An Optical Interconnection Structure Based on the Dual of a Hypercube

Yueming Li †, Jie Wu‡, and S.Q. Zheng †

†Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803

‡Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431

May 12, 1997

Abstract

A new class of interconnection networks, the hypernetworks, have been proposed recently. Hypernetworks are characterized by hypergraphs. Compared with point-to-point networks, they allow for increased resource-sharing and communication bandwidth utilization, and they are especially suitable for optical interconnects. One way to derive a hypernetwork is by finding the dual of a point-to-point network. Hypercube Q_n, where n is the dimension, is a popular point-to-point network [7]. In this paper, we consider using the dual Q_n^* of hypercube of Q_n as an interconnection network. We investigate the properties of Q_n^*, and present a set of fundamental data communication algorithms for Q_n^*. Our results indicate that the Q_n^* hypernetwork is a useful and promising interconnection structure for high-performance parallel and distributed computing systems.

Keywords: hypercubes, hypernetworks, interconnection networks, optical buses, optical interconnections, parallel and distributed computing.

1 Introduction

Designing high bandwidth, low latency and scalable interconnection networks is a great challenge in the construction of high-performance parallel computer systems. Traditionally, interconnection networks are characterized by graphs. Network topologies under graph models have been extensively investigated. Many network structures have been proposed, and some have been implemented. Observed the improving electrical bus and switching technologies and maturing optical interconnection technologies, Zheng pointed out the conventional graph structure is no longer adequate for the design and analysis of the new generation interconnection structures and proposed a new class of interconnection networks, the hypernetworks [14].

The class of hypernetworks is a generalization of point-to-point networks, and it contains point-to-point networks as a subclass. In a hypernetwork, the physical communication medium (a hyperlink) is accessible to multiple (usually, more than two) processors. Optical fibers and devices are suitable for implementing hyperlinks. The relaxation on the number of processors that can be connected by a link provides more design alternatives so that greater flexibilities in trade-offs of contradicting design goals are possible. The underlying graph theoretic tool for investigating hypernetworks is hypergraph theory [2]. Hypergraphs are used to model hypernetworks. Hypernetwork designs have been formulated as an optimization problem of constructing constrainted hypergraphs. Interested readers may refer to [14, 15, 17] for more justifications, design issues and implementation aspects of hypernetworks.

Existing results in hypergraph theory and combinatorial block design theory, which is closely related to hypergraph theory, can be used to design hypernetworks. For example, in [15], Zheng introduced several low diameter hypernetworks based on the concept of Steiner Triple System. In [17], Zheng and Wu proposed a scheme for constructing a new hypernetwork from an existing one using the concept of dual graph in hypergraph theory. They showed that the dual H^* of any given hypergraph H is a hypergraph that have some properties related to the properties of H so that one can investigate the properties of H^* based on the properties of H. Since the structure of H and its dual H^* can be drastically different, finding hypergraph duals can be considered as a general approach to the design of new hypernetworks. They investigated the structure of the dual K_n^* of an n-vertex complete point-to-point network K_n.

Hypercube is a popular point-to-point network which has many desirable features such as small diameter, symmetry, and supporting a large class of efficient parallel algorithms. In this paper, we propose a class of hypernetworks, the Q_n^* (read as Q_n star) hypernetworks. The Q_n^* hypernetwork is the dual of the n-dimensional hypercube Q_n. We discuss the topological and fault tolerance aspects of Q_n^*, and present a set of parallel data communication algorithms for Q_n^*. Our results indicate that the Q_n^* hypernetwork is a useful and promising interconnection network for high-performance parallel and distributed computing systems.

This paper is organized as follows. In Section 2, we introduce several basic concepts of hypergraphs and hypernetworks. In Section 3, we discuss the relations between a hypergraph and its dual, and show that hypergraph duals can be used to derive new hypernetworks. We then introduce the Q_n^* hypernetwork, and show that it possesses a set of desirable properties. In Section 4 we present a set of fundamental data communication algorithms for the Q_n^* hypernetwork, and analyze their performances based on the bus implementation of hyperlinks. Finally, in Section 5, we discuss the generalizations and implications of this work.

2 Preliminaries

Hypergraphs are used as underlying graph models of hypernetworks. A *hypergraph* [2] $H = (V, E)$ consists of a set $V = \{v_1, v_2, \cdots, v_n\}$ of vertices, and a set $E = \{e_1, e_2, \cdots, e_m\}$ of hyperedges such that each e_i is a non-empty subset of V and $\{v | v \in e_i, 1 \leq i \leq m\} = V$. An edge e contains a vertex v if $v \in e$. If $e_i \subseteq e_j$ implies that $i = j$, then H is a *simple hypergraph*. In this article, we only consider simple hypergraphs. When the cardinality of an edge e, denoted as $|e|$, is 1, it corresponds to a selfloop edge. If all the edges have cardinality 2, then H is a graph that corresponds to a point-to-point network. A hypergraph of n vertices and m hyperedges can also be defined by its $n \times m$ incidence matrix A with columns representing edges and rows representing vertices such that $a_{i,j} = 0$ if $v_i \notin e_j$, $a_{i,j} = 1$ if $v_i \in e_j$.

For a subset $E^{'}$ of E, we call the hypergraph $H'(V', E')$ such that $V' = \{v | v \in e, e \in E^{'}\}$ the *partial hypergraph of H generated by the set $E^{'}$*. For a subset U of V, we call the hypergraph $H''(V'', E'')$ such that $E'' = \{e_i \cap U | e_i \cap U \neq \phi, 1 \leq i \leq m\}$ and $V'' = \{v | v \in e, e \in E''\}$ the *sub-hypergraph induced by the set U*. Note that such an induced sub-hypergraph may or may not be a simple hypergraph.

The degree $d_H(v_i)$ of v_i in H is the number of edges in V that contain v_i. A hypergraph in which all the vertices have the same degree is said to be *regular*. The *degree of hypergraph* H, denoted by $\Delta(H)$, is defined as $\Delta(H) = \max_{v_i \in V} d_H(v_i)$. A regular hypergraph of degree k is called *k-regular hypergraph*. The *rank* $r(H)$ and *antirank* $s(H)$ of a hypergraph H is defined as $r(H) = \max_{1 \leq j \leq m} |e_j|$ and $s(H) = \min_{1 \leq j \leq m} |e_j|$, respectively. We say that H is a *uniform hypergraph* if $r(H) = s(H)$. A uniform hypergraph of rank k is called *k-uniform hypergraph*. A hypergraph is *vertex* (resp. *hyperedge*) *symmetric* if for any two vertices (resp. hyperedges) v_i and v_j (resp. e_i and e_j) there is an automorphism of the hypergraph that maps v_i to v_j (resp. e_i to e_j).

In a hypergraph H, a path of length q is defined as a sequence $(v_{i_1}, e_{j_1}, v_{i_2}, e_{j_2}, \cdots, e_{j_q}, v_{i_{q+1}})$ such that (1) $v_{i_1}, v_{i_2}, \cdots, v_{i_{q+1}}$ are all distinct vertices of H; (2) $e_{j_1}, e_{j_2}, \cdots, e_{j_q}$ are all distinct edges of H; and (3) $v_{i_k}, v_{i_{k+1}} \in e_{j_k}$ for $k = 1, 2, \cdots, q$. A path from v_i to v_j, $i \neq j$, is a path in H with its end vertices being v_i and v_j. A hypergraph is *connected* if there is a path connecting any two vertices. We only consider connected hypergraphs. A hypergraph is *linear* if $|e_i \cap e_j| \leq 1$ for $i \neq j$, i.e., two distinct buses share at most one common vertex. For any two distinct vertices v_i and v_j in a hypergraph H, the distance between them, denoted by $dis(v_i, v_j)$, is the length of the shortest path connecting them in H. Note that $dis(v_i, v_i) = 0$. The *diameter* of a hypergraph $H = (V, E)$, denoted by $\delta(H)$, is defined by $\delta(H) = \max_{v_i, v_j \in V} dis(v_i, v_j)$. More concepts in hypergraph theory can be found in [2].

A *hypernetwork* M is a network whose underlying structure is a hypergraph H, in which each vertex v_i corresponds to a unique processor P_i of M, and each hyperedge e_j corresponds to a *connector* that connects the processors represented by the vertices in e_j. A connector is loosely defined as an electronic or a photonic component through which messages are transmitted between

connected processors, not necessarily simultaneously. We call a connector a *hyperlink*.

Unlike a point-to-point network, in which a link is dedicated to a pair of processors, a hyperlink in a hypernetwork is shared by a set of processors. A hyperlink can be implemented by a bus or a crossbar switch. Current optical technologies allow a hyperlink to be implemented by optical waveguides in a folded-bus using time-division multiplexing (TDM). Free-space optical or optoelectronic switching devices such as bulk lens, microlens array, and spatial light modulator (SLM) can also be used to implement hyperlinks. A star coupler, which uses wavelength-division multiplexing (WDM), can be considered either as a generalized bus structure or as a photonic switch, is another implementation of a hyperlink. Similarly, an ATM switch, which uses a variant TDM, is a hyperlink. In the rest of this paper, the following pairs of terms are used interchangeably: (hyper)edges and (hyper)links, vertices and processors, point-to-point networks and graphs, and hypernetworks and hypergraphs.

The problem of designing efficient interconnection networks can be considered as a constrainted optimization problem. For example, the goal of designing point-to-point networks is to find well-structured graphs (whose ranks are fixed, as a constant 2) with small degrees and diameters. In hypernetwork design, the relaxation on the number of processors that can be connected by a hyperlink (i.e. the rank of the hyperlink) provides more design alternatives so that greater flexibilities in trade-offs of contradicting design goals are possible. The detailed discussion is beyond the scope of this paper.

3 Dual Hypernetworks and Q_n^* Hypernetworks

The *dual* of a hypergraph $H = (V, E)$ with vertex set $V = \{v_1, v_2, \cdots, v_n\}$ and hyperedge set $E = \{e_1, e_2, \cdots, e_m\}$ is a hypergraph $H^* = (V^*, E^*)$ with vertex set $V^* = \{v_1^*, v_2^*, \cdots, v_m^*\}$ and hyperedge set $E^* = \{e_1^*, e_2^*, \cdots, e_n^*\}$ such that v_j^* corresponds to e_j with hyperedges $e_i^* = \{v_j^* | v_i \in e_j$ and $e_j \in E\}$. In other words, H^* is obtained from H by interchanging of vertices and hyperedges in H. The incidence matrix of H^* is the transpose of the incidence matrix of H. Thus, $(H^*)^* = H$. The following relations between a hypergraph and its dual are apparent [17].

Proposition 1 *H is r-uniform if and only if H^* is r-regular.*

Proposition 2 *The dual of a linear hypergraph is also linear.*

Proposition 3 *A hypergraph H is vertex symmetric if and only if H^* is hyperedge symmetric .*

Proposition 4 *The dual of a sub-hypergraph of H is a partial hypergraph of the dual hypergraph H^*.*

Since $(H^*)^* = H$, all the above propositions still hold after interchanging H with H^*.

Proposition 5 $\delta(H) - 1 \leq \delta(H^*) \leq \delta(H) + 1$.

Propositions 1 - 5 show that some properties of the dual hypergraph H^* of a given hypergraph H can be derived from properties of H. For example, if H is a ring, then H^* is isomorphic to H. However, in general, the structures of H and its dual H^* can be drastically different. Finding hypergraph duals can be considered as a general approach to the design of new hypernetworks.

We consider using the dual Q_n^* of the hypercube Q_n as a hypernetwork. An n-dimensional hypercube Q_n consists of 2^n vertices, each being labeled by a unique n-bit binary number. Two vertices are connected by an edge if and only if their binary labels are distinct in one bit position. Properly labeling the vertices and hyperedges in Q_n^* can greatly simplify its use as a communication network. Vertex labels are used as processor addresses. Similarly, hyperedge labels are used as the unique names of hyperlinks.

Let I_n be the set of non-negative integers that can be represented by n-bit binary numbers. For $l, u \in I_n$, we use $d(l, u)$ to denote the number of different bits in the binary representations of l and u, i.e. $d(l, u)$ is the Hamming distance between the binary representations of l and u. We use a pair of integers to label a vertex in the Q_n^* hypernetwork.

Definition 1 Let $N_n = n2^{n-1}$ for $n \geq 2$. The Q_n^* hypernetwork is a hypergraph with vertex set $\{\langle l, u\rangle | l, u \in I_n,\ l < u,\ \text{and}\ d(l, u) = 1\}$ of N_n vertices and 2^n hyperlinks, $e_0, e_1, \cdots, e_{2^n-1}$. Each vertex $\langle l, u\rangle$ is connected to exactly two hyperedges e_l and e_u.

Example 1 *The incidence matrix A of Q_3^* is*

$$A = \begin{array}{c} \\ \langle 0,1\rangle \\ \langle 0,2\rangle \\ \langle 1,3\rangle \\ \langle 2,3\rangle \\ \langle 0,4\rangle \\ \langle 1,5\rangle \\ \langle 2,6\rangle \\ \langle 3,7\rangle \\ \langle 4,5\rangle \\ \langle 4,6\rangle \\ \langle 5,7\rangle \\ \langle 6,7\rangle \end{array} \begin{array}{c} \begin{array}{cccccccc} e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \left(\begin{array}{cccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array} \right) \end{array}.$$

The transpose of A is

$$A^T = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} e_{0,1} & e_{0,2} & e_{1,3} & e_{2,3} & e_{0,4} & e_{1,5} & e_{2,6} & e_{3,7} & e_{4,5} & e_{4,6} & e_{5,7} & e_{6,7} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Clearly, A^T is the incidence matrix of the hypercube Q_3. Figure 1 shows the bus implementation of the Q_3^* hypernetwork, whose incidence matrix A is given above. Its corresponding hypercube, whose incidence matrix is A^T, is shown in Figure 2, where each edge is labeled by its two end vertices.

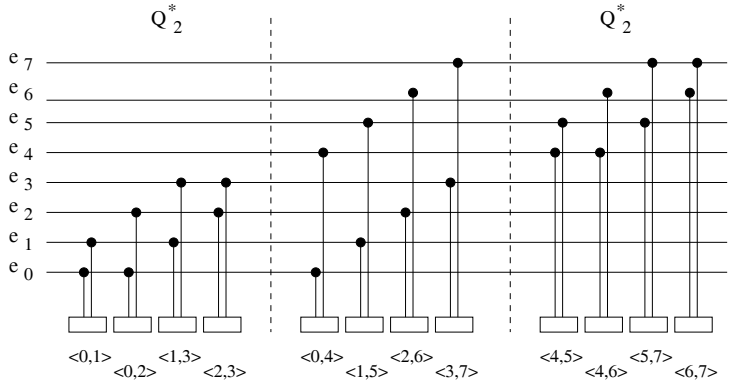

Figure 1: Bus implementation of Q_3^*.

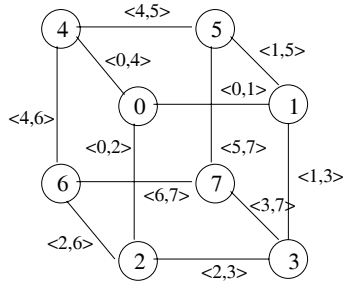

Figure 2: Hypercube Q_3 corresponding to Q_3^*.

□

The Q_n^* ($n \geq 1$) hypernetwork can also be defined in a recursive way. One can easily observe that Q_n^* can be constructed using two copies of Q_{n-1}^* and 2^{n-1} additional vertices placed in between (see Figures 1 and 3). For brevity, we omit the recursive definition of Q_n^*.

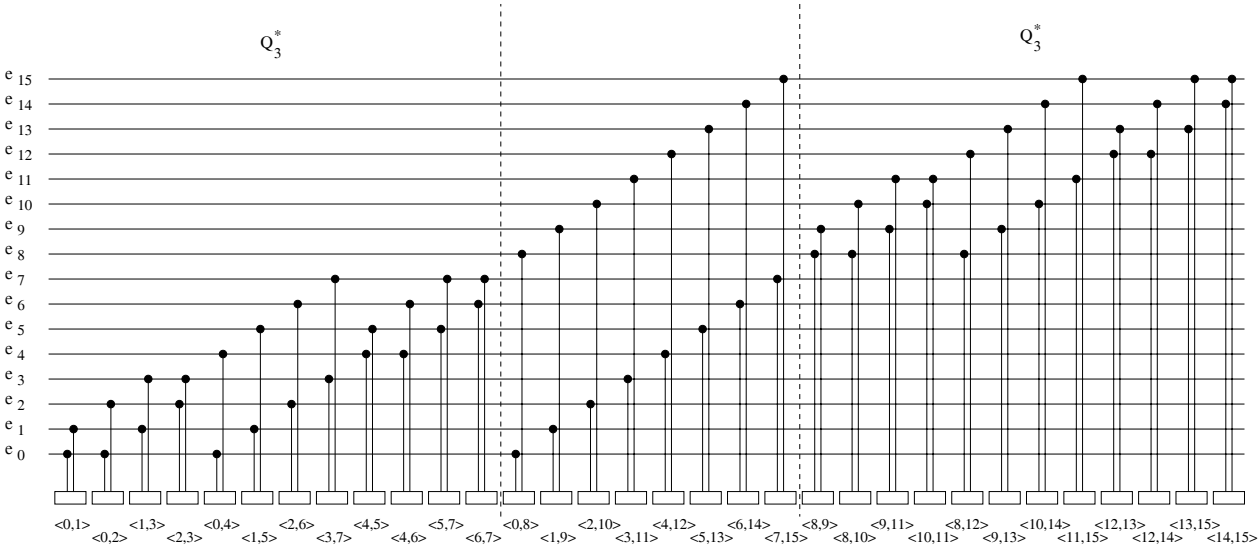

Figure 3: Bus implementation of Q_4^*.

Based on the properties of hypercube Q_n and Propositions 1 to 4, we have the following fact:

Fact 1: Q_n^* is 2-regular, n-uniform, linear, and vertex and hyperedge symmetric.

Since the diameter of Q_n is n, Property 5 indicates that the diameter of Q_n^* is at most $n+1$. We show that the diameter of Q_n^* is also n. Let $dis(\langle l, u\rangle, \langle l', u'\rangle)$ denote the distance between vertices $\langle l, u\rangle$ and $\langle l', u'\rangle$ in Q_n^*.

Lemma 1 *For any two vertices* $\langle l, u\rangle$ *and* $\langle l', u'\rangle$ *in* Q_n^*, $dis(\langle l, u\rangle, \langle l', u'\rangle) = \min\{d(l, l'), d(l, u'), d(u, l'), d(u, u')\} + 1$.

Proof. We view $\langle l, u\rangle$ and $\langle l', u'\rangle$ as two edges in Q_n. The minimal path connecting these two edges is one from one end node of $\langle l, u\rangle$ to one end node of $\langle l', u'\rangle$. Therefore, the distance is the length of a minimal path between two end nodes plus one. □

Lemma 2 *For any two vertices* $\langle l, u\rangle$ *and* $\langle l', u'\rangle$ *in* Q_n^*, $dis(\langle l, u\rangle, \langle l', u'\rangle) \leq n$.

Proof. For two hyperedges $\langle l, u\rangle$ and $\langle l', u'\rangle$ in Q_n^*, if $d(l, l') = n$ then $d(l, u') < n$. By Lemma 1, $d(\langle l, u\rangle, \langle l', u'\rangle) = \min\{d(l, l'), d(l, u'), d(u, l'), d(u, u')\}+1 \leq \min\{d(l, l'), d(l, u')\}+1 \leq (n-1)+1 \leq n$. □

Theorem 1 *The diameter of* Q_n^* *is* n.

Proof. From lemma 2, we know that the diameter is less than or equal to n. All we need to do is to prove that there are two vertices $\langle l, u\rangle$ and $\langle l', u'\rangle$ such that $d(\langle l, u\rangle, \langle l', u'\rangle) = n - 1$. Actually, it is easy to see that vertices $\langle 000...0, 100...0\rangle$ and $\langle 111...1, 011...1\rangle$ meet the above condition. □

In Q_n^*, the number of processors is $n/2$ times the number of the hyperlinks. Each processor is attached to exactly two hyperlinks, and this simplifies the processor interface circuit design. Each hyperlink connects n processors. Suppose that a 10×10 crossbar switch is implementable and cost effective, then Q_{10}^* of 5,120 processors can be implemented using 1,024 such switches as hyperlinks.

Consider the fault tolerance aspect of the Q_n^* hypernetwork. We say that a hypernetwork H is *x-processor fault-tolerant* (resp. *y-hyperlink fault-tolerant*) if it remains connected when no more than any x processors (resp. y-hyperlinks) are removed. We have the following claim.

Theorem 2 *Q_n^* is $(2n-3)$-processor fault-tolerant and 1-hyperlink fault-tolerant.*

Proof. Consider the hypercube Q_n. Take an edge e and delete all edges that share a vertex with e, then the graph becomes disconnected. This implies that it is possible to disconnect the Q_n^* hypernetwork by removing $2(n-1)$ processors. However, removing any less than $2(n-1)$ processors from Q_n will not make the remaining part of Q_n disconnected. Hence, Q_n^* is $(2n-3)$-processor fault-tolerant. Since Q_n^* is 2-regular, removing any two hyperlinks that share a vertex v will disconnect processor v. □

4 Data Communication Algorithms for Q_n^*

In this section, we use the vertex and hyperedge labels to design data communication algorithms for the Q_n^* hypernetwork. For simplicity, we assume bus implementation of hyperlinks. In the electronic domain, the bus load, i.e. the number of processors that can be connected by a bus, is limited. Using optical fibers to implement a bus, the bus load can be increased significantly. Recently, optical bus architectures have received considerable attention (e.g. [5, 8, 9, 10, 12, 16]). Since a bus is shared by all its connected processors, the performance of a bus depends on the way it is accessed by processors. For example, one way for processors to share a bus is to use time-division multiplexing (TDM), which allocates time slots to processors so that they can only access the bus during their slots. Another way is to let processors compete for bus tenure, and use an arbiter to grant the bus tenure in an on-line fashion.

We assume a synchronous mode communication. Bus allocations, although operated dynamically, are predetermined by an off-line scheduling algorithm. This bus operational mode has been used in [3] for analyzing a multiple-bus interprocessor connection structure. Assume that all messages are of the same length. The communication performance is measured in terms of parallel message steps. We adopt these assumptions for two reasons. First, under these assumptions, it is easier to assess the capability and limitation of the proposed hypernetwork structure. Secondly, the

performance results obtained can be easily used to measure other bus communication method by either adding additional overheads, which may incur in TDM transmission and asynchronous bus allocation, or deducting transmission latency saving due to pipelining effect of a pipelined optical bus.

We discuss four types of communication operations: one-to-one communications, one-to-many communications, many-to-one communications and many-to-many communications. For each type, we present an algorithm for a representative communication operation. These communication algorithms constitute a useful set of tools for designing parallel algorithms on the Q_n^* hypernetwork.

4.1 One-to-One Communication

We consider shortest path routing between two processors. where $\langle l, u\rangle$ and $\langle l^{'}, u^{'}\rangle$ to represent the source processor and the destination processor, respectively. The idea of the shortest path routing is as follows. If the two processors share one hyperlink, the message is transmitted through that hyperlink. Otherwise, the transmission is done from hyperlink e_a toward hyperlink e_b such that $a \in \{l, u\}$, $b \in \{l^{'}, u^{'}\}$, and $d(a, b)$ is the minimal. The source processor sends the message to the processor $\langle a, c\rangle$ through hyperlink e_a such that $d(c, b) = d(a, b) - 1$. This process is recursive; that is, the processor $\langle a, c\rangle$ will then relay the message toward the processor $\langle l^{'}, u^{'}\rangle$. The following is the shortest path routing algorithm.

procedure *ROUTE*($\langle l, u\rangle$, $\langle l^{'}, u^{'}\rangle$)
begin
  Let $a \in \{l, u\}$, $b \in \{l^{'}, u^{'}\}$ such that $d(a, b) = dis(\langle l, u\rangle, \langle l^{'}, u^{'}\rangle) - 1$;
  if $a = b$ **then**
    Processor $\langle l, u\rangle$ sends the message to $\langle l^{'}, u^{'}\rangle$ using e_a
  else begin
    Select c such that $d(c, b) = d(a, b) - 1$;
    Processor $\langle l, u\rangle$ sends the message to $\langle a, c\rangle$ using e_a;
    ROUTE($\langle a, c\rangle$, $\langle l^{'}, u^{'}\rangle$)
    end
end

Theorem 3 *For any given pair of processors $\langle l, u\rangle$ and $\langle l^{'}, u^{'}\rangle$ in the Q_n^* hypernetwork, algorithm ROUTE routes a message from $\langle l, u\rangle$ to $\langle l^{'}, u^{'}\rangle$ along a minimal path in $dis(\langle l, u\rangle, \langle l', u'\rangle)$ message steps.*

Proof. The theorem dirctly follows from Lemmas 1 and 2. □

4.2 One-to-Many Communication

We consider broadcasting a message from any processor $\langle l, u\rangle$ to all other processors in Q_n^*, Given $\langle l, u\rangle$, procedure *TRANSFORM* is used to transform $\langle l, u\rangle$ to $\langle 0, 1\rangle$ and all the other $\langle a, b\rangle$ in Q_n^* to $\langle a', b'\rangle$ in constant time.

procedure *TRANSFORM* $(\langle l, u\rangle)$
begin
  for all $\langle a, b\rangle$ in Q_n^* **do in parallel**
    if $a = 0$ and $b = 1$ **then** $\langle a', b'\rangle := \langle l, u\rangle$
    else if $a = 0$ **then** $\langle a', b'\rangle := \langle \min\{l, b\}, \max\{l, b\}\rangle$
    else if $a = 1$ **then** $\langle a', b'\rangle := \langle \min\{u, b\}, \max\{u, b\}\rangle$
    else if $a = l$ **then** $\langle a', b'\rangle := \langle 0, b\rangle$
    else if $b = u$ **then** $\langle a', b'\rangle := \langle 1, a\rangle$
  endfor
end

By the symmetry of the Q_n^* hypernetwork, we know that the new identities $\langle a', b'\rangle$ assigned to processors of Q_n^* satisfy the connectivities of Q_n^*. We only need to describe an algorithm which broadcasts a message from $\langle 0, 1\rangle$ to all processors in Q_n^*.

We use 0^k to represent k consecutive 0's, e.g. $0^3 = 000$. Let $Q_k = 0^{n-k-1}0*^k$ and $Q_k^{'} = 0^{n-k-1}1*^k$ denote the k-dimensional subcube of Q_n induced by all vertices whose left $n-k$ bits are $0^{n-k-1}0$ and $0^{n-k-1}1$, respectively. Here, an $*$ in a bit position stands for "don't care", and $*^k$ represents k consecutive $*$'s. We use $Q_{k+1} = Q_k + Q_k^{'}$ to denote the $(k+1)$-dimensional subcube of Q_n induced by vertices in Q_k and $Q_k^{'}$. We explain the broadcasting algorithm in Q_n^* using an n-dimensional hypercube (Q_n) by interchanging the role of vertices and edges. The idea behind our broadcasting algorithm is as follows. Assuming that initially the edge connecting vertices $0^{n-1}0$ and $0^{n-1}1$ in Q_n is colored, and all other edges in Q_n are not colored. We want to edge traversing algorithm A which systematically traverses all edges in Q_n in n steps. In the k-th step, algorithm A selects a subset E_k of edges in Q_n that satisfies the following conditions: (1) all edges in E_k are not previously traversed, (2) each edge in E_k has at least one end vertex that is an end vertex of a previously colored edge, and (3) for each edge e in in E_k assign a direction it is traversed: let u and v be the two end vertices of e, and suppose that u is an end vertex of a previously traversed edge, then traverse e from u to v. In the following, we provide a selection procedure for E_k so that all edges of Q_n are guaranteed to be traversed in n parallel steps. Obviously, such an algorithm A corresponds to a broadcasting algorithm A^* for Q_n^*.

Now, let us describe our algorithm A. Starting from Q_1 (which corresponds to the edge connecting vertices $0^{n-1}0$ and $0^{n-1}1$ in Q_n, and the source vertex $\langle 0^{n-1}0, 0^{n-1}1\rangle$ in Q_n^*), increase the

dimension of the cube by one in each step. In the first step, we consider $Q_2 = Q_1 + Q_1^{'}$, the two edges connecting vertices in Q_1 and $Q_1^{'}$ are colored. In the second step, the edge connecting $0^{n-2}10$ and $0^{n-2}11$ is traversed in the direction from $0^{n-2}10$ and $0^{n-2}11$ and the edges connecting Q_2 and $Q_2^{'}$ are traversed in the direction from Q_2 and $Q_2^{'}$. Assume that after k, $1 \leq k \leq n-2$, steps, all the edges in $Q_k = 0^{n-k-1}0*^k$ and the ones connecting Q_k and $Q_k^{'} = 0^{n-k-1}1*^k$ have been traversed, but all the edges in $Q_k^{'}$ have not been traversed. In step $k+1$, we traverse all the edges in $Q_k^{'}$ and the edges connecting Q_{k+1} and $Q_{k+1}^{'}$, where $Q_{k+1} = 0^{n-k-2}0*^{k+1}$ and $Q_{k+1}^{'} = 0^{n-k-2}1*^{k+1}$, in the direction from Q_{k+1} to $Q_{k+1}^{'}$. To traverse all the edges in $Q_k^{'}$, we randomly pick two connected vertices v and u, if $v < u$ then the link (v, u) is traversed from v to u. In step n, we only need to traverse all the edges in $Q_{n-1}^{'}$ in the direction from $Q_{n-1,0}^{'}$ to $Q_{n-1,1}^{'}$.

Let $b_{n-1}b_{n-2}\cdots b_0$ be the binary representation of b. We use $b^{(i)}$ to represent the binary number (and its corresponding decimal value) obtained by complementing the ith bit, b_i, of the binary representation of b. Translating the above hypercube edge traverse algorithm into an algorithm for traversing vertices in Q_n^*, we obtain the following algorithm.

procedure *BROADCAST*($\langle 0, 1\rangle$)
begin
  for $k = 1$ **to** $n-1$ **do**
    for all $\langle a, b\rangle$ where $a \in 0^{n-k}0*^{k-1}$ and $b \in 0^{n-k}1*^{k-1}$ **do in parallel**
      Processor $\langle a, b\rangle$ sends the message to $\langle a, a^{(k+1)}\rangle$ using e_a;
      Processor $\langle a, b\rangle$ sends the message to $\langle b, b^{(k+1)}\rangle$ using e_b;
      Processor $\langle a, b\rangle$ sends the message to $\langle b, b^{(i)}\rangle$ using e_b,
       if $b < b^{(i)}$ for $i \in \{0, 1, ..., k-1\}$
    endfor
  endfor
  for all $\langle a, b\rangle$ **do in parallel**
    if $b < b^{(i)}$ for $i \in \{0, 1, ..., n-1\}$
    then Processor $\langle a, b\rangle$ sends the message to $\langle b, b^{(i)}\rangle$ using e_b
  endfor
end

In Figure 4, we show the broadcasting tree for Q_4^*, whose bus implementation is shown in Figure 3. In Figure 4, a circle labeled by a pair of integers a and b represents a processor $\langle a, b\rangle$. A directed edge labeled by an integer c from $\langle a, b\rangle$ to $\langle a', b'\rangle$ indicates that the message is transmitted from $\langle a, b\rangle$ to $\langle a', b'\rangle$ using hyperlink e_c.

Theorem 4 *Assuming bus hyperlinks of Q_n^*, algorithm BROADCAST broadcasts a message from any processor to all other processors in n parallel message steps.*

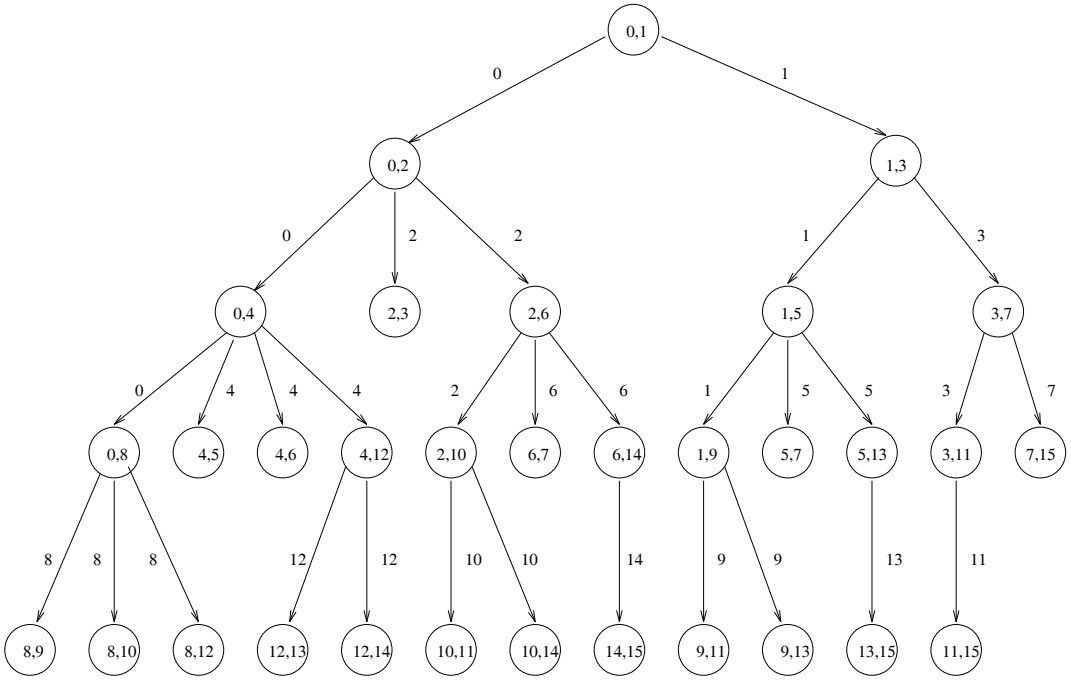

Figure 4: Data communication pattern for broadcasting from $\langle 0,1\rangle$ in Q_4^*.

Proof. The theorem follows directly from a simple induction based on the discussion predeeding *BROADCAST*. □

4.3 Many-to-One Communication

A reduction (or census, or fan-in) function is defined as a commutative and associative operation on a set of values, such as finding maximum, addition, logic or, etc. It can be carried out using a many-to-one communication operation. We only consider the case that the specified reduction operation is addition. The same algorithm can be slightly modified to perform other reduction operations.

We present an algorithm that can be used to perform a summation on a set of N_n values stores in the A registers of processors, one per processor, and putting the final result in processor $\langle l,u\rangle$. That is, the algorithm computes $\sum_{\langle a,b\rangle\in Q_n^*} A_{\langle a,b\rangle}$, and putting the final result in $A_{\langle l,u\rangle}$ of processor $\langle l,u\rangle$. We assume that each processor $\langle a,b\rangle$ has a working register $B_{\langle a,b\rangle}$. Given any processor $\langle l,u\rangle$, procedure *TRANSFORM* discussed in the previous section can be used to transform $\langle l,u\rangle$ to $\langle 0^{n-1}0, 0^{n-1}1\rangle$ (or $\langle 0,1\rangle$) and all other $\langle a,b\rangle$ in Q_n^* to $\langle a',b'\rangle$. Then, we only need to consider the summation algorithm which stores the final result in processor $\langle 0,1\rangle$.

Summation is done in two phases. In the first phase, a set of 2^{n-1} partial sums are obtained and stored in processors $\langle z, 2^{n-1}+z\rangle$, $0 \le z \le 2^{n-1}-1$. The second phase computes the sum of these partial sums and stored the final result in $\langle 0,1\rangle$.

procedure *REDUCTION*($\langle 0,1\rangle$)
begin
  /* *phase 1* */
  for $i = 1$ **to** $n-1$ **do**
    for all $\langle a,b\rangle$ **do in parallel**
      if $a_i a_{i-1} = 00$ and $b_i b_{i-1} = 01$ **then**
        Processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle a,(b^{(i)})^{(i-1)}\rangle$ using e_a;
      if $a_i a_{i-1} = 10$ and $b_i b_{i-1} = 11$ **then**
        Processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle (a^{(i)})^{(i-1)},b\rangle$ using e_b;
      if $\langle a,b\rangle$ received a value **then**
        store this value in $B_{\langle a,b\rangle}$ and perform $A_{\langle a,b\rangle} := A_{\langle a,b\rangle} + B_{\langle a,b\rangle}$
    endfor
  endfor
  /* phase 2 */
  for $i = n-1$ **down to** 1 **do**
    for all $\langle a,b\rangle$ **do in parallel**
      if $a = 0^{n-i-1}00{*}^{i-1}$ and $b = 0^{n-i-1}10{*}^{i-1}$ **then**
        Processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle a,(b^{(i)})^{(i-1)}\rangle$ using e_a;
      if $a = 0^{n-i-1}01{*}^{i-1}$ and $b = 0^{n-i-1}11{*}^{i-1}$ **then**
        Processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle (b^{(i)})^{(i-1)},a\rangle$ using e_a;
      if $\langle a,b\rangle$ received two values **then**
        store one value in $A_{\langle a,b\rangle}$ and the other in $B_{\langle a,b\rangle}$, and perform
          $A_{\langle a,b\rangle} := A_{\langle a,b\rangle} + B_{\langle a,b\rangle}$
    endfor
  endfor
end

Theorem 5 *Assuming bus hyperlinks of Q_n^*, algorithm REDUCTION carries out a reduction operation in $2(n-1)$ parallel message transmission steps.*

Proof. First, we claim that at the end of the first phase the sum of the partial sums stored in processors $\langle z, 2^{n-1}+z\rangle$, $0 \le z \le 2^{n-1}-1$, is the final sum. It is easy to verify that the claim is true for $n = 2$ and $n = 3$. Suppose that the claim is true for $n = k$, and consider the case $n = k+1$. By the algorithm, any processor $\langle a,b\rangle$ such that $a_k a_{k-1} = 00$ and $b_k b_{k-1} = 01$ has not sent and receive any value before the k-th step (iteration). Furthermore, by the hypothesis, the sum of the partial sums stored in processors $\langle z, 2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, is the sum of the values originally stored in sub-hypernetwork Q_k^* induced by all vertices $\langle a,b\rangle$ in Q_{k+1}^* such that

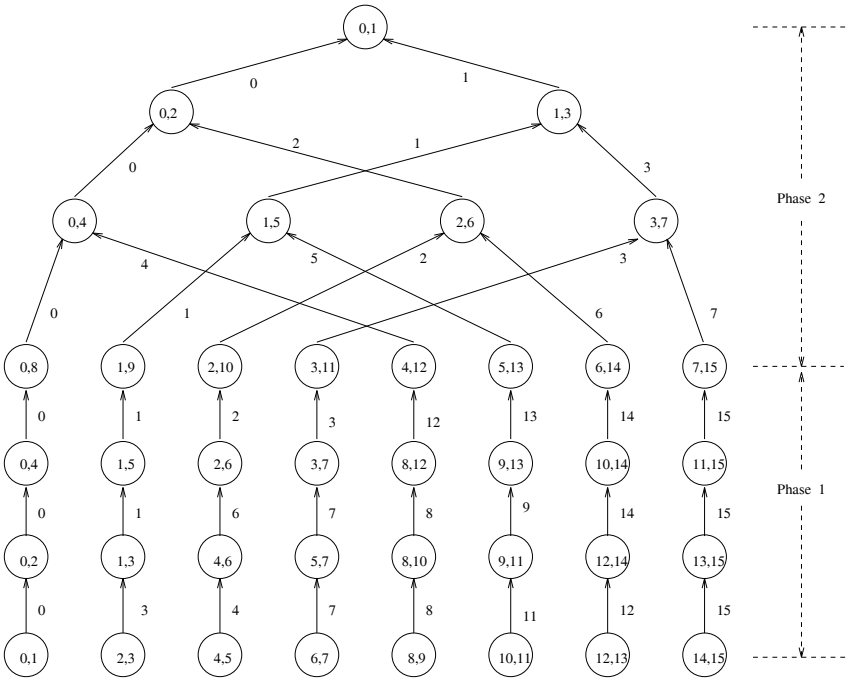

Figure 5: Data communication pattern for reduction in Q_4^*.

$a < 2^k$ and $b < 2^k$. Consider one more step (i.e. the k-th iteration) of the first phase. In this step the partial sum stored in $\langle z, 2^{k-1} + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, is sent to processor $\langle z, 2^k + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, using hyperlink e_z. By the symmetry, the sum of the partial sums stored in processors $\langle 2^k + z, 2^k + 2^{k-1} + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, is the sum of the values originally stored in sub-hypernetwork Q_k^* induced by all vertices $\langle a, b\rangle$ in Q_{k+1}^* such that $a \geq 2^k$ and $b \geq 2^k$ and after the k-th step, the partial sum stored in $\langle 2^k + z, 2^k + 2^{k-1} + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, is sent to processor $\langle 2^{k-1} + z, 2^k + 2^{k-1} + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, using hyperlink $e_{2^k 2^{k-1}+z}$. Therefore, after performing parallel additions in processors $\langle z, 2^k + z\rangle$, $0 \leq z \leq 2^k - 1$, we obtain the claimed 2^k partial sums for Q_{k+1}^*. This completes the induction.

Now, we claim that the second phase computes the sum of the partial sums stored in processors $\langle z, 2^{n-1} + z\rangle$, $0 \leq z \leq 2^{n-1} - 1$, and store the final result in $\langle 0, 1\rangle$ of Q_n^*. This claim is true for $n = 2$ and $n = 3$. Suppose that the claim is true for $n = k$, and consider the case $n = k + 1$. In the first iteration ($i = k$), each processor $\langle z, 2^{k-1} + z\rangle$, $0 \leq z \leq 2^{k-1} - 1$, receives two partial sums, one from $\langle z, 2^k + z\rangle$ via e_z, and the other from $\langle 2^{k-1} + z, 2^k + 2^{k-1} + z\rangle$ via $e_{2^{k-1}+z}$. After additions, 2^{k-1} partial sums are obtained and stored in processors $\langle z, 2^{k-1} + z\rangle$, where $0 \leq z \leq 2^{k-1} - 1$. Then, the induction hypothesis guarantees that after $k - 1$ more steps the final result will be stored in processor $\langle 0, 1\rangle$. This completes the proof of the claim for phase 2, and the proof of the theorem. □

In Figure 5, we show the communication pattern used by *REDUCTION* on Q_4^*, whose bus

implementation is shown in Figure 3. As Figure 4, in this figure, a circle labeled by a pair of integers a and b represents a processor $\langle a, b\rangle$. A directed edge labeled by an integer c from $\langle a, b\rangle$ to $\langle a', b'\rangle$ indicates that the message is transmitted from $\langle a, b\rangle$ to $\langle a', b'\rangle$ using hyperlink e_c.

4.4 Many-to-Many Communication

We consider a general case, the all-to-all communication. In an all-to-all communication, each processor sends a message to all the other processors. It is also called the *total exchange* operation.

We can obtain a total exchange communication algorithm by modifying algorithm *REDUCTION*. The operator used is set union instead of addition. After applying *REDUCTION*, all messages are collected at processor $\langle 0, 1\rangle$. Then, aby applying *BROADCAST*, processor $\langle 0, 1\rangle$ broadcasts the N_n messages to all remaining processors of Q_n^*. By Theorem 5, the second phase alone takes nN_n parallel message steps. Note that the lower bound for the time of a total exchange operation on Q_n^* is $\Omega(N_n)$, and clearly, this algorithm is not efficient.

If what follows, we present an algorithm *TOTAL_EXCHANGE* which takes $O(N_n)$ message steps to perform the total exchange operation on Q_n^*. Algorithm *TOTAL_EXCHANGE* is an all-port algorithm, i.e. the two I/O ports of each processor may participate in a message transmission step. However, each port performs either a send operation or receive operation, but not both. This algorithm can be easily converted to a single-port algorithm with the same communication complexity.

For convenience, we define that a processor $\langle i, j\rangle$ is of dimension k, $0 \leq k \leq n-1$, if $j - i = 2^k$. It is easy to verify the following facts: (i) There are exactly 2^{n-1} processors of dimension k, $0 \leq k \leq n-1$, in Q_n^*; (ii) There is exactly one processor of dimension k, $0 \leq k \leq n-1$, attached to each hyperlink in Q_n^*; and (iii) Any two processors of the same dimension are not attached to the same hyperlink in Q_n^*.

procedure *TOTAL_EXCHANGE*
begin
  for all hyperlinks e_h **do in parallel**
    All processors attached to e_h sends its message to the processor of dimension 0 that is attach to e_h using e_h;
  endfor
  Let the set of messages received by each processor $\langle i, j\rangle$ be denoted by $M_{\langle i,j\rangle}$;
  for $k = 0$ **to** $n-1$ **do**
    for all processors $\langle i, j\rangle$ of dimension k **do in parallel**
      Broadcast $M_{\langle i,j\rangle}$ to all processors attached to hyperlink e_i and e_j
    endfor
    Let the set of messages received by each processor $\langle i, j\rangle$ be denoted by $M_{\langle i,j\rangle}$;

endfor
end

The correctness of this algorithm can be verified by the following induction. It is easy to see that the algorithm is correct for Q_4^*. Suppose that the algorithm is correct for $n = m$, and consider the case of $n = m + 1$. After m iterations of the **for** loop, the total exchange operations are performed with respect to the subhypergraph of Q_{m+1}^* induced by vertices $\langle i, j\rangle$ such that $i < 2^m$ and $j < 2^m$, and the subhypergraph of Q_{m+1}^* induced by vertices $\langle i', j'\rangle$ such that $i' \geq 2^m$ and $j' \geq 2^m$. In addition, dimension m processors have received all messages in Q_{m+1}^*. In one additional iteration, each processor $\langle a, b\rangle$ of dimension m broadcasts all its received messages to processors attached to hyperlinks e_a and e_b. Then, by (i), (ii) and (iii), the total exchange operation is performed with respect to Q_{m+1}^*.

Now, let use analyze the performance of *TOTAL_EXCHANGE*. In our algorithm, we assume that we a processor broadcasts a set of messages, it broadcasts all messages it received in the previous step. As a consequence, duplicated messages are broadcast. We show that even with duplicated messages, the performance of *TOTAL_EXCHANGE* is within a constant factor of the optimal. In the first **for** statement, $2(n-1)$ messages are collected by each dimension 0 processor $\langle a, b\rangle$ using two hyperlinks e_a and e_b, and this takes $(n-1)$ message steps. Consider the **for** loop. It has n iterations. In the first iteration, $2n$ messages are broadcast from each dimension 0 processor $\langle a, b\rangle$ to all processors attached to e_a and e_b. In the second iteration, $4n$ messages are broadcast are broadcast from each dimension 1 processor $\langle a, b\rangle$ to all processors attached to e_a and e_b. In general, in the iteration with $k = m$, $2^{m+1}n$ messages need to be broadcast by each dimension m processor $\langle a, b\rangle$ to all processors attached to e_a and e_b. By (ii) and (iii) above, the iteration with $k = m$ takes no more than $2^{m+1}n$ message steps. Therefore, *TOTAL_EXCHANGE* requires no more than $(n-1)+(2+4+8+\cdots+2^n)n = (2^{n+1}-1)n-1$ parallel message steps. We summarize this analysis by the following claim.

Theorem 6 *Assuming bus hyperlinks of Q_n^*, algorithm TOTAL_EXCHANGE carries out a total exchange operation in $4N_n - n - 1$ parallel message steps.*

□

5 Discussions

We proposed a new class of hypernetworks base on the duals of hypercubes. The structures of Q_n^* and Q_n are quite different, but as we showed, many properties of Q_n^* can be directly derived from the properties of Q_n. The Q_n^* hypernetwork is suitable for exploiting the high bandwidths provided by new interconnection technologies such as optical fiber or devices. We presented a set of basic data communication algorithms for Q_n^* based on bus implementation of hyperlinks. Algorithms

ROUTE ard *BROADCAST* are optimal, and algorithms *REDUCTION* and *TOTAL_EXCHANGE* are optimal within a constant factor.

Our algorithms are closely related to the ideas behind their corresponding algorithms on the hypercube network. This leads us to a pose an open problem: is there a simulation scheme that can be used to simulates Q_n by Q_n^* efficiently? If such a scheme can be found, then all previously know hypercube algorithms can be automatically translated to algorithms for a machine using Q_n^* as the interconnection network.

Using the hypergraph dual concept, one can obtain another class of hypernetworks that contains the duals of the star graphs. The n-star graph S_n (refer to [1] for its definition) is a point-to-point network that has $n!$ vertices, $n(n-1)!/2$ edges, and its degree and the diameter are $n-1$ and $\lfloor 3(n-1)/2 \rfloor$, respectively. S_n is vertex and edge symmetric. Therefore, S_n^* has $n!(n-1)/2$ vertices and $n!$ hyperedges; S_n^* is 2-regular, $(n-1)$-uniform, linear, and vertex and hyperedge symmetric; and the diameter of S_n^* is no greater than $\lfloor (3n-1)/2 \rfloor$, respectively. Both of diameter and degree of S_n^* are sub-logarithmic functions of the number of processors and the number of hyperlinks in S_n^*. Compared with Q_n^*, S_n^* has some advantages. The topological and communication aspects of the S_n^* hypernetworks deserve further investigations.

References

[1] S. Akers, D. Harel, and B. Krishnamurthy, The Star Graph: an Attractive Alternative to the n-Cube, *Proceedings of 1987 International Conference on Parallel Processing*, pp. 393-400, 1987.

[2] C. Berge *Hypergraphs*, North-Holland, 1989.

[3] O.M. Dighe, R. Vaidyanathan, and S.Q. Zheng, The Bus-Connected Ringed Tree: A Versatile Interconnection Network, to appear in *Journal of Parallel and Distributed Computing.*

[4] P. E. Green, Jr., *Fiber Optical Networks*, Prentice Hall, 1993.

[5] Z. Guo , R. Melhem, R. Hall, D.Chiarulli and S. Levitan, Array Processors with Pipelined Optical busses, *Journal of Parallel and Distributed Computing*, **12**(3), pp. 269-282, 1991.

[6] J. Jahns and S.H. Lee (editors), *Optical Computing Hardware*, Academic Press, Inc., 1994.

[7] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercube* , Morgan Kaufmann Publishers, Inc., 1992, pp. 78-82, 239-244.

[8] Y. Li, Y. Pan and S.Q. Zheng, A Pipelined TDM Optical Bus with Conditional Delays", to appear in *Optical Engineering.*

[9] R. Melhem, D. Chiarulli, and S. Levitan, Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems, *Computer Journal*, **32**(4), pp.362-369, 1989.

[10] Y. Pan and K. Li, Linear Array with a Reconfigurable Pipelined Bus System – Concepts and Applications, *Proceedings of 1996 International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 1431-1442, 1996.

[11] C. Qiao and R. Melhem, Time-division Optical Communications in Multiprocessor Arrays, *IEEE Trans. on Computers*, **42**(5), pp. 577-590, 1993.

[12] C. Qiao, R. Melhem, D. Chiarulli and S. Levitan, Optical Multicasting in Linear Arrays, *International Journal of Optical Computing*, **2**(1), pp. 31-48, 1991.

[13] C. Partridge, *Gigabit Networking*, Addison-Wesley, 1994.

[14] S.Q. Zheng, Hypernetworks - A Class of Interconnection Networks with Increased Wire Sharing: Part I - Part IV, Technical Reports, Department of Compute Science, Louisiana State University, Baton Rouge, LA 70803, Dec., 1994.

[15] S.Q. Zheng, Sparse Hypernetworks Based on Steiner Triple Systems, *Proc of 1995 International Conf. on Parallel Processing*, pp. I.92 - I.95, 1995.

[16] S.Q. Zheng and Y. Li, A Pipelined Asynchronous TDM Optical Bus, Technical Report #97-006, Dept. of Computer Science, Louisiana State Univ., April, 1997.

[17] S.Q. Zheng and J. Wu, Dual of a Complete Graph as an Interconnection Network, *The Proceedings of 8th IEEE Symposium on Parallel and Distributed Processing*, pp. 433-442, 1996.