A CASE FOR MULTIPATH-BASED MULTICASTS ON RANDOM TOPOLOGIES

Ngo Quang Vinh[1,2], *Hoang Trang*[2], *Vu Dinh Thanh*[2], *Ikki Fujiwara*[3],

Michihiro Koibuchi[3]

Abstract

Network latency is a performance concern especially for collective communication because communication overhead for small messages dominates the system performance in both off-chip and on-chip large-scale interconnection networks. The aggregate path hops strongly affect the network latency in collective communication. In this context we try to minimize the number of total packet path hops for collective communication by introducing a combination of path-based multicastings, especially for random topologies. From our graph analysis results, multicasts based on multiple paths that include all the destinations can reduce the aggregate path hops by up to 5.5% when compared to path-based if the number of destinations is small. Our recommendation to support multicasts for a few destinations is to use multicasting based on multiple paths.

Thời gian trễ trong truyền thông mạng trên chip là một mối quan tâm đặc biệt đối với hoạt động truyền thông đa điểm vì sự truyền thông cho các gói tin nhỏ chiếm phần lớn hiệu năng của hệ thống kết nối trên chip và ngoài chip. Đối với hoạt động truyền thông đa điểm, tổng số hop cần đi qua của một gói tin ảnh hưởng rất lớn tới thời gian truyền. Trong bối cảnh này, chúng tôi cố gắng giảm thiểu số hop tổng cộng để giao tiếp đa điểm bằng phương pháp tổng hợp các đường truyền đa điểm tối ưu khác nhau, đặc biệt là cho các cấu trúc kết nối mạng dạng ngẫu nhiên.Từ kết quả phân tích đồ thị của chúng tôi, phương pháp truyền thông đa điểm dựa trên nhiều đường truyền tối ưu khác nhau đến các điểm đích có thể giảm số hop tổng cộng lên đến 5.5% so với phương pháp dựa vào một đường dẫn tối ưu duy nhất để đi đến các điểm đích khi số lượng điểm đích là nhỏ. Khuyến nghị của chúng tôi là để hỗ trợ truyền thông đa điểm khi số điểm ít là dựa vào nhiều đường truyền thông khác nhau.

Index terms

Interconnection networks, collective communication, random topology

1. Introduction

BOTH on-chip and off-chip interconnection network become latency sensitive as their number of endpoints increases, e.g., 256 cores for on-chip and 100,000 hosts for off-chip. In on-chip interconnection networks, the latency, especially of collective communications, is the primary performance factor, since it directly affects the cache access time and memory-consistency overhead that affects the performance of applications. Similar to on-chip

(1) Integrated Circuit Design Research and Education Center - Vietnam National University - Ho Chi Minh City
(2) Ho Chi Minh City University of Technology - Vietnam National University - Ho Chi Minh City
(3) National Institute of Informatics / JST, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan 101-8430

interconnection networks, in off-chip interconnection networks, large parallel applications to be deployed on next generation High Performance Computing (HPC) systems require ultra-low communication latencies that could reach hundreds of nanoseconds [1], [2], [3].

Router and switch delay is larger than link delay. For example in an on-chip network, a packet can be transferred across a small chip in a single clock cycle for the ideal case where switch delays are ignored [4]. However, on-chip router requires at least one or two clock to transfer a header flit. In an off-chip network, switch delays (e.g., about 100 nanoseconds in InfiniBand QDR) are larger than the wire and flit injection delays.

To achieve low latency in both on-chip and off-chip interconnection networks, a network topology of switches/routers should have low diameter and low average shortest path length, both measured in number of hops. Interestingly, random topologies, which are generated by augmenting classical topologies with random links, achieve low diameter, average shortest path length, and thus end-to-end network latencies [3]. Its routing is reported in [5], and random topologies are discussed for datacenter networks [6]. In addition to the theoritical studies on random structure [7], randomness receives a fair attention for designing interconnection networks.

In this work we consider multicasts, especially for a small number of destinations, in a fully random topology. It is reported that a fully random topology that does not rely on classical structure, e.g. ring or mesh is better than a random topology that includes the classical structure [8]. We thus focus on the fully random topology in this study. The case for small number of destinations is important for on-chip and off-chip HPC networks, because invalidation messages are frequently multicasted so as to maintain coherence.

If the topology is non-random, such as k-ary n-cubes, optimal or semi-optimal multicast methods have been discussed in prior works [9], [10], [11]. Lessons from the work [11] recommend us to use path-based multicasting when the number of destinations is small. The optimal multicast method is path-based in a random topology. Our recommendation is multipath-based.

The rest of this paper is organized as follows. Section 2 surveys multicasts in interconnection networks. Section 4 evaluates influences of multicast methods on path hops through the graph analysis. Finally, Section 5 concludes this paper.

2. Related Work

2.1. Multicast Communications

Hardware-, path-, and unicast-based algorithms are typical methods for multicasts in interconnection networks.

Hardware multicasts duplicate packets at intermediate switches. Since it reduces the aggregate packet hop counts in a multicast, it efficiently sends data to multiple destinations. The typical implementation usually relies on a tree structure. The packet header needs a logic to control packet duplication at intermediate switches and multiple-destination tags.

Another way to support a multicast is to do a large number of unicasts. This is called a unicast-based multicast. In a simple unicast-based multicast, each source sends packets to all destinations.

When a source scatters the same data to all destinations, a binary-tree-based multicast is practical for reducing both the number of packet contentions and the aggregate packet hops. In a binary-tree-based multicast, first a source sends data to a single destination. Then these two nodes send data to four nodes. For d destinations, $log_2(d+1)$ unicast steps are required.

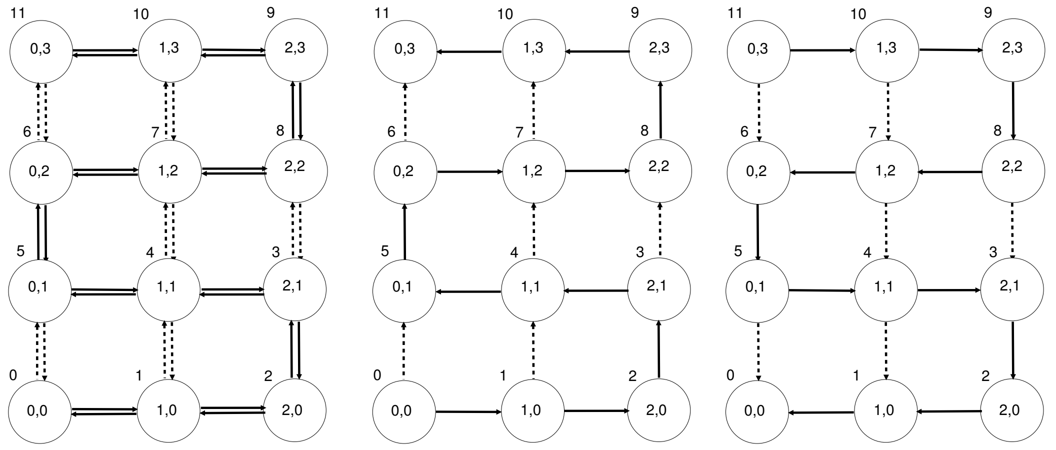
A path-based multicast sends data along a path that includes all destinations. Therefore it requires an efficient multicast-path search, such as a Hamiltonian cycle. To reduce aggregate path hops, the use of multiple paths is recently attempted in k-ary n-cubes [11].

In this work, we explore the path-based multicast mainly for random topologies, and we compare our solution to the unicast-based ones in terms of aggregate path hops.

2.2. Path-based Multicasting

The path-based method is reported as the best multicasting approach for some cases on regular network topology in terms of hop counts. This method is based on the Hamilton path which goes through all nodes in a network exactly once [15]. Figure *1* shows an example of two directed Hamilton paths in a 2-D mesh network. Every node in Figure *1* is assigned a label based on its integer coordinate using the formula in [16]. The labelling effectively divides the network in Figure *1.* (a) into two subnetworks, the *high-channel subnetwork* and the *low-channel subnetwork*, which are showed in Figure *1.* (b) and *1.* (c) respectively. Thus, multicast messages from the source node will always be sent to the destination nodes using the *high-channel subnetwork* if the labels of the destination nodes are greater than that of the source node. Vice versa, if the labels of the destination nodes are smaller than that of the source node, the multicast messages will be sent through the *low-channel subnetwork*. Besides, to reduce the path hops, the vertical links that are not part of the Hamilton path (the dash lines in Figure *1*) could be used if they have the appropriate direction.

In previous works, several path-based multicasting techniques were explored for the 2-D mesh topology, such as dual path (DP) [16], multipath (MP) [16], and column path (CP) [12]. The DP and the MP algorithms are based on Hamilton path described in Figure *1.* In the DP routing algorithm, the destination node set is partitioned into two subsets, namely D_{H} and D_{L}, as shown in Figure *2.* (a). Every node in D_{H} has a higher label number than that of the source node, whereas every node in D_{L} has a lower label number than that of



(a) Undirected Hamilton path (b) Decreasing-order directed path (c) Increasing-order directed path

Figure 1. *Example of a Hamilton path in 2D mesh*

the source node. Therefore, the source node always uses the *high-channel subnetwork* and the *low-channel network* to send the multicast messages to the destinations in D_{H} and D_{L} respectively. Figure *2.* (a) in which the label of each node is showed instead of the integer coordinate of the node illustrates an example of the DP algorithm. In this example, the source node is the node labelled 21 and the destination nodes are 2, 4, 7, 9, 11, 24, 29, 30, 32. The multicast message will follow the two directions, one from the source to the higher-label nodes labelled 24, 29, 30, 32 and one from the source to the lower-label nodes labelled 2, 4, 7, 9, 11. The solid arrows show the path and the direction that a message goes from the source to the destinations while the dash arrows provide a shorter path to go to the destinations if it is possible to do that. Consequently, the thicks arrows in Figure *2.* (a) show the path that the multicast messages go from the source node to the destinations.

The MP algorithm further reduces the aggregate path hops than that of the DP by dividing the destination node set into four parts as shown in Figure *2.* (b). The D_{H} set is further divided into two subsets considering the node location. One subset includes all destination nodes whose *x* coordinates are greater than or equal to that of the source node.The D_{L} set is also divided in a similar way. Figure *2.* (b) illustrates the same example in the Figure *2.* (a) using MP algorithm instead of DP.

In the CP algorithm, the destination set is grouped into $2k$ subsets, where k is the number of columns. An illustration of CP is shown in Figure *2.* (c) in which the source node is the thick circle and the destination nodes are gray. The multicast message is copied and sent to the columns that have the destination nodes in it. Only one copy of message is needed to be sent to a column if all the destination nodes in that column are either located below or above the source node row. If the destination nodes are located both below and above, then two copies of the message are sent to that column. However, the disadvantage of the CP algorithm is the high network latencies due to the start-up delay [11]. Besides, the performance of the network is also degraded because many multicast messages would be sent through the columns by each source node.

Yet another path-based multicast algorithm is proposed in [11] called low distance path-based (LP), which was proven to have smaller aggregate path hops than that of DP, MP and CP. The LP algorithm labels the node in different way from the DP and MP algorithm. The formula to calculate the label of each node based on its integer coordinate is shown in [11]. Then the LP algorithm groups the destination set in a similar way to the MP algorithm. Instead of following the Hamilton path, LP algorithm allows the message to find a shorter path by reaching the destinations in the order of the low-distance from the source. The distance from the destination to the source node is calculated by k = $|y - y_0|$ + $|x - x_0|$. Besides, to ensure the algorithm is deadlock free, the odd-even turn model [17] is used in LP. The Figure *2*. (d) shows the same example with DP and MP above. The messages follow the bold solid arrows to reach the destination nodes. Apparently, the average aggregate hop count of the LP algorithm is smaller than that of DP and MP. Lessons from [11] motivate us to explore multipath-based multicast algorithms for random topologies.

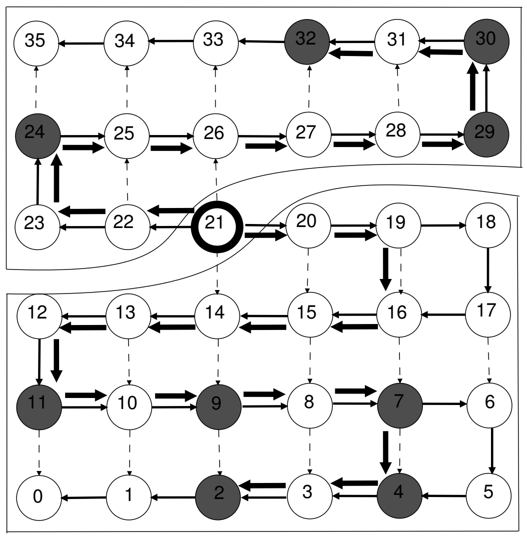
3. Multipath-based Multicasting Algorithm for Random Network Topology

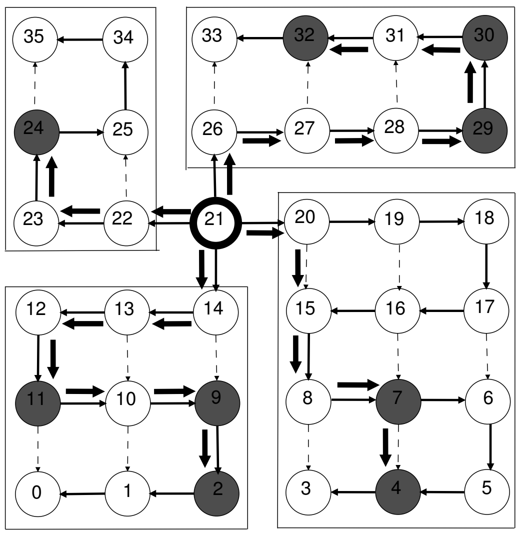
Our approach relies on brute-force search for a multipath-based algorithm. A multicast is completed by sending packets along multiple paths that include all the destinations. With a given destination set, this brute-force search algorithm groups the destinations into all possible combinations. Then by applying the breadth-first search and traveling salesman algorithm, we find the best case which gives the smallest number of aggregate hop count.

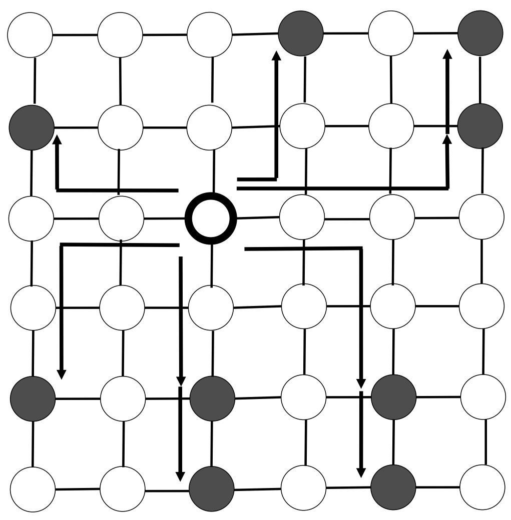
In this work, we limit the maximum number of destinations in a multicast message to be 8 on 16-node and 32-node random networks. Based on our survey, there is no practical case where a multicast message has mores than 8 destinations.

The number of possible partitions for a particular node set is known in advance. It is equal to Bell number, which counts the number of partitions of a set [14]. A partition can have at least one subset and at most n subsets, where n is the number of destinations in the destination set. We use the algorithm in [13] to partition the destination set. An example of all partitions for a 3-element set is shown in Figure *3*.

Our algorithm finds the shortest path from the source node to each subset that includes one or more destination nodes. The subsets in any specific partition can be divided into two types: a subset with only one destination and that with two or more destinations. For a single-destination subset, we use the breadth-first search (BFS) based on Dijkstra algorithm to find the shortest path from the source to that specific destination. For a multiple-destination subset, it can be regarded as a traveling salesman problem (TSP) to find the shortest path


(a) Dual path


(b) Multipath


(c) Column path

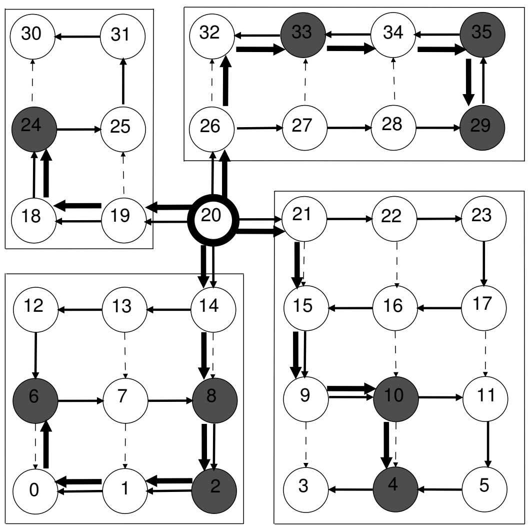

(d) Low distance path

Figure 2. Multicast techniques.

1:$\{1,2,3\}$
2:$\{1,2\}$,$\{3\}$
3:$\{1,3\}$,$\{2\}$
4:$\{1\}$,$\{2,3\}$
5:$\{1\}$,$\{2\}$,$\{3\}$

Figure 3. Partitions for a set of 3 elements: $d = \{1, 2, 3\}$.

$$
\begin{array}{l}
1{:}\{1,2,3\}\\
2{:}\{1,3,2\}\\
3{:}\{2,1,3\}\\
4{:}\{2,3,1\}\\
5{:}\{3,1,2\}\\
6{:}\{3,2,1\}
\end{array}
$$

Figure 4. Permutations for a set of 3 elements: $d = \{1, 2, 3\}$.

```
Algorithm: Finding a Hamiltonian shortest path from the source through all destinations
Input: The random network topology, Destination set D with n elements ($1 < n < 9$), Source node s
Output: The minimum hop count
Initial: $d \leftarrow P_1[1]$, $hop\_count[n!] = \{0\}$
  1: Find all the permutations $P_i$ of the destination set
  2: for each partition $P_i$ from $P_1$ to partition $P_{n!}$
  3:   for $j = 1$ to $j = n$
  4:     $hop\_count[i]$ =$hop\_count[i]$ + $BFS(s, P_i[j])$
  5:     $s \leftarrow P_i[j]$; $d \leftarrow P_i[j + 1]$
  6:     $j = j + 1$
  7:   end for
  8: end for
  9: Return the minimum hop count value in the $hop\_count[n!]$ array
```

Figure 5. BTTSP pseudo code (the number of destinations is 8).

from the source node to the destinations in the subset. In this work, we apply a brute-force traveling salesman problem algorithm (BTTSP). In the BTTSP algorithm, we take a multiple-destination subset as an input and find out all the permutations of the input subset. The number of permutation is $n!$ for a subset of n elements. An example of all permutation for a 3-element set is shown in Figure *4*.

With each permutation, we use BFS algorithm again so as to find the shortest path between each pair of source and destination. The first pair illustrates the source node and the first destination node in a specific permutation. Similarly, the next pair takes the destination node in the previous pair as the source node, and the next destination node in the specific permutation as the destination node. For example, for a subset of 8 elements, we have 8 pairs of source and destination. Figure *5* shows the pseudo code of the BTTSP algorithm.

After calculating the shortest path for all pairs, we sum them up to have the aggregate hop

```
**Algorithm**: Finding the shortest path from a source node to a destination set
**Input**: The random network topology, Destination set D, Source node s
**Output**: The shortest path, the corresponding hop count
**Initial**: $hop\_count[BELL\_number] = \{0\}$
 1: Find all the partitions $P_i$ of the destination set
 2: for each partition $P_i$ from partition $P_1$ to partition $P_{BELL\_number}$
 3:   for each subset of the partition
 4:     if the subset has only one element
 4:       $hop\_count[i]$ =$hop\_count[i]$ + $BFS(s, subset)$
 4:     else if the subset has more than one element
 4:       $hop\_count[i]$ =$hop\_count[i]$ + $BTTSP(s, subset)$
 4:     end if
 7:   end for
 8: end for
 9: Return the minimum hop count value in the $hop\_count[BELL\_number]$ array
```

Figure 6. Multipath-based multicast algorithm.

counts required to send a message from the source node to the subset of destinations. Finally, the partition which has the smallest number of aggregate hop counts provides us the best optimized path to go from the source to the given destination set. Figure *6* shows the pseudo code of our algorithm.

Because our interests are to minimize the aggregate hop counts, we do not analyze the computing complexity of our algorithm in detail. However, with a constraint of the maximum number of destination to be 8, our algorithm can finish in less than 5 seconds in total on a personal computer, including the time for the partitioning algorithm and the BTTSP algorithm.

Our random topology is generated using the a 2D matrix in which every element has a binary value. If the value of element (x, y) is 1, it means that the node number x has a connection with the node number y. Besides the number of node, the degree of the network topology d is also specified. Each node is randomly connected to d other nodes. Finally, the random topology is checked if it is strongly connected or not. If not, the topology is regenerated using another random seed. A "strongly connected" random topology is a topology that every node can find at least a way to reach each of the rest nodes in the generated topology. Therefore, to check if a generated random topology is "strongly connected" or not we simply check every node if it can find a way to reach each of the rest nodes in the generated topology. The Figure *7* shows an example 2D matrix representing an 8-node 4-degree random topology. Since our topology is undirected graph, the matrix is symmetrical.

$$\begin{matrix}
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0
\end{matrix}$$

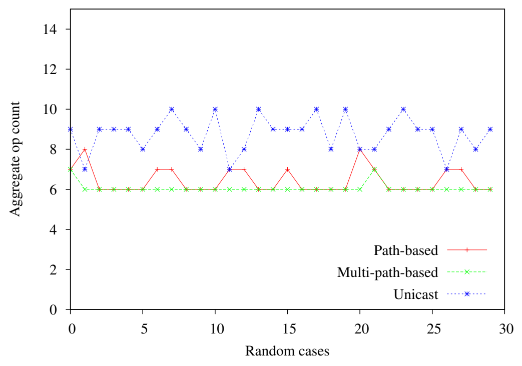
Figure 7. Random topology with 8 nodes, degree = 4.

4. Evaluation

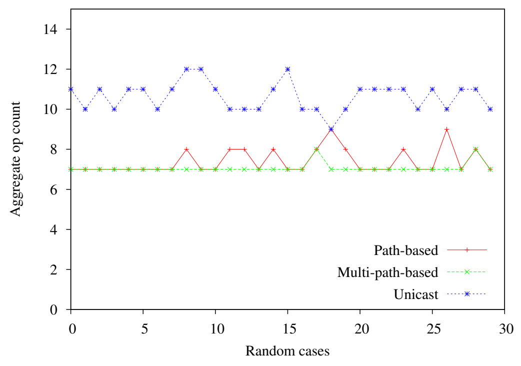
Our algorithm supports three multicasting schemes: the unicast-based, the path-based, and our proposed scheme (i.e. the multipath-based). The unicast-based scheme is simply the case when we group the destination set with n nodes into n subsets, each of which contains only one node. The path-based scheme is the case where we group the destination set into only one subset which contains all the destinations. We run our algorithm with several configurations for two main cases: 16-node and 32-node random network topologies. Table *1* shows the interesting configurations that bring the surprising result. Through the experiment, we see that if the number of the destination set is less than 4, the path-based scheme provides good result than the multipath-based scheme. When the number of destinations is between 4 and 8, the multipath-based scheme is clearly better than the path-based scheme in terms of aggregate hop count. The degree of the network is chosen to be at least 5, because degree 4 is the degree of a mesh/torus. When the degree of the network is very high, e.g., 12 in a 16-node network, the aggregate hop count of the multipath-based scheme is exactly the same as that of the path-based method. Since the random topology is different everytime we generate it, each specific configuration is run with 30 different random topologies by simply randomizing the topology 30 times using the same configuration. Figure *8* shows the results comparing the unicast-based, the path-based, and the multipath-based schemes. From the results, the unicast-based scheme is shown to be the worst in terms of aggregate hop count. Our proposed multipath-based scheme has shown to be the best, although it has some points that have the same aggregate hop count with that of the path-based scheme.

Table 1. *The input configuration.*

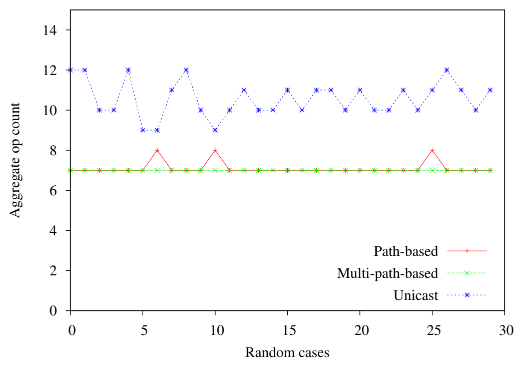
Configuration	No. of node	Degree	No. of destination
1	16	7	6
2	16	8	6
3	16	8	7
4	32	20	4



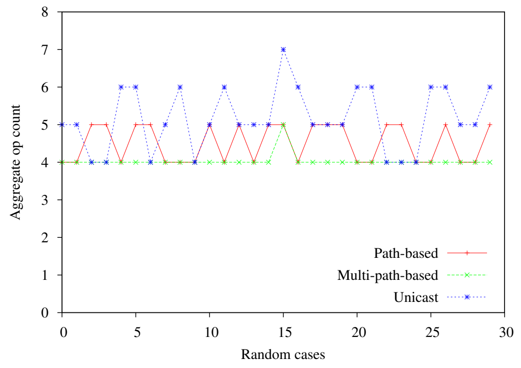
(a) 16 nodes – 6 destinations – 7 degree



(b) 16 nodes – 6 destinations – 7 degree



(c) 16 nodes – 7 destinations – 8 degree



(d) 32 nodes – 4 destinations – 20 degree

Figure 8. *Simulation results.*

A report on the mean and standard deviation for each configuration is shown in Table *2* and *3* for the cases of 16 nodes and 32 nodes, respectively. The configuration column of these tables contains the input configuration for the algorithm. For example, the configuration 1 in Table *2* means that the number of nodes is 16, the number of destinations of the multicast message is 4, and the degree of the network topology is 6. To improve the accuracy of the mean value, we generate 100 different random topologies for every configuration.

From the mean columns of Table *2* and *3*, we could see that the multipath-based scheme always has the smallest aggregate hop count on average compared with the path-based or the unicast-based schemes. For each configuration of the two tables, we calculate the difference between the mean of multipath-based and the path-based, then divide it by 100 to achieve the improvement percent. The configuration number 4 in Table *2* provides the maximum improvement percent which is 5.5. The comparison between multipath-based with unicast-based is not calculated because the unicast-based scheme is even worse than the path-based scheme. The standard deviation of the multipath-based scheme is also the best of the three schemes.

Table 2. Comparison of mean and standard deviation in the case of 16-node network.

No.	Configuration	Path-based		Multipath-based		Unicast	
		Mean	SD	Mean	SD	Mean	SD
1	16-4-5	5.24	0.84	5.22	0.77	7.11	0.96
2	16-4-6	5.02	0.54	4.89	0.54	6.53	0.79
3	16-4-7	4.75	0.49	4.66	0.46	6.26	0.73
4	16-4-8	4.58	0.50	4.33	0.28	5.76	0.58
5	16-4-9	4.34	0.28	4.19	0.17	5.68	0.66
6	16-4-10	4.20	0.16	4.06	0.06	5.20	0.62
7	16-4-11	4.12	0.11	4.06	0.06	5.00	0.64
8	16-4-12	4.07	0.07	4.00	0.00	4.58	0.38
9	16-5-5	6.39	0.66	6.24	0.74	9.18	1.81
10	16-5-6	5.92	0.55	5.65	0.49	8.13	0.81
11	16-5-7	5.63	0.53	5.39	0.34	7.69	0.73
12	16-5-8	5.42	0.36	5.26	0.21	7.48	1.05
13	16-5-9	5.19	0.19	5.06	0.06	6.94	0.80
14	16-5-10	5.13	0.11	5.01	0.01	6.49	0.83
15	16-5-11	5.03	0.05	5.00	0.00	6.15	0.63
16	16-6-5	7.17	0.60	6.87	0.57	10.84	1.73
17	16-6-6	6.77	0.68	6.44	0.33	9.76	0.90
18	16-6-7	6.36	0.31	6.15	0.13	9.29	0.79
19	16-6-8	6.16	0.17	6.03	0.03	8.65	0.97
20	16-6-9	6.07	0.07	6.00	0.00	8.27	0.80
21	16-6-10	6.05	0.05	6.00	0.00	7.70	0.79
22	16-7-6	7.85	0.65	7.66	0.46	12.68	2.14
23	16-7-6	7.41	0.30	7.25	0.23	11.44	1.05
24	16-7-7	7.19	0.17	7.07	0.07	10.84	1.03
25	16-7-8	7.09	0.08	7.01	0.01	10.16	0.83
26	16-7-9	7.03	0.03	7.00	0.00	9.65	0.91
27	16-8-5	8.61	0.56	8.27	0.24	14.22	1.39
28	16-8-6	8.19	0.19	8.10	0.09	13.13	1.41
29	16-8-7	8.06	0.06	8.03	0.03	12.37	0.93
30	16-8-8	8.02	0.02	8.00	0.00	11.53	0.97

5. Conclusion

In this work we tried to minimize the number of total packet path hops for collective communication by introducing a combination of path-based multicastings, especially for random topologies. We take a brute-force search for a multipath-based algorithm for the case where a number of destinations is small. Our graph analysis results show that our multicasts usually reduce the aggregate path hops by up to 5.5% when compared to path-based. Our recommendation to support multicasts for a few destinations is the use of multicasting based on multiple paths.

Table 3. Comparison of mean and standard deviation in the case of 32-node network.

No.	Configuration	Path-based		Multipath-based		Unicast	
		Mean	SD	Mean	SD	Mean	SD
1	32-4-6	6.68	0.78	6.51	1.19	8.40	2.26
2	32-4-7	6.26	0.87	6.17	1.00	7.85	1.71
3	32-4-8	5.76	0.76	5.62	0.92	7.27	1.14
4	32-4-9	5.69	0.77	5.59	0.80	6.95	0.99
5	32-4-10	5.52	0.75	5.37	0.87	6.83	0.90
6	32-4-11	5.21	0.59	5.05	0.69	6.54	0.73
7	32-4-12	5.12	0.85	5.05	0.79	6.57	0.83
8	32-4-13	5.01	0.69	4.88	0.55	6.30	0.87
9	32-5-6	7.78	0.89	7.60	1.08	10.50	2.49
10	32-5-7	7.19	0.93	7.10	1.03	9.83	1.40
11	32-5-8	6.79	0.89	6.65	0.87	9.30	1.53
12	32-5-9	6.74	0.93	6.47	0.91	8.78	1.13
13	32-5-10	6.68	1.00	6.33	1.12	8.47	1.05
14	32-5-11	6.41	1.00	6.23	0.92	8.45	0.99
15	32-5-12	6.07	0.87	5.84	0.75	8.12	0.91
16	32-5-13	5.90	0.69	5.71	0.55	8.08	0.97
17	32-6-6	8.58	1.38	8.45	1.59	12.60	2.70
18	32-6-7	8.26	0.89	8.09	1.04	11.84	1.63
19	32-6-8	7.97	1.31	7.79	1.25	11.18	1.65
20	32-6-9	7.84	1.15	7.53	1.09	10.52	1.23
21	32-6-10	7.33	0.90	7.15	0.99	10.40	1.20
22	32-6-11	6.91	0.76	6.72	0.62	9.95	1.05
23	32-6-12	6.86	0.78	6.65	0.63	9.72	1.08
24	32-6-13	6.61	0.44	6.39	0.32	9.45	1.03
25	32-7-5	10.40	1.52	10.34	1.78	16.95	3.89
26	32-7-6	9.75	1.25	9.46	1.41	14.77	3.02
27	32-7-7	9.09	1.12	8.90	1.05	13.90	2.25
28	32-7-8	8.85	1.35	8.57	1.31	12.84	1.33
29	32-7-9	8.43	0.93	8.02	0.80	12.46	1.53
30	32-7-10	8.13	0.89	7.94	0.84	12.22	1.35
31	32-7-11	8.02	0.82	7.73	0.60	11.69	1.23
32	32-7-12	7.75	0.73	7.54	0.51	11.35	1.41
33	32-7-13	7.45	0.35	7.23	0.20	11.11	0.96
34	32-8-6	10.56	1.33	10.26	1.59	16.96	4.38
35	32-8-7	10.11	1.52	9.75	1.45	15.65	2.71
36	32-8-8	9.75	1.15	9.36	0.93	14.55	1.89
37	32-8-9	9.28	0.76	8.92	0.57	14.36	1.39
38	32-8-10	8.87	0.81	8.69	0.61	13.83	1.14
39	32-8-11	8.74	0.69	8.48	0.51	13.37	1.57
40	32-8-12	8.46	0.45	8.18	0.23	12.71	1.53
41	32-8-13	8.37	0.33	8.24	0.22	12.69	1.73

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 25280018 and 25280043 and JST CREST. We would like to thank Professor Dang for his thorough review for the manuscript. The first author would like to thank ICDREC for a permission to join this work.

References

[1] Alam, Sadaf R. et al., "An Evaluation of the Oak Ridge National Laboratory Cray XT3," in *International Journal of High Performance Computing Applications*, vol. 22, no. 1, pp. 52-80, 2008.

[2] J. Tomkins, "Interconnects: A Buyers Point of View," ACS Workshop, 2007.

[3] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A Case for Random Shortcut Topologies for HPC Interconnects," in *Proc. of the International Symposium on Computer Architecture (ISCA)*, pp. 177–188, 2012.

[4] C.-H. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "Smart: a single-cycle reconfigurable noc for soc applications," in *DATE*, , pp. 338–3432013.

[5] Yuan, Xin and Mahapatra, Santosh and Nienaber, Wickus and Pakin, Scot t and Lang, Michael, "A New Routing Scheme for Jellyfish and Its Performance with HPC Workloads," in *Proceedings of SC13: International Conference for High Performanc e Computing, Networking, Storage and Analysis, series SC '13*, pp. 36:1–36:11, 2013.

[6] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking Data Centers Randomly," in *Proc. of USENIX Symposium on Network Design and Implementation (NSDI)*, 2012.

[7] Béla Bollobás and Fan R. K. Chung, "The Diameter of a Cycle Plus a Random Matching," in *SIAM J. Discrete Math*, vol. 1, no. 3, pp. 328-333, 1988.

[8] Sarat Yoowattana, Ikki Fujiwara, Michihiro Koibuchi, "Investigating Performance Advantages of Random Topologies on Network-on-Chip," in *SASIMI 2013 Proceedings*, 2013.

[9] W. Hu, Z. Lu, H. Liu, and A. Jantsch, "TPSS: A Flexible Hardware Support for Unicast and Multicast on Network-on-Chip," *Journal of Computers*, vol. 7, no. 7, pp. 1743-1752, 2012.

[10] N. E. Jerger, L. S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on,* IEEE, pp. 229-240, 2008.

[11] M. Daneshtalab, M. Ebrahimi, S. Mohammadi, and A. Afzali-Kusha, "Low-distance path-based multicast routing algorithm for network-on-chips," *Computers & Digital Techniques*, IET. 3, no. 5, pp. 430-442, 2009.

[12] Boppana R.V., Chalasani S., and Raghavendra C.S., "Resource deadlock and performance of wormhole multicast routing algorithms," in *IEEE Transaction Parallel Distribution System,* pp. 535-549, 1998.

[13] M. C. ER, "A fast algorithm for generating set partitions," *The Computer Journal*, vol. 31, no. 3, pp. 283-284, 1988.

[14] Gian-Carlo Rota, "The number of partitions of a set," *The American Mathematical Monthly*, vol. 7, no. 5, pp. 498-504, 1964.

[15] Xiaola Lin and Lionel M. NI, "Multicast communication in multicomputer networks," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1105-1117, 1993.

[16] X. Lin, P.K. McKinley, and L.M. Ni, "Deadlock-Free Multicast Wormhole Routing in 2D Mesh Multicomputers," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, no. 8, pp. 793-804, Aug. 1004.

[17] Chiu G., "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, pp. 729-738, 2000.

Manuscript received 16-02-2014; accepted 07-11-2014.

■

Ngo Quang Vinh received the BSc and MSc degree in electronics engineering from the HoChiMinh City University of Technology in 2006 and 2008. Since 2006, he has worked for Integrated Circuit Design Research and Education center as a hardware design engineer. His research interests include the areas of computer architecture, digital IC design.

Hoang Trang was born in NhaTrang city, Vietnam. He received the Bachelor of Engineering, and Master of Science degree in Electronics-Telecommunication Engineering from Ho Chi Minh City University of Technology in 2002 and 2004, respectively. He received the Ph.D. degree in Microelectronics-MEMS from CEA-LETI and University Joseph Fourier, France, in 2009. From 2009-2010, he did the post-doctorate research in Orange Lab- France Telecom. Since 2010, he is lecturer at Faculty of Electricals-Electronics Engineering, Ho Chi Minh City University of Technology. His field of research interest is in the domain of FPGA implementation, Speech Recognizer, MEMS, fabrication, integration of passive components and function for telecommunications, Embedded System, System-on Chip (SoC).

Vu Dinh Thanh is the Professor of Electrical Engineering at HoChiMinh City University of Technology. He received his BS (1982) from HoChiMinh City University of Technology and his M.S. (1990) and Ph.D. (1993) in electrical engineering from L'Institut National Polytechnique de Grenoble (INPG), France. His research centers on microwave engineering and on digital signal processing. He served as Dean of Department of Electrical Engineering from 2002 to 2007 and since 2007, he has been being the Rector of the HoChiMinh City University of Technology, Vietnam.

Ikki Fujiwara received the BE and ME degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2002 and 2004, respectively, and received the PhD degree from the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2012. He is currently an assistant professor in the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, Japan. His research interests include the areas of high-performance computing and optimization. He is a member of the IEEE.

Michihiro Koibuchi received the BE, ME, and PhD degrees from Keio University, Yokohama, Japan, in 2000, 2002, and 2003, respectively. He is currently an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, and the Graduate University for Advanced Studies (SOKENDAI), Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a member of the IEEE.