Journal of Information Processing Vol.23 No.5 730–743 (Sep. 2015)

[DOI: 10.2197/ipsjjip.23.730]

Regular Paper

Relaxing Heavy Congestion by State Propagation

Takashi Yokota[1,a)] Kanemitsu Ootsu[1] Takeshi Ohkawa[1]

Received: January 5, 2015, Accepted: April 25, 2015

Abstract: Interconnection network is still one of the most important key issues for building massively parallel computing systems. As a general characteristic, communication performance does not always increase as the size of network grows. Furthermore, large-scale networks suffer catastrophic performance degradation since speed of spread of congestion surpasses by far suppression speed. This paper focuses discussions on relaxation of congestion so that we can expect performance enhancement even in congested situations. This paper discusses dynamical behaviors, specifically in propagation of congestion states. When a receiver buffer becomes fully occupied, it inhibits the corresponding buffer from sending any packet to avoid loss of packet. Thus, a congested area propagates against packets' traveling direction. Based on the observation results, as the second issue of this paper, we propose a new throttling method, called State-Propagation Throttling (SPTh). The method can boost communication performance in many of typical traffic patterns in both steady and unsteady communication situations. Furthermore, this paper discusses extending the throttling method to prevent congestion from a proactive point of view. In steady communications, the proposed method improves throughput two times and latency four times. The method also improves performance of collective communication at most 1.8 times.

Keywords: parallel architectures, interconnection networks, quasi-global information, congestion control, throttling

1. Introduction

Interconnection networks in massively parallel computing systems still require performance enhancement. This paper introduces a novel viewpoint of *propagation* of congestion and proposes an effective throttling method.

Deterministic routing algorithms strictly *determine* any routes of packets. Thus, they offer simplified organizations and mechanisms in router, and they have unique advantage in keeping in-order arrival of packets. However, from the opposite viewpoint, no packet can escape from congested situations since the routing policy disallows any of alternative routes.

The alternative solution to deterministic routing is adaptive routing that allows alternative routes not to fall into congested state. Although the adaptive routing algorithms generally offer higher communication performance than deterministic ones, they also suffer performance degradation in congested situations.

From an application point of view, we can hardly control interconnection networks in their maximal performance situations. Some parallel applications involve sparse communications, so the interconnection networks are not congested. On the other hand, other (and many of) applications require frequent communications that arise dense packet flows to heavy congestion. What we should discuss for further improvement of interconnection networks is effective control method in congested situations. We focus our discussions on this problem.

Congestion appears in a group of neighboring routers whose filled buffers block packet transfers. In most cases, temporal congestion that disappears in a short period of time is not a problem. However, we have empirical knowledge that a congested area grows very rapidly, and once the area becomes large, it persists. Other empirical knowledge suggests that we can exploit the maximum communication performance at the edge of congestion, where the network is at the border state to congestion. This indicates that we should control the network state at the edge of congestion, preventing the network from falling into a congested situation. In this paper, our approach is to prevent congested areas from spreading so that we can exploit the maximal performance of network.

The first key idea in this paper is propagation of congested situations. In a macroscopic view, spread of congestion is observable as propagation of busy state of fully filled buffers in routers. The filled buffers block incoming packets from their preceding routers, and the preceding buffers also become full afterward.

Early research on parallel computing systems and interconnection networks states congestion problems. Tree saturation [1] is discussed when traffic load is unbalanced. Another representation is head-of-line blocking [2] that explains propagation of congestion. The old knowledge only suggests that congestion appears as local interactions of neighboring routers. However, few studies discuss actual congestion behaviors from a propagation point of view. This paper tries visual recognition of propagation of congestion. For that purpose, this paper introduces the space-time chart after overviewing the accomplishments of self-driven particle and general network researches in physics.

The second key idea is to apply a throttling method of packet injection. When a router detects a new congestion situation firing, it suppresses new packet injection until the congested situation is extinguished. Understanding the propagating behaviors of con-

[1] Department of Information Systems Science, Graduate School of Engineering, Utsunomiya University, Utsunomiya, Tochigi 321–8585, Japan
[a)] yokota@is.utsunomiya-u.ac.jp

gested situations drives us to an effective throttling method.

In this paper, we will discuss efficient control methods under the following assumptions: highly regular network whose topology is two-dimensional torus, deterministic routing (dimension-order routing, DOR), and virtual cut-through flow control. This paper does not assume adaptive routing, as the opposite idea of deterministic routing, since the routing principle can avoid congested area thus it might veil the net effect of our method.

The rest of this paper is organized as follows. Section 2 overviews existing researches and discusses the novel points of this paper. Section 3 shows our observation results of spatio-temporal behaviors of routers. The results conduct us to the new throttling method that Section 4 explains. Section 5 shows the effectiveness of our proposed method via simulation evaluation. Finally, Section 6 concludes this paper.

2. Related Work

This paper addresses improving communication performance in heavily congested situations. This section overviews past solutions for heavily congested situations.

2.1 Adaptive Routing

Adaptive routing principles are actively discussed so that they can improve communication performance of interconnection networks [2], [3]. Their typical feature is to select an alternative route to avoid congested situations. Many of them are successful in tolerating heavier traffic loads than those of deterministic routing policies.

Some of adaptive routing methods employ (quash-)global information for effectively control the alternative routes. One of the major ideas of this paper is propagation of congestion, and the idea implies (quasi-)global information. Thus, we state adaptive routing methods with (quash-)global information.

So et al. [4] collect information of routers within a short Manhattan distance for selecting effective routes of packets. Although the basic principle works effectively, implementation costs disallows wide range of information since computational complexity and hardware costs in collecting and evaluating the global information are considerably large.

Cross-Line [5] introduces an effective collection method of buffer status in the line of routers in packets' traveling direction. Cross-Line simply discusses the current situations in the possible directions of each packet. It does not stand on the propagation idea of congested area. However, the collection method that is employed in Cross-Line is sufficiently usable for other purposes than adaptive routing. This paper introduces the mechanism called VCinfo as described in Section 4.1.

Although adaptive methods can benefit in performance improvement, this paper deals only with deterministic routing to distinguish effectiveness of our proposed method, which is orthogonal to routing algorithms.

2.2 Throttling

As long as we disallow any packet drop in interconnection networks, routing algorithms hardly resolve heavy congestion problems, even if adaptive algorithms are applicable. Adaptive routing algorithms can tolerate higher traffic loads than those of deterministic ones. However, both of adaptive and deterministic algorithms cause congested situations in overloaded conditions anyway.

One of practical solutions, other than packet-drop, is admission control. Literature shows some representative researches of throttling [6], [7], [8], [9], [10]. All of the throttling methods proposed in these studies introduce additional information to start and to stop throttling.

Baydal et al. have proposed a set of throttling methods, called U-Channels, ALO, and INC [10]. All of these methods employ local information within a router. U-Channel and ALO use the number of unblocked buffers. INC measures packet flows in a certain period of time to predict congested situations. Throttling in these methods is based on *prediction* results of current and future congestion in the network, while this paper tries to use *actual* congestion information. As the authors state in Ref. [10], confidence level of the prediction results is a problem. It is clear that the presented method in this paper has high levels confidence in detecting congestion. Furthermore, Baydal et al.'s methods require appropriate tuning of control parameter(s), while this paper does not assume tuning. They evaluate the methods in the similar way with this paper, but, unsteady (collective) communication is not discussed [10]. These methods underlie adaptive routing methods, while this paper assumes deterministic routing. These methods are difficult to apply to deterministic routing environments. The differences come from the fundamental concept.

DRIL [6] and CLIC [7] methods also employ local information within each router. Thottethodi et al. [8], [9] use population of packets under being transferred to their destinations. Their major idea is to determine the optimal number of in-flight packets for the practical traffic situations. They introduced *meta-packet* to collect and distribute the number of in-flight packets which is global information. The meta-packet mechanism performs reduction and broadcast functions. They adopt the acquired information to throttle injection according to a certain threshold that is determined by their *hill-climbing* method. The basic idea of the meta-packet is close to that of VCinfo that is introduced in this paper. VCinfo collects buffer status in a packets' traveling direction. The global operation of the meta-packet requires higher cost than that of VCinfo.

Yokota et al. have introduced a new idea of entropy that quantitatively represents the degree of mobility [11], [12]. They have proposed a throttling method, called Entropy Throttling. Since the method requires global information to calculate entropy values, it constructs an efficient reduction network that collects the total sums of the number of buffers that contain at least one packet and the number of blocked buffers. The reduction network overlays virtual control channels that share physical communication links with ordinary channels, and the network continuously collects the information. Entropy Throttling uses average status of the network, and it does not reflect local situations.

With respect to the admission control, packet pacing [13] is a close idea to throttling. Obviously, the method does not essentially employ feedback properties.

This paper is unique in discussing throttling methods from a

novel viewpoint of congestion propagation. Its fundamental idea is rather simple. Each router suppresses a new packet injection when the injecting packet is heading to a (likely) congested area. For effective implementation of the throttling method, we should discuss how to acquire global information in the following section.

3. Spatio-Temporal Behaviors of Routers

As Section 1 stated, the first key idea of this paper is to acquire the precise behavior of congestion. For that purpose, we firstly introduce outcomes of physics research in self-driven particles and general networks. Then, we discuss possible behaviors of congested areas by introducing a simplified one-dimensional model. We clarify that our initial discussion is appropriate by observing states of routers via our simulator.

3.1 Outcomes of Physics Research

3.1.1 Knowledge from Self-Driven Particle Research

Physics research is offering attractive knowledge on general characteristics of networks and particles. Typical physics discussions show their own practical, but simplified models for representing their objective systems. Some of them offer mathematical equations for theoretical discussions and others extend the models to simulate. The idea of cellular automaton is helpful to the simplified but powerful tools, which is frequently used in many researches. Furthermore, rapid improvement of computer performance accelerates simulation studies.

One of the noticeable harvests is the idea of phase transition. In a system that has non-linear characteristics, features of the system drastically changes at certain conditions without gradual states. In interconnection networks, packet buffering and flow control offer non-linear characteristics. Thus, the networks show phase transition between uncongested and congested situations [14], [15], [16], [17], [18].

Other harvests come from car-traffic research. They also conduct to various modeling, e.g., cellular automata. They, furthermore, offer the fundamental diagram and space-time chart [19], [20]. The former models traffic situations of a road. It illustrates the relationship between density (the number of running cars per kilometer) and flux (the number of passing cars per hour). When the density is small, the road is not congested, and the traffic flux is proportional to the density. When the density exceeds a certain threshold, the traffic is jammed and the flux rapidly decreases.

Furthermore, the traffic research introduces the idea of representing spatio-temporal behaviors in a two-dimensional graph, i.e., a space-time chart. The visual representation helps us understand behaviors clearly.

3.1.2 Percolation

When general network researches deal with local interactions between neighboring items in a network, the idea of *percolation* is used frequently [21], [22]. For example, in social activities, a healthy person might get affected by virus at a certain possibility when he interacts with diseased persons. At the same time, the diseased persons might get healed up. Prior researches in percolation state that there exists a certain threshold whether the infection will disappear or lead to *pandemic*.

Qualitative behaviors of percolation are close to those of congestion in information network [23]. A congested router might infect its adjacent routers into congested situations. Intermittent congestion might be resolved soon when the packet density is under a certain threshold. However, if the packet density exceeds the threshold, chaining infection causes heavy congestion. This situation is just like pandemic of a new type of influenza.

3.2 Desktop Experiment

We discuss dynamical behaviors of congested area by means of a one-dimensional model that intends a uni-directional line of routers out of a two-dimensional torus network. **Figure 1** shows the simplified model. This figure focuses on a rightward flow of packets. At each router, the rightward flow of r_i is transferred from its left-hand side router, new packets of s_i are merged into the rightward flow, and k_i of the flow r_i (i.e., $k_i r_i$) goes outside the rightward flow[*1].

As a simple discussion, we can consider a dependency graph of packet buffers. As Fig. 1 also offers the dependency graph, i-th buffer is dependent on $(i-1)$-th buffer. The simple discussion draws a result that congested state is propagated backward in the dependency graph. However, buffer behaviors are not simple, and they sometimes show nonlinear phenomena. Sudden drastic performance drop is a typical example. Thus, before discussing a new method for eliminating congested areas, we should discuss and clarify actual dynamic behaviors of congested areas.

Figure 2 assumes some typical situations in which some of buffers are filled and other ones are not. In this figure, blue-colored packets move to horizontal direction (rightward), and red-colored packets are to escape from the rightward flow at any of the routers. In Fig. 2 (a), only one buffer (B_1) is full. This situation represents the smallest congestion. In this case, the buffer can release its packets to the next router at the moment. If the incoming flow is sufficiently small, the partial congestion quickly

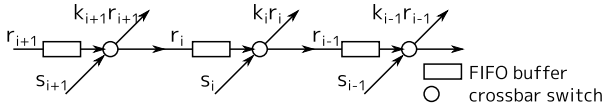
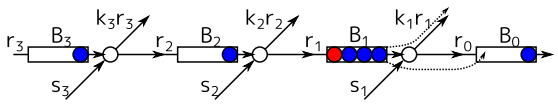
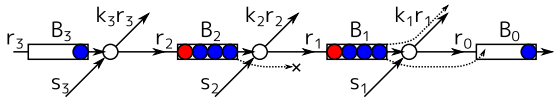


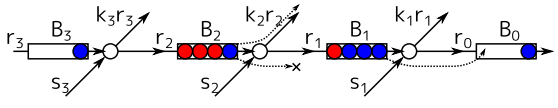
Fig. 1 Desktop experiment model.



(a) Only one buffer is full.



(b) Two consecutive buffers are full.



(c) Two consecutive buffers are full (alternative case).

Fig. 2 Temporal behaviors in desktop experiment model.

[*1] r_i and s_i represent flow rates, say [flits/cycle], and k_i represents the ratio.

disappears. However, when consecutive buffers are full, things are not so simplified.

Figure 2 (b) shows an example in which two consecutive buffers are full. The head packet in B_2 is blocked by B_1. The succeeding packets are also blocked since they cannot bypass the head packet. Thus, the congested state propagates against the packets' traveling direction.

We show an alternative case in Fig. 2 (c), where the succeeding buffer (B_2) contains escape packets. In this case, once the head packet of B_2, which is currently blocked by B_1, is transferred, congested state of B_2 will be extinguished soon. This affects the state of B_1 directly. While B_2 releases the escape packets, the input flow rate to B_1 becomes zero. Thus, the congested situation of the buffer is also extinguished. We might observe the alternative example as forward movement of congested area.

Although the results of the desktop experiment show two conflicting behaviors, their possibilities are different. Both Fig. 2 (b) and (c) assumes that consecutive buffers are full. This assumption implies that rightward traffic is sufficiently heavy so that the buffers are fully filled. Thus, frequency of Fig. 2 (b) is much larger than that of (c). This analysis is proven in the following subsection.

3.3 Observation of Behaviors

Once a network is heavily congested, communication performance is drastically degraded, since packets block each other. However, unless the network is under a deadlock situation, the performance does not fall to zero. Even in congested situations, we can guess that packets occasionally flow between adjacent routers. However, we have little knowledge on actual behaviors of routers. None of research literature on interconnection networks reports dynamical behaviors of routers in congested situations.

Physics researches often use space-time charts for ease of understanding time-series behaviors of self-driven particles. In a space-time chart, status of one dimensional space at a certain time is represented in a horizontal row, and the following horizontal rows represent the consecutive status of the corresponding space on a step by step basis. Even though a space-time chart has a limitation that it can only represent one-dimensional space, visualization offers large benefit for understanding complicated phenomena. For example, some early researches such as Refs. [19], [20] show moving behaviors of car congestion (traffic jam). Some of car drivers unconsciously slow down before entering tunnels. Thus, car density sometimes arises to the onset of congestion. The references state that a congested area does not stay still and it moves against the traveling direction of cars.

Inspired by the visualization tool, we have extended our ICN simulator to monitor time-series behaviors of routers. This extension generates a space-time chart of ready/busy status in a line of input buffers at every arbitrary interval of simulation cycles. An interconnection network with two-dimensional torus topology allows four-direction traffics. Thus, we record four space-time charts that correspond to the four traveling directions, i.e., N(orth), E(ast), S(outh), and W(est), respectively.

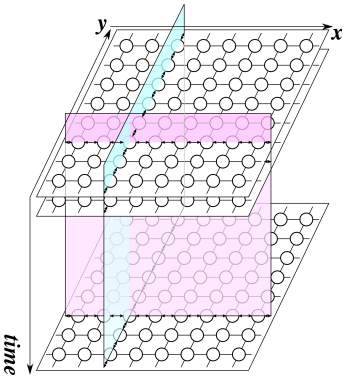


Fig. 3 Sampling space-time chart in 2D network.

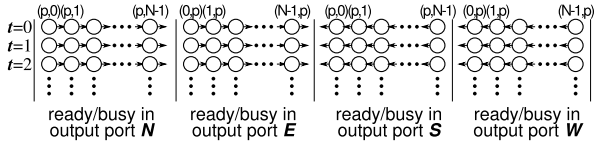


Fig. 4 Representing space-time charts of four directions.

We have tentatively use one-third positions[*2] in the horizontal and vertical axes for recording the time-series behaviors. In an $N \times N$ network, E- and W-directions in a line of routers at $(\lfloor N/3 \rfloor, j)$, $j = 0, \ldots, N-1$ are recorded to capture E- and W-direction behaviors, respectively. Similarly, N- and S-directions are recorded at routers $(i, \lfloor N/3 \rfloor)$, $i = 0, \ldots, N-1$. **Figure 3** overviews the sampling scheme of the space-time charts in four directions, and **Fig. 4** depicts the assembled space-time chart that represents ready/busy status of buffers in N-, E-, S-, and W-directions in a row.

By filling cells in Fig. 4 that correspond to busy buffers, the space-time chart clearly illustrates dynamical behaviors of congested area. **Figure 5** shows artificial examples where congested area grows against packets' traveling direction.

Figure 6 shows practical examples of space-time chars that are obtained by our simulator. This figure is in a bitmap fashion as described in Fig. 5. In this figure, two-dimensional torus network with size 32×32 is used, traffic pattern is bit-complement, and packets are generated at a given rate. Different colors show different virtual channels; red, green, and blue dots represent busy status in the corresponding 0th, 1st, and 2nd virtual channels, respectively[*3]. Tics are marked at every five simulation cycles,

[*2] Our intention of the one-third position is to avoid special-case sampling. This paper assumes the deterministic routing method (DOR) and every path is statically determined (except random traffic). When we draw all of the paths in a sheet of paper, we can recognize the geometrical distribution of the paths [24]. Each specific traffic pattern, such as *transpose*, shows unbalanced traffic loads on the inter-router links. If the sampling position is inproperly selected, the resulting space-time chart represents inappropriate communication situations. Based on the survey of the geometrical view of communication paths, this paper tentatively uses the one-third position. Other positions shows slightly different drawings of space-time charts in positions of frequently-congested areas and usage of virtual channels, although, quaitative tendencies are similar to the one-third position. Only excenption is 1/2 position, where some of necessary congestion situations are not drawn in some specific directions of sub-charts.

[*3] As described in Section 5.1, we use twice datelines at $x = 0, N/2$ and $y = 0, N/2$ in controlling virtual channel. Thus, when a channel-0 packet (in the red color) goes across the halfway dateline, its virtual channel is changed to 1 (in the green color).

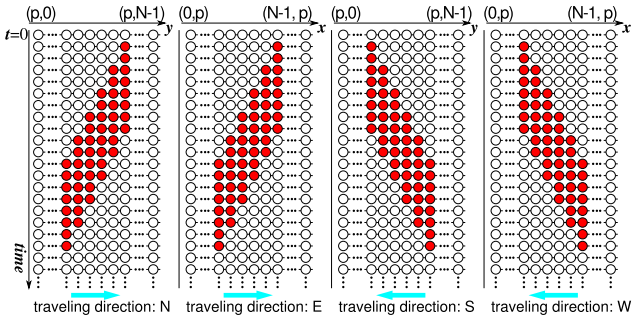


Fig. 5 Space-time chart example that represents congestion propagation.

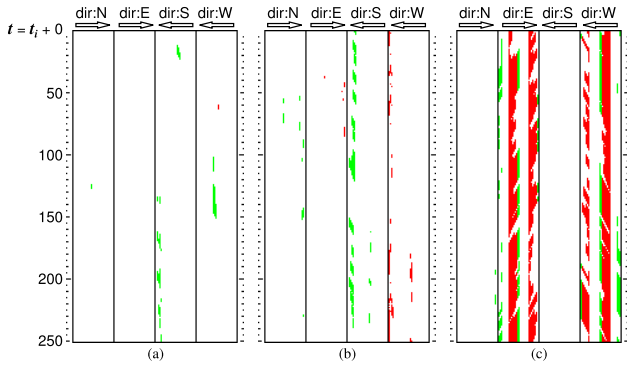


Fig. 6 Space-time chart examples (32 × 32 two-dimensional torus, bit-complement traffic): (a) uncongested, (b) critical, and (c) saturated situations. Different colors represent different virtual channels; red: channel-0, green: channel-1, and blue:channel-2. t_i indicates the starting time of the space-time chart. In (a), (b), and (c), t_i = 998,500, 1,109,500, 1,220,400 [cycles], respectively.

which are attached at the both sides of a chart. Difference in Fig. 6 (a) to (c) is packet generation rate (load ratio). Figure 6 (b) shows critical load, (a) and (c) show 90% and 110% of the critical load, respectively[*4].

Even in uncongested situations, packet buffers become fully occupied occasionally. Figure 6 (a) shows some intermittent busy status. As the figure shows, in the uncongested situations, a partial congestion, which consists of busy status of routers, arises sparsely and disappears after a short duration time. The intermittent congestion scarcely spreads or moves to other routers. Such partial and intermittent congestion affects the latency issue, not throughput, in the performance point of view. Thus, this paper claims that intermittent partial congestion does not cause fatal problems in performance.

One of our expectations was that some distinct seeds exist at the onset of congestion. However, we failed to find out the expected seeds in many cases. What we should focus on is the edge of congestion, i.e., almost congested situation but not fully congested one. At the critical level of traffic load, congested situations arise one after another (Fig. 6 (b)). Some of partially congested areas sometimes spread to neighboring routers, and the congested areas move against the packet-flow direction. However, most of congested areas do not sustain.

In overloaded situations, we can see successive congestion as shown in Fig. 6 (c). Figure 6 (a)–(c) illustrate that congested areas, i.e., consecutive busy status of routers, move against the packet-flow direction. Thus, to diminish performance degradation, we discuss efficient control methods by using busy-status information.

[*4] The critical load ratio is 0.11095 [flits/cycle] in this condition. Initial times t_i correspond to traffic load values of 0.0985, 0.11095, and 0.12204 [flits/cycle], respectively.

4. State-Propagation Throttling

Results in Section 3 reveal that congested areas propagate against packets' traveling direction. In other words, each router can predict future congestion if any of other routers in the packets' traveling direction are in a congested situation, since the congestion state will be propagated to the router.

This observation leads us to the new idea. Here, we assume that a congested area at a certain level just arises. We can easily guess that the congested area will be enlarged and will not be extinguished, if a sufficient number of packets flow into the area. Thus, we should limit the packets' flow so that the emerging congestion is extinguished.

Our basic idea of packet limitation is simple. Complete suppression of packet transmission into the congested direction is not a smart solution. If the traffic is overly controlled, communication performance is degraded at a low level. Thus, what we should discuss next is what and how each router controls the packet flow to the detected congestion.

We categorize packets in each router into three classes of flows. Assume that a packet is stored in a channel buffer and is destined to an appropriate output port to the next neighboring router. The packet follows one the follwing three cases; (a) the packet is just injected, (b) the packet was transferred via the opposite-side input port and it is destined to the same dimension (*straight* packet), and (c) the packet was input from different dimension from the currently destined output port (*turn* packet).

Then, we discuss which packet class is appropriate for suppression to resolve congested situations. All of the three classes of packet flows (a) to (c) consume packet buffers and they are destined to the congested area. Two of them (b) and (c) are different from (a) from a viewpoint of chaining congestion. For example, when a channel buffer is fully occupied by (b)-class packets, any packet-transfer for the said buffer is blocked in the preceding router. This behavior is not desirable since further congestion may be caused. Blocking of turn packets may also cause similar negative effects. On the other hand, suppression of (a)-class packets does not affect other neighboring routers. Thus, we reached our result of supression of (a) flows.

This paper proposes that each router suppresses injection of new packets that will run into a congested area, even if the congested area is distant (non-adjacent) from the router. We call the new idea of throttling method as *State-Propagation Throttling*, SPTh for short.

This paper should discuss how each router can detect congested areas that locate distantly. We should discuss how to detect the onset of congestion in the packets' traveling direction that may be sufficiently distant from the current router to control the congestion. We have a powerful tool named *VCinfo* that is one of the center mechanisms in our *Cross-Line* routing algorithm proposed in Ref. [5].

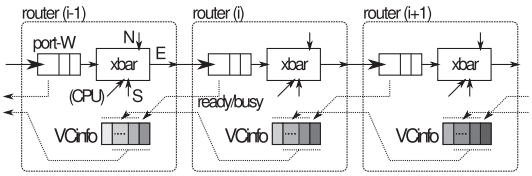


Fig. 7 VCinfo mechanism.

4.1 VCinfo Mechanism

Cross-Line is proposed as an adaptive routing algorithm. It aims at quasi-globally optimal routing to enhance communication performance in interconnection networks. The algorithm makes use of ready/busy information in each row of packets' traveling direction so that it appropriately selects the transfer direction of packets.

The distinguishing feature of the algorithm is *VCinfo*, which stands for virtual channel information. Every router has a VCinfo register for each of virtual channel in each output port. The VCinfo register consists of bit-mapped binary representation of states of buffers in the packets' traveling direction. **Figure 7** shows the basic mechanism of VCinfo. The 0-th bit (i.e., LSB) of the register corresponds to the ready(0)/busy(1) status of the adjacent buffer. The values of the VCinfo register are transferred to the neighboring router in the opposite direction, after the contents of the register are shifted left one bit. Thus, as Fig. 7 shows, n-th bit of the VCinfo register corresponds to the $n + 1$-th router in that direction.

Contents of the VCinfo registers are transferred against the packets' traveling direction. This paper assumes dedicated links for the VCinfo propagation, while the original Cross-Line propagates in a cycle-stealing fashion. This paper assumes ideal conditions for discussing possible impacts of the proposed method. Thus, as soon as a buffer state is changed, the corresponding VCinfo contents are transferred to the next neighboring router at the next simulation cycle. Furthermore, after the VCinfo contents are transferred, the corresponding VCinfo at the neighboring router is updated at the next simulation cycle. Thus, whenever any content in VCinfo register is changed, the updated information instantly propagates along the line of routers against the packets' traveling direction at every cycle.

4.2 Throttling by Means of VCinfo

The VCinfo register summarizes binary situations in the line of buffers in the traveling direction. The contents of the register are updated within short delays. Thus, each router can easily know whether any congestion arises in the traveling direction or not. Furthermore, our observation results from Section 3.3 forecast that even a distant (non-adjacent) congested area will be enlarged to the router if packets are supplied for the congested area.

Then, we reach a new throttling principle. When a router finds one or more busy situations in a VCinfo register, the router suppresses injection of new packets that is destined to the said direction. The router keeps throttling as long as the corresponding VCinfo register contains busy situations. This paper simply defines the SPTh methods as any of the VCinfo bits shows congested situation. Note that throttling of packet injection is dependent on the traveling direction of the new (injecting) packet. For example, when only the VCinfo register in E-direction has busy-bits and other VCinfo registers has no busy bits, the router only blocks specific packets that will traverse in E-direction while it accepts other packets in other directions.

4.3 Introducing Proactive Mechanism

Here, we discuss enhancement of the SPTh method, assuming that each buffer in a router has sufficient capacity and that each router follows virtual cut-through flow control [25]. Until now, we assume that VCinfo registers contain ready/busy information. 'Busy' state means that the corresponding buffer is full and the buffer cannot receive any other flits of packets. On the other hand, 'ready' state shows that the corresponding buffer can receive at least one flit of a packet, and we cannot know whether the buffer is almost full or it is empty. Thus, the 'ready' state cannot represent how the capacity of the buffer is occupied.

We reached further a new idea of occupation level. The essential purpose of the VCinfo is to represent buffer status of the routers in the forwarding row. The VCinfo mechanism does not necessarily limit only to ready/busy information constitutionally.

Suppose that a router has a buffer whose capacity is half occupied by packets. We can never expect that the half-occupied situation persists. When the traffic is sparse, the occupation will soon be dissolved since the buffer releases packets rapidly. In the opposite case, i.e., overloaded traffic, the buffer should hold incoming packets with existing packets blocked to move. Furthermore, we should recall that, once a congested area emerges, it sustains for a long time. This means that we can regard a *nearly-full* buffer as a foretaste of congestion from a preventive point of view. We have an important option of occupation level of buffers.

We extended the VCinfo registers to represent whether the occupancy of corresponding buffer exceeds a certain level or not. We call the level as occupation level. We can regard the binary state of ready/busy in the VCinfo register in the previous section (Section 4.2) as an extreme condition of the occupation level. A busy state is regarded as the buffer is fully occupied.

By using the occupation level, the VCinfo register represents foretaste of future congestion even when no packet is blocked transferring in the current situation. We can expect proactive effects in controlling congestion if packet injection is throttled by means of the foretaste in the VCinfo register.

5. Evaluation

5.1 Evaluation Environment

We have modified our interconnection network simulator to match the proposed SPTh throttling method. The simulator models a simple non-pipeline router and offers cycle-accurate simulations where all of the components operate synchronously under a global simulation clock. A packet consists of one or more flits and the simulator simulates packet transfer on the flit-by-flit basis. When a flit is not interferred by other flits, the flit is transferred to the next neighboring router at every cycle. Furthermore, alike the message packets, contents of VCinfo registers are also transferred at every cycle. We use 32×32 two-dimensional torus network with three virtual channels. Packet length is eight flits. Virtual cut-through is employed as flow control, and capacity of

Table 1 Traffic patterns used.

abbreviation	description
trns	transpose. $(X, Y) \longrightarrow (Y, X)$
shfl	perfect shuffle. $w_{2n-1}w_{2n-2}\cdots w_1w_0 \longrightarrow w_{2n-2}\cdots w_1w_0w_{2n-1}$
bcmp	bit-complement. $w_{2n-1}w_{2n-2}\cdots w_1w_0 \longrightarrow \overline{w_{2n-1}}\,\overline{w_{2n-2}}\cdots\overline{w_1}\,\overline{w_0}$
brev	bit-reverse. $w_{2n-1}w_{2n-2}\cdots w_1w_0 \longrightarrow w_0w_1\cdots w_{2n-2}w_{2n-1}$
brot	bit-rotation. $w_{2n-1}w_{2n-2}\cdots w_1w_0 \longrightarrow w_0w_{2n-1}w_{2n-2}\cdots w_1$
torn	tornado. $W \longrightarrow \mathrm{mod}(W + N/2, N^2)$
rand	random. Destination node is randomly selected.
rpar	random pair.

each buffer is sixteen flits. The evaluation uses dimension order routing (DOR) algorithm, where a packet goes along x-axis then it follows y-axis. Deadlock prevention scheme is based on *dateline* [2] where a packet changes its virtual channel to the next one in order. We use twice datelines at $x = 0, N/2$ and $y = 0, N/2$ in x and y axis, respectively, in an $N \times N$ network.

We use two major principles of communication; steady and unsteady communications. The former one controls each node to stochastically inject packets that destine the predefined node by traffic pattern. The latter one is *collective* communication in which each router starts sending certain number of packets. We use eight traffic patterns as shown in **Table 1**. In this table, assuming $N \times N$ two-dimensional torus network ($N = 2^n$), address of a node is represented as (X, Y) where $0 \le X, Y \le N - 1$, and X and Y are represented as $X = x_{n-1}x_{n-2}\cdots x_1x_0$, and $Y = y_{n-1}y_{n-2}\cdots y_1y_0$, respectively. Furthermore, the table uses unified $2n$-bit address $W = w_{2n-1}w_{2n-2}\cdots w_1w_0 = y_{n-1}y_{n-2}\cdots y_1y_0x_{n-1}x_{n-2}\cdots x_1x_0$.

5.2 Steady Communication Performance

To discuss steady communication performance, we measure throughput and average latency at a certain traffic load. This evaluation intends to compare the proposed throttling methods from the following two viewpoints. One is critical traffic load to saturate the throughput, which represents how the network can tolerate congested situations. The other one is behavior in overloaded situations where traffic load is beyond the critical level.

We use *ramp load method* [26] for precise discussions. In this evaluation method, traffic load ratio (order of [flits/cycle]) starts with zero, and it gradually increases to 0.275 during the total of 2,750,000 simulation cycles. This means that traffic load ratio is represented as a function of time (simulation cycle). In this paper's case, $g(t) = 0.1 \times 10^{-6} \cdot t$ [flits/cycle]. Throughput is measured as the number of received packets in every 100-cycle time window. Average latency is also measured as the average of ages of received packets in the 100-cycle window.

When traffic load is low, throughput is proportional to the load ratio. Throughput suddenly saturates when the load ratio exceeds a certain threshold. Critical load ratio is measured by detecting 10-percent decrements of gradient of the throughput curve [26]. To measure the critical load ratio precisely, differentiation of the throughput curve with respect to the traffic load is required. Since the ramp load method continuously changes the traffic load, it well matches to apply the differentiation. Other researches select some sparse values of traffic load as a simulation parameter, and they are hard to determine critical load ratio precisely with discrete parameters.

Figure 8 shows the results. In the figure, left vertical axis shows throughput, right axis shows average latency, and x-axis shows traffic rate in common. In each figure, red curves illustrate the conventional (no-th, non-throttling) algorithm, green (th(0)) and blue (th(0)) curves show the results of the proposed method. th(0) and th(8) indicate occupancy level that is introduced in Section 4.3. th(0) illustrates fully occupied buffers whose free space is zero. th(8) shows half-occupied buffers whose free space is less than eight flits out of sixteen flit capacity.

In ordinary performance evaluations, performance values are measured in a certain period of time after simulation situation becomes stable [2]. In this evaluation method, simulation parameters are fixed while the performance values are measured. This evaluation method requires repetitive simulation processes that slightly differ in a simulation parameter, i.e., traffic load, although, the measured performance values are sufficiently stable.

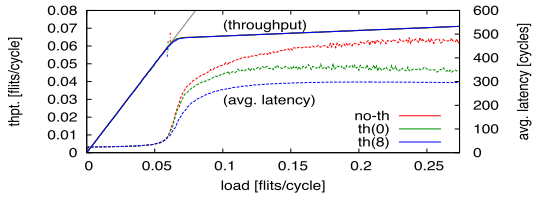
The ramp load method that is used in this paper continuously changes a simulation parameter (i.e., traffic load ratio). The method basically uses raw measured data that are not stabilized and large variability. The variability issue prevents appropriate discussions in comparing the new method. We use moving average to stabilize the measured data. Each curve in Figs. 8 and 9 is smoothed by moving average with 200-points. Performance values are measured in every 100-cycle window time in this paper. Therefore, 200 points of measured data correspond to 20,000 clock cycles. Since the traffic load varies linearly from zero to 0.275 [flits/cycle] during 2,750,000 cycles, accuracy in x-axis remains sufficiently low; only 0.09 percent in x-scale.

Figure 9 shows ratio representations of the two performance indices, i.e., throughput and average latency, for ease of comparison purpose. Performance values of the non-throttling algorithm are used as the baseline. Note that the average latency index (order of time) shows inverse of performance (order of 1/time) and that the throughput index (order of 1/time) directly shows performance. Thus, each curve in Fig. 9 directly shows performance enhancement in both throughput and average latency issues.

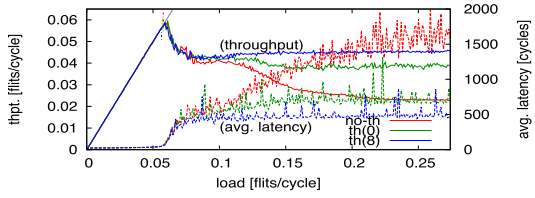
5.3 Unsteady Communication Performance

Steady communication evaluation assumes statistically stable conditions in communication situations. This evaluation method is useful in discussing the tolerance of the network method to a particular traffic pattern and load. However, in practical situations, i.e., parallel programs, many of communication situations employ collective methods where all computing nodes sends packets in unison and synchronize until all packets are received. To discuss net performance of network methods, we should evaluate the methods in the unsteady situations, i.e., collective communication. In this evaluation, every node starts generating a certain number of packets that destine a certain node defined by a given traffic pattern.

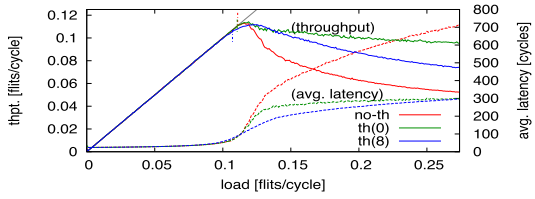
Our simulator is modified in order to detect completion of collective communication. Until the completion of a collective com-



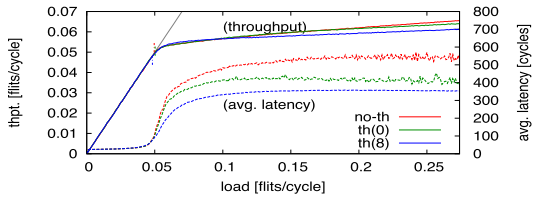
(a) transpose (trns)



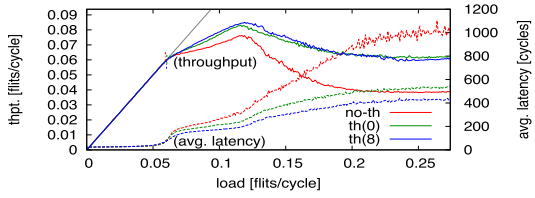
(b) perfect shuffle (shfl)



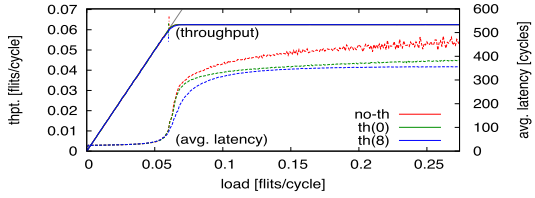
(c) bit-complement (bcmp)



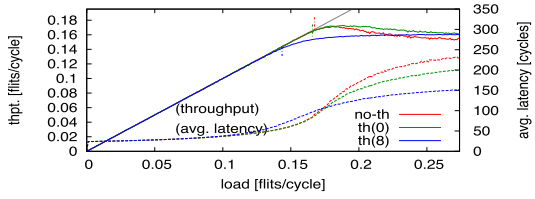
(d) bit-reverse (brev)



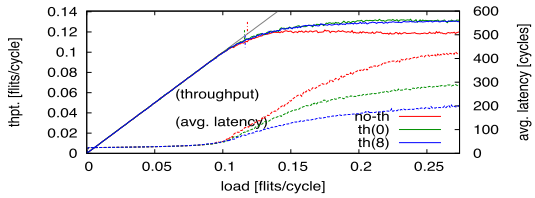
(e) bit-rotation (brot)



(f) tornado (torn)

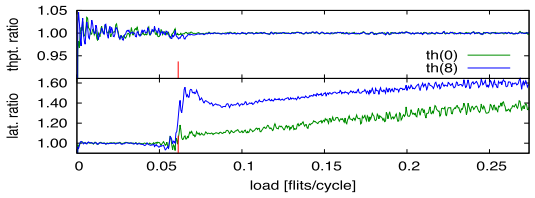


(g) random (rand)

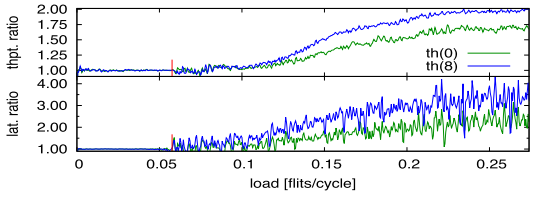


(h) random pair (rpar)

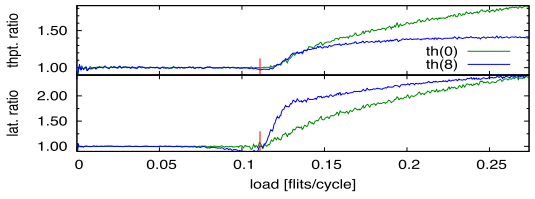
Fig. 8 Performance curves in steady communication evaluation.



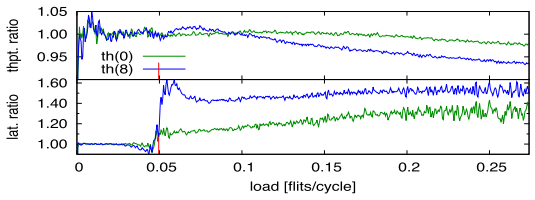
(a) transpose (trns)



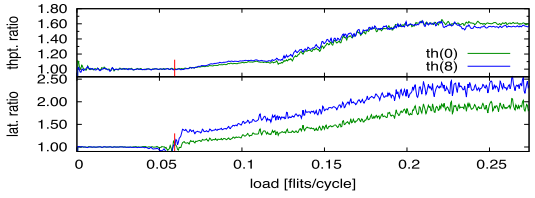
(b) perfect shuffle (shfl)



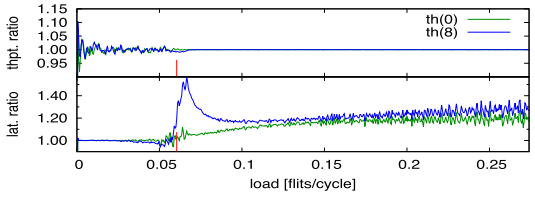
(c) bit-complement (bcmp)



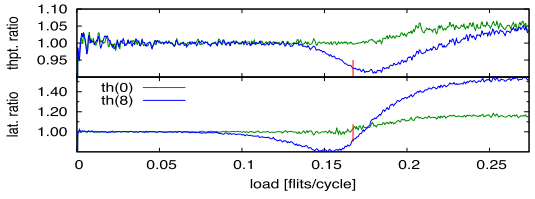
(d) bit-reverse (brev)



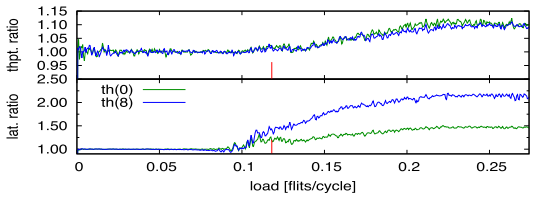
(e) bit-rotation (brot)



(f) tornado (torn)

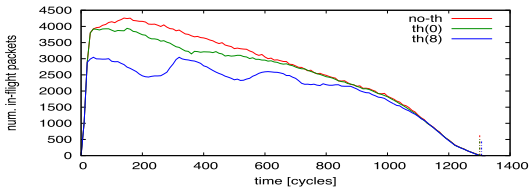


(g) random (rand)

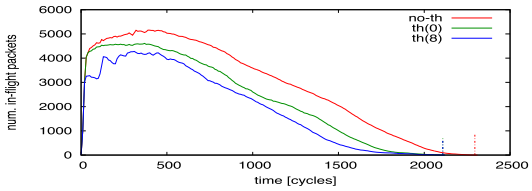


(h) random pair (rpar)

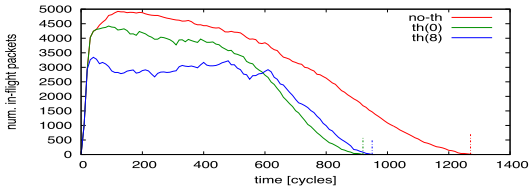
Fig. 9 Performance ratios of throughput and average latency.



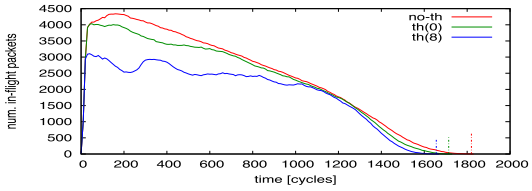
(a) transpose (trns)



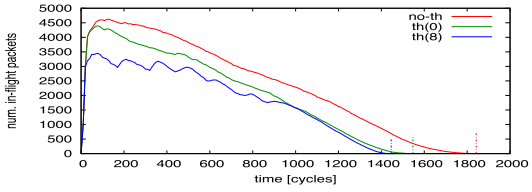
(b) perfect shuffle (shfl)



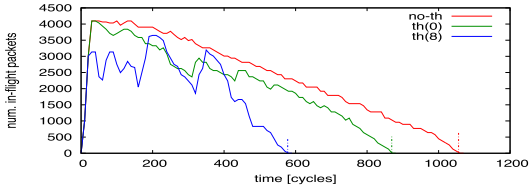
(c) bit-complement (bcmp)



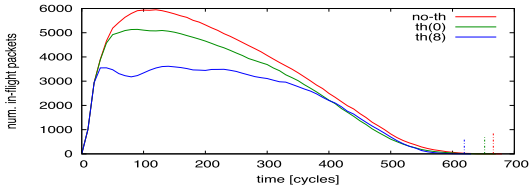
(d) bit-reverse (brev)



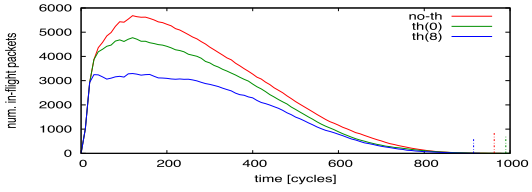
(e) bit-rotation (brot)



(f) tornado (torn)



(g) random (rand)



(h) random pair (rpar)

Fig. 10 Time series chart of the number of in-flight packets in various collective communication situations.

Table 2 Duration time of collective communication.

traffic pattern	no-th	th(0)		th(8)	
trns	1301	1301	(1.00)	1307	(.995)
shfl	2295	2108	(1.09)	2108	(1.09)
bcmp	1271	920	(1.38)	950	(1.34)
brev	1820	1714	(1.06)	1656	(1.10)
brot	1842	1547	(1.19)	1446	(1.27)
torn	1056	869	(1.22)	578	(1.83)
rand	671.5	653.3	(1.03)	633.1	(1.06)
rpar	1013.3	991.8	(1.02)	949.5	(1.07)
	[cycles]	[cycles]		[cycles]	

munication, the simulator measures the number of in-flight packets as well as throughput and average latency. This paper assumes that ten packets are transferred in each collective communication with eight-flit packet size. The simulator reports the measured values in every ten-cycle window time.

Figure 10 shows time series of the number of in-flight packets in the ordinary method (no-th), SPTh with busy information without margin (th(0)), and proactive method with eight-flit margin (th(8)). In this evaluation, a *margin* specifies the threshold level of *busy* buffer in the VCinfo registers. In the zero-margin (th(0)) case, only full buffers are marked *busy*. In the eight-flit margin (th(8)) case, buffers whose free-space is less than or equal to eight flits are marked busy.

Completion time (i.e., duration) is not clearly illustrated in the figure, we show the completion times as small vertical lines in Fig. 10. Furthermore, we summarize the measured duration times in **Table 2**. Integer values represent duration time in the number of simulation cycles, and fraction values in parentheses show the ratio to the performance of no-th. In rand and rpar cases, duration times in Table 2 are average values of ten experiments. Other traffic patterns are reproducible in duration time.

5.4 Discussion

5.4.1 Steady Communication Issues

We discuss steady communication results. As shown in Fig. 8, SPTh does not always improve critical load ratio. Its major reason is that the proposed method throttles packet injection only when congested area is detected. As Fig. 6 shows, many of congested areas disappear soon and do not persist even when the traffic load is critical to saturation (Fig. 6 (b)).

Assume that we extend the throttling method to sustain for a certain period of time even after no congestion is detected. We will leave this discussion as our future work.

Then, we discuss the extended proactive method (th(8)) that is proposed in Section 4.3. We should discuss the reason why the proactive method does not also enhance the critical load ratio. We expected the proactive method to suppress emergence of congestion, and our evaluation results proved the effectiveness in terms of observation of busy buffers as shown in **Fig. 11** (c) and Fig. 12 (c)[*5]. However, with respect to the critical load ratio, the proactive method does not show meaningful difference. This implicitly means that phase transition between congested–uncongested state is so steep that the proposed method cannot achieve considerable improvement.

[*5] Space-time charts are discussed later in this section.

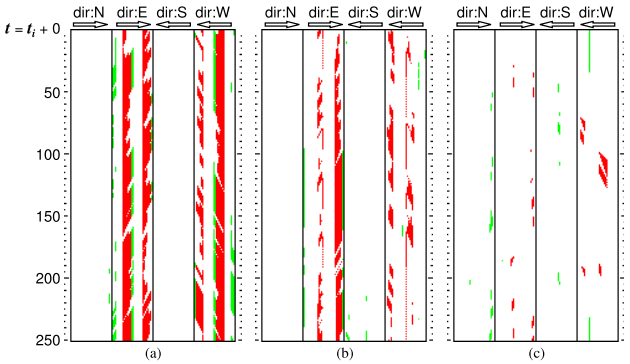


Fig. 11 Space-time charts in congested situations (32×32 two-dimensional torus, bit-complement traffic): (a) without throttling (no-th), (b) proposed method (th(0)), (c) proposed method (th(8)). t_i indicates the starting time of the space-time chart. In (a), (b), and (c), t_i = 1,220,400 [cycles].

Although the proposed method does not improve the critical load ratio, the method improves both throughput and average latency at a considerable level. We discuss the evaluation results depicted in Fig. 9. Some traffic patterns, i.e., trns and torn, show no specific improvements in throughput, and some other patterns, i.e., brev and rand, show degraded performance. Many other traffic patterns show up to 2.0 times improvement in throughput and about four times improvement in average latency. These evaluation results reveal that the proposed method resolves the initial objective of this paper, i.e., improvement of performance in over-loaded situations by relaxing congestion. Performance curves in Fig. 8 clearly reveal that the method prevents severe performance drop in over-saturated situation.

Effects of the proposed throttling method, SPTh, depend on traffic patterns. The method is based on the observation results that congested areas propagate in high-load situations. This implicitly assumes that the congestion will be extinguished if routers suppress supplying packets to the congested area. As far as a network employs deterministic routing algorithm, packets have no alternative route to reach their destinations. If a traffic pattern indicates that a packet injection affects congested area at a certain distance, the proposed throttling method works effectively. This discussion explains the performance degradation in the rand (random) pattern shown in Fig. 9 (g). We can explain that degradation alternatively. We can regard the proposed throttling method as a predictive method based on congestion detection. Many of traffic patterns are suitable for prediction for throttling, and other ones, i.e., rand, fail prediction.

5.4.2 Unsteady Communication Issues

Although steady-communication has two major issues, throughput and (average) latency, collective (unsteady) communication is discussed only with duration time.

As Fig. 10 shows, the number of packets explosively increases at the initial steps. All nodes simultaneously start injecting their packets at the initial stage of simulation. Since each buffer is empty at the beginning and is gradually filled by blocked packets, the throttling mechanism works after a certain delay, and thus, the explosive increase of packets is hardly suppressed.

The proposed throttling method absorbs the initial steep increase of packets and it succeeds in controlling the number of in-flight packets to avoid unnecessary conflict of packets that results in unnecessary performance loss. Figure 10 and Table 2 reveal performance impacts of the proposed method. The proposed method enhances collective communication performance, i.e., duration time, in most traffic patterns except trns (transpose). In unsteady communication cases, reduction of the in-flight packets does not necessarily contribute to performance enhancement. In the trns pattern, routes of packets are heavily overlapped. Since the deterministic routing algorithm offers no alternative routes for packets and the overlapped paths share a (physical) link, transmission of these packets are serialized. In terms of performance, the heaviest serialization determines the longest duration time, even if the network is optimized and well controlled.

This paper evaluates collective communication performance in which every router starts sending packets at the same time. Since the rush flows of packets soon exhaust buffer capacities, congested situation spreads before a throttling mechanism works. The rush flows encumber throttling methods, although the evaluation intends practical communication situations in parallel programs. Measuring collective communication performance is the worst-case evaluation for throttling methods.

Other existing throttling methods do not assume collective communication. However, they select other situations. Baydal et al. [10] assume a dynamic load situation in which every node starts sending a certain number of random packets in a high load rate. Thottethodi et al. [8], [9] assume a different situation of bursty load in which packet injection intervals are reduced to 1/100 in a short period of time (2,000 cycles). Although the dynamic and bursty load situations are close to the collective communication that is employed in this paper, they only assume uniform random traffic and they do not assume successive injection of packets.

5.4.3 Space-Time Chart Issues

Figures 11 and 12 show space-time charts of steady- and unsteady-communication situations, respectively. In the both figures, (a) shows the ordinary method without throttling (no-th), (b) shows the proposed method without margin (th(0)), and (c) shows the proposed proactive method with eight-flit margin (th(8)). Note that each buffer has sixteen flit capacity and the simulation uses eight-flit (fixed length) packets. And note that all space-time charts illustrated in Figs. 11 and 12 show busy buffers, not contents of VCinfo registers. Different colors show different virtual channels as described in Section 3.3; red, green, and blue dots represent busy status in the corresponding 0th, 1st, and 2nd virtual channels, respectively.

Figure 12 shows the space-time charts of unsteady communication cases. Every node starts packet generation at $t = 0$, and routers begin packet transfer at the same time. Since every buffer is vacant at $t = 0$, generated packets rush into the buffer for the initial period of time. However, blocked packets are accumulated in their corresponding buffers. Until the buffers become full, packet buffers are not recognized as busy. In Fig. 12 (a) and (b) cases, after passing the initial 26 cycles, many of buffers in E- and W-direction become busy. Vertical lengths of Fig. 12 (a), (b), and (c) illustrate duration times to complete the communication.

Figure 11 shows ready/busy behavior at 1.1 times of the critical

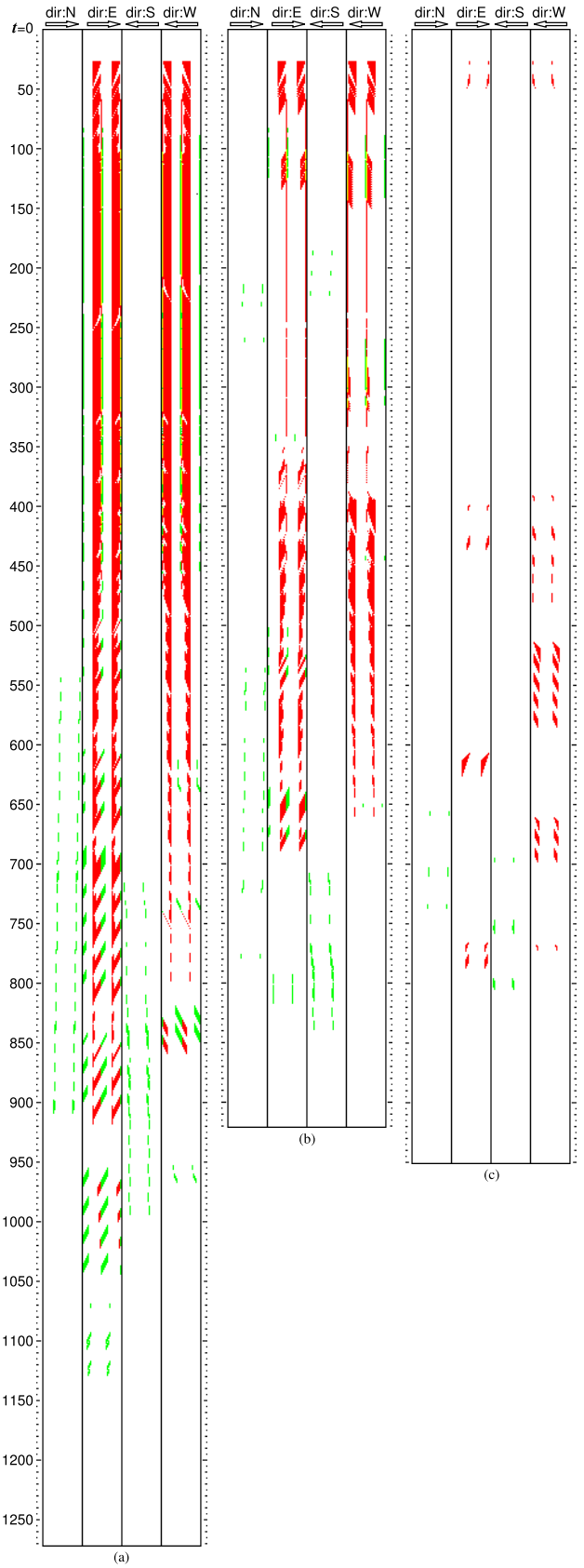


Fig. 12 Space-time charts of unsteady communication: (a) without throttling (no-th), (b) proposed method (th(0)), (c) proposed method (th(8)).

load ratio ($1.1C_r = 1.1 * 0.111 = 0.122$ [flits/cycle]). Figure 11 conforms with Fig. 6 with respect to the simulation conditions. Since the both figures represent the same traffic pattern at the same traffic load, Fig. 6 (c) is identical to Fig. 11 (a).

Figure 11 (a), (b) and (c) show the effects of the proposed throttling method. The ordinary method without throttling (Fig. 11 (a)) continuously causes heavy congestion. The proposed method with zero-margin (Fig. 11 (b)) relaxes congested situation at a certain level. However, congestion still remains whereas the proactive method with eight-flit margin sufficiently suppresses congestion.

Also in the unsteady communication cases, the proposed method sufficiently suppresses congestion as Fig. 12 shows. Each of Fig. 12 (a) to (c) shows busy buffers, conforming with Fig. 11. Vertical length of each chart represents the corresponding duration time. The proposed throttling method clearly reduces busy status of buffers as this paper aims at. Figure 12 (c), which shows the behavior in the th(8) case, shows drastic reduction of busy buffers.

5.4.4 Scale Issues

Although the previous sections reveal the effectiveness of the SPTh method, we should further discuss applicabilities to other configurations of interconnection networks. As the SPTh method is originally based on torus networks, we focus our discussions on network sizes in $N \times N$ two-dimensional torus topology and high-radix k-ary n-cube topologies.

At the first step of the discussion, we will clarify our assumptions to lead two types of packet conflicts. At the second step, we will discuss the conflict types. One of the two conflict types corresponds to the effects of the throttling method, but the other one does not directly affects the throttling method. This asymmetric nature of packet conflict presents the benefit and loss of the proposed throttling method.

This paper assumes two-dimensional torus topology, deterministic routing (dimension-order routing (DOR) algorithm), and date-line virtual channel control. We assume that every packet is transferred in the x-axis direction until the x-address of the intermittent router matches the packet's destination node, and then the packet is transferred in the y-axis direction. Therefore, most of the packets firstly move in the x-direction and turn to the y-direction. This paper also assumes an ordinary organization of router. Each router employs five input and output ports and a crossbar switch. Each input/output port corresponds to one of north-, east-, south-, and west-direction communication and CPU. Each input port employs channel buffers that support virtual channels, and every incoming packet is stored in the corresponding channel buffer. Each channel buffer is connected to a crossbar switch that connects input channel buffer and output port appropriately according to the head packet of the buffer. A packet conflict occurs when at least two of head packets simultaneously require the same output port.

As the DOR algorithm disallows any turn from y to x directions, a *straight* packet in the x-direction (which we call x-straight packet) is interfered only by a newly-injected packet that is destined to the same direction. On the other hand, a straight packet in the y-direction (y-straight packet) is interfered by turn packets from x-direction and a newly injected packet.

Here we assume a congested situation caused by successive conflicts of x-straight packets. The proposed SPTh method prop-

Table 3 SPTh effects on network sizes.

traffic	N = 8		N = 16		N = 32		N = 64		N = 128	
patt.	th(0)	th(8)	th(0)	th(8)	th(0)	th(8)	th(0)	th(8)	th(0)	th(8)
trns	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00
shfl	1.07	0.85	0.88	0.88	1.09	1.09	1.22	1.21	1.13	1.27
bcmp	1.00	1.00	1.10	1.08	1.36	1.34	1.31	1.40	1.30	1.38
brev	1.00	0.96	1.02	1.06	1.07	1.10	1.08	1.13	1.07	1.07
brot	1.00	0.95	1.02	1.06	1.20	1.27	1.14	1.39	1.11	1.03
torn	1.00	1.00	1.10	1.26	1.21	1.83	1.24	3.05	1.81	4.42
rand	0.99	0.93	1.00	1.01	1.03	1.06	1.06	1.07	1.05	1.14
rpar	0.99	0.89	1.03	1.04	1.05	1.08	1.07	1.12	1.08	1.16

agates the congestion information through the VCinfo mechanism in the reverse direction of the x-straight packets. Each router that receives the congestion information immediately suspends injection of new packets. Since the contents of VCinfo only move at most one step in every cycle, throttled areas spread on the cycle-by-cycle basis.

As the throttled area is widen, the number of x-straight packets decreases to release the congested situation. When the congestion disappears, the VCinfo mechanism carries the information to release the throttled situation of each node. Similarly to the spread of throttled area, the release spreads on the cycle-by-cycle basis. Note that nodes distant from the resolved congestion still continue to suspend packet injection until the release information arrives.

Conflicts of the x-straight packets are directly controllable by means of throttling. However, conflicts on the y-straight are not directly controllable. Throttling poorly affects the y-straight packets since most of injected packets are destined for x-direction. As the y-straight packets interfere with the turn packets, congested y-straight packets cause x-straight congestion that is connected by the turn packets. The newly caused x-straight congestion will be released by means of the SPTh mechanism, but the effect will be limited.

In a high-radix topology of network, throttling is effective only in the first dimension in the DOR routing, and congestions in other dimensions are indirectly controlled. We can estimate that the proposed SPTh method performs well in low-dimension networks and offers only limited effects in high-radix networks. This paper presents the qualitative discussions. Further evaluations are future work.

Next, we will discuss the effect of the SPTh method on various sizes of $N \times N$ torus networks. As we described above, congestion of the x-straight packets is caused by frequent conflicts of the straight and injected packets. If each packet moves only a short distance in the x-direction, packet conflict of the x-straight packet scarcely occurs. In general, small size of networks are relatively tolerable in congestion. As the network size increases, moving distance in the x-direction increases and congestion-tolerance also decreases. Large size networks are intolerable in congestion.

From the reverse point-of-view, we can estimate that the SPTh method is effective in large-size networks. To clarify the qualitative estimation, we further evaluated unsteady communication performance in various size of $N \times N$ torus networks. The length of VCinfo register is set to $N/2$, since any packet traverses at most $N/2$ hops in each of x- and y-directions and unnecessarily long VCinfo register negatively affects performance. **Table 3** summarizes the effects in N = 8, 16, 32, 64, 128 torus networks[*6]. This table shows the ratios of duration times. Each value is calculated as no-th duration time is divided by th(0) (th(8) duration time. Thus, values that exceed 1.0 mean performance benefit. Table 3 shows that the SPTh method performs well in large-size networks as the qualitative discussion estimated.

5.4.5 Optimal length of VCinfo register

Discussions and evaluation results in the previous section lead us other discussion on the length of VCinfo register. As we described above, contents of the VCinfo register are transferred on a cycle-by-cycle basis. This implies that distant routers from the congested area receive the congestion information after a certain number of simulation cycles. Furthermore, distant nodes suspend packet injection (i.e., start throttling) behind the near-nodes and in-flight packets from the distant nodes will affect the congested situation worse.

In the prior evaluations, we have assumed $N/2$ as the length of VCinfo register. But, we should evaluate the practical effect of the length of VCinfo register. We further evaluated unsteady communication performance in various lengths of VCinfo register.

Figure 13 shows the results in 32x32 torus network. Horizontal axis shows the length of VCinfo register, but zero (0) means no-throttling case (no-th). Vertical axis shows duration time. As shown in the figure, in many traffic patterns, duration time varies according to the length of VCinfo register and the optimal lengths depend on traffic patterns. The results still reveal the effectiveness of SPTh. Further discussions in determining the optimal length are our future work.

Discussion of the optimal VCinfo length leads us to further discussions on the delay issues of VCinfo propagation, since small/large propgation delays directly affect efficiencies of the proposed SPTh method. As Section 4.1 describes, this paper assumes an ideal condition for VCinfo propagation. This paper intends physical inter-router links dedicated for VCinfo propagation. Each VCinfo register is attached to the corresponging virtual channel, and we use three virtual channels in this paper. Thus, each of the dedicated physical links is shared by its correponding three VCinfo registers.

From the VCinfo mechanism given in Fig. 7 (presented in Section 4.1), a VCinfo register is updated when (1) the corresponding router detects the change of ready/busy status of the next neighboring buffer via the handshaking mechanism, and (2) new VCinfo contents are propagated from distant router(s). Since

[*6] Due to the implementation reason of the simulator, the maximum length of VCinfo register is limited to 32.

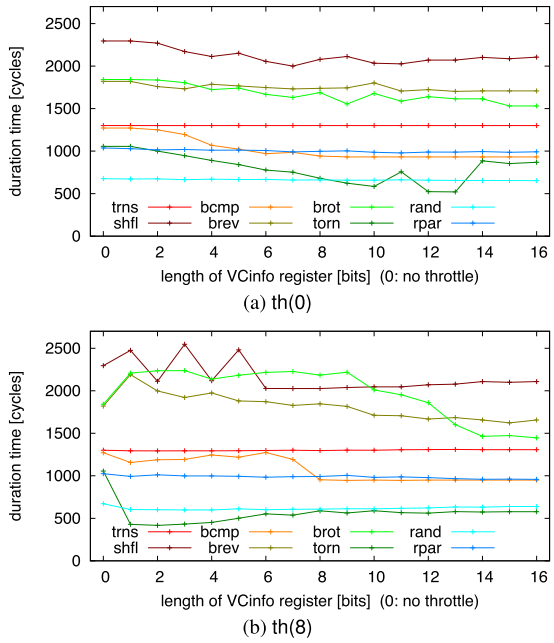


Fig. 13 Duration times of collective communication for various VCinfo length. (32x32 torus network)

each input buffer independently works, it is possible that multiple input buffers change their ready/busy status simultaneously. Thus, at most three VCinfo registers can be updated simultaneously. Since the physical link is shared by the VCinfo registers, updated information should be propagated serially.

Since contents in different VCinfo registers are serially propagated, VCinfo update caused by (2) propagation is strictly serialized (not in parallel). When both of (1) and (2) situations occur on the same VCinfo register, the update operations are merged in one cycle.

Thus, the discussions on VCinfo propgation delays are summarized as follows. The worst-case delay of VCinfo propagation is three cycles per hop. Not that this value is in the worst-case situation and practical delay strongly depends on the traffic pattern. Further detailed discussions on VCinfo delays are our future work.

5.4.6 Future Directions

As we discussed in Section 5.4.1, the proposed throttling method can be regarded as a simple prediction derived by observation of congested areas. Since practical effects of the throttling method depend on traffic pattern, the throttling method is tunable for the actual traffic patterns. The tuning issue is open for future discussions.

The next major issue is routing algorithm. This paper assumes the fundamental principle of FIFO-ness of packet arrivals, i.e., every packet from a certain node arrives in order. This paper assumes an old-styled deterministic routing algorithm (i.e., dimension-order routing, DOR). On the other hand, throttling methods are essentially orthogonal to routing algorithms. Effective combinations with various routing algorithms are expected as future work.

6. Conclusions

Interconnection network is one of the key components in massively parallel computers. The component is responsible to communication performance that directly affects parallel computing performance. Thus, exploiting maximal performance of interconnection network is a crucial issue.

Large-scale interconnection networks hardly implement centralized mechanisms due to the large number of components. Each component should work independently with others. Such decentralized manner takes us to the difficult problem, i.e., effective congestion control. We know that an interconnection network performs in proportion to the traffic load when the network is in an uncongested situation. Network performance drastically degrades as soon as congestion appears and remains for a long time. From opposite point of view, if we can obtain an effective control method to keep the network in an *edge* state to congestion, we can exploit maximal performance.

This paper discussed behaviors of routers in congested situations at variety of levels. We introduced *space-time chart* to represent spatio-temporal behaviors. Based on the observation results, we proposed a novel throttling method called *State-Propagation Throttling* (SPTh), we further discussed a proactive extension of the throttling method. We applied the proposed method to two classes of communications, steady- and unsteady-communication, with eight of typical traffic patterns. Experimental results by our interconnection network simulator reveal advantages of the proposed method. In steady-communication situations, the method increases performance in many traffic patterns. The evaluation results reveal that throughput and average latency are enhanced at most two times and four times, respectively.

The proposed method works well even in difficult situations. Collective communication, as unsteady-communication, is used as the worst-case situations, where we assume that all nodes start sending packets in unison. Even the proactive method cannot prevent explosive increase of packets in the initial cycles and resulting severe congestion, the proposed throttling method effectively works so that it can suppress unnecessary conflicts of packets, and it can enhance the performance of collective communication.

Acknowledgments This research was supported in part by Grant-in-Aid for Scientific Research ((C) 24500055, (C) 24500054) and Grant-in-Aid for Young Scientists ((B) 25730026) of Japan Society for the Promotion of Science (JSPS). We sincerely thank anonymous reviewers and editors for their valuable comments that suggests us to deep discussions.

References

[1] Pfister, G.F. and Norton, V.A.: "Hot Spot" Contention and Combining in Multistage Interconnection Networks, *IEEE Trans. Comput.*, Vol.C-34, No.10, pp.943–948 (1985).
[2] Dally, W.J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Pub. (2004).
[3] Duato, J., Yalamanchili, S. and Ni, L.: *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Pub. (2003).
[4] So, T., Oyanagi, S. and Yamazaki, K.: Speculative Selection in Adaptive Routing on Interconnection Networks, *IPSJ Trans. Advanced Computing Systems*, Vol.44, pp.147–156 (2003).
[5] Yokota, T., Nishitani, M., Ootsu, K., Furukawa, F. and Baba, T.: Cross-Line: A Novel Routing Algorithm That Uses Global Information, *IPSJ Trans. Advanced Computing Systems*, Vol.46, No.SIG 16 (ACS-12), pp.28–42 (2005). (in Japanese).
[6] López, P., Martínez, J.M. and Duato, J.: DRIL: Dynamically Reduced Message Injection Limitation Mechanism for Wormhole Networks,

Proc. 1998 International Conference on Parallel Processing, pp.535–562 (1998).
[7] Obaidat, M.S., Al-Awwami, Z.H. and Al-Mulhem, M.: A New Injection Limitation Mechanism for Wormhole Networks, *Computer Communications*, Vol.25, pp.997–1008 (2002).
[8] Thottethodi, M., Lebeck, A.R. and Mukherjee, S.S.: Self-Tuned Congestion Control for Multiprocessor Networks, *Proc. 7th International Symposium on High-Performance Computer Architecture* (*HPCA'01*), pp.107–118 (2001).
[9] Thottethodi, M., Lebeck, A.R. and Mukherjee, S.S.: Exploiting Global Knowledge to Achieve Self-Tuned Congestion Control for *k*-Ary *n*-Cube Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol.15, No.3, pp.257–272 (2004).
[10] Baydal, E., López, P. and Duato, J.: A Family of Mechanisms for Congestion Control in Wormhole Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol.16, No.9, pp.772–784 (2005).
[11] Yokota, T., Ootsu, K., Furukawa, F. and Baba, T.: Entropy Throttling: A Physical Approach for Maximizing Packet Mobility in Interconnection Networks, *Proc. 11th Asia-Pacific Computer Systems Architecture Conference* (*ACSAC 2006*), pp.309–322 (2006).
[12] Yokota, T., Ootsu, K., Furukawa, F. and Baba, T.: Entropy Throttling for Maximizing Packet Mobility in Interconnection Networks, *IPSJ Trans. Advanced Computing Systems*, Vol.47, No.SIG 12 (ACS 15), pp.1–11 (2006). (in Japanese).
[13] Shibamura, H., Miwa, H., Susukita, R., Hirao, T., Ajima, Y., Miyoshi, I., Shimizu, T., Ishihata, H. and Inoue, K.: Optimization and Simulation Evaluation of an All-to-all Communication Using Packet Pacing, *IPSJ Trans. Advanced Computing Systems*, Vol.4, No.3, pp.56–65 (2011). (in Japanese).
[14] Ohira, T. and Sawatari, R.: Phase Transition in a Computer Network Traffic Model, *Physical Review E*, Vol.58, No.1, pp.193–195 (1998).
[15] Tretyakov, A.Y., Takayasu, H. and Takayasu, M.: Phase Transition in a Computer Network Model, *Physica A*, Vol.253, pp.315–322 (1998).
[16] Takayasu, M., Takayasu, H. and Fukuda, K.: Dynamic Phase Transition Observed in the Internet Traffic Flow, *Physica A*, Vol.277, pp.248–255 (2000).
[17] Valverde, S. and Solé, R.V.: Self-organized Critical Traffic in Parallel Computer Networks, *Physica A*, Vol.3112, No.3–4, pp.636–6448 (2002).
[18] Yokota, T., Ootsu, K., Furukawa, F. and Baba, T.: Phase Transition Phenomena in Interconnection Networks of Massively Parallel Computers, *Journal of the Physical Society of Japan*, Vol.75, No.7, p.078401 (7 pages) (2006).
[19] Nagel, K. and Schreckenberg, M.: A Cellular Automaton Model for Freeway Traffic, *Journal of Physics I France*, Vol.2, pp.2221–2229 (1992).
[20] Nagatani, T.: Self-Organized Criticality and Scaling in Lifetime of Traffic Jams, *Journal of the Physical Society of Japan*, Vol.64, No.1, pp.31–34 (1995).
[21] Shante, V.K. and Kirkpatrick, S.: An Introduction to Percolation Theory, *Advances in Physics*, Vol.20, No.85, pp.325–357 (1971).
[22] Barrat, A., Marthélemy, M. and Vespignani, A.: *Dynamic Processes on Complex Networks*, Cambridge University Press (2008).
[23] Takayasu, M.: *Complex Dynamics in Communication Networks*, chapter Dynamic Complexity in the Internet Traffic, pp.329–358, Springer (2006).
[24] Yokota, T., Ootsu, K. and Baba, T.: Steady/Unsteady Communication Performance in Large-Scale Regular Networks, *Proc. 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications* (*WAINA*), pp.727–732 (2011).
[25] Kermani, P. and Kleinrock, L.: Virtual Cut-Through: A New Computer Communication Switching Technique, *Computer Networks*, Vol.3, No.4, pp.267–286 (1979).
[26] Yokota, T., Ootsu, K. and Baba, T.: A Quantitative Evaluation Methodology of Interconnection Networks, *IPSJ Trans. Advanced Computing Systems*, Vol.2, No.3, pp.58–70 (2009).

Takashi Yokota received his B.E., M.E., and Ph.D. degrees from Keio University in 1983, 1985 and 1997, respectively. He joined Mitsubishi Electric Corporation in 1985, and was engaged in several research projects in special-purpose, massively parallel and industrial computers. He was engaged in research and development of a massively parallel computer RWC-1 at Real World Computing Partnership as a senior researcher from 1993 to 1997. From 2001 to 2009, he was an associate professor at Utsunomiya University. Since 2009, he has been a professor at Utsunomiya University. His research interests include computer architecture, parallel processing, network architecture and design automation. He is a member of IPSJ, IEICE and the IEEE Computer Society.

Kanemitsu Ootsu received his B.S. and M.S. degrees from the University of Tokyo in 1993 and 1995 respectively, and later he obtained his Ph.D. in Information Science and Technology from the University of Tokyo in Japan. From 1997 to 2009, he is a research associate and then an assistant professor at Utsunomiya University. Since 2009, he is an associate professor at Utsunomiya University. His research interests include high-performance computer architecture, multi-core multithread processor architecture, mobile computing devices, binary translation, and dynamic optimization. He is a member of IPSJ, IEICE, and ISCIE.

Takeshi Ohkawa received his B.E. and M.E. and Ph.D. degrees in Electronics Engineering from Tohoku University in 1998, 2000 and 2003, respectively. He was a postdoctoral fellow at Tohoku University, where he engaged in the reconfigurable LSI and software/hardware co-design in 2003. In 2004, he joined National Institute for Advanced Industrial Science and Technology (AIST), Japan as a researcher, where he started working on distributed object technology and its application to FPGA. From 2009, he started work in start-up company TOPS systems, where he engaged in R&D of heterogeneous multi-core processor and its software platform. From 2011, he is an assistant professor in Utsunomiya University. His research interests include reconfigurable technology, software/hardware co-design, parallel and distributed architecture and component-based design technology. He is a member of ACM, IEICE, and IPSJ.