Efficient Total-Exchange in Wormhole-Routed Toroidal Cubes

Fabrizio Petrini*

Scientific Computing Group (CIC-19), MS B256

Los Alamos National Laboratory

Los Alamos, NM 87545 – USA

e-mail: fabrizio@lanl.gov

June 2, 1999

Abstract

The total-exchange is one of the most dense communication patterns and is at the heart of numerous applications and programming models in parallel computing. In this paper we present a simple randomized algorithm to efficiently schedule the total-exchange on a toroidal mesh with wormhole switching. This algorithm is based on an important property of the wormhole networks that reach high performance under uniform traffic using adaptive routing.

*The work was supported by the U.S. Department of Energy through Los Alamos National Laboratory contract W-7405-ENG-36

The experimental results, conducted on a 256 nodes bi-dimensional torus, show that this algorithm reaches a very high level of performance, around 90% of the optimal bound, and is more efficient than other algorithms presented in the literature.

Keywords. Total-Exchange, Personalized Communication, Interconnection Networks, Wormhole Routing, Collective Communication Patterns

1 Introduction

Communication between nodes in a parallel machine can generally be described as x-to-y communication where x and y can be substituted by *one*, *all* and *many*. Communication implies nodes sending and receiving messages: x being *one*, *all* and *many* respectively, indicates that only one of the nodes, that all nodes, and that only some nodes send data. Similarly, y being *one*, *all* and *many* indicates that from each of the senders one, all and many nodes receive data respectively. Communication can be further distinguished as a *broadcast/accumulation* or as a *personalized* communication. For example, one-to-all communication could be either a one-to-all broadcast (*single-node broadcast*) where a single node sends out the same message to all nodes, or a one-to-all personalized communication (*single-node scatter*) where a single node sends out different messages to each node. Communication with multiple senders and multiple receivers is also referred as *collective* communication. The nature of the messages to be sent can be also classified as *personalized* or *non-personalized*. The all-to-all personalized communication, or simply *total-exchange*, is an important communication pattern that is at the heart of many applications, such as matrix transposition and the fast Fourier transform (FFT), and programming models as the BSP [23].

The *total-exchange* is a collective communication pattern where every node has to send a distinct message to any other node. The efficient implementation of the total-exchange has been extensively studied in a variety of networks [15] [2] [14] [24] [25] [21]. These studies were motivated by several applications in the field of scientific computing [18].

Wormhole switching has been adopted by many new-generation parallel computers, such as the Intel Touchstone Delta, Intel Paragon, MIT J-Machine, Stanford Flash and the Cray T3D and T3E. In such networks, a packet is partitioned in a sequence of elementary units called *flits*, which are sent in a worm-like or pipelined manner. Network throughput of wormhole networks can be increased by organizing the flit buffers associated with each physical channel into several virtual channels [4]. These virtual channels are allocated independently to different packets and compete with each other for the physical bandwidth. This decoupling allows active messages to pass blocked messages using network bandwidth that would otherwise be wasted.

In this paper we present a simple and innovative random algorithm to schedule the total-exchange on a wormhole-routed toroidal mesh. Most of the literature that deals with the total-exchange tends to strictly divide the transmission in phases that proceed in lockstep. These algorithms often neglect important aspects of the interconnection network, as the deadlock avoidance policy or the nondeterminism of the routers when they have to route several packets. These characteristics often break the symmetries and the regular structure of many total-exchange algorithms, that are, therefore, not very robust in practice.

Our work is motivated by the fact that wormhole networks can reach a high throughput under uniform random traffic if packets are routed adaptively and if the packet size is properly chosen [11]. For this reason we adopt a randomized strategy that reproduces a uniform random traffic inside the network. The experimental results, conducted on a bi-dimensional torus with 256 nodes using a detailed simulation model show that it is possible to achieve a network throughput that is very close to optimality.

The rest of this paper is organized as follows. Section 2 provides some background information on how flow control is performed in high performance system area networks. Section 3 describes the two routing algorithms for the class of k-ary n-cubes, deterministic and minimal adaptive based on Duato's methodology, that we consider in our study. Section 4 reviews some total-exchange algorithms presented in the literature and Section 5 motivates and describes our randomized algorithm. The relevant details of the simulation models used in the experimental evaluation are shown in Section 6 and the performance results, conducted on a bi-dimensional torus with 256 nodes, are shown in Section 7. Finally, some concluding remarks are in given in Section 8.

2 Flow Control in High Performance Networks

When communication is performed without waiting for the path setup to be complete, as in packet switching, a packet may block within the network as a result of network congestion. Flow control has to regulate the movement of packets from node to node so as to utilize the network resources, buffers and communication channels, as efficiently as possible.

A simple approach to implement network flow control is *store and forward*. In this method, the entire packet (message) is buffered at each intermediate node and forwarded to the next node in its path as soon as the desired outgoing link becomes available. This switching strategy was adopted in early prototypes and commercial machines including the iPSC/1, nCUBE-1 and those based on the Transputers T414 and T800. Store and forward flow control is simple but has some drawbacks. First, each node must buffer every incoming packet, using memory space. Second (and more important), the network latency is proportional to the distance between the source and destination nodes. The network latency is, in the absence of contention, $(L/B)D$, where L is the packet length, B is the channel bandwidth, and D is the length of the path between the two nodes.

An improvement over the store and forward approach, called *virtual cut through*, avoids this problem by buffering a packet only when it encounters a busy link [17]. Cut through routing was used in the torus routing chip implemented at Caltech [6].

Buffering can be reduced to a minimum if the blocked packet can stay in multiple nodes along the path, holding the links between them. The *wormhole* approach is based on this idea [7]. A packet is divided in *flits* (flow control digits) for transmission. The header flit forces the route. As the header advances, the remaining flits follow in a pipeline fashion. If the header finds a link already in use, it is blocked until the link becomes available. The network latency for wormhole, again, in the absence of contention, is $(L_f/B)D + L/B$, where L_f is the length of each flit, B is the channel bandwidth, and L is the length of the message. If $L \gg L_f$, the path length will not significantly affect the network latency unless D it is very large. For low to moderate traffic, wormhole flow control outperforms store and forward. On the other hand, wormhole can easily introduce deadlocks by causing cyclic dependencies in the resources allocation if the path selection and the link allocation are not performed carefully. The first prototype that adopted wormhole flow control was the Ametek 2010, which used a two dimensional mesh topology. The Intel Touchstone Delta and Intel Paragon, the MIT's J-machine, the CMU iWarp, the Transputer T9000, the Meiko CS-2 and both Cray T3D and T3E, the SGI 2000 use wormhole in their communication building blocks too.

The mostly used approach to deadlock avoidance in wormhole routing consists in splitting the physical channels into virtual channels. This is obtained by multiplexing a number of virtual channels onto the same physical channel on demand. Each virtual channel is provided with its own data structures. Deadlock is prevented by constraining the allocation of the virtual channels. Through virtual channels, a physical network can be partitioned into disjoint logical networks, facilitating the implementation of adaptive routing algorithms. Also, virtual channels are useful for many other reasons. By increasing the logical degree of connectivity in the network they facilitate the mapping of applications in which processes communicate according to another logical topology. They also increase the network performance at high traffic loads.

Some intuition on how wormhole flow control is performed is given in Figure 1. Each packet is fragmented in elementary units, called flits, and these flits are propagated in a pipeline fashion from the router P_1 to router P_4. If the header flit is halted by local congestion on router P_4, the communication bandwidth along the path occupied by the packet can be utilized by another packet, that can be transmitted along the routers P_1, P_2 and P_3 when the previous packet is blocked.

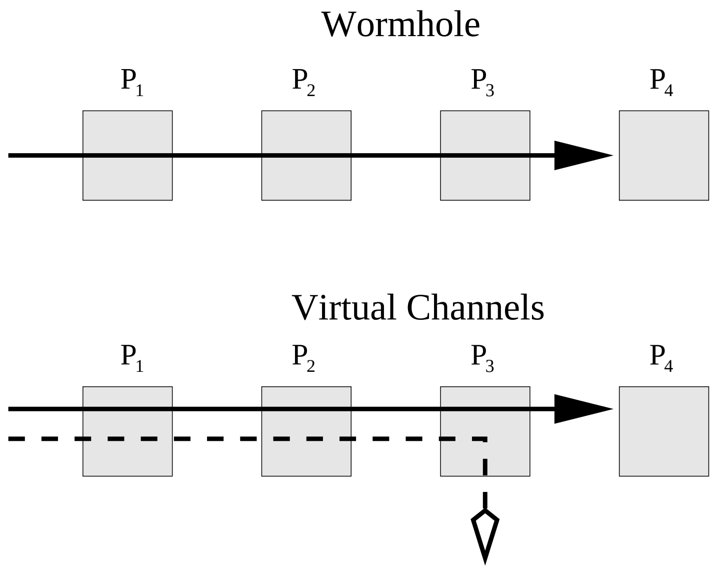

Figure 1: Wormhole flow control and virtual channels.

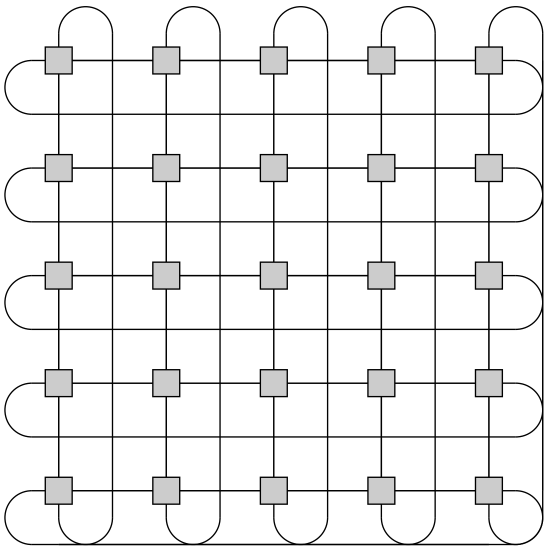

Figure 2: A 5-ary 2-cube.

3 Routing on the k-ary n-cubes

The toroidal meshes belong to the general class of the k-ary n-cube networks [3]. A k-ary n-cube is characterized by its dimension n and radix k, and has a total of k^n nodes. The k^n nodes are organized in an n-dimensional mesh, with k nodes in each dimension and wrap-around connections on the borders. The binary hypercube is a special case of k-ary n-cube with $k = 2$. Also, the two dimensional mesh is another special case with $n = 2$. Figure 2 shows an example of k-ary n-cube.

Routing algorithms on the toroidal k-ary n-cubes are deadlock-prone and require sophisticated strategies for deadlock-avoidance. In this paper we utilize two algorithms, each offering a different degree of adaptivity: a *deterministic* algorithm and minimal adaptive algorithm based on *Duato*'s methodology [8] [10].

The deterministic algorithm is a dimension order routing based on a static channel dependency graph [7]. Packets are sent to their destination along a unique minimal path, traversing the network dimensions in increasing order. The potential deadlocks caused by the wrap-around connections are avoided doubling the number of virtual channels and creating two distinct virtual networks. Packets enter the first virtual network and switch to the second one upon crossing a wrap-around connection. Our version uses four virtual channels for each physical link.

Rather than using a static channel dependency graph, Duato's methodology only requires the absence of cyclic dependencies on a connected channel subset. The remaining channels can be used in almost any way. We associate four virtual channels to each link: on two of these channels, called *adaptive* channels, packets are routed along any minimal path between source and destination. The remaining two channels are *escape* channels where packets are routed deterministically when the adaptive choice is limited by network contention [9]. A similar algorithm has been recently adopted by the Cray T3E [22].

A central point of our adaptive algorithm is the interface between the processor and the router. We assume that packets can enter the network using only a subset of the adaptive channels [19]. This limitation, known as *source throttling*, makes the network throughput stable when the network operates above saturation [5], which is the typical case when we execute a total-exchange algorithm.

4 Total-exchange algorithms

The algorithms that can be used to implement the total-exchange on a given network can be roughly classified into two classes:

- *direct* algorithms, in which data are sent directly from source to destination and
- *indirect* algorithms, in which data are sent from source to destination through one or more intermediate nodes.

The best direct algorithm for a hypercube architecture is the *pairwise* exchange algorithm, described in [15], as it guarantees no link contention at every step. This algorithm has also been shown to perform well on the fat tree architecture of the CM-5 [13]. It requires $N-1$ steps, where N is the number of nodes, and the communication schedule is as follows. In step i, $1 \leq i \leq N-1$, each node exchanges data with the node determined by taking the exclusive-or of its number with i. Therefore, this algorithm has the property that the entire communication pattern is decomposed into a sequence of pairwise exchanges. This algorithm generates conflicts on the two-dimensional meshes and tori and the maximum number of packets contending for a link at any step is limited by $k/2$ and $k/4$, respectively.

An algorithm that works for non power-of-two cubes is the *shift* exchange, and requires only $N-1$ steps for any value of N. In this algorithm node pairs do not exchange messages with each other. Instead, at step i, a node j sends data to node $(i+j) \bmod N$ and receives data from node $(N+j-i) \bmod N$.

In [25] it is shown an algorithm for wormhole-routed k-ary n-cubes that divides the transmission in a number of congestion-free phases. This algorithm uses an optimal number of phases if the arity of the cube is a multiple of eight or an asymptotically optimal number otherwise. It assumes that phases can be strictly routed in lockstep, a condition that cannot be easily achieved in a distributed environment. Also, the messages must be long enough to hide the initial delay of the potentially asymmetric message transmission in each phase.

In indirect algorithms a message is sent from the source node to the destination through one or more intermediate nodes. The *indirect pairwise* exchange algorithm [24] aims at reducing the the link contention of the pairwise exchange algorithm on the two-dimensional cubes. In this algorithm each node communicates only with the nodes in its row and column. Each exchange along a row is followed by a complete exchange along a column. Given L the message size in bytes, during the row exchange each node sends Lk bytes of data to the destination node, out of which $L(k-1)$ bytes are intended for other nodes in the same column as the destination node. This is followed by a complete exchange along the columns (involving messages of L bytes), in which data received during the row exchange is sent to the appropriate nodes in the same column. This operation requires $k(k-1)$ steps. Finally, an additional complete exchange is required along the columns for nodes to exchange their own data directly with nodes in the same columns.

5 A randomized total-exchange algorithm

The total-exchange in the k-ary n-cubes, as the uniform random traffic, is limited by the network bisection. The network capacity can be determined by considering that 50% of the traffic crosses the bisection of the network. Thus if a network has bisection bandwidth B, each of the N nodes can inject $2B/N$ traffic at the maximum load.

Our algorithm is based on an important property of the wormhole routed networks. In Figure we can see the saturation points, under uniform traffic, of a 256 nodes bi-dimensional torus, using the deterministic algorithm and the Duato's algorithm. The flit size is four bytes.

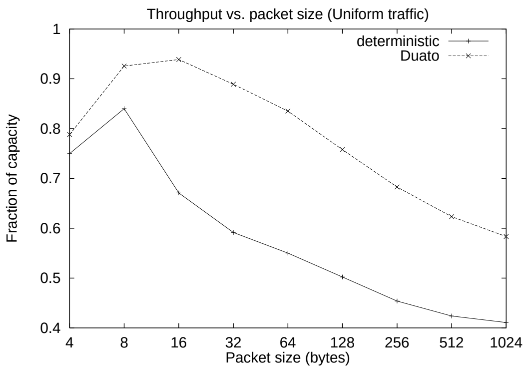

Figure 3: Network throughput varying the packet size.

We can see that the network throughput is very sensitive to the packet size.

Both algorithms prefer short packets. The adaptive algorithm reaches about 90% of the optimal performance with packets between eight and 32 bytes. The same happens for the deterministic algorithm. In this case we the have maximum throughput, around 80%, with eight bytes. Short packets reduce the network contention and increase the overall throughput. On the other hand, the injection and reception overhead of the processing nodes often suggests the use of larger packets.

While this property is well known in the wormhole routing community, it has not still been exploited to schedule collective communication patterns. A viable solution to implement the total-exchange is to synthetically reproduce inside the network a uniform random traffic, by properly choosing the packet size.

Starting from this observation, we propose a simple *randomized* algorithm to implement the total-exchange. In outline, given m the grain size of the total-exchange (i.e. the amount of information exchanged between any pair of nodes) and p the packet size, both expressed in bytes, we can logically schedule the transmission in $\lceil \frac{m}{p} \rceil$ steps. In each step i, $i \in \{0, \ldots, \lceil \frac{m}{p} \rceil - 1\}$:

1. each node j generates an independent permutation $\Pi_{i,j}$ of the remaining $N - 1$ nodes

2. and send the packets following the order suggested by the permutation.

Even if we have a sequence of steps, strict synchronization between processing nodes is not required, i. e. the processing nodes can proceed autonomously after the beginning of the communication pattern. This algorithm can be considered as complementary to the other algorithms in the literature, because it replaces a deterministic scheduling with a random scheduling. The nodes can proceed autonomously until the completion of the communication pattern.

6 Relevant details of the network model

This section presents a router model and a simulation environment, that are used in the following sections to analyze the performance of the exchange algorithms.

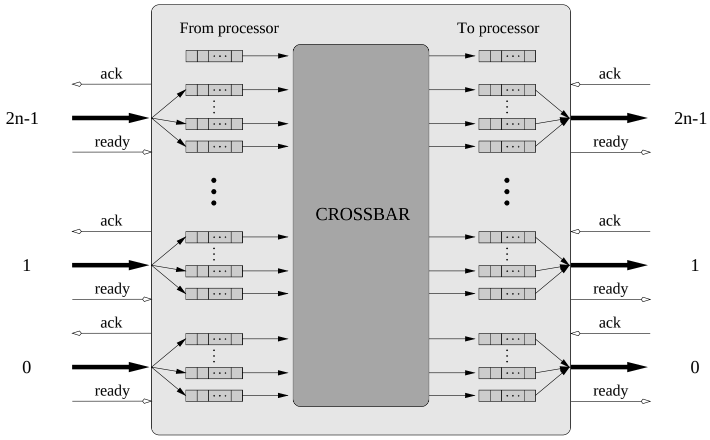

Figure 4: The internal structure of a routing switch.

Figure 4 outlines the internal structure of a routing switch. We can distinguish the external channels or links, the input and the output buffers or lanes that implement the buffer space of the virtual channels and an internal crossbar. The switch has bidirectional channels and each channel on the single direction is logically composed of three interfaces: a *data path* that transmits messages on a flit level, the *ready* lines that flag the presence of a flit on the data path and specify the virtual channel where the flit is to be stored and the *ack* lines in the reverse direction that send an acknowledgment every time buffer space is released in the input lanes. The processing nodes have a compatible interface with the same number of virtual channels.

A flit is moved from an output lane to the corresponding input lane in a neighboring node in T_{link} cycles, when there is at least a free buffer position. Each output lane has associated a counter that is initialized with the total number of buffers in the input lane, it is decremented after sending a flit and it is incremented upon receiving and acknowledgment. When multiple lanes are enabled, an arbiter picks one of them according to a fair policy.

When a header flit reaches the top of an input lane, the routing algorithm tries to establish a path in the crossbar with a suitable output lane, that is neither full nor bound to another input lane. This path will remain in action till the transmission of the tail flit of the packet. Our model allows the routing of a single header at a time every $T_{routing}$ cycles.

Although a physical link services in each direction at most one virtual channel every T_{link} cycles, multiple virtual channels can be active at the input and output ports of the crossbar. The internal flit propagation takes $T_{crossbar}$ cycles. Every time a flit is moved from an input lane to the corresponding output lane, a feedback is sent back to the neighboring switch or node to update the counter of free positions. In our experiments the three delays are equalized to a single clock cycle for both the deterministic and the minimal adaptive algorithm.

It is worth noting that the serial routing of the headers, the link arbitration and the deadlock avoidance strategies, typical characteristics of state-of-the-art routers, can break the symmetries present in a collective communication pattern. These important characteristics are often neglected in the performance analysis of the total-exchange algorithms. On the other hand, the network behavior under uniform random traffic pattern, shown in section 5, is not sensitive to these low-level aspects.

This model is evaluated in the SMART (Simulator of Massive ARchitectures and Topologies) environment [20]. Implemented in C++, SMART is an object-oriented discrete-event simulation tool for evaluating massively parallel architectures. Configuring some shell scripts, it is possible to select the network topology and the internal router policies. The simulator allows the definition of the packet length, number of virtual channels and buffers for both input and output lanes. Also, it is possible to monitor several metrics and time-dependent events, that are gathered in trace files.

7 Experimental results

We have implemented four total-exchange algorithms in the SMART simulation environment. They are the shift exchange, the pairwise exchange, the indirect pairwise exchange and our randomized algorithm. Figure 5 shows the performance of these algorithms on a 256 nodes bi-dimensional torus using the deterministic and the Duato's algorithms. We use two graphs for each routing algorithm. On the first one we show the execution time of the total-exchange algorithms and on the second graph we relate the throughput achieved at the end of the total-exchange with the bisection bandwidth. In all the graphs the input size on the x-axis represents the grain size, in bytes, of the information exchanged between any pairs of nodes. So, in order to obtain the global size of information exchanged we have to multiply for the square of the number of nodes. The graphs in Figure 5 a) and c) report a lower bound, that is computed by considering the usual topological limitation of the bisection bandwidth. In fact, half of the packets have to cross the network bisection.

For small message sizes, when we use the deterministic routing, the random exchange provides the best performance. For messages larger than 16 bytes, the random and the pairwise exchange have a similar performance. The shift and the indirect pairwise are not efficient. The shift exchange generates a type of non-uniform traffic that is not handled properly. For the indirect pairwise the loss of performance is caused by the larger packet size in the row exchange. In fact the saturation throughput decreases when we increase the packet size, as can be seen in Figure 5 b).

Things change when we use the Duato's algorithm. There is a gap between the randomized algorithm, that is very close to optimality with 16 and 32 bytes and the other algorithms. In Figure 5 d) we can see that the fraction of network throughput achieved at completion of the total-exchange is around 90% with medium sized packets. The performance of the other algorithms is between 30% and 45%.

Figure 6 provides more insight on the characteristics of the routing algorithms. It shows the network utilization, i. e. the fraction of active links, during the execution of the four total-exchange algorithms. The toroidal cube has the property that with the total-exchange (and uniform traffic in general) all the links are active when the network operates at capacity. In fact, all the $2k^{n-1}$ links of the bisection are active on both directions. Assuming that messages are long enough, each message occupies on the average a number of links equal to the average distance d_m. Thus the number of active links (in the single direction) $Links$ is

$$Links = 8d_m k^{n-1} = 2nk^n \tag{1}$$

that is exactly the number of links in the networks, because each node has $2n$ outgoing unidirectional links. From this result follows that network utilization and network throughput are strongly related.

The network utilization of the shift exchange algorithm, shown in Figure 6 a),

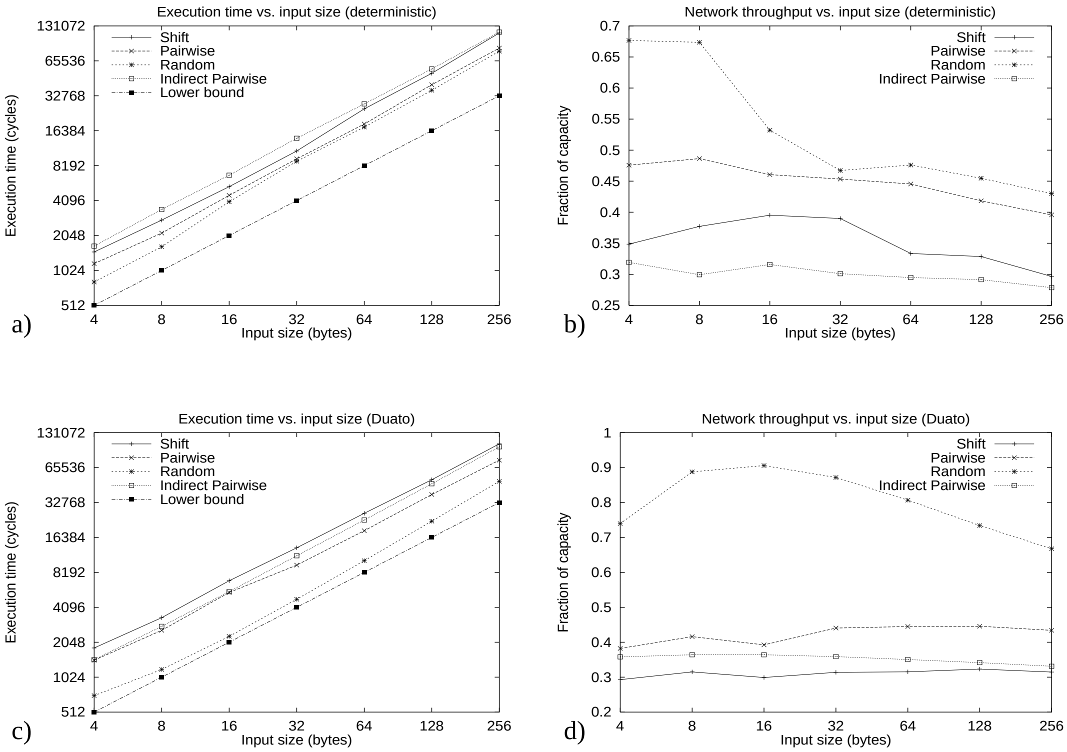

Figure 5: Execution time and fraction of the network capacity achieved at completion of the total-exchange.

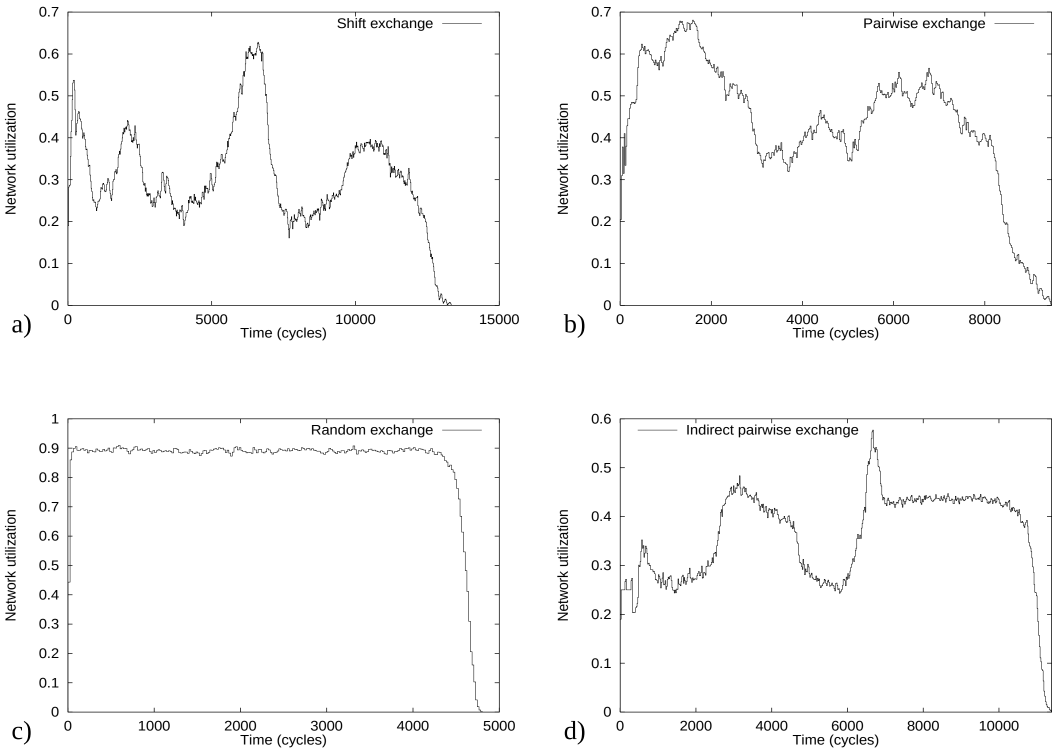

Figure 6: Network utilization during the execution of the four exchange algorithms with the Duato's routing. The packet size is 32 bytes.

alternates spikes that reach 60% with periods that fall down to 20%. The same problem exists in the initial part of the indirect pairwise algorithm. In the final part, after 7000 cycles, the communication is concentrated along the columns, using only half of the links and the network utilization converges to 45%. The pairwise exchange reduces the execution time if compared to the previous two algorithms, even if the network utilization still oscillates between 70% and 35%.

The near-optimal performance of the randomized algorithm can be explained looking at Figure 6 c). This algorithm exploits some basic characteristics of the uniformly distributed traffic with fixed packet size. When the grain size is 32 bytes,

1. the network reaches the steady state of 90% active links after very few cycles (just 25 cycles in the example);

2. once in the steady state, the network utilization and throughput are stable with small fluctuations (less than 2%): a high percentage of links are used in a profitable way;

3. at the end of the steady state packets are consumed by the network in a short period (400 cycles).

When the grain size is larger that 32 bytes, we can organize the exchange algorithm in a sequence of sweeps, each using the smaller grain size and another independent permutation. We pay the inefficiency of the initial and final periods just once, achieving 90% of the throughput. The execution time can be easily estimated by dividing the total amount of information for the steady state bandwidth. The optimal packet size is a peculiar characteristic of the routing algorithm in use. For example the deterministic and the Duato's algorithms prefer short packets, while the Chaos routing, a non minimal cut-through variant of the hot-potato routing not shown here for brevity, works well with larger packets [1].

The other algorithms, that impose a deterministic scheduling, clashes against the flow control characteristics and the router internal organization and are not very robust in practice.

These results suggest a methodology to implement all the algorithms where each node sends and receives the same amount of information on the cubes: rather than scheduling in detail the communication pattern, we can look at the problem from the flow control point of view. Each routing algorithm has a particular fingerprint in terms of network throughput with uniform traffic which tells us the maximum throughput that can be achieved for any packet size. By properly choosing a convenient size and by randomizing the message transmission we can exploit the network performance in a simple, efficient, robust and predictable way.

8 Conclusion

Many studies in the literature use a deterministic approach to structure global communication patterns [12] [16]. These studies often neglect low level aspects, as deadlock avoidance and flow control, that are very important to achieve good performance. They often are not very robust in practice, e. g. because the processing nodes loose synchronization.

We have shown that a simple randomized algorithm that exploits the low level communication characteristics of the interconnection network can be more efficient and robust than the traditional approaches on the toroidal cube. This algorithm utilizes an interesting, and not previously used, property of the interconnection network that, with the uniform traffic,

- reaches a steady state after few cycles (about 25),
- can get a stable and high throughput (90% of the capacity) in the steady state, with oscillations in the network utilization/throughput that are less than 2%,
- and is drained in few hundred cycles at the end of the communication pattern.

This scheme has several interesting features. First, it is very simple and requires no additional hardware to synchronize the processing nodes. It also exploits the characteristics of adaptive routing, which will eventually replace the deterministic routers that are currently in use in many existing multicomputers. Finally, it paves the way to a new strategy to efficiently implement dense collective communication pattern other that the total-exchange, as broadcast or personalized communication.

References

[1] Kevin Bolding. *Chaotic Routing: Design and Implementation of an Adaptive Multicomputer Network Router*. PhD thesis, University of Washington, Department of Computer Science and Engineering, Seattle, WA, July 1993.

[2] J. Bruck, C. T. Ho, S. Kipnis, and D. Weathersby. Efficient Algorithms for All-to-All Communications in Multi-Port Message-Passing Systems. In *Proceedings of the 6th ACM Symposium on Parallel Architectures and Algorithms*, pages 298–309, 1994.

[3] William J. Dally. Performance Analysis of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers*, 39(6):775–785, June 1990.

[4] William J. Dally. Virtual Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[5] William J. Dally and Hiromichi Aoki. Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–475, April 1993.

[6] William J. Dally and Charles L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1:187–196, 1986.

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

[8] José Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[9] José Duato. A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks. In *International Conference on Parallel Processing*, volume I - Architecture, pages I–142–I–149, 1994.

[10] José Duato. A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(10):1055–1067, October 1995.

[11] José Duato and Pedro López. Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes. In Kevin Bolding and Lawrence Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 45–59, Seattle, Washington, USA, May 1994.

[12] Pierre Fraigniaud and Joseph G. Peters. Structured Communication in Torus Networks. In *Proceedings of the 28th Hawaii Conference on System Science*, 1995.

[13] Steve Heller. Congestion-Free Routing on the CM-5 Data Router. In Kevin Bolding and Lawrence Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 176–184, Seattle, Washington, USA, May 1994.

[14] S. Hinrichs, C. Kosak, D O'Hallaron, T. M. Stricker, and R. Take. An Architecture for All-to-All Personalized Communication. In *Proceedings of the 6th ACM Symposium on Parallel Architectures and Algorithms*, pages 310–319, 1994.

[15] S. L. Johnsson and C. T. Ho. Optimal Broadcsting and Personalized Communication in Hypercubes. *IEEE Transactions on Computers*, 38:1249–1268, 1989.

[16] Ben H. H. Juurlink, P. S. Rao, and Jop F. Sibeyn. Worm-Hole Gossiping on Meshes. In *Second International Euro-Par Conference, Volume I*, number 1123 in LNCS, pages 361–369, Lyon, France, August 1996.

[17] P. Kermani and L. Kleinrock. A Tradeoff Study of Switching Systems in Computer Communication Networks. *IEEE Transactions on Computers*, C-29(12):1052–1060, December 1980.

[18] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.

[19] Fabrizio Petrini and Marco Vanneschi. Minimal Adaptive Routing with Limited Injection on Toroidal k-ary n-cubes. In *Supercomputing 96*, Pittsburgh, PA, November 1996.

[20] Fabrizio Petrini and Marco Vanneschi. SMART: a Simulator of Massive ARchitectures and Topologies. In *International Conference on Parallel and Distributed Systems Euro-PDS'97*, Barcelona, Spain, June 1997.

[21] Satish Rao, Torsten Suel, Thanasis Tsantilas, and Mark Goudreau. Efficient Communication Using Total-Exchange. In *Proceedings of the 9th International Parallel Processing Symposium, IPPS'95*, Santa Barbara, CA, April 1995.

[22] Steven L. Scott and Gregory M. Thorson. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. In *HOT Interconnects IV*, Stanford University, August 1996.

[23] D. B. Skillicorn, Jonathan M. D. Hill, and W. F. McColl. Questions and Answers about BSP. *Journal of Scientific Programming*, 1998.

[24] Rajeev Thakur and Alok Choudary. All-to-All Communication on Meshes with Wormhole Routing. In *Proceedings of the 8th International Parallel Processing Symposium, IPPS'94*, pages 561–565, Cancun, Mexico, April 1994.

[25] Yu-Chee Tseng and Sandeep K. S. Gupta. All-to-All Personalized Communication in a Wormhole-Routed Torus. *IEEE Transactions on Parallel and Distributed Systems*, 7(5):498–505, May 1996.