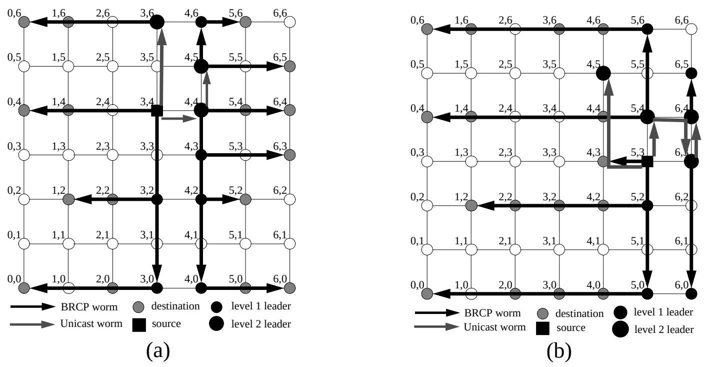

Figure 14: Message patterns for two multicasts with identical destination sets using the SCHL scheme: (a) multicast C and (b) multicast D. Multicasts C and D are from Fig 11.

Building an analytical model for the SCHL scheme is a very complex task. To estimate the node contention a small experiment was conducted to estimate the number of sequentialized start-ups that multiple multicast incurs for the SCHL Scheme with complete overlap of source-destination sets on a 8×8 mesh. Like previous experiments, all system parameters were ignored and only a high startup time was considered. The third row of Table 6 shows the results of this experiment. These results were compared with the number of steps using the SQHL scheme. It can be easily observed that SQHL produces lower latency for smaller values of D, but as the value of D increases the SCHL scheme promises much better performance than the SQHL scheme for multiple multicast.

Table 6: Experimental latency of multiple multicast (in number of steps) using the SCHL scheme and the SQHL scheme on an 8×8 mesh for different values of S and odd and even D. Complete overlap of source-destination sets is considered ($S \leq D$).

s	1	1	1	16	16	16	32	32	48	64
$D = \|\Phi\|$	16	32	64	16	32	64	32	64	64	64
SCHL	5	5	4	25	23	11	41	17	21	23
SQHL	5	5	3	20	20	16	35	28	40	48

5 Simulation Experiments and Results

We simulated and compared the performances of the various algorithms for multiple multicast. A flit-level wormhole-routed simulator WORMULSim (built using CSIM [21]) was used to model the communication steps and evaluate the algorithms. For all the experiments, we assumed system parameters representing the current trend in technology. The parameters used were: t_s (communication start-up time) = 5 microsecs, t_{phy} (link propagation time) = 5 nanosecs, t_{node} (router delay at node) = 20 nanosecs, t_{sw} (switching time across the router crossbar) = 5 nanosecs, t_{inj} (time to inject message into network) = 5 nanosecs and t_{cons} (time to consume message from network) = 5 nanosecs. The message length was assumed to be a constant 50 flits. A 16×16 mesh was used for the 2D simulations and $6 \times 6 \times 6$ mesh for the 3D simulations. The t_{node} parameter for the multidestination simulations was fixed at 40 nanoseconds, to take care of the additional routing overhead compared to unicast message passing [18]. To keep the comparison fair, the number of consumption channels was fixed at 4 for 2D and 6 for 3D in both unicast and multidestination schemes [18]. Each experiment was carried out 30 times to obtain average results.

For each simulation we considered the two cases: a) worst case scenario of total overlap of destination sets and b) random overlap of destination sets. In reality, concurrent multiple multicasts of an application could lie anywhere in the range of *random* to *complete* overlap of destination sets. For each of the above cases, we plotted multiple multicast latency against a) number of destinations (d), while keeping number of sources (S) fixed, and b) number of sources (S), while keeping d fixed. The former shows the effect of the number of multicasts on latency, while the latter shows the effect of the size of the multicasts on latency. In the complete overlap case the source-destination sets were chosen identical for $S \leq D$. For $S > D$, it can be noted that it is not possible to generate identical source-destination sets. Therefore, in our complete overlap experiments, we generated source-destination sets as close to identical as possible for $S > D$.

5.1 Multicast Algorithms using Unicast Messages

We simulated multiple multicast for different values of S and d, on 2D and 3D systems, using both Umesh and SPUmesh algorithms and compared them.

5.1.1 2D Meshes

Figures 15(a) and (b) show the comparison for complete overlap of source-destination sets. It can be observed from Fig. 15(a) that the SPUmesh algorithm outperforms the Umesh algorithm for both sizes of multicasts and for both ranges, $S \leq D$ and $S > D$. For a large number of multicasts ($S = 256$), the SPUmesh algorithm is about 5-6 times better for a large d (200), and about 4 times better for a small d (128). Even for a small number of multicasts ($S = 128$), the SPUmesh algorithm is about 5 times better. Also, the SPUmesh algorithm performs equally well for both large and small d. This is because, increasing d does not increase the node contention, as the multicast trees are perfectly staggered for identical destination sets. On the other hand, there is a marked difference between the d=128 (127) and d=200 (199) using the Umesh algorithm. This is because increasing d increases the number of startups the common center node will have to sequentialize. This trend can be seen more clearly in Fig. 15(b). As predicted in Theorem 1, the Umesh algorithm performs worse for even d (odd D) than for odd d (even D) in the $S \leq D$ range.

Figures 15(c) and (d) show the comparison for random overlap of source-destination sets. It can be observed in Fig. 15(c), that the SPUmesh algorithm still outperforms the Umesh algorithm for large d by a factor of about 2. This is clear in Fig. 15(d) where the Umesh latency shoots up

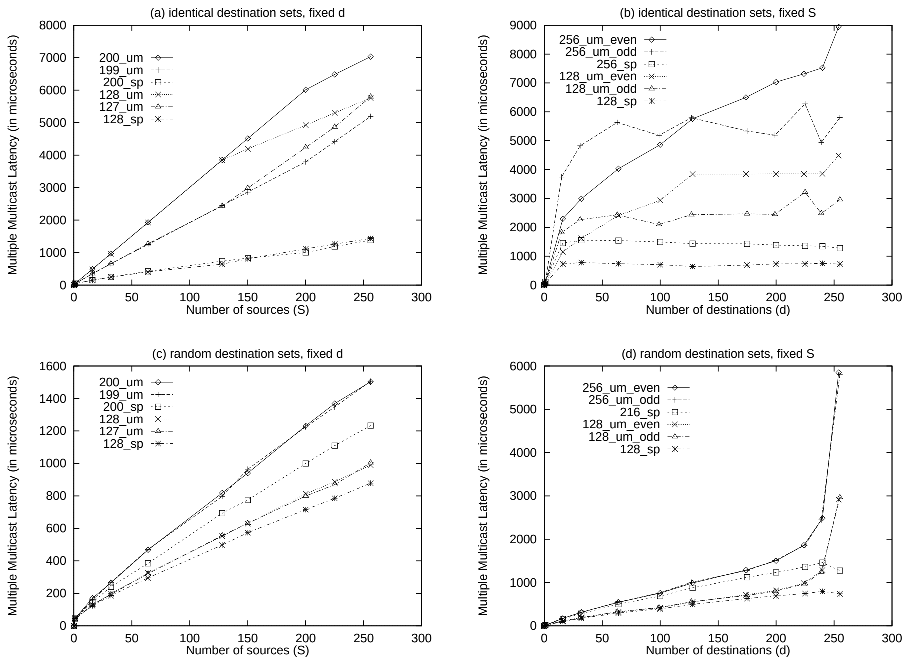

Figure 15: Multiple multicast latency on a 16×16 mesh using Umesh (um) and SPUmesh (sp) algorithms. Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 256) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 256) with varying d.

for larger d, whereas the SPUmesh latency remains more or less constant. The latency values are not as high as the identical destination case because the node contention is less (since destinations are chosen randomly). It can also be observed that, unlike the complete overlap case, the Umesh algorithm performance does not depend on whether d is odd or even.

5.1.2 3D Meshes

Figures 16(a) and (b) show the comparison for complete overlap of source-destination sets. The results are similar to the 2D case. Figure 16(a) shows that, like in the 2D case, the SPUmesh algorithm outperforms the Umesh algorithm by a factor of 5-6 for a large S (200), and by a factor of 4 for a small S (128), for both ranges of $S \leq D$ and $S > D$. Also, similar to the 2D case, there is minimal change in latency as d increases for the SPUmesh algorithm as compared to the Umesh algorithm. This is shown in Fig. 16(b). Also, as predicted, the Umesh performance is worse for even d (odd D) than for odd d (even D) in the $S \leq D$ range.

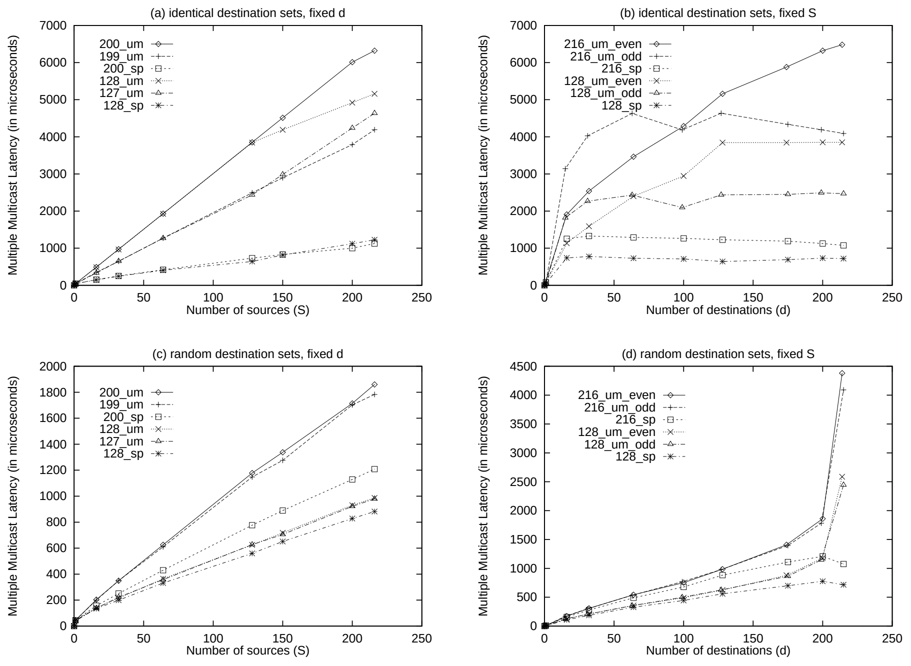

Figure 16: Multiple multicast latency on a $6 \times 6 \times 6$ mesh using Umesh (um) and SPUmesh (sp) algorithms. Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 216) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 216) with varying d.

Figures 16(c) and (d) show the comparison for random overlap of source-destination sets. Again, the results are similar to the 2D case. The figures show that the SPUmesh algorithm outperforms the Umesh algorithm by a factor of about 2, and the Umesh latency shoots up for larger degrees of multicast sets, whereas the SPUmesh latency remains more or less constant. Again, unlike the complete overlap case, the Umesh algorithm performance does not depend on whether d is odd or even. The above results lead to the conclusion that node contention in the Umesh and the SPUmesh algorithms does not vary with change in number of dimensions of the mesh. This is because the multicast tree is generated according to the dimension-ordered chain, and is not affected by the dimensionality of the mesh.

5.2 Multicast Algorithms using Multidestination Messages

We simulated multiple multicast for different values of S and d, on 2D and 3D systems, using HL, SQHL, and SCHL schemes and compared them.

5.2.1 2D Meshes

Figures 17(a) and (b) show the comparison for complete overlap of source-destination sets. It can be observed in Fig. 17(a), that for large d, the SCHL scheme is about 2 times better than the the SQHL scheme, and about 6-7 times better than the HL scheme. A property of the HL scheme is that latency reduces with increase in d [18]. This can be noted from Fig. 17(a), as the graphs for $d = 200$ are lower than the graphs for $d = 128$. This basic property of the HL scheme is best exploited by the SCHL scheme. This can be noticed in Fig. 17(b), where on increasing d the latency of SCHL drops substantially. It goes from 2000 microsec ($d = 64$) to about 250 microsec ($d = 255$), a reduction by a factor of about 10. For smaller d, the SQHL scheme outperforms the SCHL scheme, since the number of startups at the Umesh stage of the multicast is more for the SCHL scheme. A larger d offsets this overhead, and the SCHL scheme starts outperforming the SQHL scheme at d greater than 128.

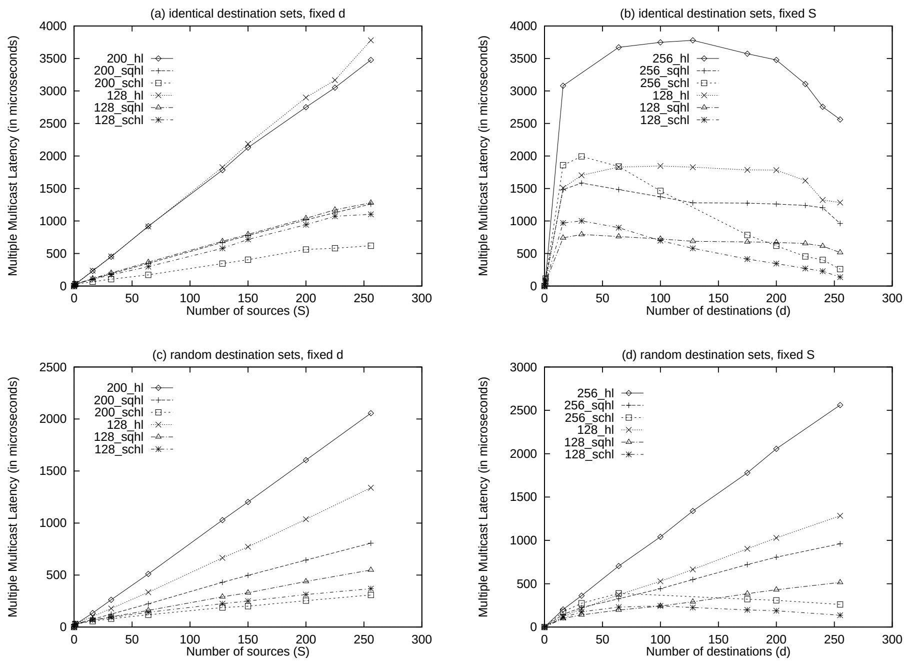

Figure 17: Multiple multicast latency on a 16 × 16 mesh using HL (hl), SQHL (sqhl) and SCHL (schl) schemes. Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 256) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 256) with varying d.

Figures 17(c) and (d) show the comparison for random overlap of source-destination sets. The

HL scheme undergoes a large degree of node contention even for randomly chosen destinations. It can be observed in Fig. 17(c) that the SCHL scheme still does about 2-3 times better than the SQHL scheme and about 5-7 times better than the HL scheme. Also, like the identical destination sets case, the SQHL scheme outperforms the SCHL scheme for small d. Again, there is a crossover point after which the SCHL scheme performs better. This can be seen Fig. 17(d).

5.2.2 3D Meshes

Figures 18(a) and (b) show the comparison for complete overlap of source-destination sets. The SCHL and SQHL schemes outperform the HL scheme. Like in the 2D case, the SCHL scheme best exploits the basic HL scheme property. This can be observed Fig. 18(a) and more clearly in Fig. 18(b). It can be seen in Fig. 18(b) that on increasing d the latency of the SCHL decreases substantially from 2000 microsec ($d = 32$) to about 500 microsec ($d = 215$). Also, as in the 2D case, the SCHL graphs crossover to outperform the SQHL graphs, but at a much larger d. This is because the radix of each dimension in the 3D mesh is only 6, which does not allow the grouping potential of the SCHL scheme to be realized completely, as compared to a 16×16 mesh.

Figures 18(c) and (d) show the comparison for random overlap of source-destination sets. As in the 2D case, the HL scheme undergoes a large amount of node contention and performs poorly. It can be observed in Fig. 18(c) that the SCHL scheme still does about 1.5 times better than the SQHL and about 5 times better than the HL scheme. Here too, the SQHL scheme outperforms the SCHL scheme for small d, but there is a crossover point after which the SCHL scheme performs better, as can be observed in Fig. 18(d).

The above results in Section 5.1 and Section 5.2 lead to the conclusion that node contention is one of the principal reasons for poor performance of existing multicast algorithms for multiple multicast. Reducing node contention should be the primary focus while designing multicast algorithms.

5.3 Comparing Unicast-based and Multidestination-based Schemes

We compared multicast algorithms using unicast messages with those using multidestination messages. This measures the potential benefits that a system provides if it supports multidestination

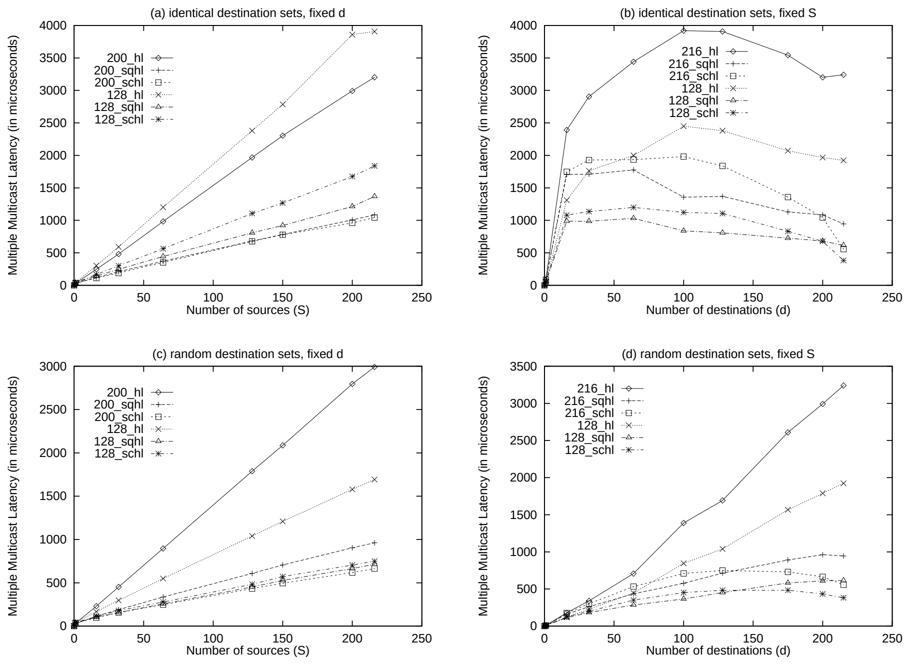

Figure 18: Multiple multicast latency on a $6 \times 6 \times 6$ mesh using HL (hl), SQHL (sqhl) and SCHL (schl) schemes. Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 216) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 216) with varying d.

wormhole routing. We compared the SPUmesh (the best unicast-based algorithm) with the SCHL (the best multidestination-based scheme) scheme.

5.3.1 2D Meshes

Figures 19(a) and (b) show the comparison for random overlap of source-destination sets. For large d, the SCHL scheme outperforms the SPUmesh algorithm as S increases, by a factor of almost 2.5. This can be observed in Fig. 19(a) and Fig. 19(b). The figures also shows that the SPUmesh algorithm shows better results for small d. There is a crossover point at around $d = 100$, after which the SCHL scheme keeps improving. The reason is, hierarchical grouping of destinations gives relatively large leader sets for small d. This is because the destinations are randomly scattered in the mesh. On the other hand, SPUmesh generates perfectly staggered trees for identical destination sets. But, as d increases, the advantage of efficient grouping overcomes the node contention. This results in better performance by the SCHL scheme. As noted earlier, the latency of SPUmesh remains steady with increase in number of destinations.

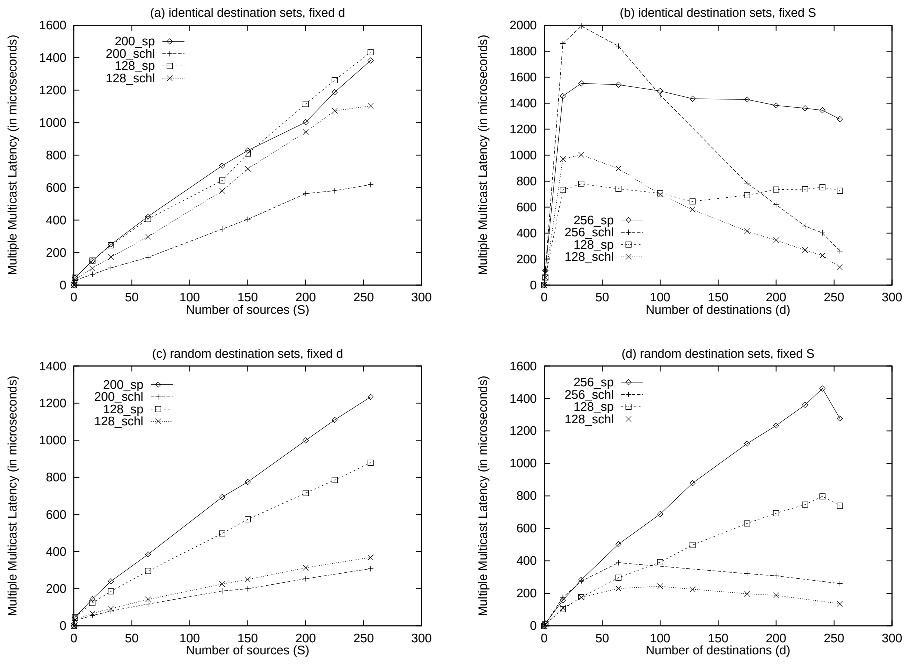

Figure 19: Multiple multicast latency on a 16×16 mesh using SPUmesh (sp) and SCHL (schl). Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 216) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 216) with varying d.

Figures 19(c) and (d) show the comparison for random overlap of source-destination sets. Unlike the identical destination set case, the SCHL scheme outperforms the SPUmesh algorithm for both large and small d. This can be seen in Figure 19(c), and Fig. 19(d) (to a larger extent). Since the destination sets are completely random, the phenomenon of identical L_1 sets does not occur for sources on the same column. Hence, there is no node contention to degrade the performance of the SCHL scheme. Also, the grouping effect takes over for larger d to give a factor of about 4-6 improvement over the SPUmesh algorithm.

5.3.2 3D Meshes

Figures 20(a) and (b) show the comparison for complete overlap of source-destination sets. Figure 20(a) shows that for large d (200) the SCHL scheme outperforms the SPUmesh algorithm, but for smaller d (128) this is not true. This is because, as explained before, the radix of each dimension in the 3D mesh is only 6, which does not allow the grouping potential of the SCHL scheme to be realized completely. Figure 20(b) shows the crossover point (d = 180) where the SCHL scheme starts performing better than the SPUmesh algorithm. The reason for the large crossover value is again the small radix of the 3D mesh.

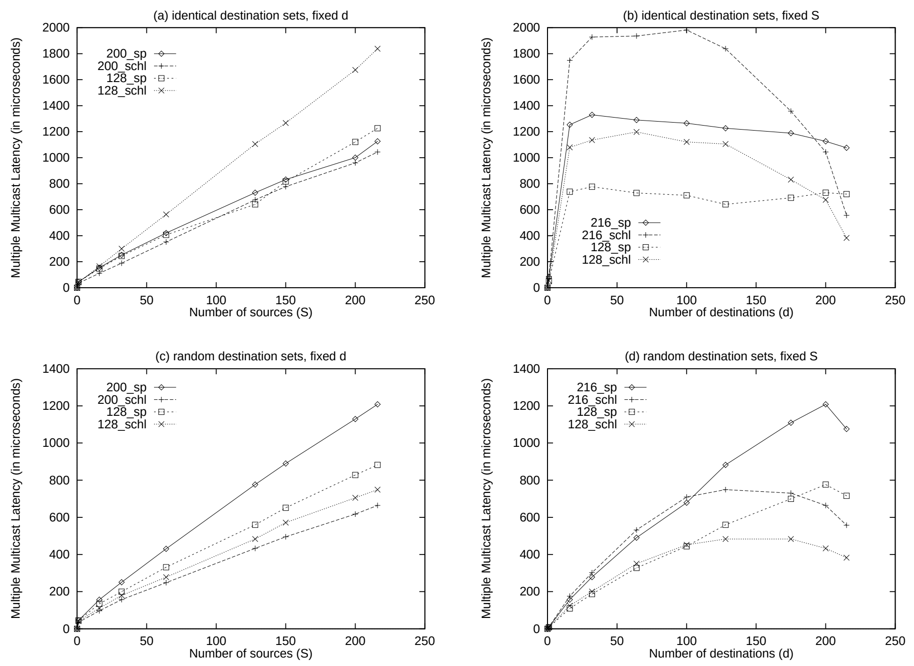

Figure 20: Multiple multicast latency on a $6 \times 6 \times 6$ mesh using SPUmesh (sp) and SCHL (schl). Complete overlap of source-destination sets with (a) fixed d (128, 200) with varying S and (b) fixed S (128, 216) with varying d. Random overlap of source-destination sets with (c) fixed d (128, 200) with varying S and (d) fixed S (128, 216) with varying d.

Figures 20(c) and (d) show the comparison for random overlap of source-destination sets. Figure 20(c) shows that the SCHL scheme outperforms the SPUmesh algorithm for both large d (200) and small d (128). Figure 20(d) clearly shows that the SPUmesh algorithm performs almost as well as the SCHL scheme till about $d = 100$. After that point, the SCHL clearly outperforms the SPUmesh. Like the 2D case, this is due to phenomenon of identical L_1 sets not occuring for sources on the same column for the SCHL case.

These results lead to the conclusion that systems providing multidestination message passing can support more efficient multicast than systems providing unicast message passing only.

5.3.3 Scalability with Increasing System Size

Finally, we compared the five multicast algorithms for varying system sizes. We simulated multiple multicast patterns using system sizes of 4×4, 8×8, 12×12 and 16×16, for a) $S = \frac{N}{2}$ and $d = N-1$, and b) $S = N$ and $d = \frac{N}{2}$, where $N = n^2$ for an $n \times n$ system size.

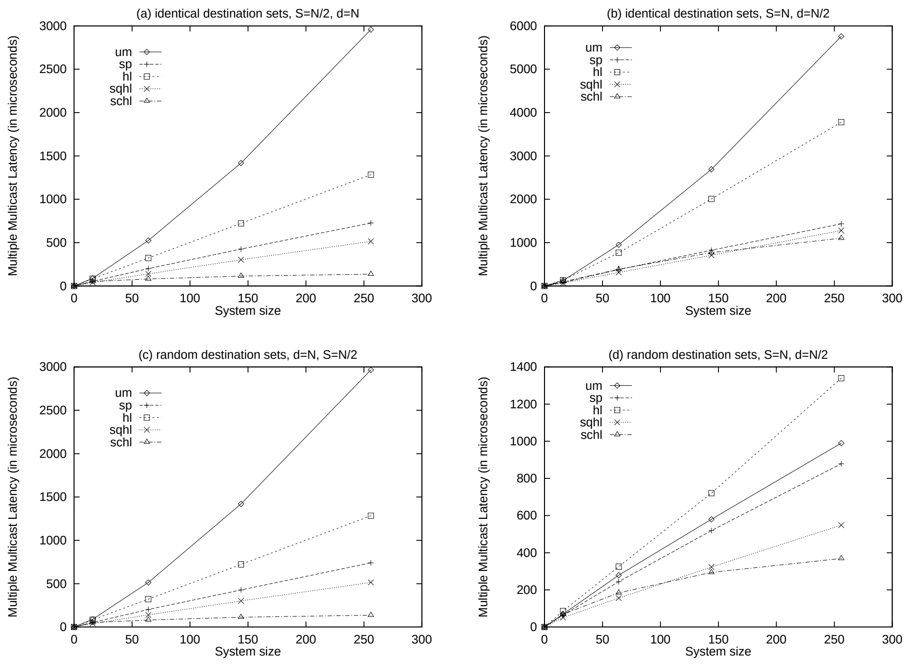

Figure 21: Multiple multicast latency on varying system sizes using Umesh (um), SPUmesh (sp), HL (hl), SQHL (sqhl) and SCHL (schl) schemes. Complete overlap of source-destination sets with (a) $d = N$ and $S = \frac{N}{2}$ and (b) $d = \frac{N}{2}$ and $S = N$. Random overlap of source-destination sets with (c) $d = N$ and $S = \frac{N}{2}$ and (d) $d = \frac{N}{2}$ and $S = N$.

Figures 21(a) and (b) show the comparison for complete overlap of source-destination sets. Figure 21(a) shows that the new algorithms scale better with increasing system size than the existing algorithms for multiple multicast. It can be observed in this graph that the SCHL scheme is the only one which shows constant performance with increasing system size for large values of d. Figure 21(b) also shows that the new algorithms are more scalable with increasing system size for large values of S.

Figures 21(c) and (d) show the comparison for random overlap of source-destination sets. Like in the complete overlap case, Fig. 21(c) shows that the new algorithms scale better with increasing system size than the existing algorithms for multiple multicast. Again, it can be observed in this graph that the SCHL scheme is the only one which shows constant performance with increasing system size for large values of d. Figure 21(d) also shows that the new algorithms are more scalable with increasing system size for large values of S.

6 Conclusions

In this paper we have introduced the concept of designing multicast algorithms with minimized node contention so that multiple multicast patterns can be implemented with reduced latency. We have shown that the existing multicast algorithms (Umesh and HL schemes) for wormhole k-ary n-cube systems lead to poor performance when multiple multicasts are involved. The poor performance has been explained by node contention. Node contention arises from the fact that the existing multicast algorithms were designed with single multicast in mind. We have presented a detailed analysis of the node contention that results when existing multicast algorithms are used for multiple multicast.

A new approach has been proposed, which uses source-specific information to create multicast trees. Using the new approach, three new multicast algorithms (SPUmesh, SQHL and SCHL) have been developed. The node contention in these algorithms have been analyzed for multiple multicast patterns, and shown to be less with respect to the existing algorithms. The new algorithms have been demonstrated to perform 5-10 times better than the Umesh and the HL schemes under multiple multicast patterns. Thus, these algorithms demonstrate a lot of potential for designing scalable collective communication libraries on current and future wormhole systems. By comparing the SPUmesh with the SCHL, it has been demonstrated that multidestination message-passing is a desired feature for future wormhole systems to implement efficient collective communication operations. In this paper, we have only emphasized on multiple multicast patterns. We are currently evaluating other patterns like multiple barriers and reduction operations. It will also be interesting to see how much benefits these new algorithms can provide when integrated with application characteristics on distributed shared memory systems with cache-coherency.

Additional Information: A number of related papers and technical reports are available electronically through the home page of *Parallel Architecture and Communication* (PAC) research group. The URL is *http://www.cis.ohio-state.edu/~panda/pac.html.* Alternatively, the papers can be obtained by anonymous ftp from: *ftp.cis.ohio-state.edu, cd pub/communication.*

References

[1] A. Bar-Noy and S. Kipnis. Multiple Message Broadcasting in the Postal Model. In *International Parallel Processing Symposium*, pages 463–470, 1993.

[2] M. Barnett, S. Gupta, D. G. Payne, L. Shuler, R. van de Geijn, and J. Watts. Interprocessor Collective Communication Library (Intercom). In *Scalable High Performance Computing Conference*, pages 357–364, 1994.

[3] N. J. Boden, D. Cohen, and et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, pages 29–35, Feb 1995.

[4] R. V. Boppana, S. Chalasani, and C. S. Raghavendra. On Multicast Wormhole Routing in Multicomputer Networks. In *Symposium on Parallel and Distributed Processing*, pages 722–729, 1994.

[5] J. Bruck, R. Cypher, and C.-T. Ho. Multiple Message Broadcasting with Generalized Fibonacci Trees. In *Symposium on Parallel and Distributed Processing*, pages 424–430, 1992.

[6] L. De Coster, N. Dewulf, and C.-T. Ho. Efficient Multi-packet Multicast Algorithms on Meshes with Wormhole and Dimension-Ordered Routing. In *International Conference on Parallel Processing*, pages III:137–141, Aug 1995.

[7] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, pages 547–553, May 1987.

[8] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[9] S. L. Johnsson and C.-T. Ho. Optimum Broadcasting and Personalized Communication in Hypercubes. *IEEE Transactions on Computers*, pages 1249–1268, September 1989.

[10] Ram Kesavan and D. K. Panda. Minimizing Node Contention in Multiple Multicast on Wormhole k-ary n-cube Networks. In *Proceedings of the International Conference on Parallel Processing*, Chicago, IL, Aug 1996.

[11] Y. Lan, A.-H. Esfahanian, and L. Ni. Multicast in Hypercube Multiprocessors. *Journal of Parallel and Distributed Computing*, 8:30–41, 1990.

[12] X. Lin and L. M. Ni. Deadlock-free Multicast Wormhole Routing in Multicomputer Networks. In *Proceedings of the International Symposium on Computer Architecture*, pages 116–124, 1991.

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

[14] P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni. Unicast-based Multicast Communication in Wormhole-routed Networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1252–1265, Dec 1994.

[15] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, Mar 1994.

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

[17] D. K. Panda. Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole-Routed Systems. In *1995 Workshop on Challenges for Parallel Processing*, pages 8–15, 1995.

[18] D. K. Panda, S. Singal, and P. Prabhakaran. Multidestination Message Passing Mechanism Conforming to Base Wormhole Routing Scheme. In *Proceedings of the Parallel Computer Routing and Communication Workshop*, pages 131–145, 1994.

[19] J. Y. L. Park, H. A. Choi, N. Nupairoj, and L. M. Ni. Construction of Optimal Multicast Trees Based on the Parameterized Communication Model. In *Proceedings of the International Conference on Parallel Processing*, Chicago, IL, Aug 1996.

[20] D. F. Robinson, P. K. McKinley, and B. H. C. Cheng. Optimal Multicast Communication in Wormhole-Routed Torus Networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1029–1042, Oct 1995.

[21] H.D. Schwetman. Using CSIM to Model Complex Systems. In *Proceedings of the 1988 Winter Simulation Conference*, pages 246–253, 1988.