Active Packet Pacing as a Congestion Avoidance Technique in Interconnection Network

Hidetomo SHIBAMURA
Institute of Systems, Information Technologies and Nanotechnologies
CREST, Japan Science and Technology Agency
2-1-22 Momochihama, Sawara-ku, Fukuoka 814-0001, Japan

Abstract. A congestion avoidance technique using active packet pacing for interconnection network is presented. In this technique, communication packets are sent intermittently by adjusting the inter-packet gap in order to match up packet flows precisely. Therefore the network congestion is avoided. The gap is pre-calculated for each communication according to the communication hop, and the value is explicitly passed to a sending function. Performance evaluations for some collective communications were performed by using an interconnect simulator and a benchmark program was executed on an actual machine. Finally, the scalability and effectiveness of packet pacing in interconnection network were confirmed.

Keywords. packet pacing, congestion avoidance, interconnection network, HPC, network simulation

Introduction

In most of HPC systems, a communication message is split into some packets and they are continuously injected into the interconnection network. When two or more different packet flows merge into a same direction link on a router, each throughput of flows is reduced and trailing packets are often blocked by packet conflict. These blocked packets are buffered on the router; however network latency increases because it must wait restart of transmission of preceding packets. In addition, the latency grows in accordance with communication hops because the situation of packet blocking propagates upstream. In such crammed packet transmission, heavy traffic communication (e.g. All-to-all communication) puts considerable workload on interconnect and may cause a slowdown of the program execution. Thus, in order to relax suppressed communication, this paper proposes aggressive use of packet pacing in interconnection network as a congestion avoidance technique.

The aim of this study is to clarify effectiveness of packet pacing in interconnection network. Communication performances of various algorithms which use packet pacing were evaluated by using an interconnect simulator. In addition, a benchmark program was executed on a supercomputer to show a possibility of packet pacing on an actual machine.

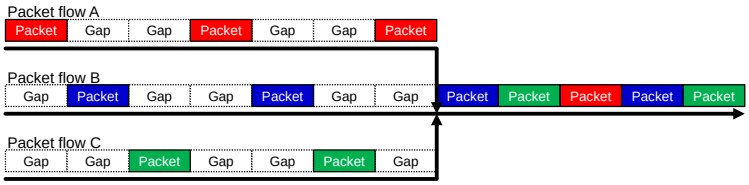


Figure 1. Merging three packet flows on a router using packet pacing.

In the following, Section 1 describes active packet pacing and Section 2 introduces an interconnect simulator NSIM. Section 3 examines the effectiveness of packet pacing by simulation, and then Section 4 also examines the effectiveness by real execution on an actual supercomputer. The concluding remarks and future directions of research are cited in Section 5.

1. Active Packet Pacing

A wide variety of congestion avoidance techniques are proposed and applied in networking. Packet pacing (or packet spacing) is one of the techniques, which throttles packet injection by inserting non-sending period (inter-packet gap) between sending packets. Since interleaved packet flows having appropriate inter-packet gap can be merged smoothly on a router as show in Figure 1, the packet blocking is resolved and network latency is also decreased.

There are many studies on packet pacing for global networking, but very few for interconnection network in contrast. Most of packet pacing are observation-based and throttle packet injection according to RTTs (Round Trip Time) [1] or ECNs (Explicit Congestion Notification) [2]. Thus, they can be defined as a category of *passive packet pacing*. In such pacing, source node takes time to detect any occurrence of network congestion because there is a delay until receiving a packet for RTT or ECN which returns from destination node. Since this delay retards the timing of packet pacing, there is a possibility to aggravate the congestion.

On the other hand, this study proposes using *active packet pacing* which controls the packet injection aggressively. In this packet pacing, inter-packet gap is pre-calculated per a route before packet injection in accordance with *message overlap degree* which is the number of messages across the link at the same time. In other words, optimum gap values are definitely found in every communication. Various communication patterns of major collective communications are simple and easily understandable, and the cost for the calculation of the gap is not so high.

2. NSIM: An Interconnect Simulator

Interconnection network, one of the important components in constructing parallel computer, is characterized by link bandwidth, topology, routing, flow control, and so on. Furthermore, the network must have a high affinity for parallel application. Designing an interconnection network with inappropriate parameters will lead to degradation of network performance and jeopardize high performance gained by parallelization. Since it

Table 1. Specification of an evaluation system

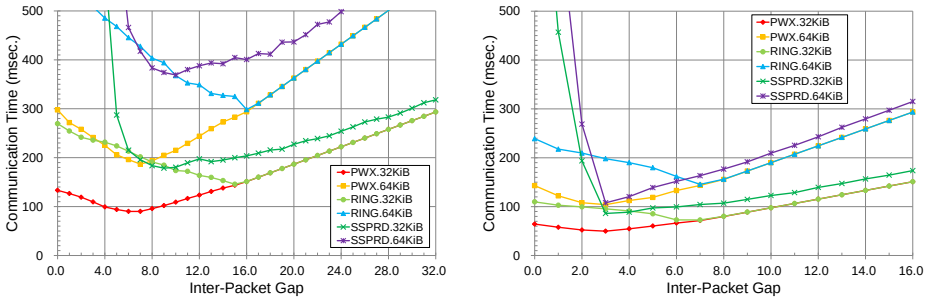
Parameter	Value
Topology	2D-torus / 3D-torus
Routing	Dimension Ordered Routing (DOR) + dateline
Flow control	Virtual cut-through, Credit base
Link bandwidth	4GB/s (unidirection)
Routing computation (RC)	4ns
Virtual-channel allocation (VA)	4ns
Switch allocation (SA)	4ns
Switch traversal (ST)	4ns
Switch latency	78ns
Cable latency	10ns
MTU	2KiB
Packet size	32B - 2KiB (MTU)
Packet header size	32B
Number of virtual channel	2
Virtual-channel buffer size	8KiB (MTU$\times$4)
Flit size	16B
Number of NIC	4(enables simaltaneous send/receive)
DMA rate	16GB/s
Memory bandwidth	16GB/s
MPI overheads	200ns

is indeed important to select the most effective parameters, interconnect simulation is indispensable to evaluate performance of interconnection network before proceeding to major steps [3].

NSIM is an interconnect simulator, which is developed to assist in various performance evaluation of exascale interconnection network [4]. Since NSIM is based on parallel discrete event simulation (PDSE) and implemented with MPI, it works on various platforms. The most recent version of NSIM supports Remote Direct Memory Access (RDMA) style communication [5]. NSIM generates communication events internally in response to actual execution of MPI-like C program without using communication log files obtained from a real machine or artificially generated. Very fine interconnect configuration and mapping of rank to node are given by dedicated files. Mesh, torus (up to 6-dimension), Tofu (K computer, FX10), and FatTree are supported. NSIM could simulate a random ring communication (HPCC) on 128K-node 3D-torus in 18 minutes by using real 128-core system for example.

3. Performance Evaluation by Simulation

In this section, some collective communications are simulated to show a possibility of packet pacing and are evaluated from the point of view of performance indexes. A target evaluation system in these simulations is assumed to have mechanisms and performance roughly correspond to recent supercomputers. The specification of the system is given in Table 1.Note that each parameter is decided from mainstream technology, and performance and the system is nonexistent.



(a) 2D-torus, 1,024 nodes (32×32) (b) 3D-torus, 1,024 nodes (16×8×8)

Figure 2. Communication time with various inte-packet gap value.

3.1. Simple Packet Pacing

Three alltoall communications were simulated by using NSIM in order to examine basic potential of packet pacing. The alltoall algorithms are pair-wise exchange (PWX), ring (RING), and simple spread (SSPRD). The simulation is performed by changing topology, node size, alltoall algorithm, message size, and inter-packet gap. The gap value is fixed in each simulation, i.e. same gap value is always used for packet injection in a simulation. Here, gap=0.0 means that there is no space between packets; therefore, the packets are continuously injected. On the other hand, gap=2.0 means that twice the time of a packet transmission time is inserted as illustrated in Figure1.

Figure 2(a) shows the communication time for each inter-packet gap on a 2D-torus system having 1,024 nodes. From the figure, the communication time of every algorithm is decreased by increasing the inter-packet gap from 0.0. However, too big gap value causes performance degradation. Figure 2(b) shows similar results on a 3D-torus.

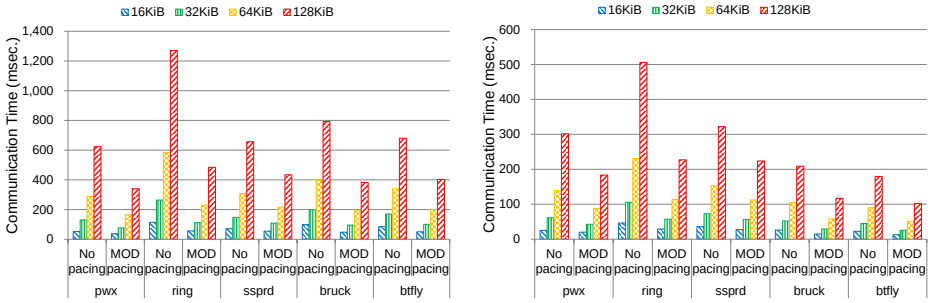
From these figures, it is clear that each algorithm has an optimum pacing point which minimizes the communication time.

3.2. MOD Packet Pacing

Five alltoall communications were simulated in order to examine the effectiveness of MOD pacing. MOD is message overlap degree described in Section 2. The inter-packet gap is defined as $hopcount - 1$ because each link is shared by messages of the number of hop-count at most. Moreover, the gap is calculated repeatedly for each communication step in each algorithm. Two alltoall communications, bruck's algorithm (bruck)[6], and butterfly (btfly), were added to the simulation.

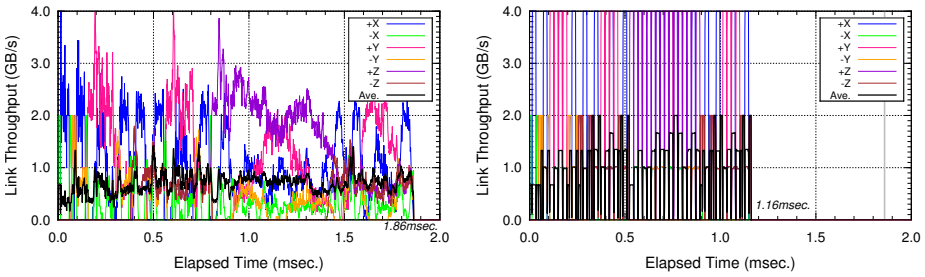
Figure 3(a) shows the communication time with several message sizes on a 2D-torus system and Figure 3(b) shows one on a 3D-torus. Every communication time is reduced drastically by using MOD pacing. Furthermore, the effectiveness of packet pacing also improves as the message size increases because long-term packet blocking caused by large-size message is suppressed.

From these results, it is confirmed that MOD pacing decreases communication time of every alltoall algorithm and works well for larger message size.



(a) 2D-torus, 1,024 nodes (32×32) (b) 3D-torus, 1,024 nodes (16×8×8)

Figure 3. Communication time with/without MOD pacing.



(a) No packet pacing (b) MOD packet pacing

Figure 4. Link throughput transitions of pairwise exchange (3D-torus 4×4×4, 40KiB message size).

3.3. Link Throughput

This subsection focuses on link throughput. A simulation of pairwise exchange algorithm was performed with 40KiB message size on a 64-node 3D-torus system. Link throughput of each elapsed time was recorded by NSIM and Figure 4 shows the transitions. Note that the link bandwidht is 4GB/s.

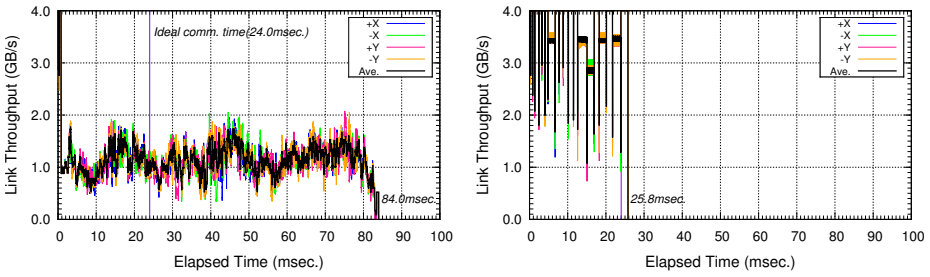
Figure 4(a) is the outcome without packet pacing, and Figure 4(b) is one with MOD pacing. By using packet pacing, some link throughput become maximum and the execution performance becomes 1.62 times faster (from 1.86 to 1.16 msec.).

A similar evaluation on A2AT, a novel alltoall algorithm for mesh/torus [7], was also performed with 1MiB message size on an 81-node 2D-torus system. Figure 5(a) is the result without packet pacing and Figure 5(b) is one with MOD pacing on a 2D-torus. Every link throughput is remarkably improved and the execution performance also becomes 3.2 times faster (from 84.0 to 25.8 msec.) and 107.5% of ideal execution time (24.0msec.) is achieved.

From these results, it is clear that packet pacing can improve utilization of network resources.

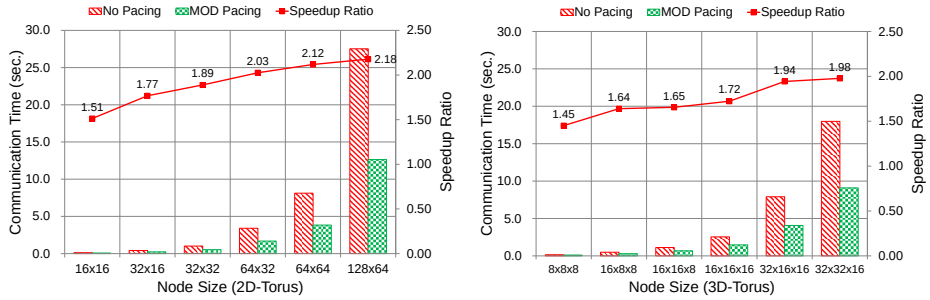
3.4. Scalability

In this subsection, speed-up ratios for each node size were investigated in order to investigate whether packet pacing has an affinity with future exascale system. Six 2D-torus



(a) No packet pacing (b) MOD packet pacing

Figure 5. Link throughput transition of A2AT (2D-torus 9×9, 1MiB message size).



(a) 2D-torus, alltoall (pairwise exchange) (b) 3DT-torus, alltoall (pairwise exchange)

Figure 6. Communication time and speedup ratio on various node size.

systems up to 8K nodes and six 3D-torus systems up to 16K nodes were simulated with pairwise exchange algorithm. Message size was 200KiB.

Figure 6(a) shows the results of 2D-torus systems and Figure 6(b) shows one of 3D-torus systems. As the number of nodes increases, the speedup ratio also grows by approximately doubled.

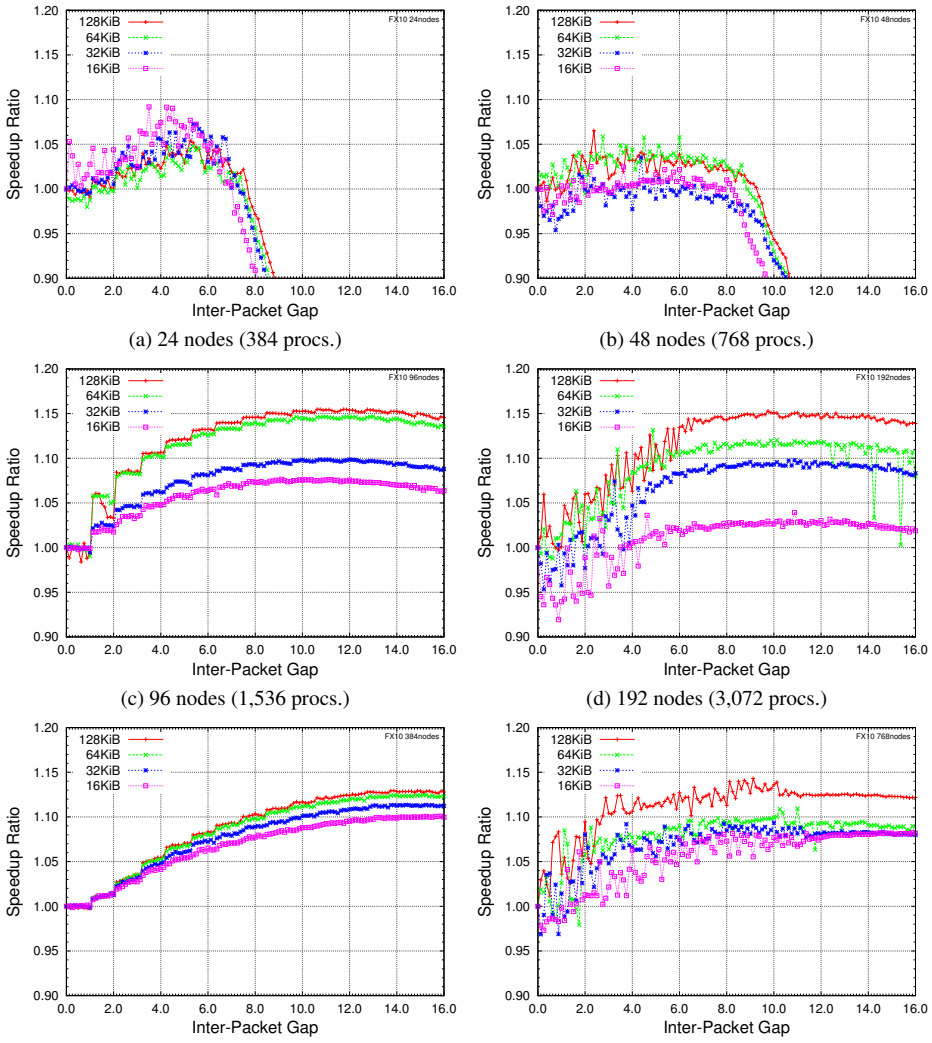
These results indicate that packet pacing has a good affinity with large-scale interconnection network.

4. Performance Evaluation on Actual Machine

This section demonstrates the effectiveness of packet pacing on an actual machine. A random ring communication program which is one of the benchmark programs in HPCC (HPC Challenge) [8] was executed on Fujitsu FX10 supercomputer system at Kyushu University in Japan. On FX10, users can adjust the inter-packet gap value via an environment variable at time of submitting job. The program was executed while varying inter-packet gap, message size, and the number of node up to 768 nodes.

Figure 7 demonstrates the results. Every graph surely represents that as the gap increases, the speedup ratio also grows. Similarly, as the message size increases, the speedup ratio also grows. Besides, good pacing point area (top of arch) can be found.

Consequently, active packet pacing may be one of effective congestion avoidance techniques for interconnection network toward future exascale system.



(a) 24 nodes (384 procs.)

(b) 48 nodes (768 procs.)

(c) 96 nodes (1,536 procs.)

(d) 192 nodes (3,072 procs.)

(e) 384 nodes (6,144 procs.)

(f) 768 nodes (12,288 procs.)

Figure 7. Effect of packet pacing on an actual machine (random ring comm. on Fujitsu FX10).

5. Conclusion

This paper presented active packet pacing which controls the packet injection aggressively as a congestion avoidance technique. Performance evaluation by interconnect simulation was performed and it was clear that proper setting of inter-packet gap greatly improves communication performance. Furthermore, the scalability and the effectiveness of packet pacing were verified on an actual supercomputer.

Future work includes applying packet pacing to not only collective communication but also practical parallel application. Performance evaluations of a few applications are in progress.

Acknowledgements

This research was supported by JST, CREST. The computational resources were provided by RIIT in the Kyushu University.

References

[1] A. Aggarwal, A. Savage, and T. Anderson: Under-standing the Performance of TCP Pacing, INFOCOM 2000 Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp.1157–1165, 2000.

[2] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato: Solving Hot Spot Contention Using InfiniBand Architecture Congestion, Proceedings HP-IPC 2005, 2005.

[3] R. Susukita, H. Ando, M. Aoyagi, H. Honda, Y. Inadomi, K. Inoue, S. Ishizuki, Y. Kimura, H. Komatsu, M. Kurokawa, K. Murakami, H. Shibamura, S. Yamamura, and Y. Yu: Performance Prediction of Large-scale Parallel System and Application using Macro-level Simulation, International Conference for High Performance Computing, Networking, Storage and Analysis (SCf08), 2008.

[4] H. Miwa, R. Susukita, H. Shibamura, T. Hirao, J. Maki, M. Yoshida, T. Kando, Y. Ajima, I. Miyoshi, T. Shimizu, Y. Oinaga, H. Ando, Y. Inadomi, K. Inoue, M. Aoyagi, and K. Murakami: NSIM: An Interconnection Network Simulator for Extreme-Scale Parallel Computers, IEICE Trans. Inf. & Syst., Vol. E94-D, No.12, pp.2298–2308, 2011.

[5] H. Shibamura: NSIM-ACE: Network Simulator for Global Memory Access, JST/CREST International Symposium on Post Petascale System Software (ISP2S2), Poster session, 2014.

[6] J. Bruck, C.T. Ho, S. Kipnis, E. Upfal, D. Weathersby: Efficient algorithms for all-to-all communications in multiport message-passing systems, Parallel and Distributed Systems, IEEE Trans. 8(11), pp.1143-1156, 1997.

[7] S. Yazaki, H. Takaue, Y. Ajima, T. Shimizu, and H. Ishihata: An Efficient All-to-all Communication Algorithm for Mesh/Torus Networks, Proc. 10th IEEE Intl. Symp. on Parallel and Distributed Processing with Applications (ISPA2012), pp.277–284, 2012.

[8] P.R. Luszczek, D.H. Bailey, J.J. Dongarra, J. Kepner, R.F. Lucas, R. Rabenseifner, and D. Takahashi: The HPC Challenge (HPCC) benchmark suite, Proc. 2006 ACM/IEEE Conference on Supercomputing, SC'06, 2006.