PAPER

The Case for Network Coding for Collective Communication on HPC Interconnection Networks

Ahmed SHALABY[†], *Nonmember*, **Ikki FUJIWARA**[††a)], *and* **Michihiro KOIBUCHI**[††], *Members*

SUMMARY Recently network bandwidth becomes a performance concern particularly for collective communication since bisection bandwidths of supercomputers become far less than their full bisection bandwidths. In this context we propose the use of a network coding technique to reduce the number of unicasts and the size of data transferred in latency-sensitive collective communications in supercomputers. Our proposed network coding scheme has a hierarchical multicasting structure with intra-group and inter-group unicasts. Quantitative analysis show that the aggregate path hop counts by our hierarchical network coding decrease as much as 94% when compared to conventional unicast-based multicasts. We validate these results by cycle-accurate network simulations. In 1,024-switch networks, the network reduces the execution time of collective communications as much as 70%. We also show that our hierarchical network coding is beneficial for any packet size.

key words: *interconnection networks, collective communication, network coding, high-performance computing*

1. Introduction

As the scale of HPC systems —such as supercomputers, custom massively-parallel computers, and PC clusters— increases, the network bandwidth per flops (floating-point operations per second) becomes low. It will be more difficult for network bisection bandwidth to reach full-bisection bandwidth in future parallel computers [1], [2]. One study recommended that the aggregate link bandwidth per flops for internal networks be greater than or equal to 0.2 bytes/sec/flops, and the bisection bandwidth per flops be greater than or equal to 0.1 bytes/sec/flops [1]. However, the link bandwidth per flops is expected to be 0.005–0.03 bytes/sec/flops in 2015 [2]. Such relatively-low bisection bandwidth affects the performance of collective communications.

Recent interconnects such as InfiniBand and Ethernet usually use unicast-based multicasts as they use commodity switches that do not support hardware multicasting, unlike the prior products such as QsNET and QsNET II, which support hardware multicasts in a fat-tree topology [3]. To implement collective communication, a large number of unicasts are simultaneously generated in such commodity interconnection networks [4]. The unicasts may introduce a large number of packet contentions that are likely to lead to

Manuscript received July 22, 2014.
Manuscript revised October 20, 2014.
Manuscript publicized December 11, 2014.
[†]The author is with E-JUST, Egypt.
[††]The authors are with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: ikki@nii.ac.jp
DOI: 10.1587/transinf.2014EDP7255

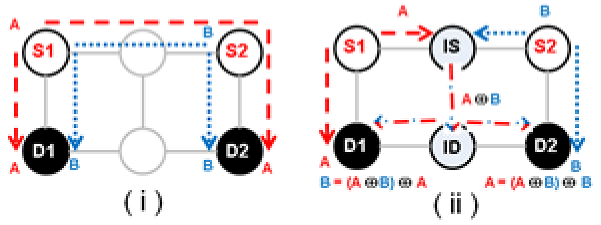


Fig. 1 (i) Unicast-based multicast and (ii) that with network coding.

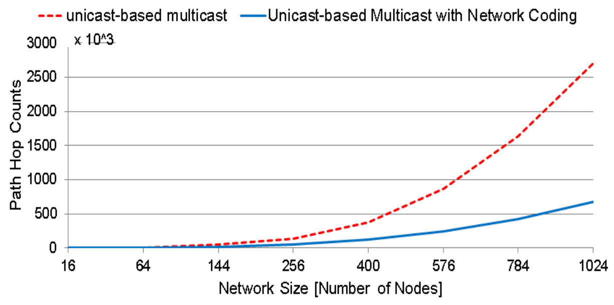


Fig. 2 Sum of unicast path hop counts of conventional broadcast vs. that with network coding in k-ary 2-mesh.

a high latency in a multicast.

In this study, we use a network coding technique to reduce the number of unicasts and the transfer data size in collective communication primarily for k-ary n-cubes.

Figure 1 shows an example of collective communication in which two sources, S1 and S2, multicast data A and B to destinations, D1 and D2, in a 3×2 2-D mesh with dimension-order routing. Figure 1 (i) shows a conventional unicast-based multicast. The two shared links may cause packet contention. In the case of network coding, as shown in Fig. 1 (ii), each source sends a unicast to a single destination. Intermediate node (IS) makes a unicast whose packet contains the results obtained by computing the XOR bit operation to two arrived unicast data. The shared link between IS and ID is used once to send the encoded packet ($A \oplus B$). Once the encoded packet is received at the destinations D1 and D2, the original data is restored by simply applying the XOR operation again with the other packet, namely $A = (A \oplus B) \oplus B$ and $B = (A \oplus B) \oplus A$.

Figure 2 illustrates the aggregate unicast path hop counts for the conventional broadcast (i.e., tree-based, as discussed in the next section) versus the same with network coding. The benefit of network coding increases exponentially as the network size increases. For example, the network coding only consumes 0.67 million aggregate unicast hop counts of packets in a 32-ary 2-mesh. When compared to the original multicast (2.7 millions), it reduces the unicast hop counts by 75%.

We propose hierarchical network coding for collective communication to reduce the number of unicasts and the transfer data size primarily for k-ary n-cube networks. This study is an extension of our prior work [25]. Our proposed network coding scheme has a hierarchical multicasting structure that consists of intra-group and inter-group multicasts so as to reduce the number of unicasts and the size of transferred data.

Our findings of this paper are as follows.

- Through our quantitative analyses, the hierarchical network coding efficiently improves the performance in various network designs. In a 4,096-switche network, this technique improves the aggregate number of unicast path hops by 94% (Sect. 4).
- Through our cycle-accurate network simulations, the hierarchical network coding constantly obtains good performance gain in (1) all the transfer data (packet) sizes evaluated and (2) various overhead latencies to compute XOR data at intermediate nodes (Sect. 5).
- We consider the design space of the hierarchical network coding as well as practical issues. As the number of multicast nodes increases, the performance gain by the hierarchical network coding becomes large (Sect. 6).

This paper is organized as follows. Section 2 describes related work. Section 3 describes our hierarchical network coding. Section 4 illustrates the network coding performance on path hop counts by using quantitative analyses. Section 5 shows the performance of broadcasts with the hierarchical network coding by using cycle-accurate network simulations. Section 6 discusses practical issues and the design space of the hierarchical network coding. Section 7 draws conclusions of our findings and states our future work.

2. Related Work

2.1 Multicast Communications

Hardware-, path-, and unicast-based algorithms are typical methods for multicasts in interconnection networks [7]. Hardware multicasts, e.g., QsNET II [3], duplicate packets at an intermediate switch for a multicast. Since it reduces the aggregate packet hop counts in a multicast, it efficiently sends data to multiple destinations. A path-based multicast sends data along a path that includes all destinations, and thus requires an efficient multicast-path search, e.g., Hamiltonian cycle. Theoretically, this is an interesting topic; however, current conventional interconnects do not always support a hardware- and path-based multicast [8], [9].

A conventional 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. This paper refers to this as an "all-at-once" multicast. It is applicable for all the multicasts occurring in a parallel programming, including MPI_Alltoall, in which a source sends different data to destinations.

When a source scatters the same data to all destinations, a tree-based multicast is practical for reducing both the number of packet contentions and the aggregate packet hops [10]. In a 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 [11]. The impact of the tree-based multicast algorithms on the execution time is evaluated in a high-performance computing (HPC) interconnect prototype [1]. In this work, we assume to use unicast-based multicasts.

2.2 Efficient Communication Methods

Network design, especially for collective communication, has been traditionally constrained by the bandwidth in supercomputers. Message combining has been proposed to avoid tree saturation in multistage interconnection networks (MINs) [13]. Message combining merges multiple packets to the same destination at an intermediate node. A similar approach is attempted in the $N \times 2N$ tori in the IBM BlueGene/L [14]. In [14], it is reported that software message concatenation improves the performance of the MPI_Alltoall function, when it is performed just before packets are turned in a dimension.

Generally, a simple overall strategy to make the best use of a limited link bandwidth is data compression. Each sender compresses the contents of packets except for their control information for routing, flow control, error-correction etc. Although the decoding and encoding overhead of nodes cannot be ignored in terms of latency, the data compression is attractive for reducing the amount of transfer data. Fortunately, in this work our network coding can work together with compression, because our network coding only operates the XOR calculation for bits of any type of contents to be transferred.

2.3 Network Coding Application

Network coding aims to optimize the data flow to improve throughput and efficiency of the network. Network coding is associated with information theory and was first introduced in 2000 [15]. Network coding has been applied in many fields, e.g., distributed storage, wireless networks, file sharing, and multimedia streaming in peer-to-peer networks [16]–[19].

Unquestionably, these applications have different characteristics from those of supercomputer interconnects. Different characteristics affect the design of optimization. (1) Supercomputer interconnects usually have a non-random fixed topology of switches and custom deadlock-free routing, e.g., dimension-order routing on k-ary n-cubes, and each cable usually has the same bandwidth. (2) Parallel applications explicitly generate multicasts at the program level, e.g., using MPI functions. These unique features allow us to use the regularity of topologies and precisely estimate the number of packet hops for optimizing network coding. (3) Another unique feature concerns low-latency requirements, i.e., order of hundreds of nanoseconds. We thus consider the simple XOR bit operation for network encoding in this paper.

To the best of our knowledge, no previous work has explored the use of network coding for efficiently use network bandwidth in supercomputer interconnects.

3. Hierarchical Network Coding

We propose to apply the network coding technique to the multicast communication scenario in supercomputers. Our proposed method has a hierarchical structure with intra-group and inter-group communications. In this section, we focus on a broadcast. However, we can naturally apply our hierarchical network coding for multicasts, or multiple sources to multiple destinations. This topic is discussed in Sect. 6.

The detailed procedure for hierarchical network coding is as follows. Figure 4 provides the pseudo code.

a) Grouping: We divide a given network into a number of groups. In the example of the 2-ary 2-mesh in Fig. 3, the nodes are divided into two groups, depicted as shaded nodes and non-shaded nodes. The details of the grouping are quantitatively discussed in the next section.

b) Intra-group broadcasts: Every node inside a group exchanges transfer data by an existing multicast algorithm, e.g., a tree-based multicast. Every node then obtains all the data of the other nodes in the group. In the example in Fig. 3, data d_1 and d_2 are exchanged between two nodes in the shaded group, whereas data d_3 and d_4 are shared in the non-shaded group.

c) Network coding: We choose one of the nodes in each group as an intermediate node that computes the XOR function to encode packets. Assume that M nodes, $1, 2, \ldots, M$, have broadcast data, $d_1, d_2, \ldots, d_M$, respectively. The intermediate node then generates $M-1$ encoded packets whose contents are $(d_1 \oplus d_2), (d_2 \oplus d_3), \ldots, (d_{M-1} \oplus d_M)$.

d) Inter-group multicasts of encoded packets: The intermediate nodes exchange all the encoded packets by an inter-group multicast between all pairs of intermediate nodes. As in step (B), an existing multicast algorithm is used to deliver the packets. In the example, the encoded packets $(d_1 \oplus d_2)$ and $(d_3 \oplus d_4)$ are exchanged between the shaded and the non-shaded groups.

e) Intra-group broadcasts of encoded packets: The intermediate node delivers the encoded packets to all the nodes in its group. In the example, $(d_1 \oplus d_2)$ is sent to the other nodes in the non-shaded group, while $(d_3 \oplus d_4)$ is distributed in the shaded group.

f) Inter-group unicasts: Every node sends its data to all the other groups. One of the nodes in the destination group receives the data. In the example, node 1 sends d_1 to node 3 while node 3 sends d_3 to node 1. Similarly, d_2 is sent from node 2 to node 4 and d_4 is sent from node 4 to node 2.

g) Decoding: Every node receives (1) the data from all the nodes in the same group (step B), (2) all the encoded packets from the other groups (step E), and (3) the data from a node in each group (step F). Then it restores the non-received data of the group by computing the XOR bit operation. For example, a node obtains $(d_1 \oplus d_2), (d_2 \oplus d_3), \ldots, (d_{M-1} \oplus d_M)$ (step D) as well as d_1 (step F); then it restores data $d_2, d_3 \ldots, d_M$ by computing the XOR bit operation. Every node starts decoding packets as soon as the required data are obtained.

Although the coded packets has a possibility to incur deadlocks of the network when allowing to perform multiple steps simultaneously in arbitrary topologies, assigning different virtual channels to communication steps essentially breaks indirect channel dependencies in the hierarchical network coding. For instance, four virtual channels are needed for four communication steps. This virtual-network approach is a traditional way to avoid deadlocks in interconnection networks and we can use it for the deadlock avoidance in any topology.

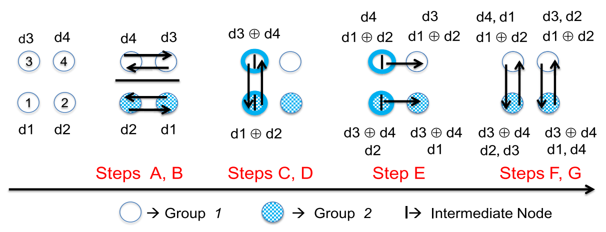


Fig. 3 Hierarchical network coding in 2-ary 2-mesh.

```
Algorithm:   hierarchical network coding
N: number of Nodes inside the Network    G: number of Groups
M: number of Nodes inside each Group     IN: Intermediate Node

- Divide N to G Groups // Step A
- for i = 1 to G {
      for j = 1 to M {
          n_{i,j} broadcast its packet to (M -1) nodes inside each group G
                                                            // Step B
          n_{i,j} sends its packet to one node in each group G // Step F }
}
- Determine intermediate nodes IN
- for i = 1 to G {
    IN_i encodes packets inside each group by computing XOR function
                                                            // Step C
    IN_i broadcasts the encoded packets to (G-1) Intermediates nodes
                                                            // Step D }
- for i = 1 to G {
    IN_i broadcasts the encoded packets to (M-1) nodes inside each
group
                                                            // Step E }
- for i = 1 to G {
      for j = 1 to M {
          n_{i,j} decodes the received encoded packets by computing
              XOR function                                  // Step G }
}
```

Fig. 4 Pseudo-code of the hierarchical network coding.

Note that the algorithm in Fig. 4 is an example of our hierarchical network coding scheme. There are variations of the algorithm for the hierarchical network coding. For example, in the step C, each node in the group generates encoded packets. Then, all nodes in the group has encoded packet. In the step D every node in the group sends encoded packet to each node in other groups. In the 2×2 mesh case, node 1 sends to node 3, and node 2 sends to node 4. In this variation the step E does nothing. The total number of generated packets in the variation is equal to that in the algorithm in Fig. 4. However, the packet hop count for inter-group communication is naturally larger than that of intra-group communication in our network-coding scheme, thus the algorithm in Fig. 4 is better in most cases.

4. Quantitative Analysis

4.1 Setup

We quantitatively evaluate our hierarchical network coding when applied to k-ary n-cube topologies with minimal routing.

In this section, we highlight such parameters of the hierarchical network coding that benefit collective communications. First, we evaluate the impact of the network size on aggregate path hop counts in a multicast. Second, we investigate the influence of the group size on the aggregate path hop counts. Third, we show that our approach improves the multicast performance in both tree-based and all-at-once multicasts. Finally, we evaluate our approach on several network topologies.

We evaluate them on a low-radix topology. Topology design has been discussed for low-radix vs. high-radix networks, especially for exascale computing systems. However, low-radix topologies have been historically used in mainstream supercomputers, because of (1) their simple management mechanisms for faults [20], [21], (2) the straightforward layout of switches with relatively short cables in a machine and (3) easiness in debugging the custom communication protocol. In this work, we use low-radix topologies, i.e., with degrees up to 6.

4.2 Network Size

We evaluate the performance of the hierarchical network coding when applied to various k-ary n-mesh topologies. Figure 5 plots the aggregate hop counts of unicasts of a tree-based multicast and that with the hierarchical network coding in various network sizes. The y-axis is logarithmic. Figure 5 shows that the aggregate hop counts of unicasts drastically increase as the network size increases in both methods. However, at each network size, we observe the benefit of the hierarchical network coding. The hierarchical network coding reduces the aggregate path hop counts by as much as 94% when compared to the original tree-based multicast.

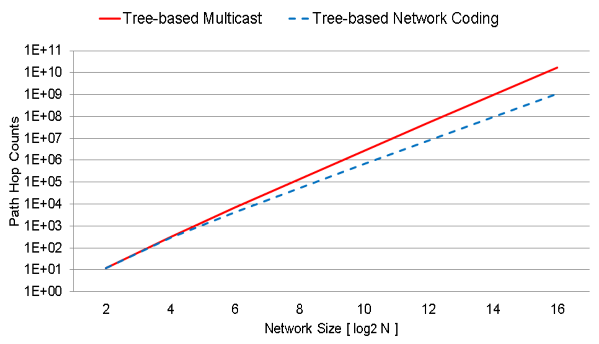


Fig. 5 Aggregate hop counts for tree-based multicast and that with hierarchical network coding on k-ary 2-mesh.

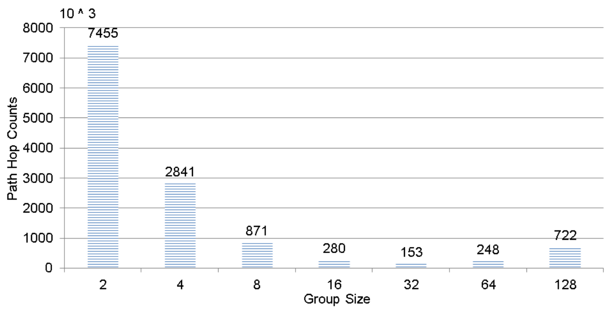


Fig. 6 Aggregate hop counts of hierarchical network coding for different group sizes in 16-ary 2-mesh.

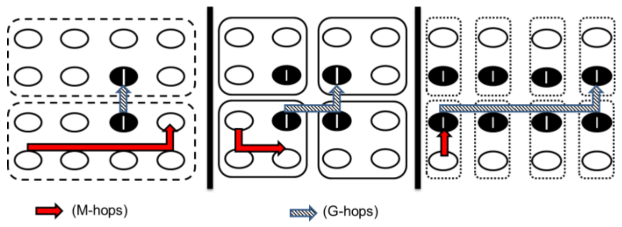


Fig. 7 Different group sizes in 4-ary 2-D mesh.

4.3 Group Size

An important concern of our hierarchical network coding is its group size, i.e., the number of nodes belonging to each group. The best group size minimizes the aggregate hop counts of unicasts. Figure 7 shows examples of group sizes in a 4-ary 2-mesh. The black nodes are the intermediate nodes. The coordinates of the intermediate nodes affect the total hop counts of unicasts when multicasting the encoded packets. We optimize the group size and the coordinates of the intermediate nodes to reduce the number of unicasts and their total hop counts in k-ary n-cubes.

Figure 6 shows the aggregate hop counts of hierarchical network coding in a 16-ary 2-mesh with $N = 256$ nodes. We varied the group size from 2 to 128. We observe that the group size plays an important role in the performance of the hierarchical network coding. The best group size is 32 nodes per group. We use the best group size from the quantitative

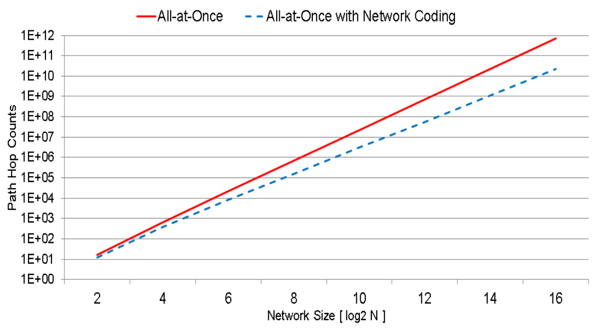


Fig. 8 Aggregate hop counts of all-at-once multicast and that with hierarchical network coding in k-ary 2-mesh.

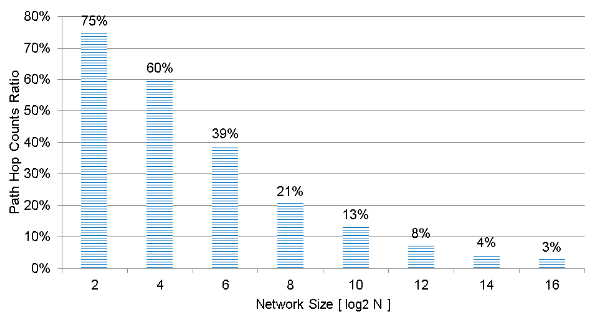


Fig. 9 Aggregate hop count ratios for all-at-once hierarchical network coding against all-at-once multicast on k-ary 2-torus.

analysis in the rest of this paper, unless otherwise stated.

Concerning the topologies for a given group, our recommendation is to reduce the longest path count for the communication inside the group (this usually leads to a small average shortest path length of the generated packets) through our analysis (we omit it, because it is obvious). For example with the group size of 32, our recommendation configuration is 8×4 or 4×8.

4.4 Multicast Algorithm

Since the hierarchical network coding uses a multicast algorithm for data exchange, we evaluate the influence of the multicast algorithm on its performance by the comparison of tree-based and all-at-once multicasts.

Figure 8 plots aggregate hop counts of an all-at-once multicast and those with the hierarchical network coding in various network sizes. Similar to the case of the tree-based multicast, we can observe that the benefit of hierarchical network coding increases as the network size increases. For example, the hop counts required by network coding for this all-to-all scenario in the 256-ary 2-mesh is 32 times less than that of the all-at-once multicast. We thus assert that our hierarchical network coding works well with any multicast algorithms in all the network sizes attempted in this quantitative analysis.

In particular, even when we apply our hierarchical network coding to a tree-based multicast, the gain of the tree-based multicast are obtained from the reduction of the number of inter-group and intra-group packet hop counts.

4.5 Topology

We finally evaluate the performance of the hierarchical network coding when applied to different k-ary n-cube topologies. In the other subsections, we use k-ary 2-meshes. The main difference between a mesh and a torus in our hierarchical network coding is the path hop counts in the inter-group unicasts step, because each unicast uses the wraparound links to reduce its path hop counts.

Figure 9 shows the hop ratios between all-at-once multicasts and those with the hierarchical network coding in the k-ary 2-tori. A good performance gain similar to the case for the k-ary 2-meshes is obtained, especially for the large network sizes. We consequently consider that the hierarchical network coding is efficient to reduce the aggregate path hop counts in k-ary n-cubes, especially for large networks.

5. Cycle-Accurate Simulation

Besides the quantitative analysis, we evaluate the performance of the hierarchical network coding more precisely by using the cycle-accurate network simulator called BookSim [22]. The quantitative analyses in the previous section attempt larger networks, whereas our cycle-accurate simulation considers the details in moderate-sized networks.

5.1 Parameters

We implement tree-based and all-at-once multicast communication scenarios with dimension-order routing on k-ary n-cube topologies. The number of virtual channels is set to four. A header flit requires at least three clock cycles to be transferred to the next router or host: one cycle for the routing computation, one cycle for allocating a virtual channel and a crossbar, and the remaining cycles for transferring the flit to the next router or host. Virtual cut-through switching is used as the switching technique on each router. The nodes inject packets independently of each other. The packet length is set to one flit as a default. Various packet sizes are evaluated in Sect. 5.4.

The default overhead to compute XOR at an intermediate node is set to one cycle as a default; however, various overheads are evaluated in Sect. 5.5. We evaluate the execution cycles (maximum end-to-end latency) of all-to-all broadcast in tree-based and all-at-once multicast algorithms. The lower value of cycles is better.

We used two measures: average packet latency and execution time of a broadcast. The average packet latency considers all the packets in all the steps in Fig. 4. The execution time of a broadcast is the sum of the maximum latency of all sequential steps.

5.2 Network Size

Figure 10 plots the execution time of all-to-all broadcasts for

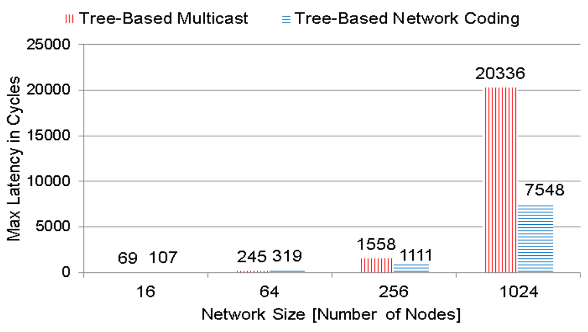


Fig. 10 Execution time for tree-based multicast and that with hierarchical network coding on *k*-ary 2-mesh.

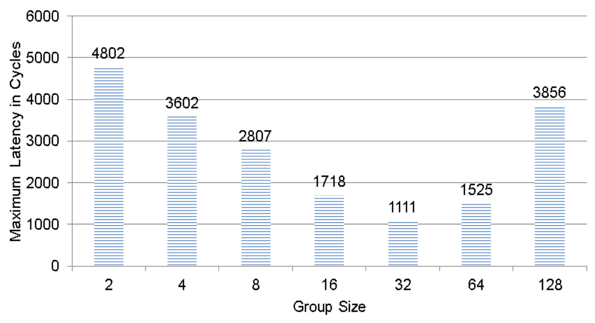


Fig. 11 Execution time for different combinations of groups compared with tree-based multicast in 16-ary 2-mesh.

the tree-based multicast and the same with the hierarchical network coding. The y-axis represents the simulation cycles. The lower values are better. The hierarchical network coding speeds up the all-to-all broadcast communications by three times. Another finding is that the performance of the tree-based multicast is better than that with the hierarchical network coding in small networks, while the hierarchical network coding achieves better performance in large networks. This is because the tree-based algorithm adds delays to the multicast communications due to synchronizations. The sum of the two overheads (the synchronizations of the hierarchical network coding and of the tree-based multicast) dominates the execution time in small networks. Thus, both methods have similar performance in a small network. In contrast, as the network size becomes larger, the synchronization delay by the tree-based multicast (without network coding) strongly affects the execution time, and thus the hierarchical network coding improves the performance drastically.

5.3 Group Size

To show the impact of group size on the performance, we implement all possible sizes of groups in a 16-ary 2-mesh. Figure 11 shows the execution time for each group size. The best group size is 32 nodes per group. These are the same results obtained in the quantitative analyses.

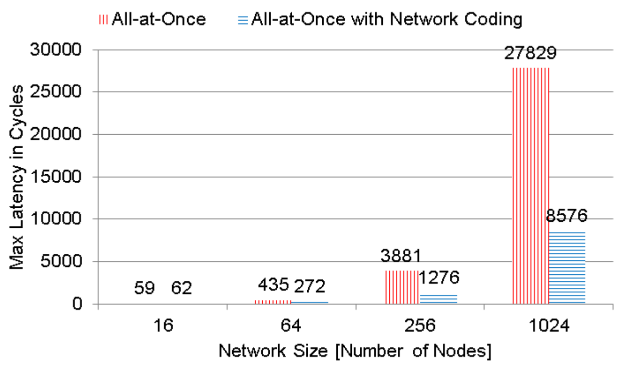


Fig. 12 Execution time in cycles for all-at-once multicast and that with hierarchical network coding on *k*-ary 2-mesh.

5.4 Multicast Algorithm

Figure 12 illustrates the execution time comparing between the all-at-once multicast and that with the hierarchical network coding. The performance tendency is consistent with that in the graph analyses. The case for our hierarchical network coding generates $(N-1)\times(N-1)$ packets simultaneously, where N is the network size. By contrast, the case for our hierarchical network coding together drastically reduces the number of packets. The hierarchical network coding speeds up the multicast communications in the all-to-all broadcast scenario by three times in the execution time for a 32-ary 2-mesh. Another important finding is that the performance is slightly improved by the hierarchical network coding in small networks up to 8-ary 2-mesh. In contrast, it achieves significantly good performance in 16-ary 2-mesh and 32-ary 2-mesh networks. We see the graph shapes of both methods in Fig. 12.

5.5 Packet Length

Generally long packets increase the possibility of incurring packet contentions under a heavy traffic load. It may seriously degrade the performance of the hierarchical network coding. We investigate the performance under various packet lengths. The hierarchical network coding with the configuration of the best grouping (32 groups) in a 16-ary 2-mesh with different packet lengths (1, 2, 4, 8, and 16 flits per packet) was evaluated. Figure 13 shows the execution time for each packet length when compared to the original tree-based multicast. The network coding always improve the execution time. As the packet length increases, it becomes more beneficial, up to 53% for the case of 16-flit packet transfer.

5.6 Latency Overhead in Network Coding

We finally evaluate the hierarchical network coding with different latency overheads to compute the XOR bit operation at the intermediate nodes. Figure 14 shows the execution time, including the overhead for computing XOR at the intermediate node in a 16-ary 2-mesh. The x-axis represents the overhead clock cycles. Surprisingly, the overhead only marginally affects the end-to-end latency. We find that it is not a bottleneck for the collective communications.

6. Discussion

6.1 Deciding Group Size

In the grouping step, the number of nodes inside each group (M), that corresponds the number of groups (G), affects the multicast performance, since our proposed network coding scheme has a hierarchical multicasting structure with a number of intra and inter-group unicasts. Thus we carefully decide group size by considering the balance between intra- and inter-group communications. For example, if group size is small, then the number of packets inside a group decreases at the expense of the increase of the number of packets between intermediate nodes. In Fig. 6 the largest hop count is the case for configuration 2M × 128G. By contrast, if the group size is large, then the number of packets by intermediate nodes (Step D) and direct nodes (Step F) are small at the expense of the increased number of packets inside the group and the number of encoded packets to be delivered to the groups.

We additionally calculate the longest path count for group nodes (inside a group) and intermediate nodes (between groups) for 16-ary 2-D mesh network, as shown in Fig. 15, where "M-d1" and "M-d2" are the dimensions of group, "M-hops" is the longest path count between nodes inside the group. "G-d1" and "G-d2" are the number of groups per 16 (1-D mesh network), "G-d1-hops" and "G-d2-hops" is the hop count per dimension between intermediate nodes, and "G-hops" is the total longest path count for the 2-D mesh groups. An example for "M-hops" and "G-hops" is shown in Fig. 7 for different group sizes in 4-ary 2-D mesh.

Figure 16 and Fig. 17 show that group size 32, which achieved the best results in quantitative analysis and cycle accurate simulation, achieved the best balance between longest path count for group nodes and intermediate nodes and so on the balance between inter and intra-group communication.

This criteria can be more generalized. For example, for

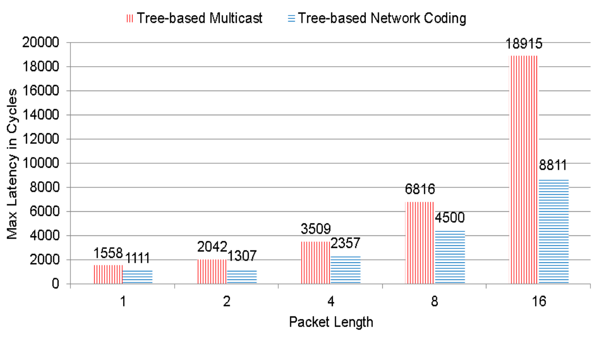


Fig. 13 Execution time for various packet lengths for tree-based multicast and that with hierarchical network coding in 16-ary 2-mesh.

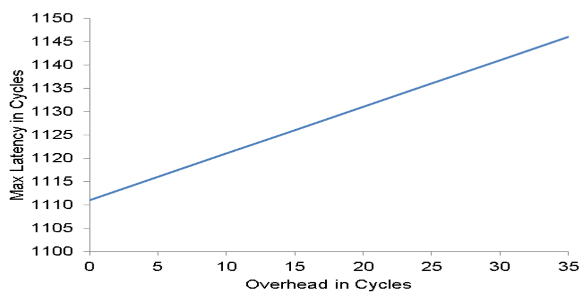
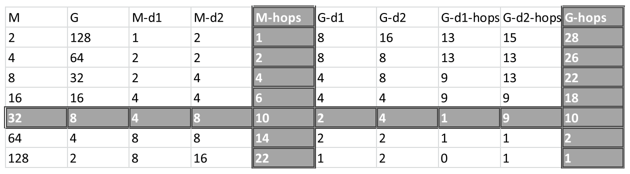


Fig. 14 Execution time for various latency overhead at intermediate nodes in 16-ary 2-mesh.



M	G	M-d1	M-d2	M-hops	G-d1	G-d2	G-d1-hops	G-d2-hops	G-hops
2	128	1	2	1	8	16	13	15	28
4	64	2	2	2	8	8	13	13	26
8	32	2	4	4	4	8	9	13	22
16	16	4	4	6	4	4	9	9	18
32	8	4	8	10	2	4	1	9	10
64	4	8	8	14	2	2	1	1	2
128	2	8	16	22	1	2	0	1	1

Fig. 15 Aggregate hop count for longest path inside and between groups, H-hops and G-hops, for different group sizes configuration in 16-ary 2-D mesh.

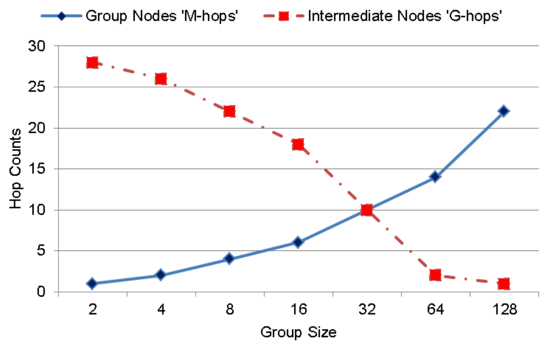


Fig. 16 Aggregate hop count for longest path inside and between groups, H-hops and G-hops, for different group sizes configuration in 16-ary 2-D mesh.

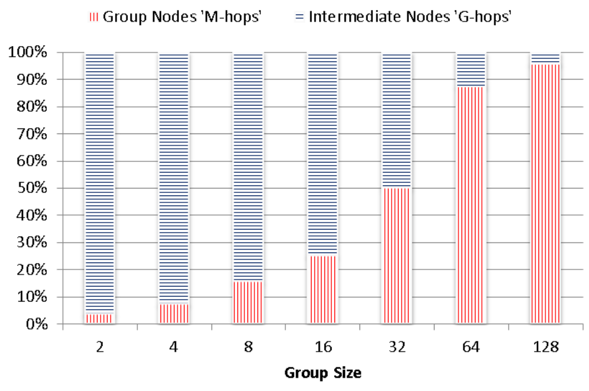


Fig. 17 Aggregate hop count ratios for longest path inside and between groups, H-hops and G-hops, for different group sizes configuration in 16-ary 2-D mesh.

4-ary 2-D mesh as shown in Fig. 7, the best group size is 4, which achieve the best balance between communications for group nodes and intermediate nodes. The longest path counts is 2 for both.

6.2 Latency Overhead at Intermediate Node in Real Deployment

The implementation of network coding is important [24], since the implementation generally affects the multicast performance. However, these implementation issues of network coding assume to take complex calculation for coding for different purposes, e.g. high reliability, supporting unknown intermediate nodes, using unknown topologies. By contrast, in our work we take a simplest XOR coding for only two packets so that a software implementation overhead does not affect the entire execution time of multicasting.

We consider the latency overhead at an intermediate node in real deployments of direct and indirect networks. In direct networks, the recent technology of many-core chips enables a router to be integrated with processors on a single chip [20]. An intermediate node (i.e., a chip) performs XOR bit computation on the same chip with a router. Its communication overhead is much smaller than that of the node-to-node hop delay. Therefore, the latency overhead at an intermediate node in direct networks can be marginal when compared to end-to-end communication latency.

In indirect networks including Gigabit Ethernet, Myrinet and InfiniBand, a commodity switch is connected to some hosts. An intermediate node (i.e., a switch and one of its attached hosts) performs XOR bit computation by transferring three packets between a switch and a host. Specifically, a host receives two packets from a switch, computes their XOR in parallel, and sends the result back to the switch. To estimate the latency overhead of this process, we consider the discussion on a mechanism proposed in [23] to avoid the deadlock of packets. The mechanism ejects a packet from a switch when it incurs a potential deadlock. The packet is then sent to one of the attached hosts, held there for a while, sent back to the switch, and re-injected into the network. This process is identical to the XOR bit computation process of the network coding. The overhead of this mechanism is quantitatively evaluated in [23] by an implementation based on Myrinet GM software. As a result, the software overhead of 125 ns is reported. This is relatively low when compared to the end-to-end communication latency as of 2001. The Myrinet network interface has some latency overhead to process a header and to start the direct memory access (DMA). The relative overall overhead of this process, including DMA, is 10% for short packets and 3% for long packets in comparison to the end-to-end communication latency in a 2-switch network. The relative overhead should be smaller in larger networks. Consequently, we expect that the latency overhead of the network coding at an intermediate node in indirect networks is marginal, especially in large-scale networks.

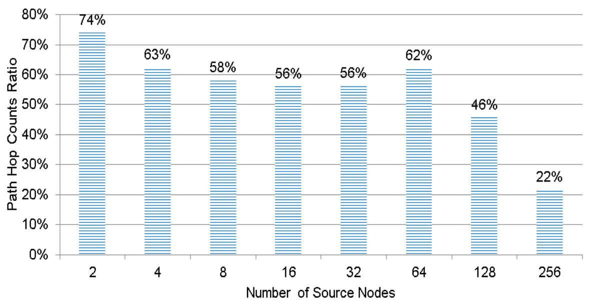


Fig. 18 Aggregate hop count ratios of all-at-once hierarchical network coding against all-at-once multicast from multiple source nodes to all on 16-ary 2-mesh.

We found that the latency overhead at intermediate node hardly affects the execution time of all-to-all broadcast with the hierarchical network coding, as seen in the simulation results in Fig. 14.

Figure 14 shows that a trivial software implementation of XOR operation will hardly affect the execution time of the entire multicast when compared to the case for *zero* XOR/memory copy overhead. This is our intention in this proposal that our hierarchical network coding works well with existing commodity network components with a slightly software update, like [23].

In the simulation, we assume that the minimum router hop delay is three cycles, whereas the overhead at an intermediate node is set to 35 cycles or less. Since the hop delay of current InfiniBand switch products reaches 100 ns, the hierarchical network coding would be beneficial in real deployment, even if its overhead is as large as some microseconds.

6.3 Many-to-All Multicast

Our hierarchical network coding is efficient not only for the all-to-all broadcast but also for the multicast in k-ary n-cubes. We extend the evaluation for multicast communication and multiple sources to all destinations. Multicast communication from 2, 4, 8, 16, 32, 64, 128, and 256 source nodes to all destination nodes is implemented by an all-at-once multicast with the hierarchical network coding in a 16-ary 2-mesh. Figure 18 shows the aggregate hop counts by hierarchical network coding relative to those by the all-at-once multicast. We can observe that the benefits of the hierarchical network coding increase as the number of sources increases. Thus, the most beneficial case is the all-to-all multicast scenario.

6.4 Recursively Applying Network Coding

Another optimization and improvement to the proposed scheme is recursive network coding, which applies a network coding one more time inside a group instead of broadcasting (step B in the network coding procedure: intra-group broadcast). With this optimization we can further reduce the number of unicasts, network resource usage, and thus the

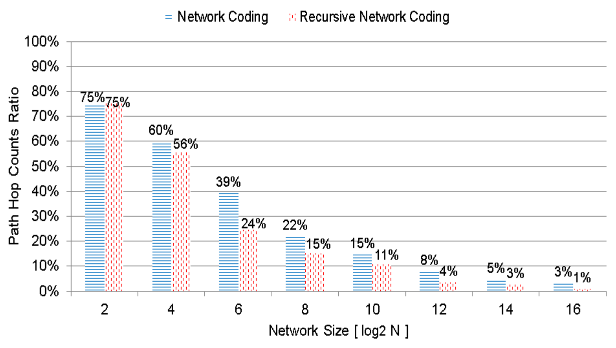


Fig. 19 Aggregate hop count ratios of recursive hierarchical network coding and hierarchical network coding on *k*-ary 2-mesh.

congestion inside a network. We extend our evaluation to measure the benefits of the recursive hierarchical network coding. We implement the recursive network coding for *k*-ary 2-meshes.

Figure 19 illustrates the aggregate hop counts by the hierarchical network coding and by the recursive hierarchical network coding relative to those by the all-at-once multicast without any network coding. We observe that the recursive network coding achieves more reduction in the aggregate hop counts; however, the reduction ratio varies along network sizes. This is because the network size affects the grouping size which plays an important role in the performance as shown in Sects. 4.3 and 5.3. We found that the advantage to recursively applying the network coding is marginal.

6.5 Message Combining vs. Network Coding

In a message combining scenario, an intermediate node in each group combines incoming multiple packets into one message, and sends the combined message to the other intermediate nodes. In the hierarchical network coding scenario, on the other hand, an intermediate node in each group generates a packet by computing the XOR function for two incoming packets. The hierarchical network coding generates a larger number of packets. However, the size of the combined packet in the hierarchical network coding is the same as the size of the incoming packets, whereas the size of the combined message in the message combining is the sum of the incoming packets to be combined. Furthermore, in the message combining scenario, the intermediate nodes must wait for all the incoming packets to start combining. Obviously, the hierarchical network coding is expected to have a lower end-to-end latency when using small packets. We extend our evaluation to compare between the two scenarios, in which the network size is 16 and the group size is 32 (best group size). From the results we observe the end-to-end latency for the hierarchical network coding is 1,276 cycles while it is 1,832 cycles for the message combining. These results confirm our expectation.

7. Conclusions

In this work we proposed the use of network coding for relaxing the relatively low network bandwidth problem in collective communication in HPC off-chip interconnects. Our network coding has a hierarchical multicast structure with intra-group and inter-group unicasts. Since it reduces both the number of unicasts and the transfer data size in multicast, good performance was obtained in various combinations of the (unicast-based) multicast algorithm, the topology, and the transfer data size when a proper group size is set. Although our hierarchical network coding can be applied recursively into its (sub-) groups, its effect is marginal.

Quantitative analysis results show that the hierarchical network coding is beneficial as the network size becomes large. A 94% improvement is obtained in a 4,096-switch network with a conventional tree-based multicast. Our network coding improves the execution time of collective communication by up to 70% in a 32-ary 2-mesh. Cycle-accurate network simulation results validate the quantitative analysis results in various topologies, multicast algorithms, packet sizes and overhead latencies to compute the XOR in intermediate nodes.

Our future work will attempt to analyze the case for complex encoding computation instead of the XOR computation at intermediate nodes in the hierarchical network coding so that more than two packets are aggregated to the resulting encoded packet. Since this may further reduce the number of unicasts in a multicast, this is theoretically an interesting topic. However, recent interconnects are definitely latency-sensitive on the order of hundreds of nanoseconds. We expect that the interconnection networks will impose an impractically large latency overhead at intermediate nodes. This extension is beyond the scope of this paper.

Acknowledgements

This work was partially supported by JST CREST and KAKENHI # 25280018 and 25280043.

References

[1] D.E. Atkins, K.K. Droegemeier, S.I. Feldman, H. Garcia-Molina, M.L. Klein, D.G. Messerschmitt, P. Messina, J.P. Ostriker, and M.H. Wright, "Revolutionizing science and engineering through cyber-infrastructure. Report of the national Science," Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, 2003.
[2] J. Tomkins, "Interconnects: A Buyers point of view," ACS Workshop, 2007.
[3] F. Petrini, S. Coll, E. Frachtenberg, and A. Hoisie, "Hardware-and software-based collective communication on the Quadrics network," IEEE International Symposium on Network Computing and Applications, NCA, pp.24–35, 2001.
[4] MPI. The Message Passing Interface. http://www.mcs.anl.gov/research/projects/mpi
[5] K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, and K.A. Yelick, "The landscape of parallel computing research: A view

from Berkeley," Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, vol.2, 2006.
[6] Top 500 Supercomputers, http://www.top500.org
[7] J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An engineering approach, Morgan Kaufmann, 2002.
[8] InfiniBand Trade Association. http://www.infinibandta.org.
[9] Myricom. http://www.myricom.com/scs/myrinet/m3switch/guide/myrinet-2000_switch_guide.pdf
[10] R. Kesavan and D. Panda, "Efficient multicast on irregular switch-based cut-through networks with up-down routing," IEEE Trans. Parallel Distrib. Syst., vol.12, no.8, pp.808–828, 2001.
[11] P. McKinley, H. Xu, A.H. Esfahanian, and L. Ni, "Unicast-based multicast communication in wormhole-routed networks," IEEE Trans. Parallel Distrib. Syst., vol.5, no.12, pp.1252–1265, 1994.
[12] M. Koibuchi, K. Watanabe, T. Otsuka, and H. Amano, "Performance evaluation of deterministic routings, multicasts, and topologies on RHiNET-2 cluster," IEEE Trans. Parallel Distrib. Syst., vol.16, no.8, pp.747–759, 2005.
[13] A. Gottlieb, R. Grishman, C.P. Kruskal, K.P. McAuliffe, L. Rudolph, and M. Snir, "The NYU ultracomputer - Designing an MIMD shared memory parallel computer," IEEE Trans. Comput., vol.32, no.2, pp.175–189, 1983.
[14] J. Doi and Y. Negishi, "Overlapping methods of all-to-all communication and FFT algorithms for torus-connected massively parallel supercomputers," Proc. ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. Ser. SC '10, pp.1–9, 2010.
[15] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," IEEE Trans. Inf. Theory, vol.46, pp.1204–1216, 2000.
[16] M. Médard and A. Sprintson, Network coding: Fundamentals and applications, Academic Press, 2012.
[17] P.A. Chou and Y. Wu, "Network coding for the Internet and wireless networks," IEEE Signal Process. Mag., vol.24, no.5, pp.77–85, 2007.
[18] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Hybrid ARQ-random network coding for wireless media streaming," Proc. 2nd International Conference on Communications and Electronics (ICCE 2008), pp.115–120, June 2008.
[19] M. Wang and B. Li, "Random push with random network coding in live peer-to-peer streaming," IEEE J. Sel. Areas Commun., vol.25, no.9, pp.1655–1666, 2007.
[20] P. Coteus, H.R. Bickford, T.M. Cipolla, P.G. Crumley, A. Gara, S.A. Hall, G.V. Kopcsay, A.P. Lanzetta, L.S. Mok, R. Rand, R. Swetz, T. Takken, P. La Rocca, C. Marroquin, P.R. Germann, and M.J. Jeanson, "Packaging the blue gene/L supercomputer," IBM Journal of Research and Development, vol.49[2/3], pp.213–248, 2005.
[21] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6D mesh/torus interconnect for exascale computers," Computer, vol.42, pp.36–40, 2009.
[22] N. Jiang, D.U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, J. Kim, and W.J. Dally, BookSim Interconnection network simulator. https://nocs.stanford.edu/cgibin/trac.cgi/wiki/Resources/BookSim
[23] J. Flich, P. López, M.P. Malumbres, J. Duato, and T. Rokicki, "Applying in-transit buffers to boost the performance of networks with source routing," IEEE Trans. Comput., vol.52, no.9, pp.1134–1153, 2003.
[24] G. Angelopoulos, M. Médard, and A.P. Chandrakasan, "Energy-aware hardware implementation of network coding," Networking Workshops, Springer Berlin Heidelberg, 2011.
[25] A. Shalaby, M. Ragab, V. Goulart, I. Fujiwara, and M. Koibuchi, "Hierarchical network coding for collective communication on HPC interconnects," 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp.98–102, Feb. 2014.

Ahmed Shalaby received the B.S. and M.S. degrees in Electrical Engineering from Alexandria University, Egypt in 2005 and 2010 respectively. He received his Ph.D. from Egypt-Japan University of Science and Technology (E-JUST) in 2013. He worked in Questa Team, Mentor Graphics in 2007, moved to Intel Mobile Communication as PHY layer Digital/DSP Engineer in 2008 till 2010. Currently he is a postdoctoral at ECE department E-JUST.

Ikki Fujiwara received the B.E. and M.E. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2002 and 2004, respectively, and received the Ph.D. degree from the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2012. He is currently a Project 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 IPSJ, IEICE and IEEE.

Michihiro Koibuchi received the B.E., M.E., and Ph.D. degrees from Keio University, Yokohama, Kanagawa, Japan, in 2000, 2002, and 2003, respectively. He is currently an Associate Professor in the Information Systems Architecture Research Division, National Institute of Informatics, and the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a member of the IPSJ, IEICE and IEEE.