Can Scatter Communication Benefit from Multidestination Message Passing? [1]

Mohammad Banikazemi 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: {banikaze,panda}@cis.ohio-state.edu

Contact Author: Dhabaleswar K. Panda

Abstract: Multiple research directions in the past have emphasized the benefits of multidestination message passing to support efficient *non-personalized* communication like broadcast and multicast. In this paper, we investigate whether *personalized* communications like *scatter* can benefit by using multidestination message-passing support, if available in a system. In order to use such support, we propose a new approach where personalized data for a set of destinations can be grouped together and sent as a single multidestination broadcast/multicast worm. This approach is shown to use less number of communication steps while achieving the lower bound for scatter communication. Using this approach, we propose two schemes, *sequential multidestination tree-based* (SMT) and *hierarchical multidestination tree-based* (HMT), to implement one-to-all scatter in k-ary n-cube wormhole systems. Performance of these two schemes are compared with two commonly used unicast-based schemes, *sequential tree-based* (ST) and *binomial tree-based* (BT). Analytical and simulation studies are carried out to determine the scatter latency under the four schemes for k-ary n-cube systems with varying communication start-up times (t_s), propagation times (t_p), and message lengths (l, size of data being sent to each destination). The study indicates that for $l \leq \frac{t_s}{kt_p}$, the scatter can be best implemented by the HMT scheme. Similarly, the SMT scheme is shown to outperform where $\frac{t_s}{kt_p} < l \leq \frac{t_s}{t_p}$. For $l > \frac{t_s}{t_p}$, both ST and SMT perform better than other algorithms. Simulation results on an 8×8 system with t_s=10.0 microsec show that the HMT and SMT schemes reduce scatter latency up to a factor of 30 and 8 compared to the ST scheme for small message lengths. These results confirm that scatter communication can in fact take advantage of multidestination message passing support.

Keywords: Parallel architecture, collective communication, scatter, personalized communication, wormhole routing, k-ary n-cubes, path-based routing, and multidestination message passing.

[1] 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 has become the trend in building parallel systems due to its inherent advantages like low-latency communication and reduced communication hardware overhead [6, 10, 11, 27, 8]. Intel Paragon [17], Cray T3D [9], Ncube-3 [12], MIT J-Machine [28], Stanford FLASH [13], Myrinet [5, 30] use the wormhole-routing switching technique. Such systems are being used for supporting either distributed-memory, shared-memory, or distributed-shared memory programming paradigms. The need for *collective communication* operations such as *broadcast, multicast, global reduction, scatter, gather, complete exchange*, and *barrier synchronization* arises frequently in parallel applications [33, 7, 2, 3, 15, 23, 26]. The usage of collective communication operations simplify the programming of parallel applications and facilitate the implementation of efficient and portable communication schemes on different machines. The Message Passing Interface Standard [25] has recently defined the importance of collective communication operations.

One of the important collective communication operations is *one-to-all personalized* transfer, or simply known as *scatter* [14, 21, 18]. In this operation, one process sends different messages (each of l bytes long) to every other process of an user-defined group. It has been shown recently that current collective communication libraries use either a *sequential tree-based* (ST) scheme or *binomial tree-based* (BT) scheme to support scatter communication. Both of these algorithms use only unicast messages. The ST scheme is used on IBM SP2 under the MPICH implementation [29, 36] and the BT scheme under the MPL implementation [36].

New generation interconnection networks are providing hardware support for multicasting. MPP systems falling into this category include Meiko [4] and NEC Cenju [20]. Networks of Workstations supporting ATM switching [1] also provide such support. For wormhole networks, multidestination message-passing scheme has been recently proposed and used to implement a set of collective communication operations using *non-personalized* message exchanges [22, 34, 32, 16, 33, 31]. These operations include multicast, broadcast, reduction, and barrier synchronization. With the availability of such hardware multicasting support in a system, an interesting question arises whether personalized communications like *scatter* can also take advantage of it and if yes, under which conditions these schemes can perform better than unicast-based schemes.

In this paper, we take on such challenges and investigate how multidestination message passing can be used to implement scatter communication efficiently. We propose a new approach where personalized data for a set of destinations can be grouped together and sent as a single multidestination broadcast/multicast worm. Using this approach, we propose two schemes, *sequential multidestination tree-based* (SMT) and *hierarchical multidestination tree-based* (HMT), to implement one-to-all scatter in k-ary n-cube wormhole systems. These schemes group destinations along base-routing-conformed-paths (BRCP) [34] in a system.

In order to develop the best possible algorithms using multidestination message-passing, we use the lower bound analysis. First we analyze the scatter latencies for unicast-based schemes and determine the conditions under which these schemes achieve the lower bound for scatter communication. Next, while designing the SMT algorithm, for different values of communication start-up time (t_s), propagation time (t_p), and message length (l), we choose appropriate size for grouping the destinations so that $t_s \leq lt_p$. This allows us to achieve the lower bound for a wide range of system parameters and message lengths. In the HMT algorithm, we use a multi-level grouping scheme and use multidestination worms to deliver the messages to the group leaders. Then, the group leaders deliver the messages to the members of their groups.

Both analytical and simulation experiments are used to evaluate all these algorithms for a range of system and technological parameters (t_s, t_p, l, and k). We also consider simulation experiments for networks with fixed packet sizes. The results indicate the HMT scheme being the best for $l \leq \frac{t_s}{kt_p}$. Similarly, the SMT scheme is shown to be the best for the following range of message lengths: $\frac{t_s}{t_p k} < l \leq \frac{t_s}{t_p}$. For message lengths larger than $\frac{t_s}{t_p}$, both SMT and ST schemes are shown to perform better. These results show that scatter communication for small and medium message lengths can in fact take advantage of multidestination message passing support to reduce scatter latency considerably.

The paper is organized as follows. The unicast-based ST and BT schemes for scatter communication are discussed in section 2. An overview of the multidestination mechanism is presented in section 3. In section 4, we develop the SMT and HMT schemes using multidestination messages and propose algorithms for k-ary n-cube systems. Simulation experiments and results are presented in section 5. Finally, we conclude the paper with conclusions and future research directions.

2 Current Approaches for Scatter Communication Using Unicast Message Passing

In this section, we define the scatter collective communication operation and derive its lower bound for wormhole-routed systems. Then, we present two unicast-based scatter algorithms which are used on current systems. We also investigate the conditions under which these algorithms can meet the lower bound.

2.1 Scatter Communication and Its Lower Bound

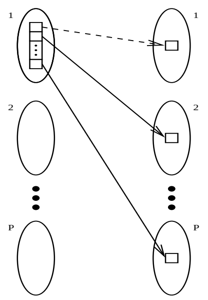

Figure 1: The semantic of scatter operation in a P-processor system.

Scatter is one of the personalized communication operations in which one processor (source node) sends a different message (each of equal length, l) to every other processor in an user-defined group (Fig. 1). Without loss of generality, and for the rest of this paper, we assume that the original data at the source node is ordered in a column-major fashion. Let us try to estimate the lower bound for this communication. Consider contemporary parallel systems like IBM SP2 [35] and Cray T3E, which provide single injection channel per node. In such systems with P processors and message size of l bytes, in the best case, at least one message needs to be sent out. On the other hand, the source node needs to inject $(P-1) \times l$ bytes of data into the interconnection network. Therefore, the lower bound (LB) for scatter operation in a wormhole routed system [2] can be derived as:

$$T_{LB} = t_s + (P-1)lt_p \tag{1}$$

where t_s is the start-up overhead and t_p is the transmission time per byte.

2.2 Sequential Tree (ST) Algorithm

The ST algorithm (also known as *separate addressing*) is based on a *sequential tree* [29] and is used on IBM SP2 under the MPICH implementation [29, 36]. In this algorithm the source node sends a separate message to each of the other processors participating in scatter. Therefore, the number of required communication steps is $P-1$. Figure 2a shows the sequential tree for a system with 9 processors. The implementation of ST on a 3×3 mesh/torus is illustrated in Fig. 2b. This algorithm is formally presented in Fig. 3.

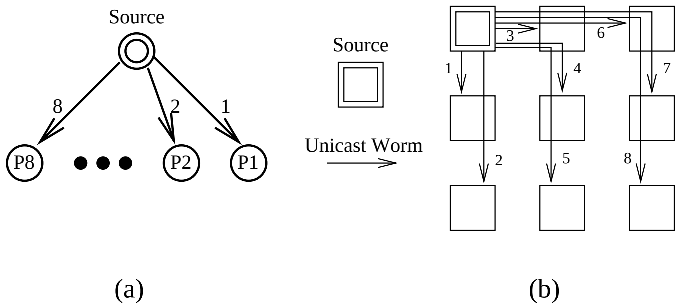

Figure 2: Sequential tree (ST) algorithm on a 3×3 mesh/torus.

Let us analyze the latency of the scatter communication under the ST algorithm. Depending on the system parameters, two different cases can be recognized: $t_s \leq lt_p$ and $t_s > lt_p$. Thus, the scatter latency can be estimated as:

$$T_{ST} = \begin{cases} t_s + (P-1)lt_p & \text{if } t_s \leq lt_p \\ (P-1)t_s + lt_p & \text{if } t_s > lt_p \end{cases} \tag{2}$$

It is obvious that only if $t_s \leq lt_p$ then the lower bound can be met. If this condition is not satisfied then the latency of the scatter operation will exceed the lower bound. Because of the $(P-1)t_s$ term, the latency of scatter can increase rapidly when the system size increases or the start-up overhead is significant. This leads to the following observations:

[2] For simplicity, we ignore the impact of router delay and distance for analytical modeling. However, in the simulation experiments in Section 5, we consider all network parameters in detail.

The ST Algorithm

Input: root: the source node of scatter; msg: the address of the message; l: message length; P: number of participating nodes.

Output: Each node receives its portion of data.

```
/* Strt returns the starting address of the message which should be transmitted to a specific
   node. */

procedure scatter(root, msg, l, P)
begin
    if (MyId = root) then
      for i = 0 to P-1 do
      begin
          if ( i != MyId ) then
          begin
              tmsg = Strt(msg, i, l);
              send tmsg to i length l;
          end
      endfor;
    else
       receive msg from root;
end
```

Figure 3: Pseudo-code for the ST algorithm.

Observation 1 *For $l \geq \frac{t_s}{t_p}$, the ST algorithm can achieve its lower bound.*

Observation 2 *With $l < \frac{t_s}{t_p}$, the latency of scatter increases rapidly with increase in system size.*

2.3 Binomial Tree (BT) Algorithm

The BT algorithm is based on the well known *binomial tree* [24] used for implementing fast broadcast/multicast. Such an algorithm is used in the MPL library [36] on IBM SP2 for scatter implementation. In this approach, the number of processors which receive the message is doubled in each communication step. Figure 4 illustrates the BT algorithm on an 8-processor system. In the first step, the source node ($P0$) sends the data for half of the processors ($P1, P2, P3$, and $P4$) to node $P1$. In the second step, $P0$ sends the data of $P5$ and $P6$ to processor $P5$. At the same time $P1$ sends the data of $P2$ and $P3$ to $P2$. Finally, processors $P0$, $P1$, $P2$, and $P5$ send the appropriate data to $P7$, $P4$, $P3$, and $P6$, respectively. It is to be observed that the message length is reduced during each successive steps. Figure 5 illustrates the implementation of the BT scatter algorithm on a 4×4 mesh/torus. The BT algorithm is formally presented in Fig. 6.

Let us analyze the scatter latency using the BT algorithm. Since the number of processors which receive their messages is doubled in each step, the number of required communication steps for the entire operation is $\log_2 P$. On the other hand, the data size decreases by a factor

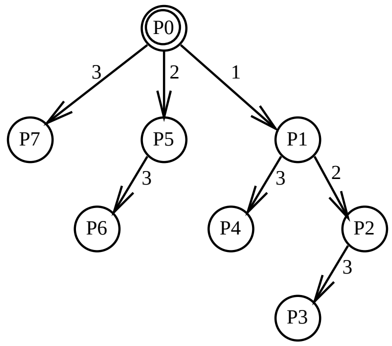

Figure 4: Binomial Tree (BT) algorithm on an 8-processor system.

of two in every successive step. Therefore, the latency of scatter using BT algorithm can be estimated as:

$$
\begin{aligned}
T_{BT} &= \sum_{i=1}^{\log_2 P} t_s + \frac{l}{2^i} t_p \\
&= \log_2 P t_s + (P-1) l t_p \qquad (3)
\end{aligned}
$$

It can be seen that the BT algorithm can never meet the lower bound for scatter communication (as derived in Eqn. 2) due to additional number of start-ups. This leads to:

Observation 3 *The BT algorithm is not capable of achieving the lower bound with respect to t_s. By increase of message size the effect of startup overhead decreases and the percentage of the difference between BT and lower bound becomes smaller and smaller.*

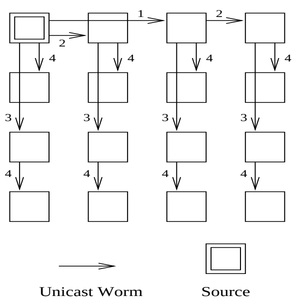

Figure 5: BT algorithm on a 4×4 mesh/torus.

3 Overview of Multidestination Message Passing

In unicast or *point-to-point* message passing, each message has exactly one destination. In contrast, systems with multidestination message passing allows a multicast/broadcast message to be absorbed at one of the destination nodes while being forwarded (i.e., *absorb-and-forward*) to the next node concurrently [22, 34]. The header of multidestination messages consist of

The BT Algorithm

Inputs: root: the source node of scatter; msg: the address of the message; l: message length; P: number of participating nodes.

Output: Each node receives its portion of data.

/* AbsID converts the relative ID of a node with respect to the root to the real ID of that node in the system. */

```
procedure scatter(root, msg, l, P)
begin
    NumSteps= log_2 P;
    for i = 1 to NumSteps do
    begin
        diff = 2 ** (NumSteps-i);
        couples = P / (2*diff);
        for j = 0 to couples-1 do
        begin
            /* AbsID converts the relative ID of a node with respect
            to the root to the real ID of that node in the system. */
            sndr = AbsID(root, 2*j*diff);
            rcvr = AbsID(root, sndr+diff);
            if (MyId = sndr) then
              send msg to rcvr length (P*l)/(2**i);
            else
              receive msg from sndr;
        endfor;
    endfor;
end
```

Figure 6: Pseudo-code for the BT algorithm.

multiple destinations as an ordered list, depending on their intended order of traversal. A Hamiltonian path-based routing scheme has been introduced in [22] to use multidestination messages. Recently a Base-Routing-Confirmed-Path (BRCP) model has been proposed in [34] to apply multidestination message passing to dimension-ordered and other adaptive routing schemes.

Formally, the BRCP model can be described as follows. Let us assume a network supporting a deadlock-free routing scheme $\mathcal{R}$ for unicast message passing. We identify $\mathcal{R}$ as the *base-routing*. A multidestination worm with an ordered destination list $d_1, d_2, \ldots, d_{n-1}, d_n$ on a network supporting routing scheme $\mathcal{R}$ conforms to the this base routing if the destination set $d_1, d_2, \ldots, d_{n-1}$ can be covered as intermediate nodes on one of the possible paths from s to d_n under the routing constraint $\mathcal{R}$. Thus, a multidestination worm gets routed in a *piece-wise* manner from the source to the final destination through intermediate destinations. Such a worm is capable of delivering data to a set of destinations using a single message passing step.

We now show how multidestination message passing can be used to implement efficient one-to-all scatter on wormhole-routed systems. Although we only consider one-to-all scatter in this paper, the technique used in [19] for multiple multicast can be used to implement all-to-all scatter.

4 Multidestination-based Scatter Algorithms

In this section we use multidestination message passing to develop better algorithms for scatter. First, we provide the motivation behind such algorithms. Next, we propose two generalized algorithms. After that we propose algorithms for 2D tori/meshes and k-ary n-cubes. Finally, we compare these algorithms with the ST and BT algorithms. As mentioned earlier, we assume that the original data at the source node is ordered in a column-major fashion. In both proposed algorithms, the set of destinations of any multidestination worm is chosen carefully so that the messages sent by one worm occupy a continuous portion of memory. This eliminates copying of data blocks into a contiguous memory space before sending a message.

4.1 Motivation

As presented in Observations 1 and 3, only ST algorithm can achieve the lower bound. Observation 1 also shows that the ST algorithm can achieve the lower bound just for limited variations of system parameters and message sizes. This lower bound is achieved only when the start-up overhead of sending messages are overlapped with the transmission time of other messages. To achieve this requirement, the $t_s \leq lt_p$ condition should hold. In other words, the size of the messages being transmitted should be large enough so that the next message can become ready for transmission no later than the time when the previous message has been completely injected to the interconnection network ($l \geq \frac{t_s}{t_p}$). In this section, we propose a couple of new algorithms that reduce the scatter latency by increasing the size of the messages which are being transmitted. The multidestination message passing mechanism helps us to achieve this goal.

4.2 Sequential Multidestination Tree (SMT) Algorithm

In the SMT algorithm, all processors which are participating in the scatter communication operation are grouped into smaller groups. The messages for all processors in each group are combined to form a multidestination message. Then, these different multidestination messages are sent out to the members of the corresponding groups. Depending on the routing scheme being used in the system and the arrangement of the data at the source node different groupings can be used. For example, as illustrated in Fig. 7, nodes in a 12-processor system are grouped into four groups (G1-G4). With such grouping, the source node uses four messages to cover these groups. If the smallest group of processors has g ($g \geq 1$) members then the size of the shortest message that is going to be transmitted is gl. Therefore, the condition under which the lower bound is met can be written as: $gl \geq \frac{t_s}{t_p}$. The minimum value of g to achieve the lower bound can be found from the following equation:

$$g_{min} = \frac{t_s}{lt_p} \tag{4}$$

Obviously, for bigger g's there are less constraints on system parameters and message size under which the lower bound can be achieved. However, the need for rearrangement of the original data and the routing constraints associated with a network don't allow selecting any arbitrary g's. Assuming that all groups have g members, the execution time of the SMT scatter algorithm can be derived as:

$$T_{SMT} = \begin{cases} t_s + (P-1)lt_p & \text{if } ts \leq lgt_p \\ \frac{P}{g}t_s + (g-1)lt_p & \text{if } ts > lgt_p \end{cases} \tag{5}$$

Observation 4 *In cases where $l \geq \frac{t_s}{gt_p}$, the SMT algorithm can achieve the lower bound.*

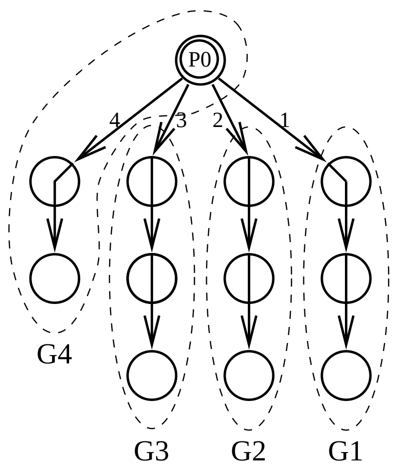

Figure 7: Illustration of the Sequential Multidestination Tree (SMT) algorithm.

Using this generalized approach we show how this algorithm can be used to implement scatter on 2D meshes/tori and higher dimensional systems with dimension-ordered routing.

4.2.1 The SMT Algorithm for 2D Tori and Meshes

In a $k \times k$ torus, it can be easily observed that the processors of each column can form a group. A multidestination message can deliver the data for all members of each group under dimension-ordered routing. The minimum value of g for which the lower bound latency is achievable can be found from Eqn. 4. The latency of scatter can be found from Eqn. 5. If $g_{min} \leq k$ then the lower bound is achieved. Figure 8 illustrates the implementation of the SMT algorithm on a 4×4 torus. Detailed steps for implementing the SMT algorithm on a $k \times k$ torus is presented in Fig. 9.

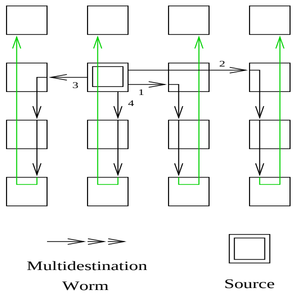

Figure 8: Sequential Multidestination Tree (SMT) algorithm on a 4×4 torus.

Let us consider applying the SMT algorithm to meshes. Exactly the same approach as used for tori, can be used for meshes when the source node is in the up-most row of the mesh (Fig. 10). In this case, the latency of scatter on a mesh will be the same as that on a torus and can be calculated from Eqn. 5. However, if the source node is not in the up-most row, an extra step is required to transfer the data to one of the nodes in the first row. This will increase the overall execution time of scatter by one additional communication step.

4.2.2 The SMT Algorithm for k-ary n-cubes

The SMT algorithm for 2D tori can be easily generalized for k-ary n-cubes. To implement the SMT algorithm on k-ary n-cubes with dimension-ordered routing, it is enough to partition the processors along the lowest dimension and select g equal to k. One multidestination message is enough for delivering the messages of all members of each group. The scatter latency can be found from Eqn. 5 and the lower bound is met when $g_{min} \leq k$. The detailed steps for implementing the SMT algorithm on a k-ary n-cube is presented in Fig. 11.

4.2.3 Performance Benefits

As mentioned in Observations 1 and 4, both the ST and SMT algorithms can achieve the lower bound under some conditions. The condition under which the latency of the ST algorithm will not exceed the lower bound is $l \geq \frac{t_s}{t_p}$. On the other hand, $l \geq \frac{t_s}{gt_p}$ condition should hold so that the SMT algorithm can achieve the lower bound latency. From these conditions, it is obvious that the SMT algorithm provides a more relaxed constraint. From Observation 4 we can conclude that smaller communication start-up overhead (t_s) will increase the number of

The SMT Algorithm for 2D Tori

Inputs: root: the source node of scatter; msg: the address of the message; l: message length; k: number of nodes in each dimension.

Output: Each node receives its portion of data.

```
/* Strt returns the starting address of the message which should be transmitted to a specific
   node.
   NodeRowCol converts the ID of a node to the row and column in which it is located.
   NodeID finds the ID of a node located at a specific row and column. */

procedure scatter(root, msg, l, k)
begin
    if (MyId = root) then
      for i = 0 to k-1 do
      begin
          NodeRowCol(MyId, row, col);
          dests = {};
          for j = 0 to k-1 do
              dests = NodeID(j, mod(col+i, k-1) ) ∪ dests;
          tmsg = Strt(buffer, NodeID(0, mod(col+i,k-1)), l);
          sendmulti tmsg to dests length k*l;
      endfor;
    else
       receive msg from root;
end
```

Figure 9: Pseudo-code for the SMT algorithm for 2D tori.

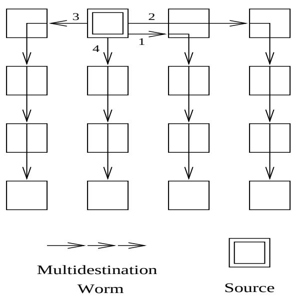

Figure 10: SMT algorithm on a 4 × 4 mesh.

The SMT Algorithm for k-ary n-cubes

Inputs: root: the source node of scatter; buffer: the address of the message; l: message length; k: number of nodes in each dimension. n: number of dimensions.

Output: Each node receives its portion of data.

```
/* Strt returns the starting address of the message which should be transmitted to a specific
   node. */

procedure scatter(root, buffer, l, k, n )
begin
  if (MyId = root) then
    for d_{n-1} = 0 to k-1 do
      for d_{n-2} = 0 to k-1 do
        •
          •
            •
              for d_1 = 0 to k-1 do
                begin
                  dests = {};
                  base = d_{n-1}, d_{n-2}, ···, d_1;
                  for d_0 = 0 to k-1 do
                    dests = dests ∪ "base,d_0";
                  /* Strt returns the starting address
                  of the message of node i */
                  msg = Strt(buffer, "base,0");
                  multisend msg to dests length k*l;
                endfor;
  else
    receive msg from root;
end
```

Figure 11: Outline of the SMT algorithm for k-ary n-cubes.

cases for which the lower bound is achievable. It can also be seen that having a bigger k will have the same effect. However, decreasing the per-byte transmission time (t_p) will have the opposite effect. The effect of changes in system parameters and size on the minimum message length for which the lower bound is achievable (l_{min-lb}) is summarized in Table 1. In the next section, we analyze and compare the effects of these parameters on the latency of the ST, BT, and SMT algorithms for a range of system and technological parameters.

4.3 Hierarchical Multidestination Tree (HMT) Algorithm

In this algorithm destination nodes are partitioned into disjoint groups with a *leader* node representing each group. The leader node is chosen in such a way that it can deliver a message to the members of its group using a single multidestination worm. These nodes can be further partitioned into disjoint groups with a new set of leaders. This process of hierarchical grouping

Table 1: The effect of changes in system parameters on the minimum message length for which the lower bound is achievable.

P	t_s	t_p	l_{min-lb}
$\uparrow$	fixed	fixed	$\downarrow$
fixed	$\downarrow$	fixed	$\downarrow$
fixed	fixed	$\downarrow$	$\uparrow$

is continued until a single group whose leader is the root of scatter operation is obtained. After the grouping is achieved each lower level leader receives a single message from its leader and forwards it to the members of its group. For example, as illustrated in Fig. 12, in the first step of grouping, nodes in a 12-processor system are grouped into four groups ($G1-1$ - $G1-4$). In the second step of grouping, leaders of these 4 groups make another group ($G2-1$) to which the scatter root node belongs. With such grouping, in the first phase of scatter, the root node will send a single multidestination worm to all members of $G2-1$ group. This message contains all information which is supposed to be delivered to the nodes in $G1-2$, $G1-3$, and $G1-4$ groups. In the second phase, leaders of $G1$ groups delivers the messages to their members using a single multidestination worm.

In general, if we have r levels of grouping then the execution time of the HMT scatter algorithm can be written as:

$$T_{HMT} \quad = \quad rt_s + (P-1)lt_p \tag{6}$$

The above estimation of latency leads to the following observation:

Observation 5 *The HMT algorithm can not achieve the lower bound for scatter communication. However, with small number of grouping levels the latency is very close to the lower bound.*

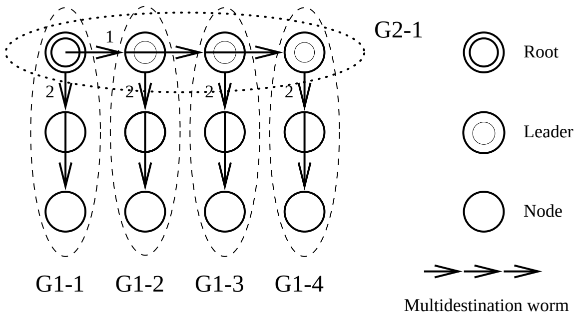

Figure 12: Illustration of the HMT algorithm.

Using this generalized approach we show how this algorithm can be used to implement scatter on 2D meshes/tori and higher dimensional systems with dimension-ordered routing.

4.3.1 The HMT Algorithm for 2D Tori and Meshes

In a $k \times k$ torus, the first level grouping can be done along columns. In each group, the node which is located in the row that scatter root is located in is chosen as the leader. All these leaders form a second level group. Then, scatter is performed in two phases. In the first phase, the root uses a single multidestination worm to deliver the messages of all nodes except the nodes which are in its column to the first level leaders. In the second phase, leaders (including the root) forward the messages to other nodes in the first level groups. Figure 13 illustrates the implementation of the HMT algorithm on a 4×4 torus. The scatter execution time can be obtained from Equation 6 and is equal to:

$$T_{HMT} \quad = \quad 2t_s + (P-1)lt_p \tag{7}$$

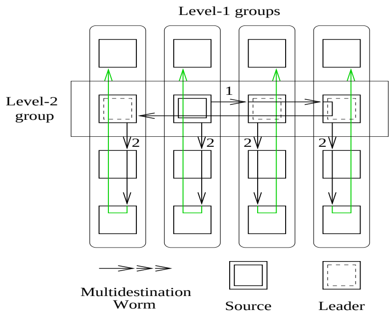

Figure 13: Hierarchical (HMT) algorithm on a 4×4 torus.

Exactly the same approach as used for tori, can be used for meshes when the source node is in the one of the four corners of the mesh. In this case, the latency of scatter on a mesh will be the same as that on a torus and can be calculated from Eqn. 7. However, if the source node is not in these columns, one or two extra steps may be required (Fig. 14). This will increase the overall execution time of scatter by at most $2 \times t_s$.

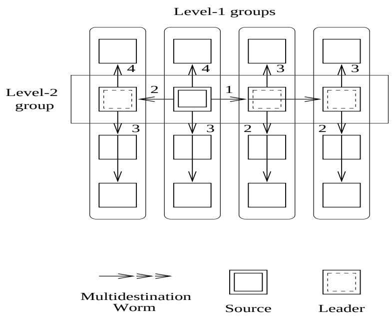

Figure 14: Hierarchical (HMT) algorithm on a 4×4 mesh.

4.3.2 The HMT Algorithm for k-ary n-cubes

The HMT algorithm for 2D tori can be easily generalized for k-ary n-cubes. To implement the HMT algorithm on k-ary n-cubes with dimension-ordered routing, n partitioning steps are required. In each partitioning step, nodes are partitioned along one dimension. The scatter operation is implemented in n phases. In each phase, multidestination messages are sent along one dimension. The execution time can be written as:

$$T_{HMT} \quad = \quad nt_s + (P-1)lt_p \tag{8}$$

The detailed steps for implementing the HMT algorithm on a k-ary n-cube is presented in Fig. 15.

The SMT Algorithm for k-ary n-cubes

Inputs: root: the source node of scatter; buffer: the address of the message; l: message length; k: number of nodes in each dimension. n: number of dimensions.

Output: Each node receives its portion of data.

```
/* Strrt returns the starting address of the message which should be transmitted to a specific
   ordered set of nodes.
   DimPos returns the position of a given node in a given dimension.
   SetDim sets the value of a given address to a new value at a specified dimension. */

procedure scatter(root, buffer, l, k, n )
begin
    for dim = 0 to n-1 do
        if (DimPos(MyId, dim) == 0) then
        begin
          dests = {};
          for grp = 1 to n-1 do
              dests = dests ∪ SetDim(MyId, dim, grp);
          msg = Strt(buffer, dests);
          multisend msg to dests length (k − 1)^(n−dim) * l;
        end
        else
          receive msg from SetDim(MyId, dim, 0);
end
```

Figure 15: Outline of the HMT algorithm for k-ary n-cubes.

4.3.3 Performance Benefits

From Observation 1 we know that the ST algorithm can achieve the lower bound if the message size is big enough and, Observation 3 suggests that BT can not achieve the lower bound. On the other hand, as mentioned in Observation 4, the SMT algorithm will achieve the lower bound for messages bigger than a certain size in a given system. By comparing Equations 2, 3, and 6 it can be seen that although BT doesn't reach the lower bound it performs better than both ST and SMT algorithms for smaller messages. Since the HMT algorithm introduces just one extra t_s (for 2D systems) to the scatter latency compared to the BT algorithm which introduces $\log_2 Pt_s$, it can be seen that the HMT algorithm performs better than all other scatter algorithms for small messages. However, the SMT algorithm outperforms the HMT algorithm for larger messages.

From Observation 1 we know that ST reaches the lower bound for messages of size $\frac{t_s}{t_p}$ and bigger. On the other hand, in Observation 4, we saw that SMT can reach the lower bound under a more relaxed constraint of $l \geq \frac{t_s}{kt_p}$ for a k-ary n-cube. For messages smaller than $l \geq \frac{t_s}{kt_p}$, we showed that although none of the presented algorithms can reach the lower bound, HMT outperforms all other algorithms. Thus, the guideline for selecting the best scatter algorithm for a specific system and a given message length is as shown in Table 2.

Table 2: Guideline for selecting the best scatter algorithm.

Message Size	$l \leq \frac{t_s}{kt_p}$	$\frac{t_s}{kt_p} < l \leq \frac{t_s}{t_p}$	$l > \frac{t_s}{t_p}$
Best Scatter Algorithm	HMT	SMT	ST or SMT

5 Simulation Experiments and Results

To compare the scatter algorithms, we simulated single source scatter on a CSIM-based simulator testbed. This testbed implements flit-level simulation and can be configured for different topologies and technological parameters. For all simulation experiments, we assumed system and technological parameters representing the current trend in technology. The following default parameters were used: t_s (communication start-up time) = 10.0 microseconds, t_p (including switching and physical link propagation time) = 15.0 nanoseconds, t_{inj} (time to inject a flit into network) = 5.0 nanoseconds and t_{cons} (time to consume a flit from network) = 5.0 nanoseconds. The header flit propagation time was set to 20.0 nanosecond for unicast worms and 40.0 nanoseconds for multidestination worms. We compared ST, BT, SMT, and HMT algorithms and investigated the effect of the variations of system parameters. Since the latency of the ST algorithm for short messages is very high in comparison with that of other algorithms, we do not show the ST latency curves in the graphs for short messages.

5.1 Effect of Communication Start-up

Figures 16 and 17 shows the comparison results for three different start-ups ($t_s = 5.0, 10.0$, and 20.0 microsec). It can be seen that the SMT algorithm performs better than the ST algorithm for all different message sizes. It can also be verified that the SMT performs better than BT for a wide range of message sizes. As t_s decreases the minimum message length for which the lower bound can be achieved decreases ($l \geq \frac{t_s}{gt_p}$). With $t_s = 10.0$ microsec, the SMT algorithm achieves the lower bound for messages of size 128 bytes and more. With $t_s = 5.0$ and 1.0 microsec, the lower bound is achievable for messages of size 64 and 16 bytes, respectively. On the other hand it can be seen that the HMT algorithm outperforms all other algorithms for short messages, and the only algorithm that perform better than HMT for long messages is the SMT algorithm. The HMT scatter latency when message size is 1 byte is 22 microsec compared to 630 microsec, 61 microsec, 80 microsec for ST, BT, and SMT algorithms, respectively.

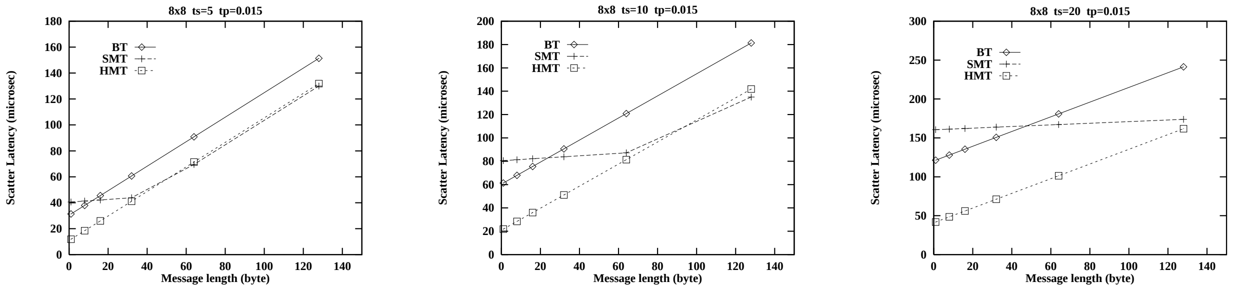

Figure 16: Latency of three different algorithms for different start-up times (small message lengths).

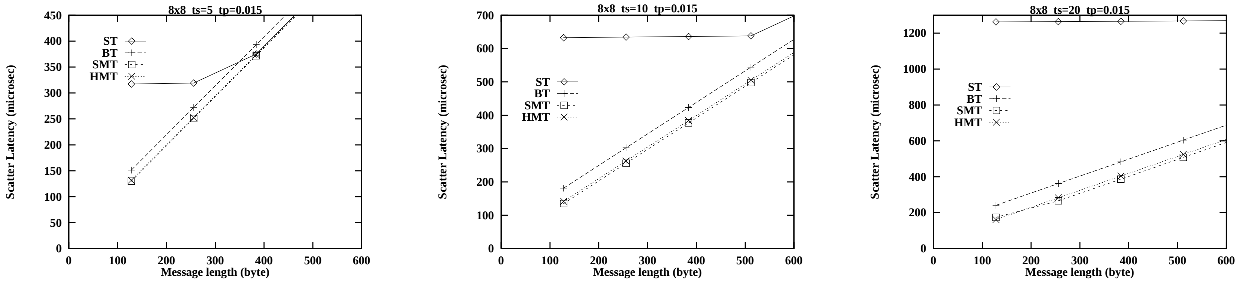

Figure 17: Latency of three different algorithms for different start-up times (large message lengths).

Figure 18 shows the factor of improvement of the SMT and the HMT algorithms over the ST algorithm for three different startups. It can be observed that the ST algorithm can never perform better than these two algorithms. For a 1-byte message size, the SMT algorithm and the HMT algorithm can reduce the scatter latency compared to the ST algorithm up to a factor of 8 and 30, respectively. As t_s increases, the multidestination-based schemes perform better than the ST scheme for a given message length. Similarly, Fig. 19 shows the factor of improvement of SMT and HMT over the BT scheme. It can be observed that the BT scheme can perform better than the SMT only for very short messages. For other message sizes, the SMT algorithm can show up to 60% improvement compared to the BT algorithm. It can be seen that as t_s reduces the maximum message size for which the BT scheme can perform better than the SMT algorithm decreases. As t_s keeps on reducing for new generation systems, these results indicate that the SMT algorithm can outperform the unicast-based algorithms (ST and BT) for a wider range of system parameters and message sizes. On the other hand, the HMT algorithm outperforms the BT scheme for all different system parameters and can show up to 200% improvement compared to the BT algorithm.

5.2 Effect of System Size

The effect of system size on the scatter algorithms is shown in Figs. 20 and 21. It is interesting to observe that by increasing the system size the minimum length requirement for achieving the lower bound in the SMT algorithm decreases. It can be seen that while the SMT algorithm can not achieve the lower bound for messages shorter than 128 bytes on a 4×4 system, the

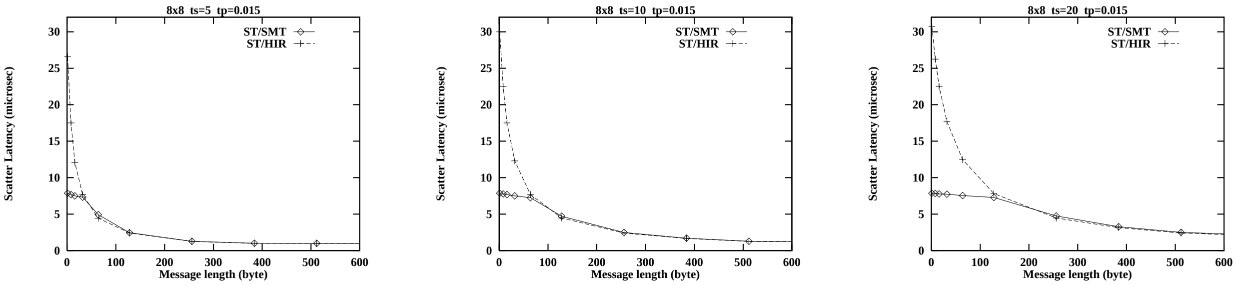

Figure 18: Factor of improvement (ST/SMT and ST/HMT) for different start-up times.

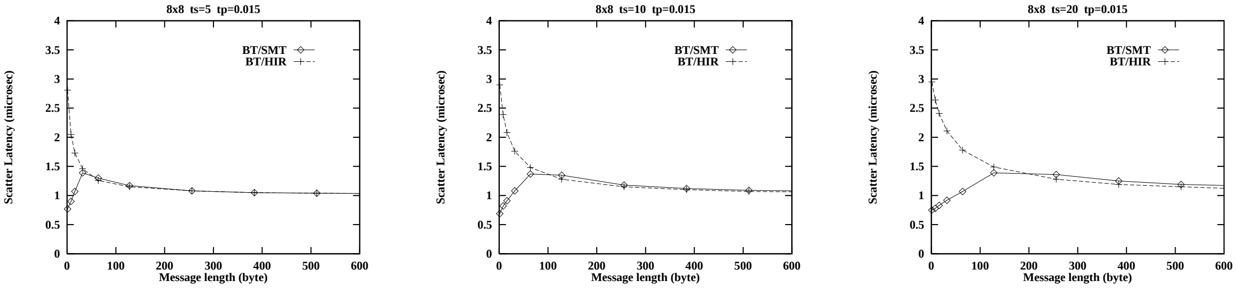

Figure 19: Factor of improvement (BT/SMT and BT/HMT) for different start-up times.

lower bound is achieved even for 64-byte and 16-byte messages on 8×8 and 16×16 systems, respectively. Such performance improvement demonstrates the scalability of the SMT algorithm with respect to increase in system size. On the other hand, the HMT algorithm outperforms the BT algorithm for different message sizes. However, as system size increases, the minimum message size for which SMT performs better than HMT decreases.

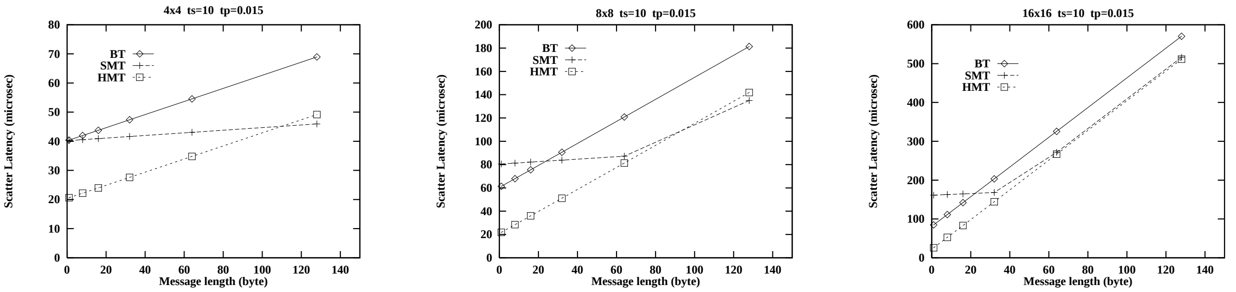

Figure 20: Latency of three different algorithms for different system sizes (small message lengths).

5.3 Effect of Propagation Cost

Figures 22 and 23 show the latency of scatter algorithms for different propagation times ($t_p = 10.0, 15.0,$ and 25.0 nsec). It can be observed that as propagation time increases, the factor of improvement obtained by using either the SMT algorithm or the HMT algorithm over the BT algorithm increases.

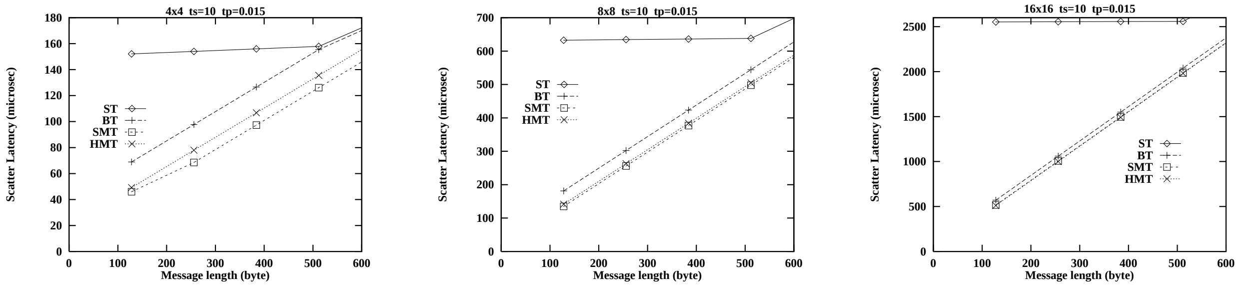

Figure 21: Latency of three different algorithms for different system sizes (large message lengths).

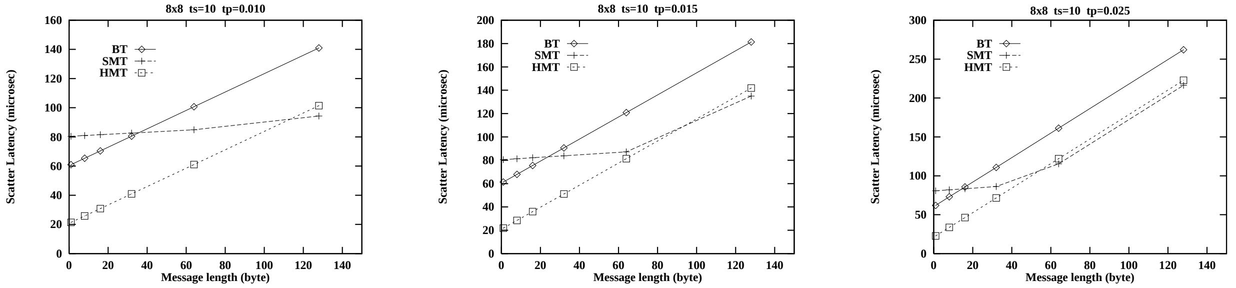

Figure 22: Latency of three different algorithms for different propagation times (small message lengths).

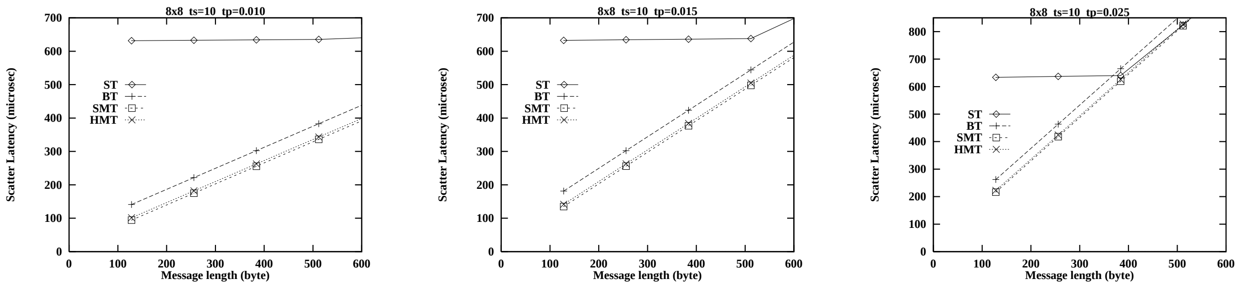

Figure 23: Latency of three different algorithms for different propagation times (large message lengths).

5.4 Effect of Packetization

All the above results assume unlimited packet size. Since many systems impose a maximum size for the packets being transmitted, long messages need to be broken into smaller packets before being transmitted. The effect of packetization on the scatter algorithms under three different contemporary maximum packet sizes (64, 128, and 256 bytes) is shown in Figs. 24 and 25. The startup overhead for these experiments have been set to 9.0 microsec and the overhead for preparing each packet in the Network Interface Card (NIC) has been set to 1.0 microsec. It can be seen that even under packetization constraint, HMT performs better than other algorithms for short messages and SMT outperforms other schemes for long messages.

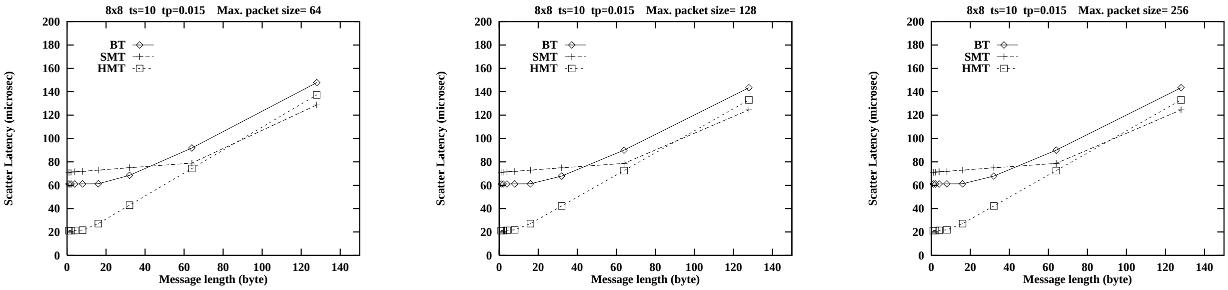

Figure 24: Latency of three different algorithms for different maximum packet sizes (small message lengths).

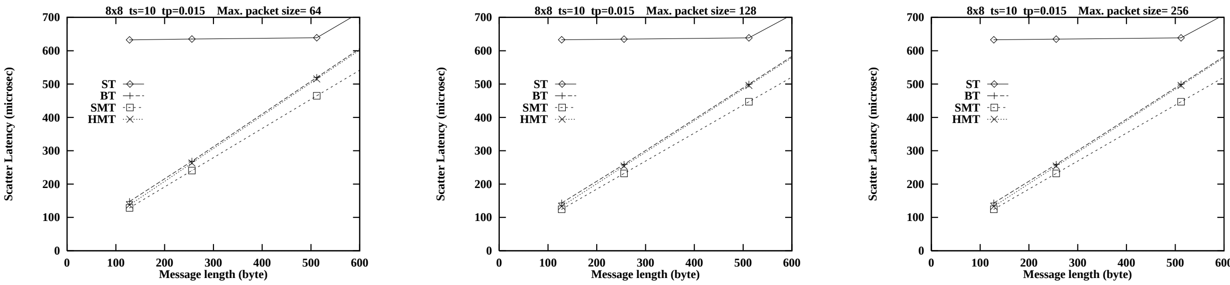

Figure 25: Latency of three different algorithms for different maximum packet sizes (large message lengths).

5.5 Overall Comparison

Simulation results confirm that the guideline presented in Table 2 can be used to choose the most efficient scatter algorithm for any given system size and message length. For an 8×8 torus with $t_s = 5.0$ microsec and $t_p = 0.015$ microsec, it is clear from Fig. 16 that HMT outperforms all other algorithms when l is less than or equal to 64 bytes. The simulation results also show that when $l \geq 512$, ST and SMT performs better than other algorithms (Fig. 17). For cases where $64 < l < 512$, SMT outperforms all other algorithms. The cutoffs observed from the graphs are little different from the guideline presented in Table 2. This is because we considered detailed network modeling (routing delay, injection delay, consumption delay, etc.) in our simulator.

6 Conclusions and Future Research

In this paper, we have investigated whether scatter communication in k-ary n-cube wormhole systems can be benefited by using multidestination message passing mechanism available in a system. We presented two new scatter algorithms using multidestination message passing and compared their performance with two previously known unicast-based algorithms. Compared to the current unicast-based schemes, the proposed schemes can meet the lower bound latency for a much wider range of system, technological, and application parameters. It has been shown that multidestination-based HMT algorithm outperforms all other scatter algorithms for short messages. Similarly SMT algorithm is shown to perform better than all other algorithms for longer messages. Exhaustive analytical and simulation studies have been performed to establish guidelines for selecting suitable algorithms for designing scalable libraries for scatter communication. The study indicates that scatter communication can in fact be benefited by using multidestination message passing if available in a system.

In this paper, we have studied the benefit of multidestination mechanism for personalized scatter communication in k-ary n-cubes under dimension-ordered routing. It will be interesting to evaluate such benefits for other topologies and routing schemes. It will also be challenging to determine whether other personalized collective communication operations like combined reduction and scatter and complete exchange can be benefited by using multidestination message passing.

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] ATM Forum. *ATM User-Network Interface Specification, Version 3.1*, September 1994.

[2] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.-T. Ho, S. Kipnis, and M. Snir. CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computers. In *Proceedings of the International Parallel Processing Symposium*, 1994.

[3] 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.

[4] J. Beecroft, M. Homewood, and M. McLaren. Meiko CS-2 Interconnect Elan-Elite Design. *Parallel Computing*, 20:1627–1638, Nov 1994.

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

[6] R. V. Boppana and S. Chalasani. A Comparison of Adaptive Wormhole Routing Algorithms. In *International Symposium on Computer Architecture*, pages 351–360, 1993.

[7] 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.

[8] Y. M. Boura and C. R. Das. Modeling Virtual Channel Flow Control in Hypercubes. In *International Symposium on High Performance Computer Architecture*, pages 166–175, 1995.

[9] Cray Research, Inc. *Cray T3D System Architecture Overview*, 1993.

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

[11] 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.

[12] B. Duzett and R. Buck. An Overview of the Ncube-3 Supercomputer. In *Proceedings of the Frontiers of Massively Parallel Computation*, pages 458–464, 1992.

[13] M. Heinrich, J. Kuskin, and Others. The performance impact of flexibility in the stanford flash multiprocessor. In *The sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI)*, San Jose, CA, October 1994.

[14] C. T. Ho and S. L. Johnsson. Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes. In *Proc. International Conference on Parallel processing*, pages 640–648, 1986.

[15] C. Huang, Y. Huang, and P. K. McKinley. A Thread-based Interface for Collective Communication on ATM Networks. In *International Conference on Distributed Computing Systems*, 1995.

[16] Y. Huang, C. C. Huang, and P. K. McKinley. Multicast Virtual Topologies for Collective Communication in MPCs and ATM Clusters. In *Proceedings of the Supercomputing*, 1995.

[17] Intel Corporation. *Paragon XP/S Product Overview*, 1991.

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

[19] R. 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*, pages I:188–195, Chicago, IL, Aug 1996.

[20] N. Koike. NEC Cenju-3: A Microprocessor-based Parallel Computer. In *Proceedings of the 8th Int'l Parallel Processing Symposium*, pages 396–401, 1994.

[21] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Cummings, 1994.

[22] 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.

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

[24] 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.

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

[26] P. Mitra, D. G. Payne, and et al. Fast Collective Communication Libraries, Please.

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

[28] M. D. Noakes, D. A. Wallach, and W. J. Dally. The J-Machine Multicomputer: An Architectural Evaluation. In *International Symposium on Computer Architecture*, pages 224–236, 1993.

[29] N. Nupairoj and L.M. Ni. Benchmarking of Multicast Communication Services. Technical Report MSU-CPS-ACS-103, Michigan State University, Sep 1995.

[30] S. Pakin, M. Lauria, and A. Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM). In *Proceedings of the Supercomputing*, 1995.

[31] 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.

[32] D. K. Panda. Global Reduction in Wormhole k-ary n-cube Networks with Multidestination Exchange Worms. In *International Parallel Processing Symposium*, pages 652–659, Apr 1995.

[33] 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.

[34] 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.

[35] C. B. Stunkel, D. G. Shea, B. Abali, et al. The SP2 High-Performance Switch. *IBM System Journal*, 34(2):185–204, 1995.

[36] Z. Xu and K. Hwang. Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2. *IEEE Parallel and Distributed Technology*, pages 9–23, Spring 1996.