Optimal Total Exchange in Cayley Graphs*

Vassilios V. Dimakopoulos *Nikitas J. Dimopoulos*

Technical Report ECE-96-1

Dept. of Electrical and Computer Engineering, University of Victoria
P.O. Box 3055, Victoria, B.C., CANADA, V8W 3P6.
Tel: (604) 721-8773, 721-8902, *Fax:* (604) 721-6052,
E-mail: {dimako, nikitas}@ece.uvic.ca

January, 1996

Abstract

Consider an interconnection network and the following situation: every node needs to send a different message to every other node. This is the *total exchange* problem, one of a number of information dissemination problems known as collective communications. Under the assumption that a node can send and receive only one message at each step (*single-port* model) it is seen that the minimum time required to solve the problem is governed by the status (or total distance) of the nodes in the network. We present here a time-optimal solution for any Cayley network. Rings, hypercubes, cube-connected cycles, butterflies are some well-known Cayley networks which can take advantage of our method. The solution is based on a class of algorithms which we call *node-invariant* algorithms and which behave uniformly across the network.

Keywords: Cayley graphs, collective communications, interconnection networks, total exchange (multiscattering)

AMS(MOS) subject classifications: 94C15, 68M10, 68R10, 68Q22, 05C25

*This research was supported in part through grants from NSERC and the University of Victoria. Sumbitted for publication.

1. Introduction

Collective communications for distributed-memory multiprocessors have recently received considerable attention, as for example is evident from their inclusion in the Message Passing Interface standard [15] and from their importance in supporting various constructs in High Performance Fortran [10, 14]. This is easily justified by their frequent appearance in parallel numerical algorithms [11, 4].

Broadcasting, scattering, gathering, multinode broadcasting (gossiping) and total exchange constitute a set of representative information dissemination problems [9] that have to be efficiently solved in order to maximize the performance of message-passing parallel programs. Out of this set, *total exchange* will be the subject of this paper. In total exchange, each node in a network has distinct messages to send to all the other nodes. The problem has often, and quite reasonably, been identified with matrix transposition. It is easy to see why: if the network has n nodes and each node stores a row of an $n \times n$ matrix then in order to transpose the matrix, each node has to distribute the elements of its row to all the other nodes. Of course the application of total exchange is not limited to matrix transposition; other data permutations occurring e.g. in FFT algorithms can also be viewed as total exchange problems. Total exchange is also known as *multiscattering* or *all-to-all personalized communication*.

Algorithms to solve the problem for a number of networks under a variety of models/assumptions have appeared in many recent works, mostly concentrating in hypercubes and tori (e.g. [18, 12, 3, 19]). Here we are going to follow the so-called *single-port* model in a store-and-forward network. Formally, our problem will be the distribution of distinct messages from every node to every other node subject to the following conditions:

- only adjacent nodes can exchange messages,
- a message requires one time unit (or *step*) in order to be transferred between two nodes,
- a node can send at most one message *and* receive at most one message in each step.

Under this model, time-optimal total exchange algorithms have been given in [4, pp. 81–83] for hypercubes and in [16] for star graphs. In this paper we are going to show that it is possible to solve the problem in the minimum time in any Cayley network. Hypercubes and star graphs belong to the class of Cayley networks, as do complete graphs, rings, cube-connected cycles, (wrapped) butterflies and many other interesting and widely studied networks whose significance in well-known [13].

The paper is organized as follows. Section 2 introduces some elementary graph-theoretic and group-theoretic notation. In Section 3 we derive a simple property of Cayley networks which will be useful for our arguments. In Section 4 we give a lower bound for the time needed to perform total exchange under the single-port model. In the same section we give sufficient conditions for achieving the lower bound. We then proceed to formally define the class of *node-invariant* algorithms and prove its optimality for the total exchange problem in Section 5. A simple node-invariant algorithm is given in Section 6, along with an example in hypercubes. Finally, Section 7 summarizes the results.

2. Graph-theoretic and group-theoretic notions

An (undirected) graph G consists of a set V of *nodes* (or *vertices*) interconnected by a set E of (undirected) *edges*. This is the usual model of representing a multiprocessor interconnection network: each processor corresponds to a node and each communication link corresponds to an edge. Thus the terms 'graph' and 'network' will be considered synonymous here. Nodes connected by an edge in E are *adjacent* to each other. Nodes adjacent to $v \in V$ are *neighbors* of v.

A *path* in G from node v to node u is a sequence of nodes

$$v = v_0, v_1, \ldots, v_\ell = u,$$

such that all vertices are distinct and for all $0 \leq i \leq \ell$, the edge $(v_i, v_{i+1}) \in E$. We say that the *length* of a path is ℓ if it contains $\ell + 1$ vertices. In a *connected* graph there exists a path between any two nodes, and this is the class of graphs we consider here. The *distance*, $dist(v,u)$, between vertices v and u is the length of a shortest path between v and u. Finally, the *eccentricity* of v, $e(v)$, is the distance to a node farthest from v, i.e.

$$e(v) = \max_{u \in V}\{dist(v,u)\}.$$

An *automorphism* of the graph is a mapping from the vertices to the vertices that preserves the edges. Formally, an automorphism of G is a permutation σ of V such that $(\sigma(v), \sigma(u)) \in E$ if and only if $(v,u) \in E$. If for any pair of vertices v, u there exists an automorphism that maps v to u then the graph is *node symmetric*.

A *group* consists of a set $\mathcal{G}$ and an associative binary operation '$\cdot$' on $\mathcal{G}$ with the following two properties. There exists an *identity* element — that is and element $\epsilon \in \mathcal{G}$ for which $a \cdot \epsilon = \epsilon \cdot a = a$ for all $a \in \mathcal{G}$ — and for each $a \in \mathcal{G}$ there exists an *inverse* element, denoted by a^{-1} — that is an element $a^{-1} \in \mathcal{G}$ for which $a \cdot a^{-1} = a^{-1} \cdot a = \epsilon$. The inverse of an element is unique. It is known that the set of automorphisms of a graph G is a group with respect to the composition operation, and we will denote it by $\Pi(G)$.

Cayley graphs [5, 1] are based on groups and constitute a large class of node symmetric networks. Given a set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_d\}$ of generators for a group $\mathcal{G}$, a *Cayley graph* has vertices corresponding to the elements of $\mathcal{G}$ and edges corresponding to the action of the generators. That is, if $v, u \in \mathcal{G}$, the edge (v, u) exists in G iff there is a generator $\gamma \in \Gamma$ such that $v \cdot \gamma = u$. A usual assumption is that the identity element of $\mathcal{G}$ does not belong to Γ (in order to avoid edges from a node to itself) and that Γ is closed under inverses (so that the graph is in effect undirected).

The class includes important networks such as the *hypercube*, the (wrapped) *butterfly*, the *cube-connected cycles* [2, 17, 8]. Also, connected *circulant graphs* [6] (which include the rings) are Cayley networks [5].

3. An automorphism property of Cayley graphs

Let G be a node symmetric network with node set $V = \{v_0, v_1, \ldots, v_{n-1}\}$, and let $\Pi(G)$ be its automorphism group. Denote by $\Pi_{v_i}(G)$ the subset of $\Pi(G)$ consisting of all automorphisms that map v_0 to v_i:

$$\Pi_{v_i}(G) = \{\sigma \;\mid\; \sigma(v_0) = v_i,\, \sigma \in \Pi(G)\}.$$

Notice that $\Pi_{v_i}(G)$ is nonempty since G is node symmetric. From each set $\Pi_{v_i}(G)$ we select one automorphism σ_{v_i} and form the set

$$\Sigma(G) = \{\sigma_{v_i} \;\mid\; \sigma_{v_i} \in \Pi_{v_i}(G),\, i = 0, 1, \ldots, n-1\}.$$

In particular, we select σ_{v_0} to be the *identity* mapping. Let $\sigma\sigma'$ be the composition of mappings σ and σ'. We insist that the selected mappings have the following property: for every neighbor v_a of node v_0 and for every $i = 0, 1, \ldots, n-1$,

$$\sigma_{\sigma_{v_i}(v_a)} = \sigma_{v_i}\sigma_{v_a}. \tag{1}$$

In simple terms, requirement (1) means that if v_a is mapped, through σ_{v_i}, to some neighbor v of v_i then σ_v can be written as the composition of σ_{v_i} and σ_{v_a}. The implications of (1) will be seen shortly; but first we have the following result.

Lemma 1 *Every Cayley network has a set $\Sigma(G)$ of automorphisms that satisfy (1).*

Proof. Consider any Cayley graph based on a group $\mathcal{G}$, and the mapping

$$\sigma_{v_i}(v_x) = v_i \cdot v_0^{-1} \cdot v_x,$$

where v_0^{-1} is the inverse element of v_0 in $\mathcal{G}$. This mapping is easily seen to be an automorphism of the graph [1] and clearly maps v_0 to v_i. We obtain

$$\begin{aligned} \sigma_{\sigma_{v_i}(v_a)}(v_x) &= \sigma_{v_i}(v_a) \cdot v_0^{-1} \cdot v_x \\ &= v_i \cdot v_0^{-1} \cdot v_a \cdot v_0^{-1} \cdot v_x \\ &= \sigma_{v_i}(v_a \cdot v_0^{-1} \cdot v_x) \\ &= \sigma_{v_i}(\sigma_{v_a}(v_x)), \end{aligned}$$

which satisfies (1). ∎

Notice that the set of automorphisms given in the proof of Lemma 1 may not be the only one which satisfies (1). Also, if the network is known, the automorphisms may obtain a (computationally) simpler form. As an example, consider a ring with n nodes. Node v_i is adjacent to nodes $v_{i\oplus 1}$ and $v_{i\ominus 1}$ where $\oplus$ and $\ominus$ denote addition and subtraction modulo n. A set $\Sigma(G)$ of automorphisms with the desired properties consists of the following mappings:

$$\sigma_{v_i}(v_x) = v_{i\oplus x},$$

$i = 0, 1, \ldots, n-1$. Clearly,

$$\sigma_{\sigma_{v_i}(v_a)}(v_x) = \sigma_{v_{i\oplus a}}(v_x) = v_{i\oplus a\oplus x} = \sigma_{v_i}(v_{a\oplus x}) = \sigma_{v_i}(\sigma_{v_a}(v_x)),$$

satisfying (1). Actually, the above mappings work for any (connected) circulant graph.

Lemma 2 *Let $\Sigma(G)$ be a set of automorphisms satisfying (1). If v_0 "picks" one of its neighbors, v_a, and every node v_i, $i = 1, 2, \ldots, n-1$, "picks" neighbor $\sigma_{v_i}(v_a)$ then* **(a)** *every node is picked by exactly one other node and* **(b)** *if v_b is the node that picks v_0 then $\sigma_{v_i}(v_b)$ is the node that picks v_i.*

Proof.

(a) For the first part, all we have to show is that $\sigma_{v_i}(v_a) \neq \sigma_{v_j}(v_a)$ for $i \neq j$. That is, no node is picked by more than one other node. Then, since there are n "picks" in total, each of the n nodes is picked by exactly one other node. Let us say that there exists some i and some $j \neq i$ for which $\sigma_{v_i}(v_a) = \sigma_{v_j}(v_a) = v_k$, for some k. Then $\sigma_{v_k} = \sigma_{\sigma_{v_i}(v_a)}$, and from (1), $\sigma_{v_k} = \sigma_{v_i}\sigma_{v_a}$. Similarly, $\sigma_{v_k} = \sigma_{v_j}\sigma_{v_a}$. Consequently, $\sigma_{v_i}\sigma_{v_a} = \sigma_{v_j}\sigma_{v_a}$, or $\sigma_{v_i} = \sigma_{v_j}$, which is absurd.

(b) Let v_b be the node that picks v_0, that is $v_0 = \sigma_{v_b}(v_a)$. Since $v_i = \sigma_{v_i}(v_0)$, we obtain $v_i = \sigma_{v_i}\big(\sigma_{v_b}(v_a)\big)$. From (1), $v_i = \sigma_{\sigma_{v_i}(v_b)}(v_a)$. This means that node $\sigma_{v_i}(v_b)$ picked v_i.

■

4. Lower bound on total exchange time

Consider some node v in the network. In the total exchange problem, node v must send one message to each of the other nodes. If there exist n_d nodes in distance d from v, where $d = 1, 2, \ldots, e(v)$, then the messages sent by v must cross

$$s(v) = \sum_{d=1}^{e(v)} d n_d$$

links in total. For all messages to be exchanged, the total number of link traversals must be

$$S_G = \sum_{v \in V} s(v).$$

The quantity $s(v)$ is known as the *total distance* or the *status* [7] of node v.

Every time a message is communicated between adjacent nodes one link traversal occurs. If nodes are allowed to transmit only one message per step, the maximum number of link traversals in a single step is at most n. Consequently, we can at best subtract n units from S_G in each step, so that a lower bound on total exchange time is

$$T \geq \frac{S_G}{n}. \tag{2}$$

Because all nodes in a node symmetric graph have the same status [7], it is seen that for such networks the lower bound is simply $T \geq s(v)$, where v is any node.

Based on the above discussion we immediately have the following sufficient conditions in order for a total exchange scheme to achieve the lower bound of (2):

$$\text{all nodes are busy all the time, and,} \tag{3}$$

$$\text{every transmitted messages gets closer to its destination.} \tag{4}$$

The conditions guarantee that n units are subtracted from S_G at every step, which is the best we can do. Notice that we must require that transmitted messages are not *derouted*, that is, they always follow minimal paths, getting closer to their destination after each link traversal.

5. Optimal algorithms

Every node v_i in the network maintains a *message queue*, Q_{v_i}, where incoming messages from neighbors are deposited until they are scheduled for transfer to some other node. If an incoming message is destined for v_i it is assumed that it does not join the message queue but is rather forwarded to the local processor for consumption. At node v_i some local algorithm $\mathcal{A}_{v_i}$ operates in order to schedule the message transfers. Whenever there exist messages in Q_{v_i}, $\mathcal{A}_{v_i}$ is responsible for selecting the message to leave in the next time unit and the neighbor of v_i to which the message will be sent.

Definition 1 A distributed total exchange algorithm $\mathcal{A} = (\mathcal{A}_{v_0}, \mathcal{A}_{v_1}, \ldots, \mathcal{A}_{v_{n-1}})$ is a collection of local algorithms, algorithm $\mathcal{A}_{v_i}$ running on node v_i, $i = 0, 1, \ldots, n-1$. Algorithm $\mathcal{A}_{v_i}$ is written as $\mathcal{A}_{v_i} = (f_{v_i}, w_{v_i})$, where, given a message queue Q_{v_i}, f_{v_i} selects a message $f_{v_i}(Q_{v_i}) = m$ and w_{v_i} selects a neighbor $w_{v_i}(m)$ of v_i.

The idea now is to let every node v_i select a message "corresponding" to the message selected by node v_0 and to send it to a neighbor "corresponding" to the neighbor selected by v_0. This way we expect that the algorithm will behave uniformly across the network. The implication of such a behavior will be that all nodes have "corresponding" message queues at each step, hence queues that have the same size. We will then be able to guarantee that all queues become empty at the same time. This is exactly the time when total exchange is completed, and condition (3) will have been satisfied.

In order to describe algorithms with a uniform behavior, we need the following notation. Let $m_{v_x}(v_y)$ be the message of node v_x (source) meant for node v_y (destination). For an automorphism $\sigma \in \Pi(G)$, let $\sigma\big(m_{v_x}(v_y)\big)$ be the message of node $\sigma(v_x)$ destined for node $\sigma(v_y)$, i.e.

$$\sigma\big(m_{v_x}(v_y)\big) \stackrel{\text{def}}{=} m_{\sigma(v_x)}\big(\sigma(v_y)\big).$$

Finally, let Q be a set of messages. We define

$$\sigma(Q) \stackrel{\text{def}}{=} \Big\{\sigma\big(m_{v_x}(v_y)\big) \;\mid\; m_{v_x}(v_y) \in Q\Big\}.$$

Definition 2 Let G be a Cayley graph and let $\Sigma(G)$ be a set of automorphisms that satisfy (1). A total exchange algorithm $\mathcal{A} = (\mathcal{A}_{v_0}, \ldots, \mathcal{A}_{v_{n-1}})$ where $\mathcal{A}_{v_i} = (f_{v_i}, w_{v_i})$, $i = 0, 1, \ldots, n-1$, will be called *node-invariant* if for any message queue Q and any message m it satisfies

$$\begin{aligned} f_{v_i}\big(\sigma_{v_i}(Q)\big) &= \sigma_{v_i}\big(f_{v_0}(Q)\big) \\ w_{v_i}\big(\sigma_{v_i}(m)\big) &= \sigma_{v_i}\big(w_{v_0}(m)\big). \end{aligned}$$

Lemma 3 *If $Q_{v_i}(t)$ is the queue of node v_i at time t, $i = 0, 1, \ldots, n-1$, then any node-invariant algorithm guarantees that*

$$Q_{v_i}(t) = \sigma_{v_i}\big(Q_{v_0}(t)\big),$$

for all $t \geq 0$.

Proof. The proof is by induction on t. Initially $(t = 0)$ we have that

$$Q_{v_0} = \{m_{v_0}(v_j) \;\mid\; j = 1, 2, \ldots, n-1\}.$$

Because automorphisms are bijections $\sigma_{v_i}(v_k) \neq \sigma_{v_i}(v_\ell)$ if $k \neq \ell$. Consequently, the set $\{\sigma_{v_i}(v_j) \;\mid\; j = 1, 2, \ldots, n-1\}$ contains all nodes of G except node v_i (since $\sigma_{v_i}(v_0) = v_i$ and $j \neq 0$). Thus the message set $S = \{m_{v_i}\big(\sigma_{v_i}(v_j)\big) \;\mid\; j = 1, 2, \ldots, n-1\}$ is the same as the set $S' = \{m_{v_i}(v_k) \;\mid\; k = 0, 1, \ldots, n-1,\, k \neq i\}$. Notice that $S' = Q_{v_i}(0)$. If we write v_i as $\sigma_{v_i}(v_0)$, we obtain

$$\begin{aligned} S &= \{m_{\sigma_{v_i}(v_0)}\big(\sigma_{v_i}(v_j)\big) \;\mid\; j = 1, 2, \ldots, n-1\} \\ &= \{\sigma_{v_i}\big(m_{v_0}(v_j)\big) \;\mid\; j = 1, 2, \ldots, n-1\} \\ &= \sigma_{v_i}\Big(\{m_{v_0}(v_j) \;\mid\; j = 1, 2, \ldots, n-1\}\Big) \\ &= \sigma_{v_i}\big(Q_{v_0}(0)\big), \end{aligned}$$

showing that $Q_{v_i}(0) = \sigma_{v_i}(Q_{v_0}(0))$.

Next, assume as an induction hypothesis that for some $t \geq 0$

$$Q_{v_i}(t) = \sigma_{v_i}(Q_{v_0}(t)). \tag{5}$$

For time $t+1$ we proceed as follows. Let for simplicity $m_{s(v_i)} = f_{v_i}(Q_{v_i}(t))$ and $v_{s(v_i)} = w_{v_i}(m_{s(v_i)})$. That is, $m_{s(v_i)}$ is the message selected by v_i, and $v_{s(v_i)}$ is the neighbor of v_i to which the selected message is *sent*. From (5) and the definition of node-invariant algorithms it is easily seen that

$$m_{s(v_i)} = \sigma_{v_i}(m_{s(v_0)}), \tag{6}$$

$$v_{s(v_i)} = \sigma_{v_i}(v_{s(v_0)}). \tag{7}$$

Now notice that $v_{s(v_0)}$ is the neighbor v_0 "picked" to send the message to. From (7) it is seen that Lemma 2 applies to show that every node receives exactly one message, and that, if $v_{r(v_0)}$ is the neighbor from which v_0 *receives* a message then

$$v_{r(v_i)} = \sigma_{v_i}(v_{r(v_0)}) \tag{8}$$

is the neighbor from which v_i receives its (unique) message. Moreover, if $m_{r(v_i)}$ is the message received by v_i, we obtain

$$\begin{aligned} m_{r(v_i)} &= m_{s(v_{r(v_i)})} \\ &= \sigma_{v_{r(v_i)}}(m_{s(v_0)}) \\ &= \sigma_{\sigma_{v_i}(v_{r(v_0)})}(m_{s(v_0)}) \\ &= \sigma_{v_i}(\sigma_{v_{r(v_0)}}(m_{s(v_0)})), \end{aligned}$$

and since $m_{r(v_0)} = m_{s(v_{r(v_0)})} = \sigma_{v_{r(v_0)}}(m_{s(v_0)})$,

$$m_{r(v_i)} = \sigma_{v_i}(m_{r(v_0)}). \tag{9}$$

To recapitulate, any node v_i selects a message $m_{s(v_i)}$ given by (6), sends it to some node $v_{s(v_i)}$ given by (7) and receives a message $m_{r(v_i)}$ given by (9) from some node $v_{r(v_i)}$ given by (8). If the destination of $m_{r(v_0)}$ is node v_0, then from (9) it is seen that the destination of $m_{r(v_i)}$ is node v_i. Conversely, if $m_{r(v_0)}$ is not meant for v_0 then $m_{r(v_i)}$ is not meant for v_i. In the first case at node v_0 we will have

$$Q_{v_0}(t+1) = Q_{v_0}(t) \setminus \{m_{s(v_0)}\},$$

since $m_{r(v_0)}$ does not join the queue, and in the second case,

$$Q_{v_0}(t+1) = Q_{v_0}(t) \cup \{m_{r(v_0)}\} \setminus \{m_{s(v_0)}\}, \tag{10}$$

where '\' is the set-theoretic difference. In the second case (the first case is treated identically), for node v_i we have

$$Q_{v_i}(t+1) = Q_{v_i}(t) \cup \{m_{r(v_i)}\} \setminus \{m_{s(v_i)}\}.$$

Using (5), (6), (9) and (10),

$$\begin{aligned} Q_{v_i}(t+1) &= \sigma_{v_i}(Q_{v_0}(t)) \cup \{\sigma_{v_i}(m_{r(v_0)})\} \setminus \{\sigma_{v_i}(m_{s(v_0)})\} \\ &= \sigma_{v_i}\Big(Q_{v_0}(t) \cup \{m_{r(v_0)}\} \setminus \{m_{s(v_0)}\}\Big) \\ &= \sigma_{v_i}(Q_{v_0}(t+1)), \end{aligned}$$

concluding the induction. ■

Lemma 4 *If node v_0 never deroutes a message then the same is true for every other node v_i, $i = 1, 2, \ldots, n-1$.*

Proof. If at some time t node v_0 selects message $m_{v_x}(v_y)$ out of its queue and sends it to some neighbor v_s, then any node v_i selects message $\sigma_{v_i}(m_{v_x}(v_y))$ and sends it to neighbor $\sigma_{v_i}(v_s)$ as we have already seen (equations (6)–(7)). All we have to show is that if v_s is on a shortest path from v_0 to v_y (i.e. v_0 does not deroute the message) then $\sigma_{v_i}(v_s)$ is on a shortest path from v_i to $\sigma_{v_i}(v_y)$.

This is easy to do because automorphisms preserve distances [5]. That is, if σ is an automorphism of a graph G then $dist(v,u) = dist(\sigma(v), \sigma(u))$ for any two vertices v and u of G. If v_0 does not deroute then $dist(v_0, v_y) = dist(v_s, v_y) + 1$. Then, we must have $dist(v_i = \sigma_{v_i}(v_0), \sigma_{v_i}(v_y)) = dist(\sigma_{v_i}(v_s), \sigma_{v_i}(v_y)) + 1$ and $\sigma_{v_i}(v_s)$ indeed lies on a shortest path from v_i to $\sigma_{v_i}(v_y)$. ■

Theorem 1 *Any node-invariant algorithm for which function w_0 selects shortest paths is an optimal total exchange algorithm for Cayley graphs.*

Proof. From Lemma 3 it is seen that all nodes have the same queue size at any step. Thus all nodes become idle (all queues are empty, hence total exchange is completed) at the same time. From Lemma 4 no message is derouted if w_0 selects shortest paths. Consequently, both conditions (3) and (4) are satisfied and the algorithm solves the problem optimally. ■

Summarizing, we just showed that there exists a class of algorithms, called node-invariant algorithms, which are able to solve the total exchange problem optimally in any Cayley network. Most reasonable algorithms, such as furthest-first, closest-first, etc. schemes are valid candidates, as long as they do not stay idle when a queue contains messages and they are replicated "consistently" at all nodes in the network. In the next section we provide a particularly simple node-invariant algorithm and we give a complete example in the context of hypercubes.

6. A simple node-invariant algorithm

Assume that we have an algorithm $\mathcal{W}$ which takes a message, looks at its destination and picks a neighbor of v_0 which lies on a shortest path from v_0 to the destination of the message. It is always possible to construct such an algorithm $\mathcal{W}$ for any network, e.g. using something like a table look-up procedure. More efficient schemes of course are possible if the structure of the network is known. For example, in a ring R_n we can have

$$\mathcal{W}(m_{v_x}(v_y)) = \begin{cases} v_1 & \text{if } y \leq n/2 \\ v_{n-1} & \text{otherwise} \end{cases}$$

(nodes v_1 and v_{n-1} are the two neighbors of node v_0).

Let us treat a message queue as a set of messages that behaves as a FIFO queue. At node v_0 we initially sort destinations in any desired order. For instance,

$$Q_{v_0}(0) = \{m_{v_0}(v_1), m_{v_0}(v_2), \ldots, m_{v_0}(v_{n-1})\}.$$

Suppose that the right end is the head of the FIFO queue and the left end is its tail. Departing messages will leave from the head of the queue. Arriving messages will join at the tail of the queue as long as they are not destined for the current node; otherwise they are immediately forwarded to the local processor. We have to guarantee that initially $Q_{v_i}(0)$ is equal to $\sigma_{v_i}(Q_{v_0}(0))$, so we let

$$Q_{v_i}(0) = \{m_{v_i}(\sigma_{v_i}(v_1)), m_{v_i}(\sigma_{v_i}(v_2)), \ldots, m_{v_i}(\sigma_{v_i}(v_{n-1}))\}.$$

$\mathcal{A}_{v_i}$: $(i = 0, 1, \ldots, n-1)$

At $t = 0$ set

$$Q_{v_i} = \Big\{ m_{v_i}(\sigma_{v_i}(v_1)), m_{v_i}(\sigma_{v_i}(v_2)), \ldots, m_{v_i}(\sigma_{v_i}(v_{n-1})) \Big\},$$

and let

$f_{v_i}(Q_{v_i})$: select the message at the head of the queue Q_{v_i},

$w_{v_i}(m)$: if $m = f_{v_i}(Q_{v_i})$, select neighbor $\sigma_{v_i}\Big(\mathcal{W}(\sigma_{v_i}^{-1}(m))\Big)$,

Figure 1. *An optimal total exchange algorithm for Cayley networks. The queues are FIFO. Messages join at the left end and depart from the right end of the queue.*

The local algorithm $\mathcal{A}_{v_i} = (f_{v_i}, w_{v_i})$ is defined as follows:

$$f_{v_i}(Q): \quad \text{select the message at the head of the queue } Q.$$

It is trivial to see that $f_{v_i}(\sigma_{v_i}(Q)) = \sigma_{v_i}(f_{v_0}(Q))$: if m is the message at the head of Q then $\sigma_{v_i}(m)$ is obviously the message at the head of $\sigma_{v_i}(Q)$. Since $m = f_{v_0}(Q)$ and $\sigma_{v_i}(m) = f_{v_i}(\sigma_{v_i}(Q))$, it is derived that $\sigma_{v_i}(f_{v_0}(Q)) = f_{v_i}(\sigma_{v_i}(Q))$.

Finally, let σ^{-1} be the inverse mapping of $\sigma \in \Pi(G)$. The existence and the uniqueness of σ^{-1} is guaranteed by the fact the $\Pi(G)$ is a group. Given $\mathcal{W}$ we define

$$w_{v_i}(m): \quad \text{for message } m \text{ select neighbor } \sigma_{v_i}\Big(\mathcal{W}(\sigma_{v_i}^{-1}(m))\Big).$$

We only have to show that $w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}(w_{v_0}(m))$, for any message m. Notice that σ_{v_0} is taken to be the identity mapping so that w_{v_0} is actually the same as $\mathcal{W}$. Thus we have to show that $w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}(\mathcal{W}(m))$. Indeed, from the description of w_{v_i} above, we have

$$w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}\Big(\mathcal{W}(\sigma_{v_i}^{-1}(\sigma_{v_i}(m)))\Big) = \sigma_{v_i}(\mathcal{W}(m)),$$

since $\sigma_{v_i}^{-1}\sigma_{v_i}$ is the identity.

In summary, the algorithm shown in Fig. 1 is, based on Definition 2, node-invariant. Therefore, it is an optimal total exchange algorithm for any Cayley network, according to Theorem 1.

6.1. An example in hypercubes

To illustrate the theory developed in the previous sections we will construct an algorithm for hypercubes, based on the algorithm in Fig. 1. An optimal algorithm was given in [4,

$\mathcal{A}_i$: $(i = 0, 1, \ldots, n-1)$

At $t = 0$ set

$Q_i = \Big\{ m_i(i \oplus_b 1), m_i(i \oplus_b 2), \ldots, m_i(i \oplus_b (n-1)) \Big\}.$

At any step $t \geq 0$,

- select the message at the head of Q_i (say $m_x(y)$)
- send it to node $i \oplus_b 2^k$ where k is the leftmost non-zero bit position of $i \oplus_b y$.

Figure 2. *An optimal total exchange algorithm for d-dimensional hypercubes. The standard e-cube routing paths are followed at every transmission.*

pp. 81–83] but is not in explicit form, and it is based on a rather involved algorithm for the multiport model (where a node may send messages to all its neighbors simultaneously).

Let $\oplus$ be the exclusive-or (addition modulo 2) operation. If the binary representation of x is $(x_{d-1}, \ldots, x_1, x_0)$ then the bitwise exclusive-or operation, $\oplus_b$, is defined as

$$x \oplus_b y = (x_{d-1} \oplus y_{d-1}, \ldots, x_1 \oplus y_1, x_0 \oplus y_0).$$

Dropping 'v' from the name of node v_i, a hypercube Q_d has node set $V = \{0, 1, \ldots, 2^d - 1\}$. A node i has neighbors $i \oplus_b 2^0$, $i \oplus_b 2^1$, ..., $i \oplus_b 2^{d-1}$. The following is an automorphism of the hypercube [13] that maps node 0 to node i:

$$\sigma_i(x) = i \oplus_b x. \tag{11}$$

Because of the associativity of exclusive-or, it is seen that

$$\sigma_{\sigma_i(a)}(x) = i \oplus_b a \oplus_b x = \sigma_i\big(\sigma_a(x)\big),$$

for any node a, so that the set of automorphisms given by (11) for $i = 0, 1, \ldots, 2^d - 1$ satisfy (1). Because $i \oplus_b i = 0$, it is seen that $\sigma_i^{-1} = \sigma_i$. Finally, it is known that if in the binary representation of y, $y_k = 1$ for some k then neighbor 2^k of node 0 lies on a shortest path from 0 to y, that is $\mathcal{W}\big(m_x(y)\big) = 2^k$. Usually, k is selected to be the leftmost non-zero bit position of y in order to comply with the standard e-cube routing. Consequently, the algorithm of the last section takes the simple form shown in Fig. 2.

7. Discussion

We considered the total exchange problem under the single-port model in the setting of Cayley graphs. It was shown that as long as every node sends a message at every step and the message is not derouted, the optimal completion time is guaranteed. A particular type of algorithms, which we named node-invariant algorithms, always satisfy these optimality conditions and hence constitute optimal solutions to the total exchange problem.

The only requirement for our arguments to work was that the network possesses a set of isomorphisms that satisfy (1). In any network which has this property (Cayley graphs do) node invariant algorithms can be defined and utilized for the total exchange problem. We would like to see what other networks, apart from Cayley ones, possess property (1). Is (1) satisfied in any node symmetric networkΓ

As a last note, it is interesting to mention that total exchange can be viewed as a specific case of *isotropic* communication problems, as originally considered by Varvarigos and Bertsekas [19]. In our setting, a communication problem will be named isotropic if whenever node v_0 has $k_i \geq 0$ messages to send to node v_i, node v_x has k_i messages to send to $\sigma_{v_x}(v_i)$, for all $i, x = 1, 2, \ldots, n-1$. In effect, all that is required for a communication problem to be isotropic is that at time $t = 0$, $Q_{v_i} = \sigma_{v_i}(Q_{v_0})$. All our arguments and all our results are immediately applicable to any isotropic communication problem. An optimal algorithm still has to satisfy conditions (3)–(4) and any node-invariant algorithm does. Consequently, as long as Q_{v_i} is appropriately set at time $t = 0$, the algorithm in Fig. 1 is an optimal algorithm for any problem of the isotropic type.

References

[1] S. B. Akers and B. Krishnamurthy, *A group-theoretic model for symmetric interconnection networks*, IEEE Trans. Comput., 38 (1989), pp. 555–566.

[2] F. Annexstein, M. Baumslag and A. L. Rosenberg, *Group action graphs and parallel architectures*, SIAM J. Comput., 19 (1990), pp. 544–569.

[3] D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng and J. N. Tsitsiklis, *Optimal communication algorithms for hypercubes*, J. Parallel Distrib. Comput., 11 (1991), pp. 263–275.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice - Hall, Englewoods Cliffs, N.J., 1989.

[5] N. Biggs, *Algebraic Graph Theory (2nd edition)*, Cambridge University Press, Cambridge, G.B., 1993.

[6] F. Boesch and R. Tindell, *Circulants and their connectivities*, Journal of Graph Theory, 8 (1984), pp. 487–499.

[7] F. Buckley and F. Harary, *Distance in Graphs*, Addison - Wesley, Reading, Mass., 1990.

[8] G. E. Carlsson, J. E. Cruthirds, H. B. Sexton and C. G. Wright, *Interconnection networks based on a generalization of cube-connected cycles*, IEEE Trans. Comput., C-34 (1985), pp. 769–772.

[9] P. Fraigniaud and E. Lazard, *Methods and problems of communication in usual networks*, Discrete Appl. Math., 53 (1994), pp. 79–133.

[10] S. Hiranandani, K. Kennedy and C. - W. Tseng, *Compiling Fortran D for MIMD distributed-memory machines*, Commun. ACM, 35 (1992), pp. 66–80.

[11] S. L. Johnsson, *Communication efficient basic linear algebra computations on hypercube architectures*, J. Parallel Distrib. Comput., 4 (1987), pp. 133–172.

[12] S. L. Johnsson and C. - T. Ho, *Optimum broadcasting and personalized communication in hypercubes*, IEEE Trans. Comput., 38 (1989), pp. 1249–1268.

[13] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Diego, CA, 1992.

[14] D. B. Loveman, *High Performance Fortran*, IEEE Parallel Distrib. Tech., 1 (1993), pp. 25–42.

[15] Message Passing Interface Forum, *MPI: A message-passing interface standard*, Technical Report CS-94-230, University of Tennessee, Apr. 1994.

[16] J. Misŝić and Z. Jovanović, *Communication aspects of the star graph interconnection network*, IEEE Trans. Parall. Distrib. Syst., 5 (1994), pp. 678–687.

[17] F. P. Preparata and J. Vuillemin, *The cube-connected cycles: a versatile network for parallel computation*, Commun. ACM, 24 (1981), pp. 300–309.

[18] Y. Saad and M. H. Schultz, *Data communications in hypercubes*, J. Parallel Distrib. Comput., 6 (1989), pp. 115–135.

[19] E. A. Varvarigos and D. P. Bertsekas, *Communication algorithms for isotropic tasks in hypercubes and wraparound meshes*, Parallel Comput., 18 (1992), pp. 1233–1257.