Optimal Total Exchange in Cayley Graphs*

VASSILIOS V. DIMAKOPOULOS
Dept. of Computer Science,
University of Ioannina,
P.O. Box 1186, Ioannina, Greece, GR-45110
Tel: +30-651-98809, Fax: +30-651-98890
E-mail: dimako@cs.uoi.gr

NIKITAS J. DIMOPOULOS
Dept. of Electrical and Computer Engineering,
University of Victoria
P.O. Box 3055, Victoria, B.C., Canada, V8W 3P6.
Tel: +1-250-7218902, Fax: +1-250-7216052,
E-mail: nikitas@ece.uvic.ca

*This research was supported in part through grants from NSERC and the University of Victoria. A preliminary version of this work was presented in the European Conference on Parallel Processing, EUROPAR' 96, Lyon, France, Aug. 1996
Optimal Total Exchange in Cayley Graphs

VASSILIOS V. DIMAKOPOULOS (IEEE Member)
Department of Computer Science,
University of Ioannina

NIKTAS J. DIMOPOULOS (Senior IEEE Member)
Department of Electrical and Computer Engineering,
University of Victoria

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 or all-to-all personalized communication 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, node-invariant algorithms, total exchange (all-to-all personalized communication)
1 Introduction

Collective communications for distributed-memory multiprocessors have received considerable attention, as for example is evident from their inclusion in the Message Passing Interface standard [17] and from their importance in supporting various constructs in High Performance Fortran [12, 16]. This is easily justified by their frequent appearance in parallel numerical algorithms [13, 5].

Broadcasting, scattering, gathering, multinode broadcasting (sometimes called gossiping) and total exchange constitute a set of representative information dissemination problems 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 the literature mostly concentrating in hypercubes and tori (e.g. [20, 14, 4, 21, 10]). 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 [11]:

- 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 [5, pp. 81-83] for hypercubes (although highly involved), in [18] for star graphs, and in [10] for general cartesian product networks.

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 is well-known [15]. Communication algorithms for recently proposed Cayley graphs either do not address the total exchange problem (e.g. in [3] for stars and pancakes, and in [23] for cyclic-cubes) or are not strictly optimal under the model we consider (e.g. the proposed total exchange algorithm for the macro-stars in [22]). In contrast, our method achieves absolute optimality as far as completion time is concerned. In the case of hypercubes and star graphs, where optimal solutions are already known, our method can still be important since it leads to much simpler algorithms, as shown in Section 6. Furthermore, what is more important is that the developed theory is not tied to a particular topology; it is quite general and applies to any Cayley graph.

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 $\ell$ 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 $\sigma$ 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 $G$ and an associative binary operation ‘$\cdot$’ on $G$ with the following two properties. There exists an identity element — that is an element $\epsilon \in G$ for which $a \cdot \epsilon = \epsilon \cdot a = a$ for all $a \in G$ — and for each $a \in G$ there exists an inverse element, denoted by $a^{-1}$ — that is an element $a^{-1} \in G$ 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 [6, 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 \( \Gamma \) (in order to avoid edges from a node to itself) and that \( \Gamma \) is closed under inverses (so that the graph is in effect undirected).

The class includes quite important networks such as the hypercube, the (wrapped) butterfly, the cube-connected cycles [2, 19, 9]. Also, connected circulant graphs [7] (which include the rings) are Cayley networks [6]. More recently proposed Cayley graphs include the cyclic-cubes [23] and the macro-stars [22].

3 An automorphism property of Cayley graphs

Consider a Cayley graph \( G \) with node set \( V = \mathcal{G} = \{ v_0, v_1, \ldots, v_{n-1} \} \), 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 \( V \). It is easily seen that this mapping is an automorphism of the graph [1]. Let \( \Sigma(G) \) be the set of the \( n \) mappings defined by (1) for \( i = 0, 1, \ldots, n - 1 \):

\[
\Sigma(G) = \{ \sigma_{v_i} \mid i = 0, 1, \ldots, n - 1 \}.
\]

The mappings in \( \Sigma(G) \) have the following properties:

- \( \sigma_{v_i} \) maps \( v_0 \) to \( v_i \)
- \( \sigma_{v_0} \) is the identity mapping
- If \( i \neq j \), then:

\[
\sigma_{\sigma_{v_i}(v_j)}(v_x) = \sigma_{v_i}(v_j) \cdot v_0^{-1} \cdot v_x
\]

6
\[ v_i \cdot v_0^{-1} \cdot v_j \cdot v_0^{-1} \cdot v_x = \sigma_{v_i}(v_j \cdot v_0^{-1} \cdot v_x) = \sigma_{v_i}(\sigma_{v_j}(v_x)), \]
that is,
\[ \sigma_{\sigma_{v_i}(v_j)} = \sigma_{v_i} \sigma_{v_j}, \quad (2) \]
the composition of mappings \( \sigma_{v_i} \) and \( \sigma_{v_j} \).

Notice that \( \Sigma(G) \) may not be the only set of automorphisms which satisfy (2). 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 \( n \). An easy 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. Actually, the above mappings work for any (connected) circulant graph.

During total exchange nodes are required to send messages to various destinations. If a node holds a number of messages to be forwarded, at each step it must select one of them and send it to one of its neighboring nodes. Thus, before the selected message is transmitted the node must choose a neighbor according to some predefined rules. What we would like to establish is that at any step all nodes in the network choose “equivalent” neighbors. This way we can expect that all nodes operate in a uniform manner, and whatever occurs at node \( v_0 \) occurs “equivalently” at all the other nodes. The preceding comments are formalized in the following lemma.

**Lemma 1** Let \( v_0 \) pick one of its neighbors, \( v_a \), and let every other node \( v_i \), \( i = 1, 2, \ldots, n - 1 \), pick 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 \( i \neq j \).
Let us assume that for some \( j \neq i \) we have \( \sigma_{v_i}(v_a) = \sigma_{v_j}(v_a) = v_k \), for some
\( k \). Then \( \sigma_{v_k} = \sigma_{\sigma_{v_i}(v_a)} \), and from (2), \( \sigma_{v_k} = \sigma_{v_j} \sigma_{v_a} \). Similarly, \( \sigma_{v_k} = \sigma_{v_k} \sigma_{v_a} \).
Consequently, \( \sigma_{v_i} \sigma_{v_a} = \sigma_{v_j} \sigma_{v_a} \), or \( \sigma_{v_k} = \sigma_{v_j} \), which cannot hold.

(b) Let \( v_b \) be the node that picks \( v_{b_0} \) that is \( v_0 = \sigma_{v_b}(v_a) \). Since \( v_i = \sigma_{v_i}(v_0) \)
(\( \sigma_{v_i} \) maps \( v_0 \) to \( v_i \)), we obtain \( v_i = \sigma_{v_i}(\sigma_{v_b}(v_a)) \). From (2) we get \( v_i = \sigma_{\sigma_{v_i}(v_b)}(v_a) \). This means that node \( \sigma_{v_i}(v_b) \) picked \( v_i \).

\[ \square \]

4 Lower bound on total exchange time

In the total exchange problem, every node \( v \) has to send \( n-1 \) distinct messages,
one to each of the other nodes in an \( n \)-node network. 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

\[ s(v) = \sum_{d=1}^{e(v)} dn_d \]

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 [8] 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

\[ T \geq \frac{S_G}{n}. \]  

(3)
Because all nodes in a node symmetric graph have the same status [8], 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 (3):

\begin{align}
\text{all nodes are busy all the time, and,} & \quad (4) \\
\text{every transmitted message gets closer to its destination.} & \quad (5)
\end{align}

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 \textit{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 \textit{message queue}, $Q_{vi}$, where incoming messages from neighbors are deposited until they are scheduled for transfer to some other node. Initially, $Q_{vi}$ contains the $n - 1$ messages of $v_i$ for the other nodes. As time passes, messages originating from other nodes join this queue on their way to their destination. 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 $\mathcal{A}_{vi}$ operates in order to schedule the message transfers. Whenever there exist messages in $Q_{vi}$, algorithm $\mathcal{A}_{vi}$ is responsible for selecting:

(i) the message to leave in the next time unit, and,

(ii) the neighbor of $v_i$ to which the message will be sent.

\textbf{Definition 1} A distributed total exchange algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{n-1})$ is a collection of local algorithms, algorithm $\mathcal{A}_{vi}$ running on node $v_i$, $i = 0, 1, \ldots, n - 1$. Algorithm $\mathcal{A}_{vi}$ is written as $\mathcal{A}_{vi} = (f_{vi}, w_{vi})$, where, given a
message queue $Q_{v_i}$, procedure $f_{v_i}$ 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 fix a node in the network (say $v_0$) and to make all the other nodes behave in a similar way with $v_0$. We will design the algorithms in such a way that every node $v_i$ selects a message “corresponding” to the message selected by node $v_0$ and sends 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. This uniformity is highly desirable because it will force all nodes to have “corresponding” messages queues at each step; hence we can argue that message queues always 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 (4) 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 $\sigma \in \Sigma(G)$, let $\sigma(m_{v_x}(v_y))$ be the message of node $\sigma(v_x)$ destined for node $\sigma(v_y)$, i.e.

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

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

$$\sigma(Q) \overset{\text{def}}{=} \{ \sigma(m_{v_x}(v_y)) \mid m_{v_x}(v_y) \in Q \}.$$

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

$$f_{v_i}(\sigma_{v_i}(Q)) = \sigma_{v_i}(f_{v_0}(Q)),$$

$$w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}(w_{v_0}(m)).$$
Lemma 2 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

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

for all $t \geq 0$, where $\sigma_{v_i}$ is as given in Definition 2.

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 for $j = 0$, $\sigma_{v_i}(v_0) = v_i$). Thus the message set $S = \{m_{v_0}(\sigma_{v_i}(v_j)) \mid j = 1, 2, \ldots, n-1\}$ is the same as the set $S' = \{m_{v_0}(v_k) \mid k = 0, 1, \ldots, n-1, k \neq i\}$. Notice that $S' = Q_{v_i}(0)$. If we write $v_i$ as $\sigma_{v_i}(v_0)$, and use (2) it is straightforward to derive that $S = \sigma_{v_i}(Q_{v_0}(0))$, showing that $Q_{v_i}(0) = \sigma_{v_i}(Q_{v_0}(0))$.

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

$$Q_{v_i}(t) = \sigma_{v_i}(Q_{v_0}(t)). \quad (6)$$

For time $t + 1$ we proceed as follows. For simplicity, let $m_{s(v_i)} = f_{v_i}(Q_{v_i}(t))$ 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 will be sent. From (6) and the definition of node-invariant algorithms it is easily seen that

$$m_{s(v_i)} = \sigma_{v_i}(m_{s(v_0)}), \quad (7)$$

$$v_{s(v_i)} = \sigma_{v_i}(v_{s(v_0)}). \quad (8)$$

Now notice that $v_{s(v_0)}$ is the neighbor $v_0$ picked to send the message to. From (8) it is seen that Lemma 1 applies so 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

$$v_{r(v_i)} = \sigma_{v_i}(v_{r(v_0)}). \quad (9)$$
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

\[
m_{r(v_i)} = m_{s(v_r(v_i))} \\
= \sigma_{v_r(v_i)}(m_{s(v_0)}) \\
= \sigma_{\sigma_{v_r(v_0)}(m_{s(v_0)})}(m_{s(v_0)}) \\
= \sigma_{v_i}(\sigma_{v_r(v_0)}(m_{s(v_0)})),
\]

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

\[
m_{r(v_i)} = \sigma_{v_i}(m_{r(v_0)}).
\]

To recapitulate, any node \( v_i \) selects a message \( m_{s(v_i)} \) given by (7), sends it to some node \( v_{s(v_i)} \) given by (8) and receives a message \( m_{r(v_i)} \) given by (10) from some node \( v_{r(v_i)} \) given by (9). If the destination of \( m_{r(v_0)} \) is node \( v_0 \), then from (10) 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,

\[
Q_{v_0}(t + 1) = Q_{v_0}(t) \cup \{m_{r(v_0)}\} \setminus \{m_{s(v_0)}\},
\]

where ‘\( \setminus \)’ 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 (6), (7), (10) and (11),

\[
Q_{v_i}(t + 1) = \sigma_{v_i}(Q_{v_i}(t) \cup \{m_{r(v_0)}\}) \setminus \{\sigma_{v_i}(m_{s(v_0)})\} \\
= \sigma_{v_i}(Q_{v_i}(t) \cup \{m_{r(v_0)}\} \setminus \{m_{s(v_0)}\}) \\
= \sigma_{v_i}(Q_{v_i}(t + 1)),
\]

concluding the induction. \( \square \)
Lemma 3 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 $m_{v_i}(v_y)$ out of its queue and sends it to some neighbor $v_s$, then any node $v_i$ selects message $\sigma_{v_i}(m_{v_i}(v_y))$ and sends it to neighbor $\sigma_{v_i}(v_s)$ as we have already seen (equations (7)–(8)). 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 [6]. That is, if $\sigma$ is an automorphism of a graph $G$ then $\text{dist}(v, u) = \text{dist}(\sigma(v), \sigma(u))$ for any two vertices $v$ and $u$ of $G$. If $v_0$ does not deroute then $\text{dist}(v_0, v_y) = \text{dist}(v_i, v_y) + 1$. Then, we must have $\text{dist}(v_i = \sigma_{v_i}(v_0), \sigma_{v_i}(v_y)) = \text{dist}(\sigma_{v_i}(v_s), \sigma_{v_i}(v_y)) + 1$ and $\sigma_{v_i}(v_s)$ indeed lies on a shortest path from $v_i$ to $\sigma_{v_i}(v_y)$. \hfill $\square$

Theorem 1 Any node-invariant algorithm for which $w_{\text{in}}$ selects shortest paths is an optimal total exchange algorithm for Cayley graphs.

Proof. From Lemma 2 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 3 no message is derouted if $w_{\text{in}}$ selects shortest paths. Consequently, both conditions (4) and (5) are satisfied and the algorithm solves the problem optimally. \hfill $\square$

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 $\mathcal{W}$ that knows the shortest routes from node $v_0$ to any other node. In other words, $\mathcal{W}$ 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 a table look-up procedure. More efficient schemes are possible if the structure of the network is known. For example, in a ring $R_n$ we can have

$$\mathcal{W}(m_{v_0}(v_y)) = \begin{cases} v_1 & \text{if } y \leq n/2 \\ v_{n-1} & \text{otherwise} \end{cases}$$

(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 $Q_{v_0}(0)$ is equal to $\sigma_{v_0}(Q_{v_0}(0))$, so we let

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

The local algorithm $A_{v_i} = (f_{v_i}, w_{v_i})$ is defined as follows:

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

It is trivial to see that $f_{v_i}(\sigma_{v_i}(Q)) = \sigma_{v_i}(f_{v_i}(Q))$: 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 $\sigma^{-1}$ be the inverse mapping of $\sigma$. The existence and the uniqueness of $\sigma^{-1}$ is guaranteed by the fact the the set $\Pi(G)$ of the automorphisms of
the graph is a group. Given \( \mathcal{W} \) we define:

\[
      w_{v_i}(m) : \text{ for message } m \text{ select neighbor } \sigma_{v_i}(\mathcal{W}(\sigma^{-1}_{v_i}(m))).
\]

We only have to show that \( w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}(w_{v_i}(m)) \), for any message \( m \).
Notice that \( \sigma_{v_i} \) is taken to be the identity mapping so that \( w_{v_i} \) 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:

\[
      w_{v_i}(\sigma_{v_i}(m)) = \sigma_{v_i}(\mathcal{W}(\sigma^{-1}_{v_i}(\sigma_{v_i}(m)))) = \sigma_{v_i}(\mathcal{W}(m)),
\]

since \( \sigma^{-1}_{v_i} \sigma_{v_i} \) 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: 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 [3, 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 \( i \oplus_b 2^0, i \oplus_b 2^1, \ldots, i \oplus_b 2^{d-1} \). In order to apply the algorithm in Fig. 1 we need to identify three quantities:

- **Defining a simple \( \Sigma(G) \):**
  - The following is an automorphism of the hypercube [15] that maps node
0 to node $i$:

$$\sigma_i(x) = i \oplus_b x. \quad (12)$$

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

$$\sigma_{\sigma_{i}(j)}(x) = i \oplus_b j \oplus_b x = \sigma_i(\sigma_j(x)),$$

for any node $j$, so that the set of automorphisms given by (12) for $i = 0, 1, \ldots, 2^d - 1$ satisfy (2).

- **Obtaining $\sigma_i^{-1}$:**
  Because $i \oplus_b i = 0$, it is seen that $\sigma_i^{-1} = \sigma_i$.

- **Constructing $W$:**
  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 $W(m_x(y)) = 2^k$. 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 and constitutes an optimal total exchange algorithm for hypercubes.

## 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 (2). 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 (2). Is (2) satisfied in any node symmetric network?

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 [21]. 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 $v_e$ has $k_i$ messages to send to $\sigma_{v_i}(v_i)$, 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 (4)–(5) 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.

A interesting direction of future research is the development of total exchange algorithms for multiport Cayley networks. In such a setting, each node has the capabilities to communicate with all its neighbors simultaneously. Although node-invariant algorithms could still be significant, it seems that they are not sufficient to enforce optimality. It is not enough to keep all nodes busy; one must rather keep all links busy. In such a case edge symmetries should play a more important role than node symmetries.

References


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

At \( t = 0 \) set
\[
Q_{v_i} = \left\{ m_{v_i}(\sigma_{v_i}(v_1)), m_{v_i}(\sigma_{v_i}(v_2)), \ldots, m_{v_i}(\sigma_{v_i}(v_{n-1})) \right\},
\]
and let
\[
f_{v_i}(Q_{v_i}): \text{ select the message at the head of the queue } Q_{v_i},
\]
\[
w_{v_i}(m): \text{ if } m = f_{v_i}(Q_{v_i}), \text{ select neighbor } \sigma_{v_i} \left( W(\sigma_{v_i}^{-1}(m)) \right),
\]

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

At $t = 0$ set

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

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.