Powerful and Feasible Processor Interconnections With an Evaluation of Their Communications Capabilities *

Qian Wang and **Sotirios G. Ziavras**
Departments of Electrical and Computer Engineering,
and Computer and Information Science
New Jersey Institute of Technology
Newark, NJ 07102

Address for Correspondence
Professor Sotirios G. Ziavras
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, New Jersey 07102

Phone: (973) 596-5651
Fax: (973) 596-5680
Email: ziavras@oak.njit.edu

*The work presented in this research was supported in part jointly by NSF and DARPA under the New Millennium Computing Point Design Grant ASC-9634775.

Abstract

The majority of existing interprocessor connection networks are plagued by poor topological properties that result in large memory latencies for DSM (Distributed Shared-Memory) computers. On the other hand, scalable networks with very good topological properties are often impossible to build because of their prohibitively high VLSI (e.g., wiring) complexity. Such a network is the generalized hypercube (GH). The GH supports full-connectivity of all its nodes in each dimension and is characterized by outstanding topological properties. Also, low-dimensional GHs have very large bisection widths. We propose here a new class of scalable interprocessor connections, namely HOWs (Highly-Overlapping Windows), capable of lower complexity than GHs and comparable performance. HOWs are obtained from GHs by uniformly removing edges to produce feasible systems of lower wiring complexity. Resulting systems contain numerous highly-overlapping GHs of smaller size. Also, the classical GH belongs to this new class of interconnections. We analyze the communications capabilities of 2-D HOWs and demonstrate that in practical cases they perform much better than binary hypercubes for important communications patterns. They also perform comparably to GHs. We develop algorithms for one-to-one unicasting, one-to-all broadcasting, all-to-all broadcasting, one-to-all personalized, and all-to-all personalized communications, under various communications models.

1 Introduction

The demand for ever greater performance by many computation problems has been the driving force for the development of computers with hundreds or thousands of processors. The massively-parallel processing field is currently dominated by: (a) high-level parallel languages supporting a shared address space (for DSM computers), and (b) point-to-point interconnection networks (for workstation-like nodes). Near PetaFLOPS (i.e., 10^{15} FLOPS) and more performance is required by many applications. However, this objective seems to be a distant dream primarily because of the, as currently viewed, unsurmountable difficulty in developing low-complexity, high-bisection bandwidth, and low-latency networks to interconnect thousands of processors.

Several interconnection networks have been proposed for massively-parallel computers, including, among others, regular meshes and tori, enhanced meshes [11], (direct binary) hypercubes, and hypercube variations [1] [6]. The hypercube dominated the high-performance computing field in the 1980's because it has good topological properties and rather rich interconnectivity that permits efficient emulation of many topologies frequently employed in the development of algorithms [8]. Nevertheless, these properties come at the cost of often prohibitively high VLSI (primarily wiring) complexity due to a dramatic increase in the number of communication channels with any increase in the number of PEs (processing elements). The high VLSI complexity is undoubtedly its dominant drawback, that limits scalability [8] and does not permit the construction of powerful, massively-parallel systems.

To support scalability, current approaches to massively-parallel processing use bounded-degree networks, such as meshes or k-ary n-cubes (i.e., tori), with low node degree. However, low-degree networks result in large diameter, large average internode distance, and small bisection width. The high VLSI complexity problem is unbearable for generalized hypercubes (GHs). Contrary to nearest-neighbor k-ary n-cubes that form rings with k nodes in each dimension, GHs implement fully-connected systems with k nodes in each dimension [10]. The n-D (symmetric) generalized hypercube $GH(k,n)$ contains k^n nodes. The address of a node is $x_{n-1}x_{n-2}...x_1x_0$, where x_i is a radix-k digit with $0 \leq x_i \leq k-1$. This node is a neighbor to the nodes with addresses $x_{n-1}x_{n-2}...x'_i...x_1x_0$, for all $0 \leq i \leq n-1$ and $x'_i \neq x_i$. Therefore, two nodes are neighbors if and only if their n-digit addresses differ in a single digit. For the sake of simplicity, we restrict our discussion to symmetric generalized hypercubes where the nodes have the same number of neighbors in all dimensions. Therefore, each node has $k-1$ neighbors in each dimension, for a total of $n \cdot (k-1)$ neighbors per node. The n-D $GH(k,n)$ has diameter equal to only n. For $n=2$ and k an even number, the diameter of the GH is only 2 and its bisection width is the immense $k^3/4$. The increased VLSI/wiring cost of GHs results in outstanding performance that permits optimal emulation of hypercubes and k-ary n-cubes, and efficient implementation of complex communications patterns [3] [2] [13].

To reduce the number of communications channels in the generalized hypercube, the spanning bus hypercube uses a shared bus for the implementation of each fully-connected subsystem in a given dimension [6]. However, shared buses result in significant performance degradation because of the overhead imposed by the protocol that determines each time ownership of the bus. Similarly, hypergraph architectures implement all possible permutations of their nodes in each dimension by employing crossbar switches or optical networks [5] [12]. Reconfigurable generalized hypercubes interconnect all nodes in each dimension dynamically via a scalable mesh of very simple, low-cost programmable switches [9]. However, all these proposed reductions in hardware complexity may not be sufficient for very high performance computing.

To summarize, low-dimensional massively-parallel computers with full node connectivity in each dimension, such as generalized hypercubes, are very desirable because of their outstanding topological properties, but their electronic implementation is a Herculean task because of packaging (primarily wiring) constraints. We propose in this paper a new class of architectures, namely HOWs (Highly-Overlapping Windows), that can be implemented effectively with current technology, contain many highly-overlapping GHs, are scalable, and possess outstanding topological properties. HOWs require fewer and shorter interconnections between processors than GHs, and therefore their construction may be feasible.

Section 2 introduces HOWs. Sections 3 and 4 propose and analyze popular communications operations for 1-D and 2-D HOWs, respectively; relevant comparisons with regular GHs are also included. Finally, Section 5 presents performance and cost comparisons of binary and generalized hypercubes, and 2-D HOWs.

2 HOW Architectures

We first introduce 1-D HOWs. $HOW(p,w,1)$ denotes a 1-D HOW system with p nodes and window size w. In this system, each node with unique address k, where $0 \leq k \leq (p-1)$, is directly connected to all nodes with addresses $0 \leq k \pm i \leq (p-1)$, for $i = 1,2,3,...,w$. Each processor(node) has up to $2 \cdot w$ neighbors. Therefore, all connections could be short and rather local; the length of channel wires depends heavily on the value of w. The same configuration is also obtained by removing edges from the 1-D fully-connected system (a GH). The n-D $HOW(p,w,n)$ is constructed recursively, so that each node can have up to $2 \cdot w \cdot n$ neighbors. This system contains many generalized hypercubes $GH(w+1,n)$. Also, it can be derived from the $GH(p,n)$ by uniformly removing some edges. Figure 1 shows examples of 2-D HOW systems.

Table 1 compares the numbers of channels in the binary hypercube (i.e., m-cube), the k-ary n-cube (i.e., n-D torus), the generalized hypercube $GH(k,n)$, the 2-D $HOW(2^{\frac{m}{2}},w,2)$, and the n-D $HOW(k,w,n)$, all with the same number N of processors. We assume bidirectional data channels for full-duplex communications, and that $N = k^n = 2^m$ (therefore, $k = N^{1/n} = 2^{m/n}$). This paper focuses on 2-D HOW systems because of their simplicity, high bisection width, and ease of implementation.

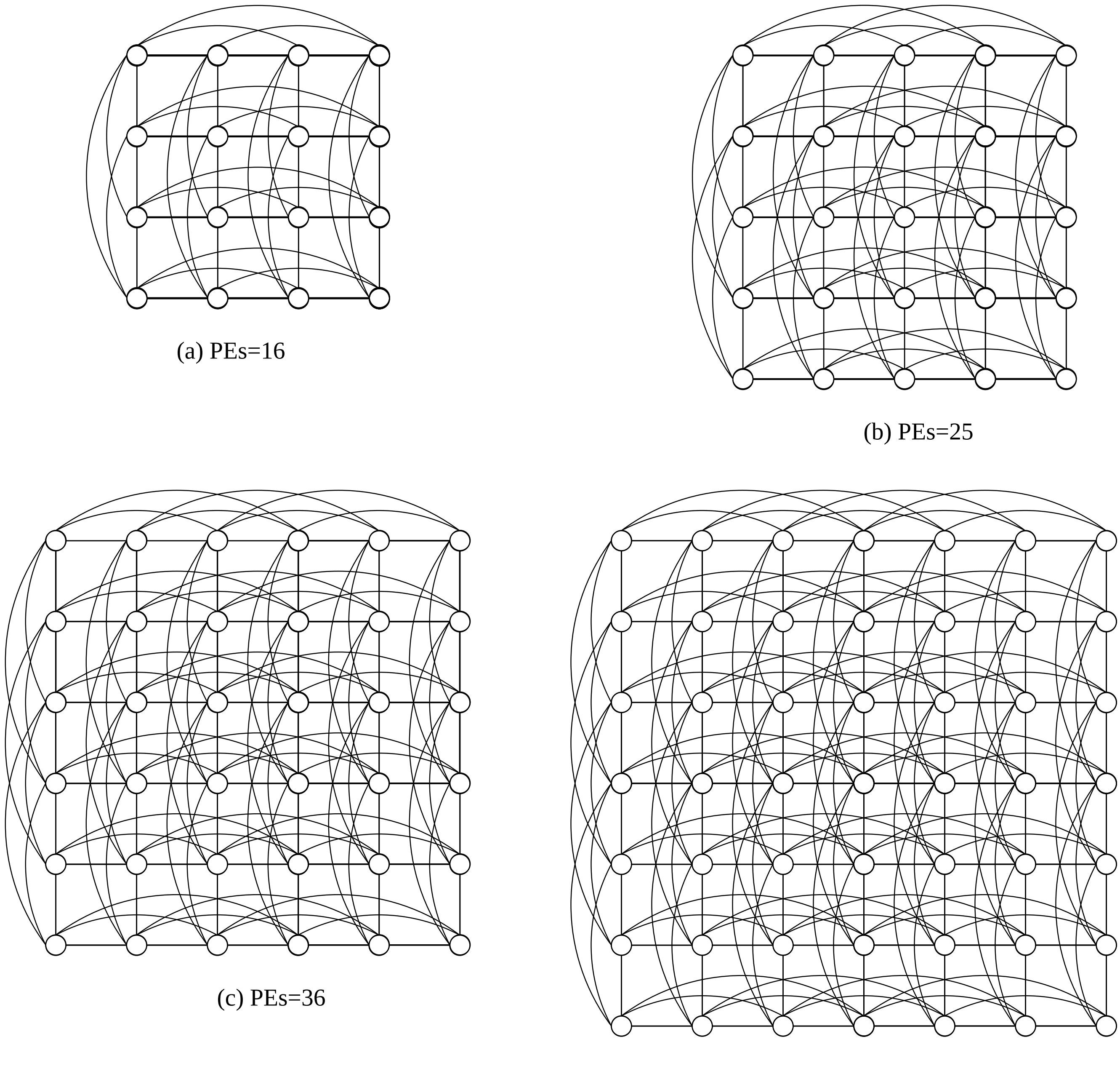

Figure 1: Examples of 2-D HOW systems with window size $w = 3$. (a) $HOW(4,3,2)$. (b) $HOW(5,3,2)$. (c) $HOW(6,3,2)$. (d) $HOW(7,3,2)$.

HOWs have reduced pin-outs compared to GHs. Also, the wiring of HOW channels requires fewer VLSI layers than GHs if the number of wire crossovers has to be minimized. Therefore, HOWs are more feasible to implement than GHs. Their scalability is supported by current and expected electronic and/or optical technologies; free-space optics can support efficient broadcasting of information within local windows of processors. In contrast, binary hypercubes are not scalable [8] and are more difficult to build because of long wires. HOWs also improve dramatically the topological properties of k-ary n-cubes; the $HOW(k, 1, n)$ is the n-D mesh. We also show later in this paper that HOW systems perform much better than binary hypercubes for many demanding communications operations.

Network	Number of channels	Diameter
m-cube	$m * 2^m$	$m = n * \log_2 k$
k-ary n-cube	$2 * n * 2^m$	$n * \lfloor \frac{k}{2} \rfloor$
$GH(k, n)$	$(k-1) * n * 2^m$	n
$HOW(2^{\frac{m}{2}}, w, 2)$	$2^{\frac{m}{2}+1} * w * (2 * k - w - 1)$	$2 * \lceil \frac{k-1}{w} \rceil$
$HOW(k, w, n)$	$n * k^{n-1} * w * (2 * k - w - 1)$	$n * \lceil \frac{k-1}{w} \rceil$

Table 1: Comparison of interconnection networks, assuming full-duplex bidirectional data channels.

3 Communications Operations on 1-D HOW Systems

Our focus is 2-D HOWs. However, for the sake of simplicity we first develop communications routines for 1-D HOWs. Before we propose algorithms for implementing various communications operations, some definitions are pertinent to evaluate their performance. The *communication latency*, that is the time taken to communicate a message between two processors in the system, depends on the following parameters [4]:

- *Startup time (t_s)*: the time consumed by the sending processor. It comprises the time to prepare the message (producing the header, trailer, and error correction information), the time to run the routing algorithm at the source, and the time to send the first part of the message to the appropriate communication port.

- *Per-word transfer time (t_w)*: the time taken by a word to traverse a channel. If the channel bandwidth is b words per second, then each word takes time $t_w = 1/b$.

- *Combining time (t_c)*: the time consumed by an intermediate node to switch a message from an input to an output port; it also includes the time to combine incoming messages, if needed, and send them to the appropriate output port.

We calculate only the time taken by a message to reach the input port of the destination. Additional time may be needed to get the data from that port. In **store-and-forward (SF) routing**, with a message traversing a path with multiple links, each intermediate processor forwards the message to the next processor in the path after it has received the entire message. To increase the utilization of communication resources and reduce communication time, **wormhole routing** divides a message into **flits** (**flow-control digits**). As the header flit advances along the chosen path, the remaining flits follow in the same path in pipelined fashion. If the header flit encounters a channel already in use, this flit is blocked until the channel becomes available [4]. Normally, the flit size coincides with the channel width. The combining time t_c is ignored in wormhole routing.

We develop algorithms under three communications models. For all of the models, each processor can receive more than one message at a time in different input ports. These models differ in how they can use their output ports.

- **Model-1**: Each processor can use only one output port at a time.
- **Model-2**: Each processor can use multiple output ports simultaneously, as long as all output ports contain the same value.
- **Model-3**: Each processor can use multiple output ports simultaneously, and different output ports can have different values.

In the following subsections we develop algorithms for various communications operations on 1-D HOW systems and derive corresponding execution times for the aforementioned models. The analysis is done each time for SF and wormhole routing, in this order. These operations are very frequently used in parallel processing [7] [3].

3.1 One-to-One Communication

This basic operation sends a message from one processor to another.

With SF routing, sending a single message containing m words takes $t_s + mt_w l + t_c(l-1)$ time, where l is the number of links traversed by the message. For a 1-D HOW system with p processors and window size w, l is at most $\lceil \frac{p-1}{w} \rceil$, and therefore the time for a single message transfer has the *upper bound* of

$$T_{one\_to\_one} = t_s + mt_w \lceil \frac{p-1}{w} \rceil + t_c(\lceil \frac{p-1}{w} \rceil - 1) = O(m\frac{p}{w})$$

assuming no contention with other messages at intermediate processors.

With wormhole routing, assume that the flit is one word, and therefore the flit transfer time is t_w. If the message traverses l links, then the header of the message takes $t_s + lt_w$ time to reach the destination. If the message is m words long, then the remaining flits will reach the destination in time $(m-1)t_w$ after the arrival of the header. Therefore, the *upper bound* is

$$T(WR)_{one\_to\_one} = t_s + t_w \lceil \frac{p-1}{w} \rceil + (m-1)t_w = O(m + \frac{p}{w})$$

3.2 One-to-All Broadcasting

One-to-all broadcasting is an operation where a single processor sends the same data of m words to all other processors. Initially, only the source processor has the data of size m that needs to be broadcast. At the termination of the procedure, there are $p-1$ copies of the initial data, one copy residing in each of the other processors. The naive way to perform one-to-all broadcasting is to sequentially send $p-1$ messages from the source to the other $p-1$ processors. For the sake of efficiency, every processor could keep a copy of the message it receives from a neighbor, and then could forward this message to one or more of its other neighbors.

3.2.1 Model-1

Since there is only one output port "available" for each processor at each transfer step, we consider two different stages. We assume that the leftmost processor is the source, for worst case timing. In the first stage, we copy the data to all processors (PEs) in the source's window of size w. In the second stage, the data in the leftmost window is propagated to the right, one window size at a time.

We introduce two parameters here: s_1 represents the number of transfer steps needed to fill the first window, and s_2 represents the number of transfer steps needed in the second stage to copy the values in the first window into the remaining windows. In the first stage, the propagation doubles each time the number of PEs that receive the message, and therefore the processors within the window are assumed to form a binary tree. We have the following relations among s_1,s_2, and w.

$$\begin{aligned} s_1 &= \lceil \log(w+1) \rceil \\ s_2 &= \lceil (p - 2^{s_1})/w \rceil \end{aligned}$$

All logarithms in this paper are in the base 2. The communication time for one-to-all broadcasting under model-1 and SF routing has the *upper bound* of

$$T_{one\_to\_all,1} = \begin{cases} t_s + mt_w\lceil \log p \rceil + t_c(\lceil \log p \rceil - 1) = O(m \log p) & \text{if } (p-1) \le w \\ t_s + mt_w(s_1 + s_2) + t_c(s_1 + s_2 - 1) = O(m \log w + m\frac{p}{w}) & \text{if } (p-1) > w \end{cases}$$

This asymptotic time is optimal.

With wormhole routing, the *upper bound* is

$$T(WR)_{one\_to\_all,1} = \begin{cases} t_s + t_w\lceil \log p \rceil + (m-1)t_w = O(m + \log p) & \text{if } (p-1) \le w \\ t_s + t_w(s_1 + s_2) + (m-1)t_w = O(m + logw + \frac{p}{w}) & \text{if } (p-1) > w \end{cases}$$

assuming that incoming data can simultaneously be stored locally and also be transferred to the next PE in the path.

3.2.2 Model-2 and Model-3

For one-to-all broadcasting, there is only one value to be sent, and therefore the procedures for this operation are identical under model-2 and model-3. Assume the leftmost PE as the source. Model-2 is not inferior to model-3 because up to w output ports are "available" to the right of each processor at each transfer step as long as these ports transfer the same value, which is the case here. The first stage now consumes one transfer step and the total number of transfer steps is $\lceil (p-1)/w \rceil$. The communication time has the *upper bound* of

$$T_{one\_to\_all,2} = t_s + mt_w\lceil \frac{p-1}{w} \rceil + t_c(\lceil \frac{p-1}{w} \rceil - 1) = O(m\frac{p}{w})$$

This asymptotic time is optimal.

With wormhole routing, the *upper bound* is

$$T(WR)_{one\_to\_all,2} = t_s + t_w\lceil \frac{p-1}{w} \rceil + (m-1)t_w = O(m + \frac{p}{w})$$

3.3 All-to-All Broadcasting

In all-to-all broadcasting, which is a generalization of one-to-all broadcasting, all p processors simultaneously initiate a broadcast. A processor sends the same m-word message to every other processor, but different processors may broadcast different messages.

3.3.1 Model-1

For model-1, there is only one output port of each processor we can use at a time. To let every processor pass information to a neighbor in each step, we deliberately choose those channels that form a ring, as shown in Figure 2. If communication is performed circularly in a single direction, then each processor receives all $(p-1)$ pieces of information from all other processors in $(p-1)$ steps. The time taken by the entire operation is

$$T_{all\_to\_all,1} = t_s + mt_w(p-1) + t_c(p-2) = O(mp)$$

This asymptotic time is optimal because each processor can use only one output port at a time, and therefore each message must make $p-1 = O(p)$ hops.

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all,1} = t_s + mt_w(p-1) = O(mp)$$

because the header of each message is blocked at each intermediate node until the previous message has completely departed.

3.3.2 Model-2

The broadcasting procedure follows:

- First stage: Each PE sends its message to all of its neighbors.
- Remaining stages: Assume the stage i, where $i = 1, 2, ..., \lceil \frac{p-1}{w} \rceil - 1$. In one direction, beginning from position iw and also involving all its successors, send the messages from the PEs $0, 1, ..., (p-1-iw-1)$ through all possible channels. In the other direction, beginning from position $(p-1-iw)$ and also involving all its predecessors, send the messages from the PEs $p-1, p-2, ..., (iw+1)$. If there is an overlap between these two directions, then split this stage into two steps in order to make sure that every PE sends just one value at a time. From all the messages it contains, each time a PE sends out the message received earlier from its most distant PE.

The total time taken by this operation is

$$T_{all\_to\_all,2} = t_s + mt_w(\lceil \frac{p-1}{w} \rceil + x) + t_c(\lceil \frac{p-1}{w} \rceil + x - 1)$$

where x is the number of stages needed to be split into two steps, and x should satisfy the condition $xw < p - 1 - xw$. So x is the largest integer less than $\frac{p-1}{2w}$. Therefore,

$$T_{all\_to\_all,2} = O(m\frac{p}{w})$$

This asymptotic time is optimal because the diameter of the system is $O(\frac{p}{w})$.

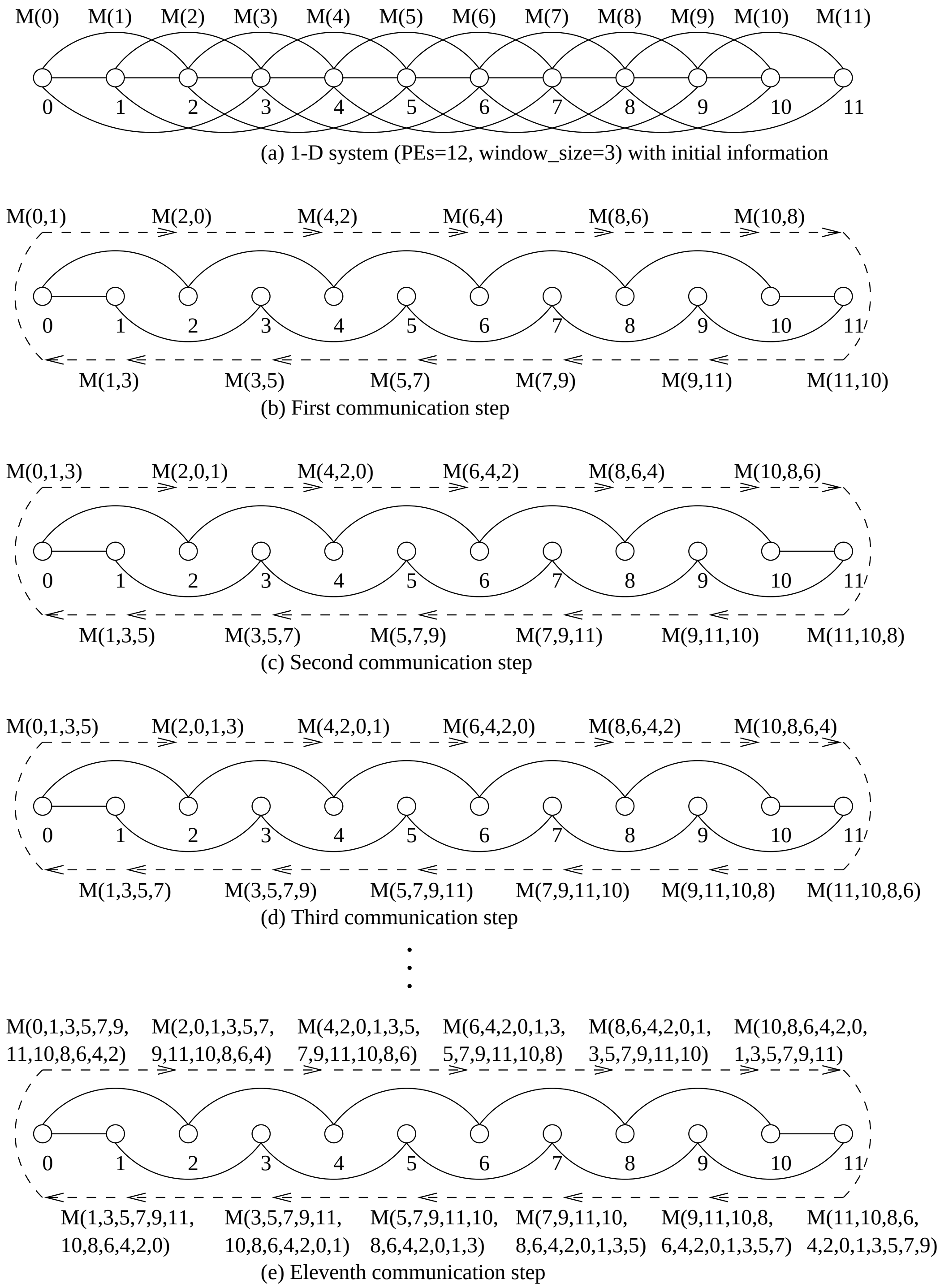

Figure 2: All-to-all broadcasting under model-1 with 12 processors and window size of 3. The numbers in parentheses for each processor are the labels of source processors from which data was received prior to the current communication step.

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all,2} = t_s + mt_w(\lceil \frac{p-1}{w} \rceil + x) = O(m\frac{p}{w})$$

because of message blocking on reused channels.

3.3.3 Model-3

This procedure is very similar to that for model-2. Since each individual processor can send different messages at the same time, we do not need to split any step. The total time taken by this operation is optimal and given by

$$T_{all\_to\_all,3} = t_s + mt_w\lceil \frac{p-1}{w} \rceil + t_c(\lceil \frac{p-1}{w} \rceil - 1) = O(m\frac{p}{w})$$

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all,3} = t_s + mt_w\lceil \frac{p-1}{w} \rceil = O(m\frac{p}{w})$$

3.4 One-to-All Personalized Communication

One-to-all personalized communication is an operation where the source processor sends $(p-1)$ unique messages, each one destined for a different processor in the system. Unlike one-to-all broadcasting, one-to-all personalized communication does not involve any duplication of data. However, the communications patterns for one-to-all broadcasting and one-to-all personalized communications are identical; only the sizes and contents of messages are different.

3.4.1 Model-1 and Model-2

Even though under model-2 each processor has multiple outports available in each step, all the outports are supposed to transport the same message. But for one-to-all personalized communication, the source processor has different messages to be transmitted. In this case, the communications procedures are exactly the same for both model-1 and model-2. For these two models, no matter what the window size is, it will take $(p-1)$ transfer steps for this communications operation. A ring structure is used to communicate values, as shown in Figure 2. Messages going farther have higher priority of transmission. The total time taken by this operation is

$$T_{one\_to\_all\_pers,1} = t_s + mt_w(p-1) + t_c(\lceil \frac{p-1}{2} \rceil - 1) = O(mp)$$

This is similar to the asymptotic time consumed by the source, and therefore it is optimal. The shortest paths in the ring are chosen to reach respective destinations. For the sake of simplicity, assume that the source is p_0. To reach the PE p_x, where $1 \leq x \leq (p-1)$, the message makes $\lceil \frac{x}{2} \rceil$ hops. Assume that the source first sends out the messages destined for the odd-numbered PEs. It then transmits messages to the even-numbered PEs. Assume for the second case the PE p_x with $x = 2y$. This PE will receive its message with delay $t_c(y-1) + mt_w(y-1)$ after it was transmitted by the source. The time left for the source to complete the entire operation is $mt_w(y-1)$, because $(y-1)$ is the number of messages still to be transmitted. Therefore, the "combining time" term used in the equation is for the worst case, where $y = \lceil \frac{p-1}{2} \rceil$.

With wormhole routing, the total number of flits to be transferred by the source is $(p-1)m$. Messages going farther have higher priority of transmission. The communication time is

$$T(WR)_{one\_to\_all\_pers,1} = t_s + mt_w(p-1) = O(mp)$$

This also represents the time consumed by the source because of the pipelining of messages and the chosen priority for message transmission.

3.4.2 Model-3

Under model-3, the one-to-all personalized communication operation can be done as follows. For the worst case, we assume p_0 to be the source:

- First, the processor p_0 passes the w most distant messages to its w neighbors, so that a destination processor with higher address gets a message for a higher-addressed processor.
- Second, the processor p_0 similarly passes the next w most distant messages to its window, while all processors that received an intermediate message earlier pass that message to their neighbor at distance w in the next window (i.e., window to their right).
- The second step repeats until all processors receive their own message.

The total time taken by this operation is

$$T_{one\_to\_all\_pers,3} = t_s + mt_w\lceil\frac{p-1}{w}\rceil + t_c(\lceil\frac{p-1}{w}\rceil - 1) = O(m\frac{p}{w})$$

which has the same asymptotic complexity with the time consumed by the source, and therefore it is optimal.

With wormhole routing, all processors receive their messages simultaneously in time $t_s + mt_w\lceil\frac{p-1}{w}\rceil$, because of message pipelining and message blocking resulting from the m-flit messages. Therefore, the total communication time is

$$T(WR)_{one\_to\_all\_pers,3} = t_s + m\lceil\frac{p-1}{w}\rceil t_w = O(m\frac{p}{w})$$

which is again optimal because it is identical to the time consumed by the source with peak utilization of its communication ports and no data duplication.

3.5 All-to-All Personalized Communication

In all-to-all personalized communication, also known as *total exchange*, each processor sends a distinct message of size m to every other processor. Unlike all-to-all broadcasting, all-to-all personalized communication does not involve any duplication of data.

3.5.1 Model-1 and Model-2

For all-to-all personalized communication, the source processor has different messages to be transmitted. Even though model-2 has multiple outports available, all the outports are supposed to transport the same message. Therefore, the communications procedures are exactly the same for both model-1 and model-2.

We form a ring here, as in Figure 2. In each transfer step every processor transfers the m-word message destined for its farthest remaining processor. If only one direction in the ring is used for all transfers, then the total number of transfer steps is equal to $\sum_{i=1}^{p-1}(p-i) = \sum_{i=1}^{p-1} i = \frac{(p-1)p}{2}$. The total time taken by this operation is

$$\begin{aligned} T_{all\_to\_all\_pers,1} &= t_s + \sum_{i=1}^{p-1} m t_w (p-i) + \sum_{i=1}^{p-1} t_c (p-i-1) \\ &= t_s + m t_w \frac{(p-1)p}{2} + t_c \frac{(p-1)(p-2)}{2} = O(mp^2) \end{aligned}$$

However, for the shortest paths, and therefore for smaller communication time, both directions in the ring should be used. In this case, there are $\lceil \frac{p-1}{2} \rceil$ "large" communications stages. In the i-th "large" stage, where $i = 1, 2, ..., \lceil \frac{p-1}{2} \rceil$, each processor transmits the respective messages to the processors at the same distance i to its left and to its right, exclusively in this order. If p is even, then the $\lceil \frac{p-1}{2} \rceil$-th "large" stage implements transmissions in only one of the two directions in the ring. Therefore, the total number of transfer steps to neighbors is equal to

$$2 \sum_{i=1}^{\lceil \frac{p-1}{2} \rceil} i - (\lceil \frac{p-1}{2} \rceil - \lfloor \frac{p-1}{2} \rfloor) = 2\frac{1}{2} \lceil \frac{p-1}{2} \rceil (\lceil \frac{p-1}{2} \rceil + 1) - (\lceil \frac{p-1}{2} \rceil - \lfloor \frac{p-1}{2} \rfloor) = \lceil \frac{p-1}{2} \rceil^2 + \lfloor \frac{p-1}{2} \rfloor$$

The total time is

$$\begin{aligned} T_{all\_to\_all\_pers,1} &= t_s + m t_w (\lceil \frac{p-1}{2} \rceil^2 + \lfloor \frac{p-1}{2} \rfloor) + 2 \sum_{i=0}^{\lceil \frac{p-1}{2} \rceil - 1} t_c\, i - (\lceil \frac{p-1}{2} \rceil - \lfloor \frac{p-1}{2} \rfloor) t_c \\ &= t_s + m t_w (\lceil \frac{p-1}{2} \rceil^2 + \lfloor \frac{p-1}{2} \rfloor) + t_c (\lceil \frac{p-1}{2} \rceil^2 - 2 \lceil \frac{p-1}{2} \rceil + \lfloor \frac{p-1}{2} \rfloor) = O(mp^2) \end{aligned}$$

which is asymptotically optimal because each processor sends out $O(p)$ messages of m words each, and the average distance travelled is $O(p)$.

With wormhole routing, the communication time is

$$\begin{aligned} T(WR)_{all\_to\_all\_pers,1} &= t_s + 2 \sum_{i=1}^{\lceil \frac{p-1}{2} \rceil} m\, t_w\, i - m t_w (\lceil \frac{p-1}{2} \rceil - \lfloor \frac{p-1}{2} \rfloor) \\ &= t_s + m t_w (\lceil \frac{p-1}{2} \rceil^2 + \lfloor \frac{p-1}{2} \rfloor) = O(mp^2) \end{aligned}$$

3.5.2 Model-3

The all-to-all personalized communication operation involves a lot of message transfers. We will not necessarily derive the most efficient procedure here, because such a procedure can be of a very complex nature. We present a simple procedure that comprises two stages. The basic idea is to use the largest possible number of linear arrays for pipelined message transfers, with the smallest possible number of nodes per such array. Figure 3 shows the chosen linear arrays in the $HOW(10, 3, 1)$.

- First stage: this is the initialization stage where local transfers are employed to move messages to processors that belong to the aforementioned linear arrays. Every processor passes all

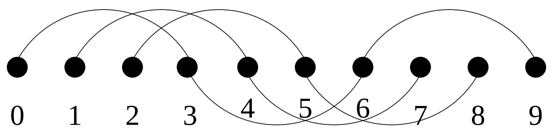

Figure 3: Chosen linear arrays in the $HOW(10, 3, 1)$ for all-to-all personalized communication.

relative messages to neighbors in its window(s). For a given destination message, it passes that message to its neighbor that belongs to a linear array containing that destination; if two such neighbors exist, the one closer to the destination is chosen. It takes up to $s_1 = \lceil \frac{p-1}{w} \rceil$ cycles to finish the initialization, which is the same as the maximum number of values to be sent from a processor to another one.

- Second stage: the linear arrays are used to transfer the values. There are w linear arrays to be used. We need up to $s_2 = \lceil \frac{p-1}{w} \rceil - 1$ cycles to finish the broadcasting along the linear arrays, which is the same as the maximum number of values a processor has to send in a single dimension; messages going farther have higher priority.

The total time taken by this operation is

$$T_{all\_to\_all\_pers,3} = t_s + m(s_1+s_2)t_w + m(s_1+s_2-1)t_c = t_s + 2\ mt_w \lceil \frac{p-1}{w} \rceil + mt_c(2\ \lceil \frac{p-1}{w} \rceil - 1) = O(m\frac{p}{w})$$

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all\_pers,3} = t_s + 2\ mt_w \lceil \frac{p-1}{w} \rceil = O(m\frac{p}{w})$$

4 Communication Operations on 2-D HOW Systems

Assume symmetric 2-D HOW systems with p processors. Also, p_{ij} denotes the processor in row i and column j, where $i, j = 0, 1, \cdots, \sqrt{p} - 1$.

4.1 One-to-One Communication

We assume, without loss of generality, that p_{00} is the source processor and that the destination is at distance l.

With SF routing, sending a single message containing m words takes $t_s + mt_w l + t_c(l-1)$ time, where l is the number of links traversed by the message. For a 2-D HOW system with a total of p processors (having $\sqrt{p}$ rows and $\sqrt{p}$ columns) and window size w, l is at most $2\lceil \frac{\sqrt{p}-1}{w} \rceil$, and therefore the time for a single message transfer has the *upper bound* of

$$T_{one\_to\_one} = t_s + 2mt_w \lceil \frac{\sqrt{p}-1}{w} \rceil + t_c(2\lceil \frac{\sqrt{p}-1}{w} \rceil - 1) = O(m\frac{\sqrt{p}}{w})$$

assuming no contention with other messages at intermediate processors.

With wormhole routing, for a single message transfer on the 2-D HOW system the *upper bound* is

$$T(WR)_{one\_to\_one} = t_s + 2t_w \lceil \frac{\sqrt{p}-1}{w} \rceil + (m-1)t_w = O(m\frac{\sqrt{p}}{w})$$

For the $GH(\sqrt{p}, 2)$ we have $w = O(\sqrt{p})$, and therefore both times are $O(m)$.

4.2 One-to-All Broadcasting

4.2.1 Model-1

For the best possible performance, we first have to determine which of the row or column window the source belongs to is closer to the center of that row or column, respectively. If it is the row window, then the source broadcasts within that row, and this is followed by broadcasting from those row PEs into all columns. Otherwise, we begin with column broadcasting. However, here we assume the worst case, where the source PE is in the first window of the corresponding 1-D HOW row and column subsystems. Using the same notations as for the 1-D HOW system, s_1 represents the number of transfer steps needed to fill the first window in this row and s_2 represents the number of transfer steps needed in the second stage to copy the values from the first window into the remaining windows of this row. We already know the following relations among s_1,s_2, and w

$$s_1 = \lceil \log(w+1) \rceil$$
$$s_2 = \lceil (\sqrt{p} - 2^{s_1})/w \rceil$$

This operation is done by first broadcasting within the aforementioned row and then from that row within all the columns. The communication time under model-1 with SF routing has the *upper bound*

$$T_{one\_to\_all,1} = \begin{cases} t_s + 2mt_w \lceil \log \sqrt{p} \rceil + t_c(2\lceil \log \sqrt{p} \rceil - 1) = O(m \log p) & \text{if } (\sqrt{p}-1) \leq w \\ t_s + 2mt_w(s_1+s_2) + t_c(2(s_1+s_2)-1) = O(m \log w + m\frac{\sqrt{p}}{w}) & \text{if } (\sqrt{p}-1) > w \end{cases}$$

We need time $O(m \log p)$ for the $GH(\sqrt{p}, 2)$.

With wormhole routing, the *upper bound* is

$$T(WR)_{one\_to\_all,1} = \begin{cases} t_s + 2t_w \lceil \log \sqrt{p} \rceil + (m-1)t_w = O(m + \log \sqrt{p}) & \text{if } (\sqrt{p}-1) \leq w \\ t_s + 2t_w(s_1+s_2) + (m-1)t_w = O(m + \log w + \frac{\sqrt{p}}{w}) & \text{if } (\sqrt{p}-1) > w \end{cases}$$

assuming that incoming data can be stored locally and can simultaneously be transferred to the next PE in the path. We need time $O(m + \log p)$ for the $GH(\sqrt{p}, 2)$.

4.2.2 Model-2 and Model-3

For the one-to-all broadcasting operation, there is only one value to be sent, and therefore the whole procedure for model-3 is exactly the same as that for model-2. Figure 4 shows two different methods used for one-to-all broadcasting. The numbers of communications steps for the two methods are the same. However, method (b) is easier to program, because it is an extension of the respective method for the 1-D HOW system. This method first broadcasts within the row and then within all columns. The *upper bound* on the total time taken by this operation is

$$T_{one\_to\_all,2} = t_s + 2mt_w \lceil \frac{\sqrt{p}-1}{w} \rceil + t_c(2\lceil \frac{\sqrt{p}-1}{w} \rceil - 1) = O(m\frac{\sqrt{p}}{w})$$

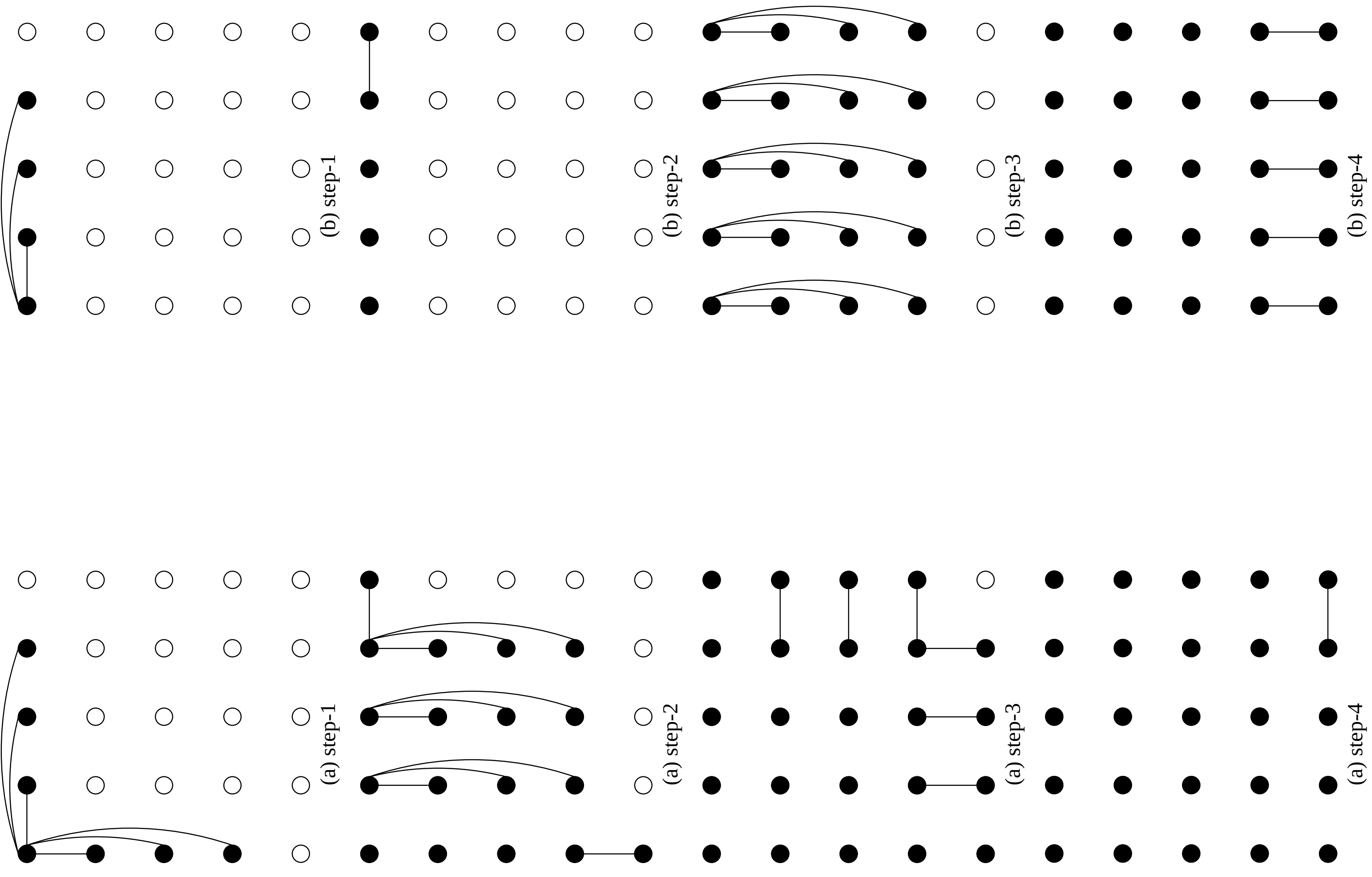

Figure 4: One-to-all broadcasting under model-2 and model-3 with two different methods, both of which have the same number of communication steps. A filled circle means that the current processor has already received the message broadcast by the source. All communications steps are shown here. We assume that w=3. For the worst case, we assume p_{00} to be the source.

With wormhole routing, the *upper bound* is

$$T(WR)_{one\_to\_all,2} = t_s + 2t_w\lceil\frac{\sqrt{p}-1}{w}\rceil + (m-1)t_w = O(m + \frac{\sqrt{p}}{w})$$

assuming that the dimension to be traversed is changed just after the first flit is received. Both times are $O(m)$ for the $GH(\sqrt{p}, 2)$.

4.3 All-to-All Broadcasting

The procedure repeats many times the corresponding procedure for the 1-D HOW system. That is, processors first exchange messages along rows, so that each processor has $\sqrt{p}$ messages at the end for the processors in its own column. Then, processors exchange their $\sqrt{p}$ messages along columns by repeating the same procedure $\sqrt{p}$ times within the columns.

4.3.1 Model-1

For model-1, there is only one output port of each processor we can use at a time. In order to let every processor pass some information to a neighbor, we deliberately choose some channels to form a ring in each row/column. We assume pipelining of messages along rows and columns.

We start with all-to-all row broadcasting that takes time $t_s + \tau = t_s + (\sqrt{p}-1)mt_w + t_c(\sqrt{p}-2)$, as derived for the 1-D HOW system in Subsection 3.1.1. The $\sqrt{p}$ column broadcasts then take time $\sqrt{p}\tau$, because all-to-all 1-D HOW broadcasting is repeated $\sqrt{p}$ times. The time taken by the entire operation is

$$\begin{aligned} T_{all\_to\_all,1} &= t_s + (1+\sqrt{p})mt_w(\sqrt{p}-1) + (1+\sqrt{p})t_c(\sqrt{p}-2) + t_c \\ &= t_s + (p-1)mt_w + (p-\sqrt{p}-1)t_c = O(mp) \end{aligned}$$

The last t_c term is for switching from row broadcasting into column broadcasting. This asymptotic time is optimal because each processor can use only one output port at a time, and therefore each message will make $O(p)$ hops to visit all $O(p)$ processors.

With wormhole routing, within each row, the entire time is $t_s + m(\sqrt{p}-1)t_w$, assuming the formation of a ring. This is because each processor starts receiving flits with the first data transfer, pipelining of messages is applied, and the total number of flits each processor receives is $m(\sqrt{p}-1)$. Similarly, for columns the time is $m\sqrt{p}(\sqrt{p}-1)t_w$. The total time is

$$T(WR)_{all\_to\_all,1} = t_s + m(1+\sqrt{p})(\sqrt{p}-1)t_w = t_s + m(p-1)t_w = O(mp)$$

We need the same times for the $GH(\sqrt{p}, 2)$.

4.3.2 Model-2

Based on the algorithm proposed for the 1-D HOW system, the total time taken by this operation is

$$T_{all\_to\_all,2} = t_s + (1+\sqrt{p})mt_w(\lceil\frac{\sqrt{p}-1}{w}\rceil + x) + t_c(1+\sqrt{p})(\lceil\frac{\sqrt{p}-1}{w}\rceil + x - 1) = O(m\frac{p}{w})$$

where x is the largest integer less than $\frac{\sqrt{p}-1}{2w}$. The algorithm for the 1-D HOW system is used $(1+\sqrt{p})$ times, once for the rows and $\sqrt{p}$ times for the columns.

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all,2} = t_s + 2mt_w(\lceil \frac{\sqrt{p}-1}{w} \rceil + x)(1+\sqrt{p}) = O(m\frac{p}{w})$$

Both times are $O(m\sqrt{p})$ for the $GH(\sqrt{p},2)$.

4.3.3 Model-3

It is very similar to the procedure for model-2. Since each individual processor can send different messages at the same time, we do not need to split any stage. The total time taken by this operation is

$$T_{all\_to\_all,3} = t_s + (1+\sqrt{p})mt_w \lceil \frac{\sqrt{p}-1}{w} \rceil + t_c(1+\sqrt{p})(\lceil \frac{\sqrt{p}-1}{w} \rceil - 1) = O(m\frac{p}{w})$$

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all,3} = t_s + mt_w \lceil \frac{\sqrt{p}-1}{w} \rceil (1+\sqrt{p}) = O(m\frac{p}{w})$$

Both times are $O(m\sqrt{p})$ for the $GH(\sqrt{p},2)$.

4.4 One-to-All Personalized Communication

4.4.1 Model-1 and Model-2

Because of personalized data, the same procedure is applied for model-1 and model-2. Restricted by the availability of only one output port at a time for each processor, independently of the window size it will take $(\sqrt{p}-1)$ transfer steps along a row or a column for a processor to send personalized data to all other processors. In the first phase, the source processor, assume p_{00}, passes messages within its row for all processors in the corresponding columns. Messages going farther have higher priority of transmission. This process is implemented as $\sqrt{p}$ one-to-all personalized communications within the row (i.e., 1-D HOW system). At the end of the first phase each of the first row processors will have $\sqrt{p}$ messages. All $\sqrt{p}$ messages of each first row processor will be transferred in the second phase along the corresponding column applying again one-to-all personalized communication. The total time taken by this operation is

$$\begin{aligned} T_{one\_to\_all\_pers,1} &= t_s + (\sqrt{p}+1)mt_w(\sqrt{p}-1) + t_c(1+\sqrt{p})(\lceil \frac{\sqrt{p}-1}{2} \rceil - 1) \\ &= t_s + (p-1)mt_w + t_c(1+\sqrt{p})(\lceil \frac{\sqrt{p}-1}{2} \rceil - 1) = O(mp) \end{aligned}$$

With wormhole routing, the communication time is

$$T(WR)_{one\_to\_all\_pers,1} = t_s + (1+\sqrt{p})mt_w(\sqrt{p}-1) = t_s + mt_w(p-1) = O(mp)$$

Both times are $O(mp)$ for the $GH(\sqrt{p},2)$.

4.4.2 Model-3

We first send the messages that must travel the longest distance using simultaneously all column and row connections. (Note: it is a different method than that used for model-1.) The number of message transfer steps is $2\lceil\frac{\sqrt{p}-1}{w}\rceil$, the same as the diameter of the system. The *upper bound* on the total time is

$$T_{one\_to\_all\_pers,3} = t_s + mt_w 2\lceil\frac{\sqrt{p}-1}{w}\rceil + t_c(2\lceil\frac{\sqrt{p}-1}{w}\rceil - 1) = O(m\frac{\sqrt{p}}{w})$$

which is optimal.

With wormhole routing, the *upper bound* is

$$T(WR)_{one\_to\_all\_pers,3} = t_s + mt_w 2\lceil\frac{\sqrt{p}-1}{w}\rceil = O(m\frac{\sqrt{p}}{w})$$

Both times are $O(m)$ for the $GH(\sqrt{p}, 2)$.

4.5 All-to-All Personalized Communication

Two phases are implemented again.

4.5.1 Model-1 and Model-2

We form rings in rows and columns. In each transfer step the message size is m words and every processor tries to transfer the message(s) destined for its farthest processor. We start with row transfers and continue with $\sqrt{p}$ all-to-all personalized communications within columns. Based on the implementation of $(\sqrt{p}+1)$ all-to-all personalized 1-D HOW operations, we get

$$T_{all\_to\_all\_pers,1} = t_s + (\sqrt{p}+1)mt_w(\lceil\frac{\sqrt{p}-1}{2}\rceil^2 + \lfloor\frac{\sqrt{p}-1}{2}\rfloor) +$$
$$(\sqrt{p}+1)t_c(\lceil\frac{\sqrt{p}-1}{2}\rceil^2 - 2\lceil\frac{\sqrt{p}-1}{2}\rceil + \lfloor\frac{\sqrt{p}-1}{2}\rfloor) = O(mp^{3/2})$$

With wormhole routing, the communication time is

$$T(WR)_{all\_to\_all\_pers,1} = t_s + (\sqrt{p}+1)mt_w(\lceil\frac{\sqrt{p}-1}{2}\rceil^2 + \lfloor\frac{\sqrt{p}-1}{2}\rfloor) = O(mp^{3/2})$$

We need similar times for the $GH(\sqrt{p}, 2)$.

4.5.2 Model-3

The implementation of this operation requires $\sqrt{p} + (p - \sqrt{p}) = p$ all-to-all personalized 1-D HOW communications. Each processor in the first phase sends $\sqrt{p}$ personalized messages to each processor in its own row; these values are for the processors in the column where the latter processor resides. After the first phase, each processor contains p personalized messages for the processors in its own column; actually, $\sqrt{p}$ of these messages where destined for the particular processor. Therefore, $p - \sqrt{p}$ operations are implemented in the second phase. The total time is

$$T_{all\_to\_all\_pers,3} = t_s + pmt_w 2\lceil\frac{\sqrt{p}-1}{w}\rceil + pmt_c(2\lceil\frac{\sqrt{p}-1}{w}\rceil - 1) = O(m\frac{p^{3/2}}{w})$$

With wormhole routing, the time is

$$T(WR)_{all\_to\_all\_pers,3} = t_s + pmt_w 2\lceil\frac{\sqrt{p}-1}{w}\rceil = O(m\frac{p^{3/2}}{w})$$

Both times are $O(mp)$ for the $GH(\sqrt{p}, 2)$.

5 Performance Comparisons

In this section we compare the communications capabilities of 2-D HOW systems, binary hypercubes [4], and 2-D generalized hypercubes, all with the same number p of processors. We consider communications under model-3 for store-and-forward routing. Table 2 summarizes their performance and compares them using as cost measure the product of the communication time and the node pin-out; systems with lower cost are preferrable. The pin-out for a network is the number of wires per node; it is the product of the node degree and the channel width. It is a very widely used measure of the VLSI cost.

Network	HOW	Binary Hypercube	Generalized Hypercube
Pin-out	$O(w)$	$O(\log p)$	$O(\sqrt{p})$
$T_{one\_to\_one,3}$	$O(m\frac{\sqrt{p}}{w})$	$O(m \log p)$	$O(m)$
$T_{one\_to\_all,3}$	$O(m\frac{\sqrt{p}}{w})$	$O(m \log p)$	$O(m)$
$T_{all\_to\_all,3}$	$O(m\frac{p}{w})$	$O(mp)$	$O(m\sqrt{p})$
$T_{one\_to\_all\_pers,3}$	$O(m\frac{\sqrt{p}}{w})$	$O(mp)$	$O(m)$
$T_{all\_to\_all\_pers,3}$	$O(m\frac{p^{3/2}}{w})$	$O(mp \log p)$	$O(mp)$
$cost_{one\_to\_one,3}$	$O(m\sqrt{p})$	$O(mlog^2 p)$	$O(m\sqrt{p})$
$cost_{one\_to\_all,3}$	$O(m\sqrt{p})$	$O(mlog^2 p)$	$O(m\sqrt{p})$
$cost_{all\_to\_all,3}$	$O(mp)$	$O(mp \log p)$	$O(mp)$
$cost_{one\_to\_all\_pers,3}$	$O(m\sqrt{p})$	$O(mp \log p)$	$O(m\sqrt{p})$
$cost_{all\_to\_all\_pers,3}$	$O(mp^{3/2})$	$O(mplog^2 p)$	$O(mp^{3/2})$

Table 2: Cost comparison of interconnection networks.

It becomes obvious that HOWs perform asymptotically better than binary hypercubes in one-to-all personalized communication and all-to-all broadcasting. In the other two types of communications, the result of the comparison depends on the value of w. The cost of implementing all these communications operations is asymptotically identical for HOWs and generalized hypercubes; this is very important as HOWs are much easier to implement than generalized hypercubes. Therefore, HOWs are proven viable networks in the field of very high performance computing.

6 Conclusions

We introduced a new class of scalable architectures and relevant communications algorithms for popular communications models. We compared the communications performance and cost of these architectures with those of generalized and binary hypercubes. Our results prove the viability of our scalable architectures for the very high performance computing field.

References

[1] S. G. Ziavras, "RH: A Versatile Family of Reduced Hypercube Interconnection Networks," *IEEE Trans. Paral. Distr. Systems*, Vol. 5, No. 11, Nov. 1994, pp. 1210-1220.

[2] J. K. Antonio, L. Lin, and R. C. Metzger, "Complexity of Intensive Communications on Balanced Generalized Hypercubes," *Intern. Paral. Proces. Symp.*, 1993, pp. 387-394.

[3] P. Fragopoulou, S. G. Akl, and H. Meijer, "Optimal Communication Primitives on the Generalized Hypercube Network," *Journ. Paral. Distr. Comput. 32,* 1996, pp. 173-187.

[4] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms.* The Benjamin/Cummings Publishing Company, Inc. 1994.

[5] T. Szymanski, ""Hypermeshes": Optical Interconnection Networks for Parallel Computing," *Journ. Paral. Distr. Comput.*, Vol. 26, 1995, pp. 1-23.

[6] L.D. Wittie, "Communication Structures for Large Networks of Multicomputers," *IEEE Trans. Comput.* C-30(4), 1981.

[7] S.G. Ziavras and A. Mukherjee, "Data Broadcasting and Reduction, Prefix Computation, and Sorting on Reduced Hypercube Parallel Computers," *Parallel Computing* 22, 1996, pp. 595-606.

[8] S.G. Ziavras, "On the Problem of Expanding Hypercube-Based Systems," *Journ. Paral. Distr. Comp.* 16(1), 1992, pp. 41-53.

[9] S.G. Ziavras, "Scalable Multifolded Hypercubes for Versatile Parallel Computers," *Paral. Proc. Letts.* 5(2), 1995, pp. 241-250.

[10] L.N. Bhuyan and D.P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Trans. Comput.* 33 (4), 1984, pp. 323-333.

[11] S.G. Ziavras, "Investigation of Various Mesh Architectures with Broadcast Buses for High-Performance Computing," *VLSI Design*, Jan. 1999.

[12] S.G. Ziavras, et al., "A Low-Complexity Parallel System for Gracious, Scalable Performance. Case Study for Near PetaFLOPS Computing," *6th Symp. Frontiers Massively Paral. Comput.*, 1996, pp. 363-370.

[13] S.G. Ziavras and S. Krishnamurthy, "Evaluating the Communications Capabilities of the Generalized Hypercube Interconnection Network," *Concurrency: Practice and Experience*, accepted for publication.