Performance Evaluation of Deterministic Routings, Multicasts, and Topologies on RHiNET-2 Cluster

Michihiro Koibuchi, Konosuke Watanabe, Tomohiro Otsuka, Hideharu Amano, *Member, IEEE,*

***Abstract*— System Area Networks (SANs), which usually accept arbitrary topologies, have been used to connect nodes in PC/WS clusters or high-performance storage systems. Although deadlock-free routings, multicasts, and topologies for SANs have been widely developed, their evaluation on real PC clusters was rarely done. Thus, the evaluation of routings, multicasts and topologies in real systems is important to analyze their impact on the total systems and validate their simulation results. In this paper, we implement and evaluate deadlock-free routings and unicast-based multicasts under various topologies and channel buffer sizes on a PC cluster called RHiNET-2 with 64 hosts. Execution results show that descending layers (DL) routing and structured channel pools improve up to 57% of bandwidth and 34% of barrier synchronization time compared with up*/down* routing. They also show that, by visiting hosts in numerical order, execution time of unicast-based barrier synchronization is improved up to 28% compared with that in random order. However, channel buffer sizes don't affect the bandwidth in the RHiNET-2 cluster. In addition to fundamental evaluation, we appraise them using NAS Parallel Benchmarks, and the DL routing achieves 3.2% improvement on their execution time compared with up*/down* routing.**

***Index Terms*— Deterministic routing, multicast, topology, performance evaluation, System Area Networks, RHiNET, interconnection networks, PC clusters**

I. Introduction

Network-based parallel processing using commodity components, such as personal computers, has received attention as an important parallel-computing environment [1] [2] [3] [4]. System Area Network (SAN), which consists of switches connected with point-to-point links, is a key technology of such an environment. Unlike Local Area Networks (LANs), wormhole[5] or virtual cut-through[6], in which deadlock-free routing is essential, is used in SANs for low-latency direct-communication. On the other hand, unlike interconnection networks used in massively parallel computers [7] [8], most of SANs usually accept arbitrary topologies, thus requiring generic routing algorithms[9] [3].

Topological generality introduces difficulty on guarantee of connectivity and deadlock-free packet transfer. Thus, the following simple deterministic routings have been used as practical solutions: 1) spanning-tree based routings, which exploit the connectivity and acyclicity of tree structure (e.g. up*/down*[10]); 2) ones using virtual channels to eliminate deadlocks (e.g. structured buffer pools[11], LASH[12], DL[13]). Deadlock-free multicast is also difficult to be implemented on such irregular networks. In order to cope with complicated topologies, algorithms for unicast-based multicast have been proposed and used[14] [15].

There are a large number of researches on performance evaluation of such routing algorithms, however, most of them use a computer simulation with probabilistic models[16], or execution driven models[17]. Thus, the evaluation of routings, multicasts and topologies in real systems is important to analyze their impact on the total systems and validate their simulation results.

In this paper, we evaluate the performance of various deadlock-free routings and unicast-based multicasts on various network topologies in a real PC cluster called RHiNET-2 with 64 hosts[18][19]. RHiNET-2 is an experimental SAN consisting of the following components: 1) RHiNET-2/NI, that is a network interface built in a host, and supports a user-level direct-memory-communication completely by its hard-wired logic; 2) 8 Gbps optical link; 3) RHiNET-2/SW, that is a 64 Gbps cut-through switch. Like Myrinet[1] and InfiniBand[3], RHiNET-2 supports only deterministic routings. However, it provides a simple re-writable routing table and sixteen virtual channels, which enable us to implement various kinds of deterministic routings and topologies. Since RHiNET-2 provides MPI library via a low-level communication library PMv2[20], a large number of scientific applications can be executed.

The rest of this paper is organized as follows. In Section II, existing deadlock-free routings and multicasts are introduced, and their implementation on the RHiNET-2 cluster is described in Section III. In Section IV, the execution results and the related works are shown, and in Section V, we conclude this paper.

II. Routing algorithms

A. Deterministic Routings

Unlike adaptive routing which dynamically changes paths of packets, a path is fixed statically in deterministic routing, and it has the following advantages: 1) switch complexity is decreased, because adaptive routing must include a logic to dynamically select an output channel from alternative channels at packet routing; 2) it guarantees in-order packet delivery, which several message passing libraries require. Since applications usually use message passing libraries, adaptive routing requires a sorting mechanism at hosts or network interfaces. Thus, most of real SANs (e.g. Myrinet, RHiNET, InfiniBand and QsNet) currently adopt deterministic routings as a compact network architecture.

Manuscript received January 20, 2002; revised August 13, 2002.

M. Koibuchi, K. Watanabe, T. Otsuka, and H. Amano are with the Department of Information and Computer Science, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223-8522, Japan. E-mail: {koibuchi, nosuke, terry, hunga}@am.ics.keio.ac.jp.

The simplest deterministic routing used in SANs is up*/down* routing [10]. Up*/down* routing exploits a spanning-tree based directed-graph in which *up* or *down* direction is assigned to each network channel. There are two alternative methods to build a spanning tree: breadth-first search (BFS) and depth-first search (DFS)[21]. Prefix routing[22], which cuts down the size of routing table, is also based on a spanning tree.

On the other hand, routing methods using virtual channels have been also proposed[11] [12] [13]. In structured channel pools (SBP), a packet stored in the virtual network i is transferred to the virtual network $(i+1)$ in order to guarantee both minimal paths and deadlock-free routing. However, it needs $(D+1)$ virtual channels, where D indicates the diameter of SANs. Layered shortest path (LASH) routing proposed by Skeie et al. also guarantees minimal paths by dividing the physical network into a set of virtual networks[12]. Descending layers (DL) routing proposed by us divides the target network into virtual networks with the same topology consisting of layers of virtual channels like the LASH routing, and it establishes a large number of paths across virtual networks in order to reduce the path length and path congestion.

Table I shows path hops, which are the number of intermediate switches between hosts, of routing algorithms on topologies shown in Figure 1. Although SANs usually accept arbitrary topologies, complete randomized connections are not realistic. Thus, we take the topologies with a certain degree of regularity. These four topologies are used in execution in Section IV.

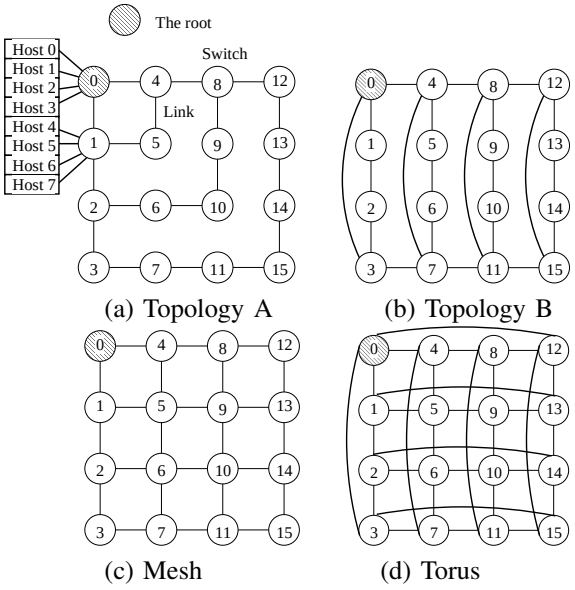


Fig. 1. Topologies considered in execution

"BFS U/D" represents up*/down* routing based on the BFS spanning tree, whose root is switch $zero$ in Table I, while "DFS U/D" represents that based on the DFS spanning tree with a heuristic rule[21]. "Topl." shows the topological (physical) path hops. On both Topology A and B in Table I, prefix routing and both of up*/down* routings must accept some non-minimal paths, while the DL routing and the SBP can use minimal paths. On the other hand, in the mesh and torus, a packet takes a minimal path in all the routings except prefix routing.

TABLE I
PATH HOPS

	Topology A		Topology B		Mesh		Torus	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
Topl.	4.09	7	3.63	7	3.50	7	3.00	5
Prefix	4.89	11	3.95	8	4.47	11	3.43	7
BFS U/D	4.56	11	3.83	8	3.50	7	3.00	5
DFS U/D	4.58	11	3.75	7	3.50	7	3.00	5
SBP	4.09	7	3.63	7	3.50	7	3.00	5
DL(2vch)	4.09	7	3.63	7	3.50	7	3.00	5

All the routings except prefix routing originally provide alternative paths. In order to implement as a deterministic routing, it is necessary to statically choose a single path[23] [24]. The simple selection algorithm chooses a path with a smaller port number at a switch. In this paper, this is called "low port first"[24]. However, low port first may introduce unbalanced paths. Thus, a traffic balancing algorithm using a static analysis of paths has been proposed by Sancho et al[23].

B. Unicast-Based Multicast Algorithms

Tree-, path-, and unicast-based algorithms are typical multicast methods[16].

Tree-based (hardware) multicast requires a logic to control packet duplication at switches and multiple-destination tags, whose size could be large in massive systems, at packet header. Tree-based multicast is efficient especially at a broadcast operation in regular topologies like fat tree. Although path-based multicast requires an efficient multicast-path search, SANs usually accept arbitrary topologies which may introduce difficulty on efficient multicast-paths.

Unicast-based multicast requires $\lceil log_2(d+1) \rceil$ unicast steps for d destinations[25], and its key issues are to decrease both their unicast hops and contentions[14]. The contention means that a packet overlaps with another packet on a physical channel at the same time, and it would introduce a large latency. The contention can be avoided by renewing routes. In an example of the contention of the two packets in Figure 2, it can be solved by changing a route from host 3 to host 17 to one via switch d. However, it is difficult that a unicast-based multicast spatially or temporally schedules and avoids all the contentions under any topology.

The simplest multicast algorithm is the low host-ID order (HIO), that delivers data to each host in the host-ID order. Another is the random order (RO), that delivers data to each host in random order. Although the RO algorithm is often used in MPI implementation[14], it has possibility to generate an inefficient visiting order. For example, assuming a multicast from source host 0 to destination host $1, 2, \ldots, 11$ and $16, 17, \ldots, 19$ shown in Figure 2, the RO algorithm may generate packets with maximum hops from host $4, 5, 6$ and 7 to host $16, 17, 18$ and 19 respectively.

To address this problem, switch-based hierarchical-order (SHO) algorithm, which groups hosts connected to the same switch, uses two-step hierarchical multicast (intraswitch and

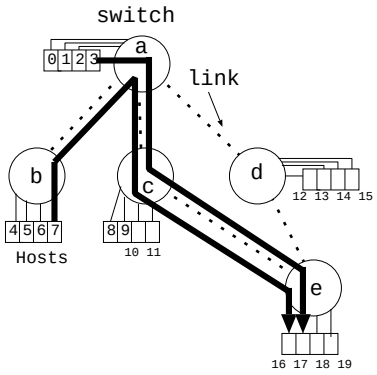


Fig. 2. An example of contention in SANs

Fig. 3. RHiNET-2 cluster with 64 hosts

interswitch)[14]. After delivering data to a single destination host in each group, the SHO algorithm delivers data to remaining destination hosts from their group hosts. For example, assuming the same multicast using the SHO algorithm in Figure 2, after host 0 sends data to host 8, host 0 and 8 send data to host 4 and 16 respectively. Finally, host $0, 4, 8$ and 16 send data to the other destination hosts in their groups. Thus, the SHO algorithm decreases its unicast hops and contentions compared with the RO algorithm.

III. THE RHINET-2 CLUSTER

In this section, the RHiNET-2 cluster used in the evaluation is introduced.

A. Components of the RHiNET-2 Cluster

Real World Computing Partnership (RWCP) carried out a research project called RHiNET[2] for establishing high-end system area networks in collaboration with Hitachi Co. Ltd. and Keio University. RHiNET-2 is a network designed not only for dedicated clusters but also parallel computing environments using personal computers distributed within one or more floors of a building.

The RHiNET-2 cluster with 64 hosts, that is a prototype of such systems shown in Figure 3, consists of hosts with specially designed network interfaces (RHiNET-2/NI) and switches (RHiNET-2/SW) connected with 8 Gbps optical interconnects[18] [19]. Table II shows specifications of each host in the RHiNET-2 cluster.

TABLE II
SPECIFICATIONS OF THE HOST

CPU	Intel Pentium III 933MHz × 2 (SMP)
Memory	PC133 SDRAM 1GByte
Chipset	Serverworks ServerSet III HE-SL
PCI	64bit/66MHz
OS	RedHat Linux 7.2 (kernel 2.4.18)

a) Network Interface RHiNET-2/NI: A network interface RHiNET-2/NI carries a network controller chip Martini[19], 256 MByte SDRAM, and optical interfaces for 800 MHz interconnects. It is put into a common personal computer with 64bit/66MHz PCI bus. Martini is an ASIC chip which manages fundamental zero-copy communications only with a hard-wired logic including complicated processing for address conversion and memory protection. It also provides an on-chip processor compatible with MIPS R3000 for exceptional processing. Martini is fabricated by Hitachi Device Development Center using 0.14μm CMOS embedded array technology.

Martini provides two types of primitive communication methods. One is a remote DMA transfer for high-bandwidth communication: PUSH (remote write) and PULL (remote read). The other is a PIO-based transfer for low-latency communication. Since latency of address conversion or DMA setup cannot be ignored for a small-sized DMA transfer, the PIO-based transfer is suitable for sending a small-sized data. In both cases, each packet, whose header and tailer sizes are 40 Byte (5 flits) in total, is transferred by splitting into 8-Byte flits.

b) Network Switch RHiNET-2/SW: A network switch RHiNET-2/SW[26] provides eight input/output ports each of which is connected to an 8 Gbps optical link. The core of RHiNET-2/SW is one-chip switching fabric with 0.18μm CMOS embedded array technology. It provides 800 Mbit/sec-per-signal high-speed low-voltage differential signaling (LVDS) I/O. Its aggregate throughput is 64 Gbps. However, in this evaluation, the frequency of optical modules is decreased from 800 MHz to 600 MHz in order to cope with heat dissipation problem. That is, the link bandwidth is 6 Gbps, and the total throughput of RHiNET-2/SW is actually 48 Gbps at present.

Sixteen virtual channels at each port are provided, and each virtual channel has a 4 KByte buffer on a chip so that Go & Stop flow control supports up to 200 m link length.

B. Routing in RHiNET-2/SW

RHiNET-2/SW supports deterministic routings with a table look-up manner, in which a packet finds its output port by referring a routing table at each switch. At the table reference, only destination tag is used to get output port. Each virtual channel transition is determined by a pair of input port and output port at the table reference. As mentioned before, sixteen virtual channels are provided in RHiNET-2/SW. Eight of them numbered from 0 to 7 are reserved for data-transmission packets, while those numbered from 8 to 15 are reserved for

system control packets. The same routing table is shared with the two types of virtual channels.

Since the routing mechanism in RHiNET-2/SW is simple[1], it accepts various topologies and routing algorithms by rewriting the routing table. Each switch is equipped with a simple control processor, and the content of routing table is loaded from the processor at initialization.

C. System Software

It is difficult that MPI function is directly associated with PUSH/PULL or PIO-based primitives[18]. Thus, a low-level communication library PMv2[20], and SCore cluster system software[27] are implemented on the RHiNET-2 cluster[18]. SCore is a high-performance parallel programming environment for PC clusters, and it provides MPI library (MPICH-SCore). SCore and MPICH-SCore are open source software, and they improve bandwidth and latency compared with GM software on Myrinet[20]. The detail of the RHiNET-2 cluster and its performance are described in [18] and [19].

IV. Performance Evaluation

In this section, we will evaluate performance of unicast routings, multicasts, topologies, and channel buffer sizes on the RHiNET-2 cluster.

A. Environment

1) Implementation of Deterministic Routings: We implemented up*/down, prefix, the SBP, and the DL routings considering a large number of virtual channels[2] in the RHiNET-2 cluster.

We used four topologies for routing evaluation shown in Figure 1. In the irregular topologies, since routing algorithms take different path hops, we can investigate the impact of routing hops on system performance. Also, in order to investigate the impact of virtual channels, up*/down* routing is evaluated about the case where the number of virtual channels is 1, or 2 as follows; each path does not change a number of its virtual channels at intermediate switches so as to simply distribute the traffic uniformly among virtual channels. When there are alternative paths, the traffic balancing algorithm is mostly used to select a path[23]. We employed two virtual channels in the DL routing, and their properties on the topologies are shown in Table I.

2) Implementation of Multicasts: We implemented a barrier synchronization operation using the unicast-based multicasts as a foundation of collective communication operations. The barrier synchronization is done by (1) the root host gathers the request messages from all hosts, and (2) the root host distributes the release messages to all hosts. Thus, the barrier synchronization with 64 hosts requires 12 unicast steps consisting of its collection for 6 steps and its release for 6 steps.

Collective operations, such as MPI_Allreduce, usually consist of each of the multicast operation, whose access pattern is similar to one in barrier synchronization, and their specific calculation. Consequently, barrier synchronization can be considered as a primitive collective communication.

The RO, SHO, HIO, Switch-based Contention Order (SCO), 1SHO-3RO and Switch-based Host-ID Order (SHIO) algorithms are used. The RO, SHO, and HIO algorithms have been mentioned in Section II. The SCO, 1SHO-3RO, and SHIO algorithms are introduced only to investigate performance factors of unicast-based multicasts.

a) The SCO Algorithm: The SCO algorithm is introduced to investigate the effect of decreasing unicast contentions on the SHO algorithm. In order to increase contentions, the SCO algorithm reverses the order of a two-step hierarchical multicast (intraswitch and interswitch) of the SHO algorithm. That is, after sending packets to intraswitch hosts, it sends packets to hosts via some switches. Since there are four hosts connected to a single switch in the RHiNET-2 cluster, the SCO algorithm would cause the contention with four packets in a channel between switches.

b) The 1SHO-3RO Algorithm: The 1SHO-3RO algorithm is introduced to investigate the effect of reducing unicast hops within a group (intraswitch) on the SHO algorithm. First, the 1SHO-3RO algorithm delivers data to a single host in each group as well as the SHO algorithm. Second, unlike the SHO algorithm, it delivers the remaining destination hosts in a random visiting order from hosts already received data. Through comparing the SHO and 1SHO-3RO algorithms, the effect of intraswitch communication in the SHO algorithm is analyzed.

c) The SHIO Algorithm: The SHIO algorithm is the same as the SHO algorithm except for interswitch unicast order. The SHIO algorithm sends data to a single host at each switch in a visiting order of low switch-ID (identifier), while the SHO algorithm sends in a random order.

All algorithms except the SHIO and HIO can take various visiting orders including the root selection. Thus, ten different visiting orders are randomly generated and evaluated.

3) Topologies: We evaluated various topologies so as to investigate the impact of them on the RHiNET-2 cluster. Since SANs accept both direct and indirect networks, we evaluated topologies without distinction of direct and indirect networks.

First, we use typical direct networks, two-dimensional mesh and torus, shown in Figure 1. They are easy to establish well-distributed paths because of their symmetric connections.

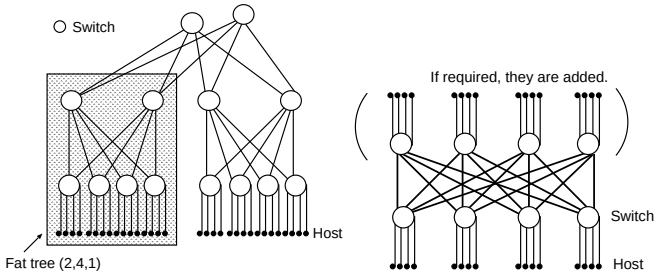
Second, the layered indirect network is used as shown in Figure 4(a). Fat tree mitigates the tree problem that the traffic concentrates around the root by introducing multiple tree structures.

Finally, we try the flat topology called Myrinet Clos as shown in Figure 4(b). Myrinet Clos network is close to the perfect connection. Path hops and bisection width of these topologies are shown in Table I and III.

4) Measures: First, we measured bandwidth and barrier synchronization time, which are essential for supporting par-

[1] RHiNET-2/SW is originally designed for a simple deadlock-free deterministic routing called the modified structured channel[26] for supporting arbitrary topologies.

[2] "The number of virtual channels" in this evaluation represents the number of virtual channels used by data-transmission packets. Notice that system control packets will also use the same number of virtual channels numbered from 8 to 15.



(a) Fat tree(2,4,2) (b) Myrinet Clos (4×4)

Fig. 4. Indirect networks considered in execution

TABLE III
PATH HOPS AND BISECTION WIDTH

Topology (switch)	Avg. hops	Max. hops	Bisection width
Ring(8)	3.00	5	2W
Mesh (4×2)	2.75	5	2W
Torus(4×2)	2.50	4	4W
Myrinet-Clos(8)	2.25	3	8W
Fat-Tree(14)	3.75	5	4W

allel processing. The former is the average bandwidth under the condition that each host repeatedly injects packets using PUSH primitive as short intervals as possible. The following typical traffic patterns are used:

- *bit-reversal*
  A host with the identifier $(a_0, a_1, \cdots, a_{n-1})$ sends a packet to the host whose identifier is the bit reversal $(a_{n-1}, \cdots, a_1, a_0)$ of the source host.
- *matrix transpose*
  A host (x, y) sends a packet to the host $(k-y-1, k-x-1)(k$ is the number of hosts in each dimension) or $(k-x-1, k-y-1)$ when $x+y=k-1$.

Packet sizes are set to be up to 229 flits (1,832 Byte in total). On the other hand, the latter is the average execution time of unicast-based barrier synchronization using PIO-based communication on 100,000 trials. Packet sizes are set as the smallest length (17 flits,136 Byte) and 69 flits (552 Byte) [3].

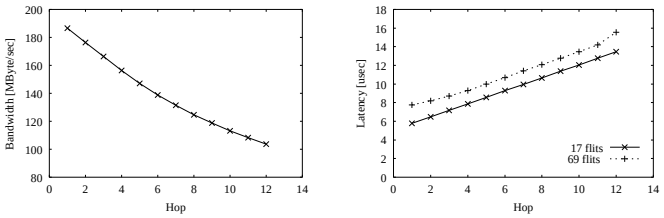
Second, we measured execution time of BT (Block Tridiagonal solver), CG (Conjugate Gradient), IS (Integer Sort), LU (LU-decomposition), MG (Multi-Grid solver) and SP (Scalar Pentadiagonal solver) from NAS parallel benchmarks 2.3[28] [29]. The number of hosts (processes) is 16, 32, or 64 under using 1, 2, or 4 hosts per switch. Since the RHiNET-2 cluster are intended to run scientific applications, we also evaluated routing algorithms using Himeno Benchmark[30], which is a kernel in a linear solver of pressure Poisson equation which appears in an incompressible Navier-Stokes solver. However, the RHiNET-2 cluster with any routing algorithm achieves 824 MFLOPS in 16 hosts under the small set. We consider that it is memory-bandwidth bounded in the RHiNET-2 cluster. Thus, we mainly focus on NAS parallel benchmarks as parallel applications.

[3]The former consists of header and tailer for total 5 flits, raw data for 1 flit and hardware padding for remaining flits. On the other hand, the latter includes header and tailer for 5 flits, and 512-Byte raw data for 64 flits.

B. Routing Impact

In this subsection, we focus on performance evaluation of routing algorithms. Figure 5 shows the bandwidth of each routing algorithm. In this figure, the vertical axis represents bandwidth, while the horizontal one represents the data size of each packet. "SBP(6vch)" shows the structured channel pools using six virtual channels, while "Up*/Down* (1vch)" shows up*/down* routing based on the BFS spanning tree using one virtual channel. Dimension-order routing[5], which uses y-dimension channels after using x-dimension channels, is also evaluated so as to compare path distribution in the mesh and torus. In dimension-order routing in the torus, we employ two virtual channels to avoid deadlocks. In each dimension, virtual channel 1 is firstly used. Only when a wrap-around channel is used, a number of virtual channel is changed from 1 to 0. Figure 5(i) shows the minimum and maximum bandwidth of individual source hosts. The DL routing has smaller gap between minimum and maximum bandwidth than that of up*/down* routing, because the DL routing takes minimal well-distributed paths. Thus, it can be said the DL routing provides more stable communication.

As shown in Figure 5, the DL routing and the SBP achieve up to 57 % improvement compared with up*/down* routing, while the bandwidth of the DL routing and the SBP is almost the same. Thus, we consider that a deadlock-free routing, that 1) takes shorter paths and 2) distributes them more uniformly, achieves higher bandwidth.

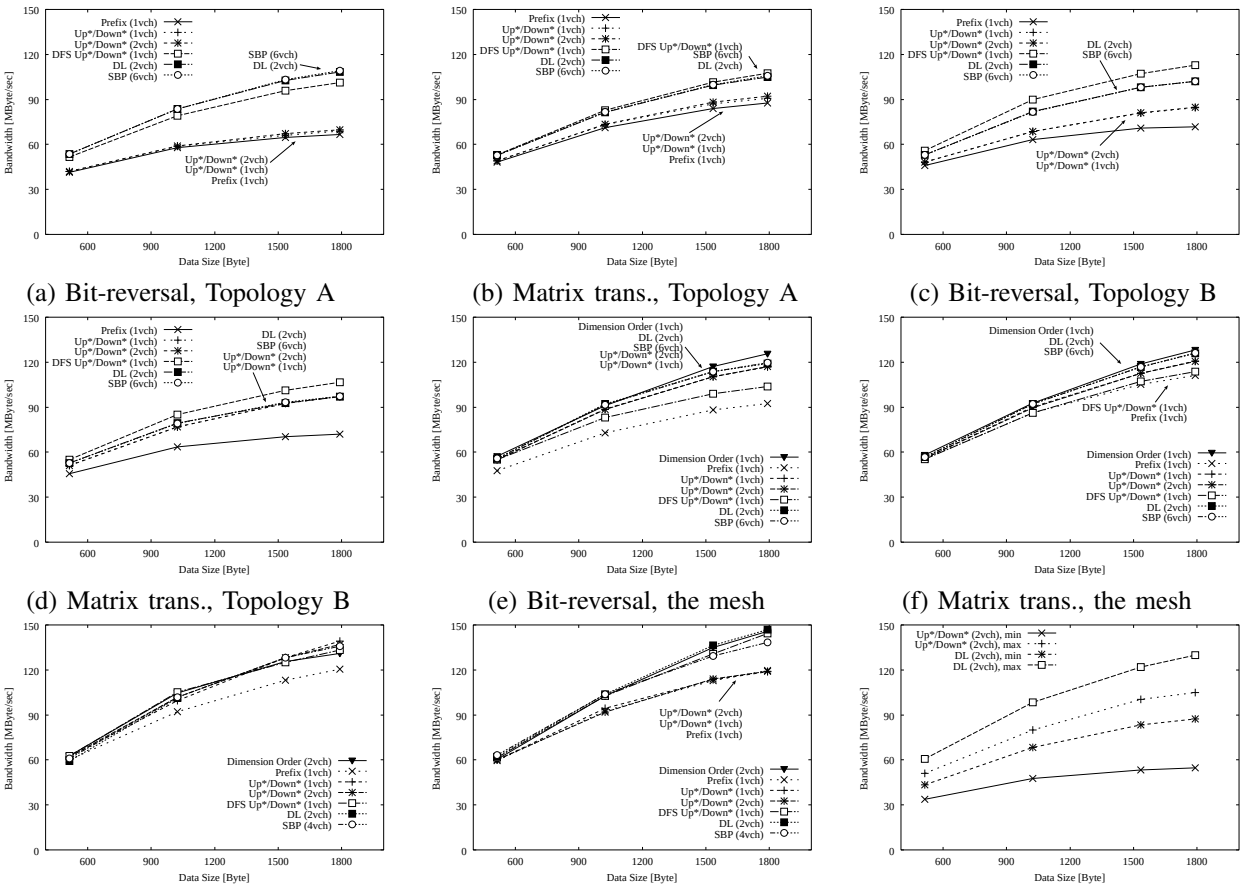


(a) Bandwidth (b) Latency of barrier synchronization

Fig. 6. Bandwidth and latency between two hosts (μsec)

In order to investigate the former —shorter paths—, we shows a relation between the bandwidth of two hosts and their distance under 229-flits-length packets in Figure 6(a). Bandwidth decreases by about 7.5 MByte/sec per hop, because a packet is injected after receiving an acknowledgment of the previous packet. We employ the simple reception protocol as a fundamental evaluation of bandwidth, because it is a fundamental reliable communication protocol between hosts. In addition, a large number of negative acknowledgments, and re-sent packets would be consecutively generated, which would seriously degrade bandwidth, when a destination host occurs a trouble under the no-ack-wait protocol. We evaluate and compare the acknowledge reception protocols at Section 4.5.

As well as path hops, the latter —path distribution— effects on the bandwidth, because dimension-order routing, which distributes paths most uniformly, achieves higher bandwidth than the other minimal routings under the mesh and torus in Figure 5. The DFS up*/down* routing outperforms the BFS



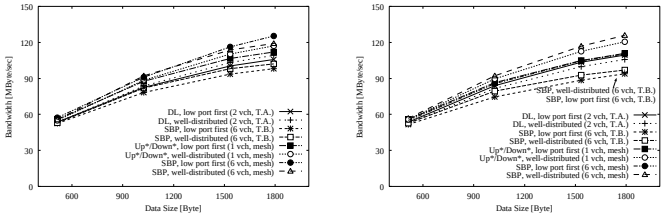
(a) Bit-reversal, Topology A (b) Matrix trans., Topology A (c) Bit-reversal, Topology B

(d) Matrix trans., Topology B (e) Bit-reversal, the mesh (f) Matrix trans., the mesh

(g) Bit-reversal, the torus (h) Matrix trans., the torus (i) Bit-rev, Min,Max, Topology A

Fig. 5. Routing bandwidth under 32 hosts

up*/down* routing, which provides smaller path hops than the DFS one in Topology A. Since the DFS up*/down* routing efficiently avoids unbalanced paths, it also indicates that path distribution can not be ignored.



(a) Bit-reversal traffic (b) Matrix transpose traffic

Fig. 7. Bandwidth of path selection algorithms

Here, Figure 7 shows the impact of the path selection policy at initial path search on each routing algorithm. "Well-distributed" is the traffic balancing algorithm[23], and it used the same path search method as shown in Figure 5. On the other hand, "low port first" chooses a path with a smaller port number at a switch[24]. Figure 7 demonstrates that the path selection influences up to 15% of total bandwidth. It can be said that path distribution has an effect on bandwidth especially at topologies with the larger number of links. Each routing with a well-distribution achieves higher bandwidth than that with low port first in all the cases except for the SBP in the mesh. The SBP with low port first is suitable for the mesh topology, since it selects the similar physical paths to dimension-order routing which efficiently distributes paths using the simple rule.

On the contrary, as shown in Figure 5, the number of virtual channels gives only small impact on bandwidth in every deadlock-free routing. Since a virtual channel mechanism can reduce head-of-line (HOL) blocks, it improves bandwidth by increasing channel utilization. However, in deterministic routing which can't dynamically select an output virtual-channel number at a switch, a virtual channel mechanism is infirm of avoiding HOL blocks compared with adaptive routing.

Consequently, we confirm that 1) path length and 2) path distribution are the main factors that influence the routing bandwidth. From both viewpoints, the DL routing and the SBP outperform up*/down* routing, and the DL routing is similar to the SBP. Since the impact of virtual channels is small, the DL routing is better than the SBP, which wastes the larger number of virtual channels.

The execution time of barrier synchronization with 17-flits-

TABLE IV
LATENCY OF BARRIER SYNCHRONIZATION USING THE RO MULTICAST ALGORITHM(μSEC)

	Topology A		Topology B	Mesh
	Avg	Max - Min	Avg	Avg
Prefix (1vch)	52.39(60.54)	2.10	49.70	51.49
Up*/Down*(1vch)	50.49 (59.77)	2.44	48.65	45.78
Up*/Down*(2vch)	50.53 (59.87)	2.10	48.68	45.77
DFS U/D (1vch)	51.59 (44.65)	0.86	46.75	44.72
SBP	47.83 (44.58)	1.46	46.61	45.61
DL(2vch)	47.86 (44.60)	2.03	46.65	45.62

packets using the RO algorithm is shown in Table IV. The parenthesized number indicates results under the RO visiting order, in which there is the largest difference on routing performance, in Table IV. "MIN" is the minimum barrier-complete time of individual hosts. Table IV illustrates the following properties; 1) in the mesh, the three deadlock-free routings require the same average hops, and they achieve almost the same latency of barrier synchronization: 2) in Topology A and B, the barrier synchronization time of the DL routing and the SBP are almost the same, and up to 34% better than that of up*/down* routing. Thus, we consider that the routing latency of barrier synchronization is influenced with the average hops of packets. Here, Figure 6(b) shows a relation between barrier synchronization time of two hosts and their distance, and it demonstrates that latency increases about 0.7 μsec per hop. Since a unicast-based multicast with 64 hosts requires 12 unicast steps, an important issue of barrier synchronization is how the average hops of unicast are decreased. In this sense, the DL routing and the SBP, which take minimal paths in such topologies, are desirable.

Table V, VI and VII show that the execution time of NAS parallel benchmarks. A small problem, class S, is sometimes used in each benchmark, because RHiNET-2/SW has a bug at flow control, that sometimes discards data in a specific communication. Then, the RHiNET-2 cluster is difficult to completely run large-sized applications. However, the communication ratio against computation is relatively increased under small-sized parallel applications. From the viewpoint of routing evaluation, it is also valuable.

As shown in them, the DL routing improves 3.2% execution time compared with up*/down* routing. We investigate the detailed itemization of the computation and communication time under using two virtual channels in Table VIII. Each item is defined as follows: 1) "MPI overhead" is software overhead in MPI library; 2) "PM overhead" is software overhead in PM library; 3) "send ready" is waiting time for send buffer allocation; 4) "pkt" is waiting time for packet; 5) "NIC" ready is waiting time for a network interface to be ready to process the next packet; 6) "ACK" is waiting time for receiving an acknowledgment of previously sent packet.

"MPI overhead" and "PM overhead" are pure software overhead, which routing performance doesn't affect. In PM message communication, memory space is statically allocated for message send/receive buffer. Therefore, when a send buffer is full, a sender process waits until the buffer becomes free again. Likewise, if no packet has been received yet, a receiver process waits for a packet. Send ready and packet waiting times include overhead of network, however most of them come from a timing gap of send/receive operations. Since "NIC ready" is overhead of a network interface, network performance including routing algorithm doesn't affect it. On the other hand, "ACK" is time after a packet is passed to a network interface until an acknowledgment comes back. Thus, it is greatly influenced by the network performance including routing algorithm.

We measured the itemization of communication at PM library level. We inserted Intel CPU's read-time stamp counter (RDTSC) instructions into PM library code, and recorded elapsed clocks in each itemized section. The RDTSC instruction takes about fifty clocks, so latency of the RDTSC instruction is about 54 nanoseconds in the evaluation environment (933MHz CPU). This value is sufficiently small.

Table VIII illustrates that the execution time is strongly influenced by the communication operation. However, a period of flying packets is short, while the software overhead including MPI and PM is large. Since we consider that network bandwidth is hardly saturated under any routing algorithm, the routing impact on the execution time is limited.

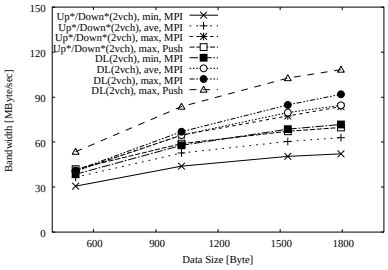


Fig. 8. Routing bandwidth under MPI level, bit-reversal, Topology A

Here, Figure 8 shows routing bandwidth using MPI_Send/MPI_Recv operations under bit-reversal traffic, and it illustrates the DL routing increases 33% of bandwidth compared with up*/down* routing. A packet is inserted in the shortest interval at each host on bandwidth measurement, while the small number of packets are partially and spatially generated on NAS parallel benchmarks. Thus, we consider that the impact of routing algorithms on bandwidth measurement is enhanced compared with that in NAS parallel benchmarks.

From the viewpoint of system balance in PC clusters, in general, it can be said that the network bandwidth (6Gbps) is enhanced compared with CPU computation power(933MHz) and memory size(1GB). This leads the impact of routing algorithms tends to be small in the RHiNET-2 cluster. Nevertheless, the routing algorithm affects the execution time in the RHiNET-2 cluster. We consider that the routing algorithm increasingly influences in most of real PC clusters.

C. Multicast Impact

In this subsection, we focus on performance evaluation of unicast-based multicasts. Table IX, X, and XI show execution time of barrier synchronization with 17-flits-packets. They demonstrate that the RO algorithm wastes up to 112% latency

TABLE V
EXECUTION TIME OF CG, LU, IS, SP AND BT BENCHMARKS UNDER TOPOLOGY A AND B(SEC)

	CG.S.16	LU.W.16	IS.S.16	IS.S.32	IS.S.64	SP.S.16	BT.S.16
Up*/Down* (1vch, T.A)	0.20600	7.78050	0.01886	0.02169	0.11798	0.224	0.179
Up*/Down* (2vch, T.A)	0.20633	7.81633	0.01883	0.02157	0.11798	0.224	0.180
DL (2vch, T.A)	0.20450	7.76525	0.01850	0.02091	0.11751	0.224	0.179
Up*/Down* (1vch, T.B)	0.20325	7.79975	0.01846	0.02087	0.11704	0.224	0.180
Up*/Down* (2vch, T.B)	0.20450	7.79450	0.01843	0.02077	0.11704	0.224	0.178
DL (2vch, T.B)	0.20400	7.77257	0.01838	0.02052	0.11681	0.223	0.178

TABLE VI
EXECUTION TIME OF CG, LU, IS, SP, AND BT BENCHMARKS UNDER THE MESH AND TORUS(SEC)

	CG.S.16	LU.W.16	IS.S.16	IS.S.32	IS.S.64	SP.S.16	BT.S.16
Up*/Down* (1vch, the mesh)	0.20433	7.78100	0.01842	0.02039	0.11686	0.224	0.178
Up*/Down* (2vch, the mesh)	0.20325	7.78433	0.01854	0.02043	0.11670	0.224	0.179
DL (2vch, the mesh)	0.20325	7.76200	0.01836	0.02052	0.11673	0.222	0.177
Up*/Down*(1vch, the torus)	0.20367	7.77100	0.01837	0.02013	0.11580	0.224	0.180
Up*/Down*(2vch, the torus)	0.20367	7.75350	0.01837	0.02010	0.11631	0.222	0.177
DL (2vch, the torus)	0.20367	7.75225	0.01820	0.02008	0.11631	0.221	0.177

TABLE VII
EXECUTION TIME OF MG AND CG BENCHMARKS UNDER THE MESH AND TORUS WITH 16 HOSTS (SEC)

	MG.S	MG.W	MG.A			CG.W	CG.A
	Max	Max	Min	Avg	Max	Max	Max
U/D(2vch, the mesh)	0.016	0.553	3.256	3.256	3.256	0.759	2.083
DL(2vch, the mesh)	0.016	0.551	3.251	3.251	3.251	0.761	2.083
U/D(2vch, the torus)	0.016	0.550	3.246	3.247	3.247	0.756	2.080
DL(2vch, the torus)	0.016	0.551	3.228	3.229	3.229	0.749	2.048

TABLE VIII
ITEMIZED STATEMENT OF IS, SP, BT, MG, AND CG BENCHMARKS (SEC)

	Comp.	Communication						
		Total	MPI	PM	Send ready	Pkt	NIC ready	ACK
DL(IS.S.16, torus)	0.002	0.016	0.002	0.001	0.000	0.008	0.000	0.005
U/D(SP.S.16,T.A)	0.103	0.121	0.035	0.022	0.000	0.043	0.003	0.018
DL (BT.S.16,T.A)	0.088	0.091	0.062	0.007	0.000	0.015	0.001	0.005
DL (MG.W.16, mesh)	0.377	0.174	0.050	0.044	0.000	0.055	0.006	0.019
U/D(MG.A.16, torus)	3.047	0.200	0.036	0.038	0.001	0.087	0.005	0.034
DL (CG.A.16, torus)	1.460	0.588	0.124	0.091	0.001	0.258	0.010	0.104

compared with the SHO algorithm. The SHO algorithm decreases contentions and unicast hops because of its grouping, while the RO algorithm takes unicasts with the average routing hops.

First, we compare the SCO and 1SHO-3RO algorithms with the SHO algorithm in order to investigate the impact of their unicast hops and contentions. Although the SCO algorithm would generate heavy contentions of four packets in a channel between switches, the SCO algorithm increases up to only 5.5% of execution time, which is smaller than the performance gap between the RO and SHO algorithms. Thus, we consider that the contention is not a main factor of latency in this case. On the other hand, the 1SHO-3RO and RO algorithms have the similar latency, however they degrade 12% of latency compared with the SHO algorithm. The SHO and 1SHO-3RO algorithms take different-hops unicasts, and Figure 6(b) shows that latency increases by 0.7 μsec per hop. Thus, it can be said that unicast hops are the main factor that improves multicast algorithms under small-length-packets.

Second, the SHIO and HIO algorithms are compared with the others. The RHiNET-2 cluster assigns the ID to each host in the order of low switch-coordinates number. Also in general, this simple host-ID allocation may be used because of its simplicity. Under such an allocation, the SHIO and HIO algorithms improve up to 28% execution time compared with the RO algorithms as shown in Table IX, X, and XI, because

TABLE IX
LATENCY OF BARRIER SYNCHRONIZATION UNDER TOPOLOGY A (μSEC)

	Prefix	Up*/Down*		DL	SBP
	Avg	Avg	Max - Min	Avg	Avg
RO	52.39	50.53	2.10	47.86	47.83
SHO	47.31	45.09	0.23	43.97	43.98
SCO	49.59	45.79	2.07	44.33	44.37
1SHO-3RO	53.02	49.98	5.66	48.05	48.04
SHIO	44.35	38.99	0.15	39.04	39.04
HIO	44.95	39.62	0.78	39.38	39.32

TABLE X
AVERAGE LATENCY OF BARRIER SYNCHRONIZATION UNDER TOPOLOGY B AND THE MESH (μSEC)

	Topology B				Mesh			
	Prefix	Up*/Down*	DL	SBP	Prefix	Up*/Down*	DL	SBP
RO	49.70	48.68	46.65	46.61	51.49	45.77	45.62	45.61
SHO	44.54	43.95	43.14	43.13	46.20	42.19	42.10	42.10
SCO	47.00	45.02	43.79	43.85	48.52	42.09	42.05	42.07
1SHO-3RO	49.18	48.14	47.07	47.05	51.53	45.92	45.97	45.96
SHIO	40.90	39.86	38.87	38.87	44.32	38.83	38.84	38.85
HIO	43.97	42.85	40.00	40.09	46.14	38.92	38.89	38.92

TABLE XI
AVERAGE LATENCY OF BARRIER SYNCHRONIZATION WITH 16 OR 32 HOSTS(μSEC)

	Fat Tree (16host,6sw)	Myrinet Clos (16host,8sw)	Myrinet Clos (32host,8sw)	Mesh (32host,4×2sw)
RO	27.68	27.62	33.60	35.00
SHO	26.07	25.89	31.90	32.59
SCO	25.90	25.89	31.91	32.55
1SHO-3RO	28.32	28.10	33.32	35.06
SHIO	25.86	25.86	32.42	31.64
HIO	25.85	25.89	32.38	31.65

the SHIO and HIO algorithms generate a large number of unicasts via only one or two switches. By employing SHIO algorithm, the variance of execution time in the individual hosts is reduced.

Third, we focus on the influence of topology and routing algorithm upon the multicast algorithms. Since the average hops of unicasts are determined with a pair of topology and routing algorithm, each multicast algorithm with the DL routing or the SBP, which takes minimal paths, achieves up to 19% improvement compared with that with up*/down* or prefix routing. Similarly, topologies with small diameter improve execution time of each multicast algorithm.

Finally, Table XII shows execution results of broadcast using unicasts with 512 Byte data (total 69 flits). We use 6-switches(2,4,1), and 14-switches(2,4,2) as fat trees shown in Figure 4. Table XII shows the SHO and SHIO algorithms outperform all of the other algorithms, because contentions are more critical for 69-flits-packets than 17-flits-packets. Compared with 6-switches fat tree, 14-switches fat-tree increases the execution time under the same number of hosts. This is because 14-switches fat tree makes larger unicast hops than 6-switches one. Consequently, it can be said that reducing unicast hops has an effect on latency of small-sized multicasts. On the other hand, decreasing contentions is crucial to middle-sized multicasts.

TABLE XII
LATENCY OF BROADCAST WITH 512 BYTE DATA (μSEC)

	Mesh(64hosts,16sw)		Fat tree (16hosts,14sw)	Fat tree (16hosts,6sw)
	Avg	Max - Min	Avg	Avg
RO	65.60	6.78	46.81	37.86
SHO	56.21	2.09	39.06	35.03
SCO	63.08	8.95	40.07	35.73
1SHO-3RO	65.31	8.49	41.05	37.65
SHIO	52.95	0.92	37.29	34.79
HIO	61.65	9.20	38.46	35.65

D. Topology Impact

In this subsection, we focus on the impact of topologies on the RHiNET-2 cluster. Figure 9 shows bandwidth of direct networks using the SBP and indirect networks using minimal routing with no virtual channels. As shown in Figure 9,

the torus achieves up to 91% improvement compared with Topology A. This is because, by using the larger number of links, packet hops are reduced, and paths can be well-distributed in the torus. Similarly, Myrinet Clos network, which takes the shortest well-distributed paths, achieves the highest bandwidth in Figure 9.

The 14-switches fat tree takes smaller bandwidth than the 8-switches torus and mesh. This means that increasing the number of switches doesn't always improve bandwidth, while only packet hops and path distribution are important. Notice that, as shown in Figure 6(a) and 9 the torus bandwidth under complement traffic is similar to bandwidth between two hosts, whose distance is 3-hops, since the SBP takes three-hops paths for all packets with contention-free under complement traffic.

Table XIII shows that execution results of barrier synchronization, and the parenthesized number indicates the number of switches. Two hosts per switch are used for 16-hosts evaluation in the 14-switches fat tree. As shown in Table XIII, topologies, that have the smaller average distance between hosts, achieve the smaller latency. In particular, the 6-switches fat tree outperforms the 14-switches one. Here, we focus on topologies with the same average distances— the 8-switches ring and 8-switches mesh, the Myrinet Clos and 6-switches fat tree. Then, the former —the ring or Myrinet Clos— decreases latency compared with the latter —the mesh or 6-switches fat tree— respectively, because the former has the larger number of links to distribute paths uniformly.

Finally, Table XIV and XV show the execution results of CG, LU, IS, SP, BT and MG benchmarks under the DL routing. The DL routing and the SBP provide the same bandwidth on the 16-switches networks as illustrated in Figure 5. Table XIV and XV demonstrate that the torus improves up to 4.1% execution time compared with Topology A. As shown in Table XIV, the fat tree increases 26% of CG execution time under class S compared with torus. In this implementation, process allocation to hosts is done in the order of low switch coordinates. In order to send a packet to a host (process) connected to the neighbor switch, the Myrinet Clos network and fat tree require the larger hops than the mesh and torus. Since CG benchmark generates a large number of packets to a neighboring process in this implementation, we consider that topologies influence with the execution time of CG benchmark.

E. The Impact of Channel Buffer Size, Link Bandwidth, and Acknowledgment Reception Protocol

Here, we vary the execution environment: channel buffer size, link bandwidth, and acknowledgment reception protocol. Figure 10 shows their evaluation results under Topology A. The "8Gbps" represents the condition in which each of link bandwidth is 8Gbps. The "VCT" represents that the channel buffer is fully used, and the "Stop,Go=1" represents that stop signal is issued as soon as receiving a header flit.

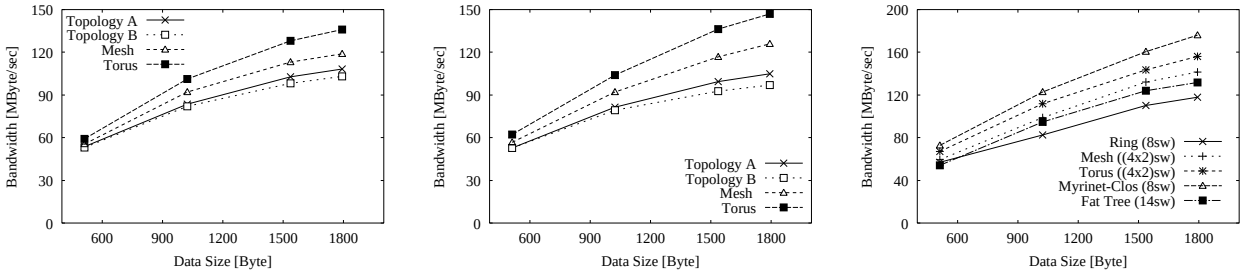
First, we focus on the buffer size. Because of RHiNET-2/SW circuit delay and link delay, we can only vary within the limits of more than 75-flits channel buffer sizes. That is, a destination switch requires $1\mu s$ to send stop signal to a source switch after receiving a header flit. Thus, until stopping the data flit at the source switch, it has already finished to send more than 600 Byte data. As shown in Figure10(a) and (b), the impact of buffer sizes is quite small within the limits of such channel sizes. This means that large channel buffers are required to support a long link length rather than improving bandwidth.

Second, we discuss on the link bandwidth against host I/O bus. In the RHiNET-2 cluster, with the 66MHz/64bit PCI bus, the peak bandwidth (4.22 Gbps) is smaller than the link bandwidth (6Gbps). It seems that a serious host I/O bottleneck may be introduced. However, as shown in Figure10(a) and (b), each 8Gbps-link condition outperforms the 6Gbps-link condition by 12% of average bandwidth. Thus, it can be said that the impact of I/O bus on average bandwidth is limited. In addition, [31] reports that application performance is not affected significantly when switching from PCI-X bus to PCI bus. In general, CPU performance and network bandwidth linearly speed up every year, while the host I/O bus is hardly changed because it is required to be standardized. This leads that I/O bandwidth tends to be small compared with network bandwidth in PC clusters as well as the RHiNET-2 cluster. However, Figure 10(a) and (b) illustrates that routing algorithm affects bandwidth even in such an environment.

Third, we focus on the acknowledgment reception protocol. We employed the simple reception protocol, which waits for an acknowledgment of the previous packet before inserting a packet, as a fundamental evaluation of bandwidth. Here, we show another protocol that can insert a packet before receiving any acknowledgment. In Figure 10(c), (d) and (e), "Ack-Wait" represents the simple protocol waiting an acknowledgment, while "No-Ack-Wait" represents the no-wait protocol. By employing the no-wait protocol, bandwidth between two hosts is constant under various packet hops. However, even when employing the no-wait protocol, path hops influence 32 % of routing bandwidth under each synthetic access pattern. In up*/down* routing, a packet consumes a larger number of switches and links, which increase the probability of packet blocks. We consider that this drastically degrades bandwidth. Thus, routing algorithm greatly influences the bandwidth under any acknowledgment reception protocol.

F. Recapitulation of Results

Figure 5 and 9 show that routing algorithms and topologies influence up to 57 % and 91% of the bandwidth respectively. On the other hand, they slightly influence the latency of barrier synchronization and execution time of NAS parallel benchmarks in many cases. Thus, in the RHiNET-2 cluster, routing algorithms and topologies don't always affect the performance of real applications. RHiNET-2 can be characterized by the function to connect personal computers within one or more floors of a building. The function includes address transform and protection mechanisms that enable zero-copy data transfer between main memory space of PCs. However, the hardware mechanism built in the network controller chip Martini treats this function with almost similar or smaller delay compared with other existing network interfaces[19]. In addition, the



(a) Bit-reversal, 32 hosts (b) Matrix trans., 32 hosts (c) Bit-reversal, 16 hosts

Fig. 9. Topology bandwidth

TABLE XIII
TOPOLOGY LATENCY OF BARRIER SYNCHRONIZATION (μSEC)

64 hosts		32 hosts		16 hosts	
Topology	Latency	Topology	Latency	Topology	Latency
Topology A	47.83	Ring(8sw)	36.51	Ring (4sw)	26.33
Topology B	46.61	Mesh ((4×2)sw)	35.00	Fat Tree(2,4,2)(14sw)	32.03
Mesh((4×4)sw)	45.61	Torus((4×2)sw)	34.02	Fat Tree(2,4,1)(6sw)	27.68
Torus((4×4)sw)	44.09	Myrinet-Clos(8sw)	33.60	Myrinet-Clos(8sw)	27.62

TABLE XIV
TOPOLOGY EXECUTION TIME OF CG, LU, AND IS BENCHMARKS (SEC)

	CG.S.16	LU.W.16	IS.S.16	IS.S.32	IS.S.64
T. A	0.20450	7.76525	0.01850	0.02091	0.11751
T. B	0.20400	7.77257	0.01838	0.02052	0.11674
Mesh ((4×4)sw)	0.20325	7.76200	0.01836	0.02052	0.11673
Torus ((4×4)sw)	0.20367	7.75225	0.01820	0.02008	0.11631
Ring(8sw)	0.16300	7.44067	0.01629	0.01736	—
Mesh ((4×2)sw)	0.16300	7.40233	0.01620	0.01735	—
Torus((4×2)sw)	0.16233	7.43900	0.01611	0.01702	—
Myrinet Clos(8sw)	0.20200	7.79100	0.01840	0.01994	—
Fat Tree(6sw)	0.20433	7.77700	0.01832	—	—

TABLE XV
TOPOLOGY EXECUTION TIME OF SP, BT, MG, AND CG BENCHMARKS UNDER 16-SWITCHES TOPOLOGIES(SEC)

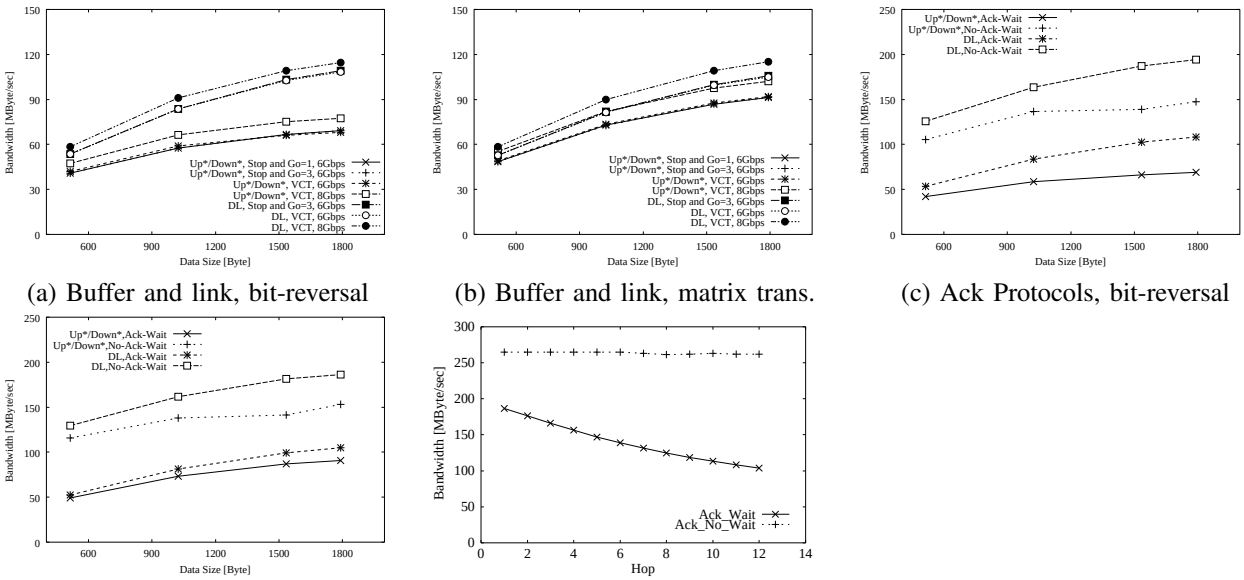
	SP.S.16	BT.S.16	MG.S.16	MG.W.16	MG.A.16	CG.A.16
T. A	0.224	0.179	0.017	0.557	—	—
T. B	0.223	0.178	0.017	0.557	—	—
Mesh	0.222	0.177	0.016	0.551	3.251	2.083
Torus	0.221	0.177	0.016	0.551	3.229	2.048

fundamental technologies in RHiNET-2 are similar to those used in other SANs as shown in Table XVI.

The RHiNET-2 cluster uses the same-specification hosts with short (2m and 5m) cables under no other application load. Its construction and usage are the same as those of common PC clusters. Thus, these results can be considered as one of general results using SANs.

G. Comparison with Results from Simulation Study

Routing, topologies and other network issues of PC clusters have been analyzed by the following techniques: 1) theoretical analysis; 2) probabilistic simulation; 3) execution driven simulation; and 4) execution on real machines. The theoretical analysis would be difficult to model large complicated network systems in detail. Although the probabilistic simulation has been frequently used for analysis of routing algorithms, traffic pattern is not usually based on real applications. Recently, sophisticated execution driven simulation environments have been developed[32], and network can be analyzed with precisely modeled network interfaces and switches under real traffic. Although it could take a long time to simulate a minutely modeled host with operating system especially for



(a) Buffer and link, bit-reversal (b) Buffer and link, matrix trans. (c) Ack Protocols, bit-reversal

(d) Ack Protocols, matrix trans. (e) Ack Protocols between two hosts

Fig. 10. Bandwidth of various buffer sizes, link speed, and ack protocols

TABLE XVI
FEATURES OF SANS

	RHiNET-2	Myrinet-2k	QsNET II	InfiniBand
Routing	Distributed	Source	Source	Distributed
Multicast	Unicast-	Unicast-	Tree- and unicast-	Tree- and unicast-
Topology	Free	Free	Fat Tree	Free
Flow control	Go/Stop	Go/Stop	Wormhole	Credit Based
Link speed	8Gbps	2Gbps	8Gbps	10Gbps(4x)

large systems, performance evaluation and analysis of PC clusters with monitoring tools have been done[33][34]. However, most of these performance evaluation studies focus on fundamental performance of network interfaces and switches. Also, there are a large number of researches on performance evaluation with real PC clusters[35][36][31], and most of them focus on the network performance itself with a fixed network configuration.

A simulation study using a probabilistic model has reported that the impact of the routing performance on SANs is larger than that shown here[13]. For example, our simulation results[13] show that the DL routing improves up to 67% of throughput compared with up*/down* routing in 16-switches irregular networks. Also, it has reported the path selection using the static analysis of paths improves up to 71% of throughput. Notice that this simulation condition—virtual cut-through, three virtual channels, and 128-flit packet length—is similar to that in the RHiNET-2 cluster. The RHiNET-2 cluster shows the following different results from the simulation results: 1) the path selection influences only 15% of bandwidth. 2) the number of virtual channels doesn't affect the performance so much. Both demonstrate that path length is relatively important in the real machines. However, we can't ignore the difference between the measure of real systems and simulations one. Throughput is usually defined as maximum accepted flits per host per clock cycle in simulations[16], whereas bandwidth B is measured in real systems including the RHiNET-2 as follows.

$$B = D/T$$

Where D is the data size and T is the time until receiving the acknowledgment. Thus, path length indirectly affects throughput in simulations, while it greatly influences the bandwidth in real machines. This indicates the impact of both path selection and path length is smaller than that in the simulation studies. Since such simulations usually simplify host functions, the packet interval is usually determined regardless of host processing. On the other hand, the packet interval of the RHiNET-2 cluster includes the processing time (e.g. the DMA transfer to memory and the generation of acknowledge packets at each host). Thus, we consider that the effect of routing algorithms in simulations is enhanced compared with that in the RHiNET-2 cluster.

V. CONCLUSIONS

In this paper, we implemented and evaluated deadlock-free routings and unicast-based multicasts under various topologies and channel buffer sizes on the RHiNET-2 cluster with 64 hosts. Execution results show that descending layers (DL) routing and structured channel pools improve up to 57% of bandwidth and 34% of barrier synchronization time compared with up*/down* routing. They also show that, by visiting hosts in numerical order, execution time of unicast-based barrier synchronization is improved up to 28% compared with that in random order. Regarding topology, the torus achieves up to 91% improvement on bandwidth compared with a simple topology including irregularity. However, channel buffer sizes don't affect the bandwidth in the RHiNET-2

cluster. In addition to fundamental evaluation, we appraise them using NAS Parallel Benchmarks, and the DL routing achieves 3.2% improvement on their execution time compared with up*/down* routing.

VI. Acknowledgments

The authors would like to thank the anonymous referees for their fruitful comments to improve this paper. The authors would also like to thank Dr. Tomohiro Kudoh at National Institute of Advanced Industrial Science and Technology (AIST), Dr. Hiroaki Nishi, Mr. Akiya Jouraku, Mr. Akira Kitamura at Keio University, and all researchers at Real World Computing Partnership (RWCP) for their contribution to the RHiNET-2 project.

References

[1] N.J.Boden and et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–35, 1995.

[2] T.Kudoh, S.Nishimura, J.Yamamoto, H.Nishi, O.Tatebe, and H.Amano. RHiNET: A network for high performance parallel computing using locally distributed computing. In *Proceedings of IWIA*, pages 69–73, November 1999.

[3] I.T.Association. Infiniband architecture. specification volumen 1,release 1.0.a. *available from the InfiniBand Trade Association, http://www.infinibandta.com*, June 2001.

[4] F. Petrini, W.C. Feng, and A. Hoisie. The Quadrics network (QsNet): high-performance clustering technology. In *Proceedings of Hot Interconnects*, pages 125–130, August 2001.

[5] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, 36(5):547–553, May 1987.

[6] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching techniques. *Computer Networks*, 3(4):267–286, 1979.

[7] S. L. Scott and G. T.Horson. The Cray T3E network: adaptive routing in a high performance 3D torus. In *Proceedings of Hot Interconnects IV*, pages 147–156, August 1996.

[8] Y.Yang, A.Funahashi, A.Jouraku, H.Nishi, H.Amano, and T.Sueyoshi. Recursive Diagonal Torus: an interconnection network for massively parallel computers. *IEEE Transaction on Parallel and Distributed Systems*, 12(7):701–715, 2001.

[9] Myricom, Inc. http://www.myri.com/.

[10] M.D.Schroeder and al et. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9:1318–1335, 1991.

[11] M.P.Merlin and J.P.Schweitzer. Deadlock Avoidance in Store-and-Forward Networks. *IEEE Transactions on Computers*, COM-28(3):345–354, 1980.

[12] T. Skeie, O. Lysne, and I. Theiss. Layered Shortest Path (LASH) Routing in Irregular System Area Networks. In *Proceedings of International Parallel and Distributed Processing Symposium*, pages 162–169, April 2002.

[13] M.Koibuchi, A.Jouraku, , K.Watanabe, and H.Amano. Descending Layers Routing: A Deadlock-Free Deterministic Routing using Virtual Channels in System Area Networks with Irregular Topologies. In *Proceedings of the International Conference on Parallel Processing*, pages 527–536, October 2003.

[14] R. Kesavan and D.K. Panda. Efficient Multicast on Irregular Switch-Based Cut-Through Networks with Up-Down Routing. *IEEE Transactions on Parallel and Distributed Systems*, 12(8):808–828, August 2001.

[15] M. Koibuchi, K. Watanabe, K. Kono, A. Jouraku, and A. Amano. Performance Evaluation of Routing Algorithms in RHiNET-2 Cluster. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 395–402, December 2003.

[16] J.Duato, S.Yalamanchili, and L.Ni. *Interconnection Networks: an engineering approach*. Morgan Kaufmann, 2002.

[17] J.Flich, M.P.Malumbres, P.Lopez, and J.Duato. Performance evaluation of networks of workstations with hardware shared memory model using execution-driven simulation. In *Proceedings of the International Conference on Parallel Processing*, pages 146–153, October 1999.

[18] T. Otsuka, K. Watanabe, J. Tsuchiya, H. Harada, J. Yamamoto, H. Nishi, T. Kudoh, and H. Amano. Performance Evaluation of a Prototype of RHiNET-2: A Network-based Distributed Parallel Computing System. In *the IASTED International Conference on Applied Imformatics*, pages 738–743, February 2003.

[19] K. Watanabe, T. Otsuka, J. Tsuchiya, H. Harada, J. Yamamoto, H. Nishi, T. Kudoh and H. Amano. Performance Evaluation of RHiNET-2/NI: A Network Interface for Distributed Parallel Computing Systems. In *Proceedings of International Symposium on Cluster Computing and the Grid*, pages 318–325, May 2003.

[20] T.Takahashi, S.Sumimoto, A.Hori, H.Harada, and Y.Ishikawa. PM2: High Performance Communication Middleware for Heterogeneous Network Environment. In *SC2000*, pages 52–53, November 2000.

[21] J.C.Sancho, A.Robles, and J.Duato. An effective methodology to improve the performance of the up*/down* routing algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 15(8):740–767, 2004.

[22] J.Wu and L.Sheng. Deadlock-Free Routing in Irregular Networks Using Prefix Routing. In *Proceedings of Parallel and Distributed Computing Systems*, pages 424–430, August 1999.

[23] J.C.Sancho and A.Robles. Improving the Up*/Down* Routing Scheme for Networks of Workstations. In *Proceedings of the European Conference on Parallel Computing*, pages 882–889, August 2000.

[24] M.Koibuchi, A.Jouraku, and H.Amano. The Impact of Path Selection Algorithm of Adaptive Routing for Implementing Deterministic Routing. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1431–1437, June 2002.

[25] 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, December 1994.

[26] S.Nishimura, T.Kudoh, H.Nishi, J.Yamamoto, K.Harasawa, N.Matsudaira, S.Akutsu, K.Tasho, and H.Amano. High-speed network switch RHiNET-2/SW and its implementation with optical interconnections. In *Hot Interconect*, pages 31–38, August 2000.

[27] Y.Ishikawa, H.Tezuka, A.Hori, S.Sumimoto, T. Takahashi, F. O'Carroll, and H. Harada. RWC PC Cluster II and SCore Cluster System Software – High Performance Linux Cluster. In *5th Annual Linux Expo*, pages 55–62, May 1999.

[28] D.Bailey, T.Harris, W.Saphir, R.Wijngaart, A.Woo, and M.Yarrow. The NAS Parallel Benchmarks 2.0. *NAS Technical Report, NAS-95-020*, December 1995.

[29] D.Bailey, T.Harris, W.Saphir, R.Wijngaart, A.Woo, and M.Yarrow. New Implementations and Results for the NAS Parallel Benchmarks 2. *PP97*, March 1997.

[30] Himeno benchmark. *http://w3cic.riken.go.jp/HPC/HimenoBMT/program2.htm*.

[31] J. Liu, B. Chandrasekaran, J.Wu, W.Jiang, S.Kini, W.Yu, D.Buntinas, P.Wyckoff, and D.K. Panda. Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics. In *SC2000*, 2003.

[32] http://www.opnet.com/products/home.html.

[33] M.Taufer and T.Stricker. A performance monitor based on virtual global time for clusters of pcs. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 64–72, December 2003.

[34] NetPIPE Team. A Network Protocol Independent Performance Evaluator. *http://www/scl.ameslab.gov/netpipe/*.

[35] Z.Lan and P.Deshikachar. Performance analysis of a large scale cosmology application on three cluaster systems. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 56–63, December 2003.

[36] J.Beecroft, D.Addison, F.Petrini, and M.MclLaren. QsNETII: An Interconnect for Supercomputing Applications. *http://www.quadrics.com/*.

PLACE PHOTO HERE

Michihiro Koibuchi received the B.E., M.E. and Ph.D degrees from Keio University, Japan, in 2000, 2002 and 2003. He is currently a visiting researcher in Keio University, and Research Fellow of the Japan Society for the Promotion of Science. His research interests include the area of interconnection networks and parallel processing.

PLACE
PHOTO
HERE

Konosuke Watanabe received the M.E. degree from Keio University in 2003. He is currently a Ph.D candidate in Keio University. His research interests include the area of interconnection networks and parallel processing.

PLACE
PHOTO
HERE

Tomohiro Otsuka received the M.E. degree from Keio University in 2003. He is currently a Ph.D candidate in Keio University. His research interests include the area of interconnection networks and communication middleware.

PLACE
PHOTO
HERE

Hideharu Amano received the Ph.D degree from Keio University 1986. He is currently a Professor in the Department of Information and Computer Science, Keio University, His research interests include the area of parallel processing and reconfigurable systems.