International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 6, Number 2, pages 368–394, July 2016

# Routing and Broadcasting Algorithms for Generalized-Star Cube

Daiki Arai
Graduate School of CIS
Hosei University
Tokyo 184-8584 Japan

Yamin Li
Faculty of Computer and Information Sciences
Hosei University
Tokyo 184-8584 Japan

Received: February 15, 2016
Revised: May 6, 2016
Accepted: July 6 2016
Communicated by Yoshiaki Kakuda

**Abstract**

In this paper, another version of the star cube called the generalized-star cube, GSC$(n, k, m)$, is presented as a three level interconnection topology. GSC$(n, k, m)$ is a product graph of the $(n, k)$-star graph and the $m$-dimensional hypercube ($m$-cube). It can be constructed in one of two ways: to replace each node in an $m$-cube with an $(n, k)$-star graph, or to replace each node in an $(n, k)$-star graph with an $m$-cube. Because there are three parameters $m$, $n$, and $k$, the network size of GSC$(n, k, m)$ can be changed more flexibly than the star graph, star-cube, and $(n, k)$-star graph. We first investigate the topological properties of the GSC$(n, k, m)$, such as the node degree, diameter, average distance, and cost. Also, the regularity and node symmetry of the GSC$(n, k, m)$ are derived. Next, we present a formal shortest-path routing algorithm. Then, we give broadcasting algorithms for both of the single-port and all-port models. To develop these algorithms, we use the spanning binomial tree, the neighborhood broadcasting algorithm, and the minimum dominating set. The complexities of the routing and broadcasting algorithms are also examined.

## 1 Introduction

In recent years, study of parallel and distributed computing has been featured as one of the important research themes. Especially, there is increasing interest in large scale parallel computing. For such parallel computing systems, the wide variety of interconnection topologies were proposed. Among them, the hypercube [22] structure has been widely used because of its elegant topological properties and the ability to emulate a wide variety of other frequently used networks.

However, conventional hypercube network is not a good candidate for such large scale networks because hypercube has a major drawback. That is, the number of communication links for each node is a logarithmic function of the number of nodes in the network. To alleviate this drawback, several variations of the hypercube have been proposed in the literature. Cube-connected cycles [20]

and reduced hypercube [25] focused on the reduction of the number of links of the hypercube. Hierarchical cubic network [12] focused on reductions of the number of links and the diameter of the hypercube. These topologies are the modification of the hypercube in one way or another with motivation to improve some of its properties.

In such circumstances, [4] and [3] pointed out that many of these properties of the hypercube are in fact group theoretic properties possessed by a large class of networks called Cayley graphs. Some Cayley graphs not only possess all these properties but even offer a better degree and diameter than the hypercube. The star graph is an important class of Cayley graph and an attractive alternative to the hypercube in lower degree and shorter diameter [3]. Additionally, we can find multiple disjoint paths in $n$-star graph and its each data or replica (copy data) can be placed in a distinct $(n-1)$-star, so we can realize balanced data/replica distribution among peers on the network. These fascinating properties have promoted extension of the star graph such as the contracted star graph for P2P overlay networks by Fujita [10]. Furthermore, Fujita provided a P2P DHT as his advanced work [11]. In the work, Fujita gave a scheme of mapping from a given key to a vertex on the proposed overlay network.

However, star graph has also a major drawback such that the network size is restricted on the choice of the total number of nodes by $n!$. To mitigate the restriction of the significant gap between the two consecutive sizes of nodes $n!$ in the $n$-star graph, the incomplete star [15] and arrangement graphs [8] have been proposed. However, the incomplete star is a non-symmetric and irregular graph and the arrangement graphs have a problem of the very high node degree. These problems restrict the adoption of these topologies to the practical system design. To solve these problems, $(n, k)$-star graph [6] and star-cube [24] are proposed.

By generalizing the star graph with another parameter $k$, we can obtain the $(n, k)$-star graph. In this graph, two parameters $n$ and $k$ are used to control the number of nodes, thus making it convenient to design a network with a desirable size and a better degree/diameter trade-off than the star graph. The star-cube is a product graph based on the star graph and the hypercube and inherits all the attractive properties from both topologies. In this graph, two parameters of star graph $n$ and hypercube $m$ are used to control the network size. Therefore, its size grows in smaller steps than the star graph.

For any interconnection network, we can classify it as either a single-port model or all-port model, depending on how a node communicates with its neighbors. In the single-port model, in one step, a node can send (receive) a message to (from) one and only one of its neighbor nodes. Meanwhile, in the all-port model, in one step, a node can send (receive) messages to (from) all of its neighbor nodes.

One of the simplest and most fundamental collective communication operations is one-to-all broadcasting. In one-to-all broadcast, a source node sends a message to all nodes. Nowadays, it is a standard component of parallel software libraries and is supported in hardware in many commercial parallel computers. One-to-all broadcast algorithm is used in many parallel algorithms, such as vector-matrix multiplication, Gaussian elimination, LU (lower upper) factorization, and Householder transformations [18].

A similar problem which has been studied is the problem of neighborhood broadcasting. It is an algorithm to send a message from a node to its all neighbors. It is clear for any interconnection network with $N$ nodes, on a single-port model, that the problem of broadcasting has a trivial lower bound of $\mathbf{\Omega}(\log N)$ because in one step, the number of informed nodes can double at most. Similarly, the problem of neighborhood broadcasting has a trivial lower bound of $\mathbf{\Omega}(\log n)$ where $n$ is the degree of the source node. It is also clear for any interconnection network, on an all-port model, that a trivial lower bound for the problem of broadcasting is the diameter of the network and the neighborhood broadcasting can be done in one step.

In this paper, we propose a new interconnection network called the generalized-star cube (GSC) with three parameters $n$, $k$, and $m$. A GSC$(n, k, m)$ network consists of $2^m n!/(n-k)!$ nodes with a degree of $m+n-1$ and a diameter of $m+2k-1$ for $k \leq \lfloor n/2 \rfloor$ and $m+k+\lfloor (n-1)/2 \rfloor$ for $k > \lfloor n/2 \rfloor$. GSC$(n, k, m)$ is a product graph based on the $(n, k)$-star graph and $m$-dimensional hypercube. Using these three parameters, compared to star graph, star-cube, and $(n, k)$-star, the network size of GSC$(n, k, m)$ can be changed flexibly. We also address graph-theoretic properties of

GSC$(n, k, m)$. Additionally, a shortest-path routing algorithm and broadcasting algorithms for both of the single-port and all-port models for GSC$(n, k, m)$ are established. For these routing algorithms, we separate them into hypercube part and $(n, k)$-star graph part and use existing optimal algorithms. In the shortest-path routing, both algorithms for hypercube and $(n, k)$-star graph are simple, so the routing algorithm for the product graph of them is also simple. In broadcasting algorithms at both of the single-port and all-port models, we use spanning binomial tree for hypercube part. On the other hand, in $(n, k)$-star graph part, on the single-port model, we use a neighborhood broadcasting algorithm, and on the all-port model, we use minimum dominating set. As a result, we derived optimal algorithms for these three-type routing problems.

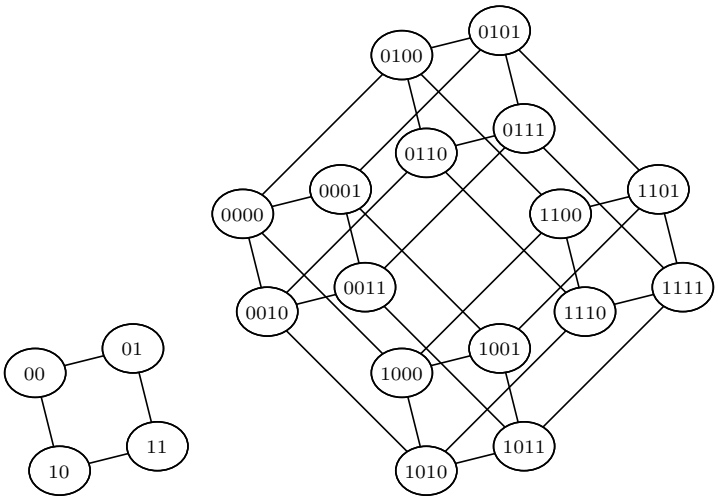
The rest of this paper is organized as follows. Section 2 introduces the related works. Section 3 proposes the generalized-star cube and investigates properties of the generalized-star cube. Section 4 gives a shortest-path routing algorithm. Sections 5 and 6 describes two broadcasting algorithms on single-port and all-port models, respectively. And the final section concludes the paper.

# 2 Preliminaries

This section introduces some interconnection topologies which have a close relation with the proposed generalized-star cube topology.

## 2.1 Hypercube

The $n$-dimensional hypercube [22], $n$-cube for short, has $2^n$ nodes and $n$ edges per node. Each $n$-cube node is assigned with a unique $n$-bit binary address. Two nodes are connected through an edge if and only if their binary addresses differ in a single bit. In addition to this property, $n$-cube has other elegant topological properties. For example, the strong connectivity, regularity, symmetry, embeddability, partitionability, maximal fault-tolerance, strong resilience, and simple routing are attractive properties of the hypercube. Figures 1(a) and (b) show a 2-cube and a 4-cube, respectively.



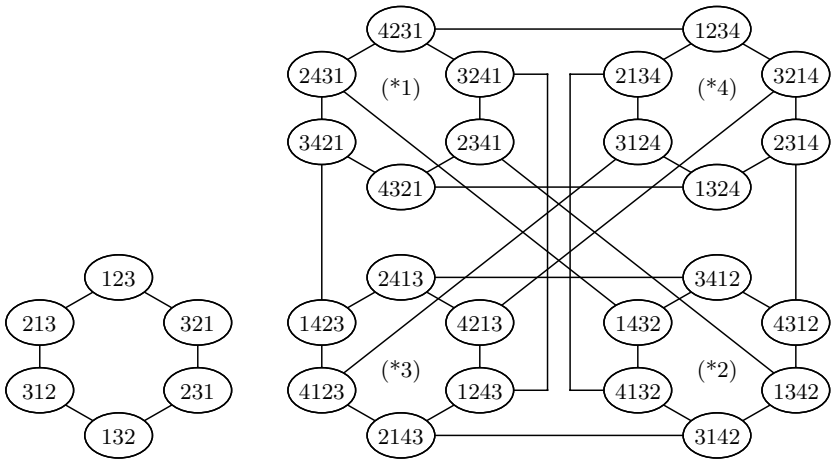

(a) 2-cube (b) 4-cube

Figure 1: Hypercubes

## 2.2 Star Graph

The $n$-dimensional star graph ($n$-star) [3] is composed of an integer set $\{x_1, x_2, x_3, \ldots, x_n\}$, where $x_i \neq x_j$ for $i, j = 1, 2, 3, \ldots, n$. In this graph, each node is represented as an $n$-permutation so that the total number nodes is $n!$. The node degree is $n - 1$ and the diameter is $\lfloor 3(n-1)/2 \rfloor$.

A node in an $n$-star is adjacent to $n-1$ neighbors if their symbols differ in the first and another positions. That is, two permutations, $\langle a_1 a_2 \ldots a_n \rangle$ and $\langle b_1 b_2 \ldots b_n \rangle$, are connected if and only if $a_1 = b_i$, $a_i = b_1$, and $a_j = b_j$ for $i \neq 1$ and $j \notin \{1, i\}$. Some of important features of the star graph are fault-tolerance, partitionability, node-disjoint parallel paths, vertex/edge symmetry, and strongly hierarchical structure, as are the case with the properties of the hypercube. Figures 2(a) and (b) show a 3-star and a 4-star, respectively. Compared to hypercube the star graph has a lower node degree, a shorter diameter, and a smaller average distance in a system with similar number of nodes.




(a) 3-star (b) 4-star

Figure 2: Star graphs

## 2.3 $(n, k)$-Star Graph

The $(n, k)$-star graph is a generalization of the star graph topology [6, 7]. An $(n, k)$-star graph, denoted by $\mathrm{S}(n, k)$, is regular of degree $n-1$ and specified by two integers $n$ and $k$ with $1 \leq k < n$. The node set of $\mathrm{S}(n, k)$ is a set of all $k$-permutations of $n$, denoted by $\{p_1 p_2 \ldots p_k \mid p_i \in \langle n \rangle$ and $p_i \neq p_j$ for $i \neq j\}$, where $\langle n \rangle = \{1, 2, \ldots, n\}$. A node $p = \langle p_1 p_2 \ldots p_i \ldots p_k \rangle$ is adjacent to the neighbors as follows: (1) $\langle p_i p_2 \ldots p_1 \ldots p_k \rangle$ through an edge of dimension $i$, where $2 \leq i \leq k$ (swap $p_1$ and $p_i$); this kind of edges are referred to as $i$-edges, and (2) $\langle x p_2 \ldots p_i \ldots p_k \rangle$ through an edge of dimension 1, where $x \in \langle n \rangle - \{p_i \mid 1 \leq i \leq k\}$; this kind of edges are referred to as 1-edges. The number of nodes in the $(n, k)$-star is $n!/(n-k)!$ and the diameter is $2k-1$ for $k \leq \lfloor n/2 \rfloor$ and $k + \lfloor (n-1)/2 \rfloor$ for $k \geq \lfloor n/2 \rfloor + 1$. Figure 3 shows an S(4,2). The $(n, k)$-star graph shares many attractive properties with the $n$-star graph such as node symmetry, low degree, small diameter, hierarchical structure, maximal fault tolerance, and simple shortest path routing. In addition, when $k = n-1$, $\mathrm{S}(n, n-1)$ is isomorphic to the $n$-star. This implies that the $n$-star graph is a special case of the $(n, k)$-star graph.

## 2.4 Star-Cube

The star-cube is a product graph based on the star graph and hypercube and defined by two parameters $m$ and $n$ [24]. A star-cube, denoted by $\mathrm{SC}(n, m)$, also known as cube-star $\mathrm{CS}(m, n)$, is regular of degree $m + n - 1$. The number of nodes is $2^m n!$ and the diameter is $m + \lfloor 3(n-1)/2 \rfloor$. In this graph, the address of each node has two parts as $\langle x_m x_{m-1} \ldots x_2 x_1 \, , \, y_1 y_2 \ldots y_{n-1} y_n \rangle$. $x_m \ldots x_1$ represents hypercube part and $y_1 \ldots y_n$ represents star graph part. Therefore, each node is adjacent to two types of neighbors, named as the cube-neighbors and star-neighbors with node addresses $\langle x_m \ldots \tilde{x}_i \ldots x_1, y_1 \ldots y_n \rangle$ for $1 \leq i \leq m$ ($\tilde{}$ means a bit inversion operation) and $\langle x_m \ldots x_1, y_j \ldots y_1 \ldots y_n \rangle$

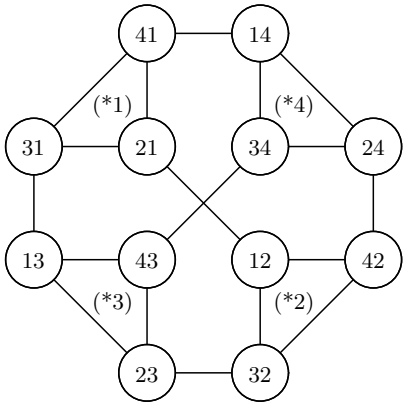



Figure 3: GS(4,2): A generalized star graph ((4,2)-star)

for $2 \leq j \leq n$, respectively. This graph possesses all the attractive properties which are common to both star graph and hypercube such as being regular, vertex/edge-symmetric, maximally fault-tolerant, and simple shortest path routing. In addition, this topology has interesting features that the nodes with the same hypercube-label form an $n$-star whereas the nodes with the same star graph-label form an $m$-cube, as shown in Figure 4 and Figure 5, respectively.

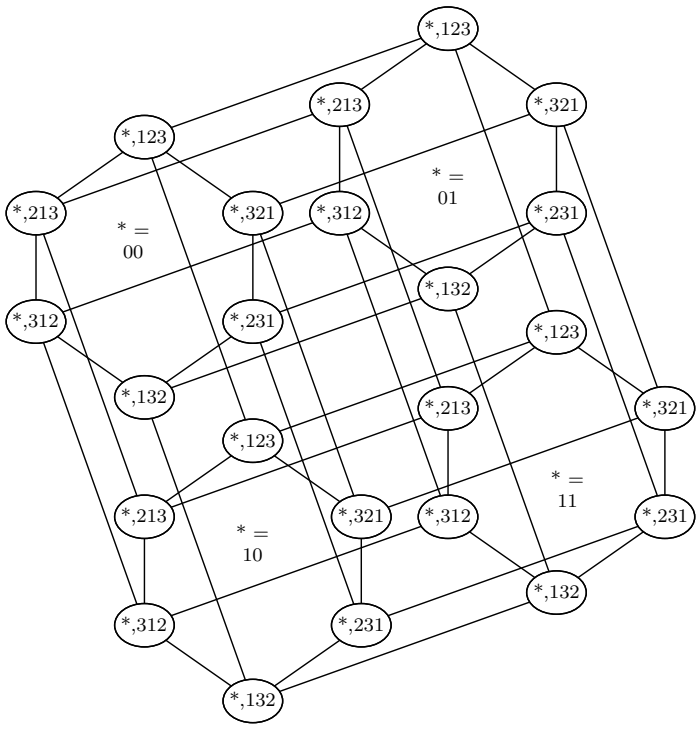



Figure 4: CS(2,3): A cube-connected-star (2-cube $\times$ 3-star)

# 3 Generalized-Star Cube and its Properties

This section describes our new proposed interconnection topology, generalized-star cube, and its topological properties.

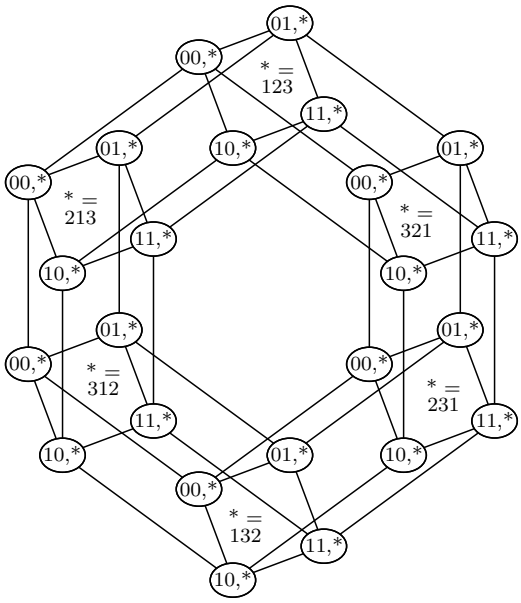



Figure 5: SC(3,2): A star-connected-cube (3-star × 2-cube)

## 3.1 Generalized-Star Cube

The generalized-star cube, denoted by GSC$(n, k, m)$, is a product graph of the S$(n, k)$ and $m$-cube. In a GSC$(n, k, m)$, the node address of each vertex can be separated into two part labels $\langle x_m x_{m-1} \ldots x_2 x_1 \, , \, y_1 y_2 \ldots y_{k-1} y_k \rangle$, where the label of $x_m \ldots x_1$ signifies the $m$-cube part (cube-label) and $y_1 \ldots y_k$ signifies the $(n, k)$-star graph part ($(n, k)$-star-label). Each node will be adjacent to two types of neighbors, namely the cube-neighbors and $(n, k)$-star-neighbors, respectively. The node addresses of cube-neighbors are represented as $\langle x_m \ldots \tilde{x}_i \ldots x_1 \, , \, y_1 \ldots y_k \rangle$ for $1 \leq i \leq m$ ($\tilde{}$ means a bit inversion operation) and of $(n, k)$-star-neighbors are represented as (1) $\langle x_m \ldots x_1 \, , \, y_j \ldots y_1 \ldots y_k \rangle$ for $2 \leq j \leq k$, or (2) $\langle x_m \ldots x_1 \, , \, y' \ldots y_j \ldots y_k \rangle$ for $y' \in \{1, 2, \ldots, n\} - \{y_j \mid 1 \leq j \leq k\}$. The edges of kind (1) are referred to as $j$-edges and (2) are referred to as 1-edges. In this graph, an $(n, k)$-star graph replaces each vertex of the $m$-cube as the vertex of the GSC$(n, k, m)$ and an $m$-cube replaces each vertex of the $(n, k)$-star graph. This means that there are $n!/(n-k)!$ $m$-cube subgraphs in the GSC$(n, k, m)$, where the nodes of each $m$-cube are assigned with the same $(n, k)$-star-label. These subgraphs can be distinguished by their $(n, k)$-star-labels (Figure 6). Similarly, the GSC$(n, k, m)$ can be considered as having $2^m$ $(n, k)$-star graphs, where the nodes of each $(n, k)$-star graph are assigned with the same cube-label. These sub-graphs can be distinguished by their cube-labels (Figure 7).

## 3.2 Basic Terminologies

Before illustrating the topological properties of the proposed GSC$(n, k, m)$, the basic terminologies of the interconnection network are explained below. In this paper, the interconnection network is thought of as an undirected graph. Therefore, the vertices correspond to the processors and the edges correspond to the bidirectional communication links.

**Definition 1** *The interconnection network is a finite graph $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$, where $\boldsymbol{V}$ and $\boldsymbol{E}$ are a set of vertices (or nodes) and a set of edges (or links), respectively.*

**Definition 2** *The degree of a vertex $v$ in $\boldsymbol{G}$ is equal to the number of edges incident on $v$.*

**Definition 3** *The diameter of a graph $\boldsymbol{G}$ denoted as $D_G$ is defined to be $max\{d_G(u, v) \mid u, v \in \boldsymbol{V}\}$, where $d_G$ is the distance between two nodes $u$ and $v$.*

**Definition 4** *A graph is called regular if all of its vertices have the same degree.*

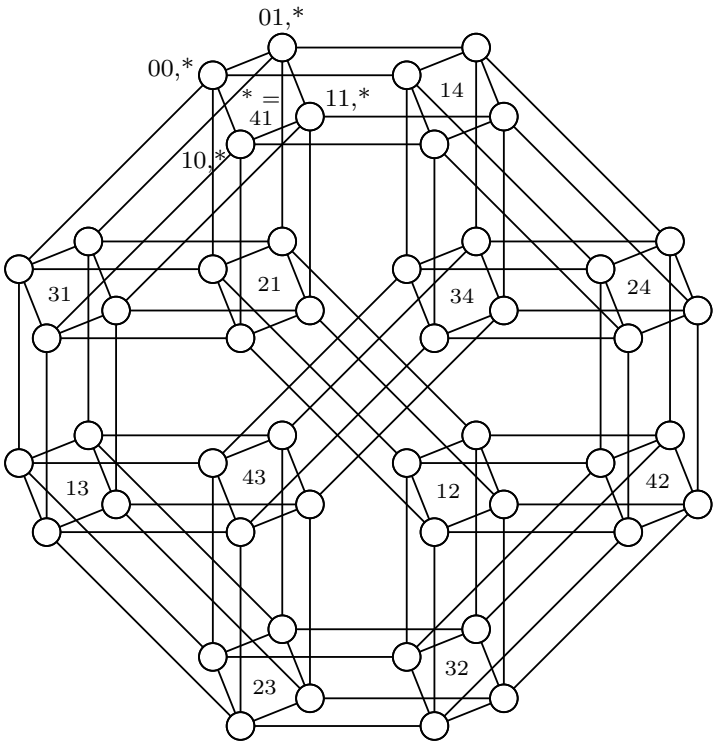



Figure 6: GSC(4,2,2): A generalized star-connected-cube ((4,2)-star $\times$ 2-cube)

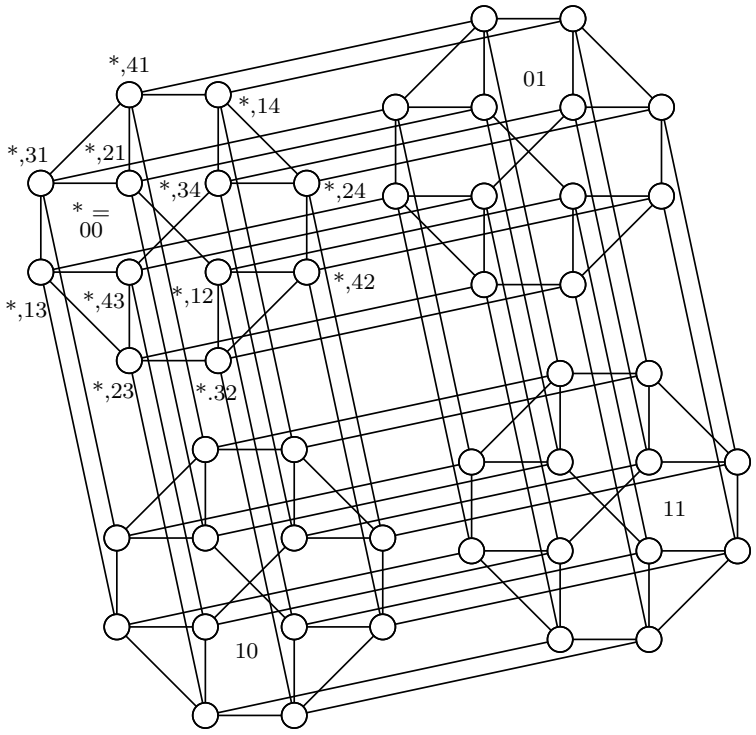



Figure 7: GSC(4,2,2): A generalized cube-connected-star (2-cube $\times$ (4,2)-star)

**Definition 5** *A graph* $\boldsymbol{G(V, E)}$ *is vertex symmetric if for any arbitrary pair of vertices,* $u$ *and* $v$*, there exists an automorphism of the graph that maps* $u$ *into* $v$ $(u, v \in \boldsymbol{V})$.

### 3.3 Topological Properties

The following are some of the topological properties of the proposed generalized-star cube and these results are summarized in Table 1.

**Theorem 1** *The total number of nodes in GSC*$(n, k, m)$ *is* $2^m n!/(n-k)!$.

*Proof:* The hypercube of dimension $m$ has $2^m$ nodes and the $(n, k)$-star graph has $n!/(n-k)!$ nodes. Because GSC$(n, k, m)$ is a product graph of them, the total number of nodes is given by $p = 2^m \times n!/(n-k)!$. □

**Theorem 2** *The degree of the GSC*$(n, k, m)$ *is* $(m + n - 1)$.

*Proof:* In an $m$-cube, each node has $m$ edges and $(n-1)$ edges are incident on the $(n, k)$-star graph. Hence, the degree of each node in the GSC$(n, k, m)$ is $(m + n - 1)$. □

**Theorem 3** *The total number of links in the GSC*$(n, k, m)$ *is* $2^{m-1}n!/(n-k)!(m+n-1)$.

*Proof:* The GSC$(n, k, m)$ can be considered as $n!/(n-k)!$ $m$-cubes connected in the $(n, k)$-star graph form. Therefore, the total number of links in cube part is $(m2^m)/2 \times n!/(n-k)! = m2^{m-1}n!/(n-k)!$. $(m2^m)/2$ means that each node of $2^m$ is incident on $m$ neighbors but all edges are counted double, so it is divided by 2. Next, each cube part node is linked to $n-1$ neighbors which have same cube part label through $(n, k)$-star edges. Therefore, there are $(n-1)2^m/2 \times n!/(n-k)! = (n-1)2^{m-1}n!/(n-k)!$ such $(n, k)$-star edges. Hence, the total number of links in GSC$(n, k, m)$ is given by $E = m2^{m-1}n!/(n-k)! + (n-1)2^{m-1}n!/(n-k)! = 2^{m-1}n!/(n-k)!(m+n-1)$. □

**Theorem 4** *The diameter of the GSC*$(n, k, m)$ *is* $m+2k-1$ *for* $1 \le k \le \lfloor n/2 \rfloor$ *and* $m+k+\lfloor n-1/2 \rfloor$ *for* $\lfloor n/2 \rfloor + 1 \le k \le n-1$.

*Proof:* Let $(\boldsymbol{s}, \boldsymbol{u})$ and $(\boldsymbol{t}, \boldsymbol{v})$ be a source node and a destination node, respectively, in the GSC$(n, k, m)$, where $\boldsymbol{s}$ and $\boldsymbol{t}$ are cube part labels and $\boldsymbol{u}$ and $\boldsymbol{v}$ are $(n, k)$-star part labels. Traveling from the node $(\boldsymbol{s}, \boldsymbol{u})$ to the node $(\boldsymbol{t}, \boldsymbol{u})$, one can arrive in at most $m$ steps because the nodes with the same $(n, k)$-star part label form an $m$-cube. Then, traveling from the node $(\boldsymbol{t}, \boldsymbol{u})$, one can reach the node $(\boldsymbol{t}, \boldsymbol{v})$ in $2k-1$ or $k + \lfloor n - 1/2 \rfloor$ steps for $1 \le k \le \lfloor n/2 \rfloor$ and $\lfloor n/2 \rfloor + 1 \le k \le n-1$, respectively, because the nodes with the same cube part label form an $(n, k)$-star graph. Therefore, in $m+2k-1$ $(1 \le k \le \lfloor n/2 \rfloor)$ or $m + k + \lfloor n - 1/2 \rfloor$ $(\lfloor n/2 \rfloor + 1 \le k \le n-1)$ steps, we can reach the destination node $(\boldsymbol{t}, \boldsymbol{v})$ from the source node $(\boldsymbol{s}, \boldsymbol{u})$. □

**Theorem 5** *The cost of the GSC*$(n, k, m)$ *is* $(m + n - 1)(m + 2k - 1)$ *for* $1 \le k \le \lfloor n/2 \rfloor$ *and* $(m + n - 1)(m + k + \lfloor n - 1/2 \rfloor)$ *for* $\lfloor n/2 \rfloor + 1 \le k \le n-1$.

*Proof:* The cost of a network is defined as $\xi$ = degree $\times$ diameter. From Theorem 2, the degree of the GSC$(n, k, m)$ is $(m+n-1)$. From Theorem 4, the diameter is $m+2k-1$ and $m+k+\lfloor n-1/2 \rfloor$ for $1 \le k \le \lfloor n/2 \rfloor$ and $\lfloor n/2 \rfloor + 1 \le k \le n-1$, respectively. Therefore, the cost of the GSC$(n, k, m)$ is $\xi = (m + n - 1)(m + 2k - 1)$ for $1 \le k \le \lfloor n/2 \rfloor$ and $(m + n - 1)(m + k + \lfloor n - 1/2 \rfloor)$ for $\lfloor n/2 \rfloor + 1 \le k \le n-1$. □

**Theorem 6** *The average distance of the GSC*$(n, k, m)$ *is* $m/2 + k - 1 + \sum_{i=1}^{k} 1/i - 2(k-1)/n - k!(n-k)!/n!$.

*Proof:* The average distance for the hypercube of dimension $m$ is $\sum_{i=0}^{m} i\binom{m}{i}/2^m = m/2$ and the average distance for the $(n, k)$-star graph is shown to be $k-1+\sum_{i=1}^{k} 1/i - 2(k-1)/n - k!(n-k)!/n!$ in [7]. Therefore, the average distance of the GSC$(n, k, m)$ is given by $\overline{d}_{gsc} = m/2+k-1+\sum_{i=1}^{k} 1/i - 2(k-1)/n - k!(n-k)!/n!$. □

**Theorem 7** *The GSC$(n,k,m)$ is a regular graph.*

*Proof:* The $m$-cube and $(n,k)$-star graph are regular graphs. Then, GSC$(n,k,m)$ is the product graph of them, so from Definition 4, the GSC$(n,k,m)$ is a regular graph. □

**Theorem 8** *The GSC$(n,k,m)$ is vertex symmetric.*

*Proof:* The $m$-cube and $(n,k)$-star graph are vertex symmetric. Then, GSC$(n,k,m)$ is the product graph of them, so from Definition 5, the GSC$(n,k,m)$ is vertex symmetric. However, GSC$(n,k,m)$ could not be edge symmetric. For example, in Figure 6, each 2-cube edge belongs to a cycle of length at least 4, but each 1-edge of the $(4,2)$-star graph may belong to a cycle of length at least 3. □

Table 1: Comparison of the topological parameters for different networks

| Parameters | HC($m$) | $n$-Star | $(n,k)$-Star | SC$(n,m)$ | GSC$(n,k,m)$ |
|---|---|---|---|---|---|
| Nodes | $2^m$ | $n!$ | $\frac{n!}{(n-k)!}$ | $2^m n!$ | $2^m \frac{n!}{(n-k)!}$ |
| Degree | $m$ | $n-1$ | $n-1$ | $m+n-1$ | $m+n-1$ |
| Links | $m2^{m-1}$ | $n!\frac{n-1}{2}$ | $\frac{n!}{(n-k)!}\frac{n-1}{2}$ | $2^{m-1}n!(m+n-1)$ | $2^{m-1}\frac{n!}{(n-k)!}(m+n-1)$ |
| Diameter | $m$ | $\lfloor\frac{3(n-1)}{2}\rfloor$ | $2k-1$ (if $1 \le k \le \lfloor\frac{n}{2}\rfloor$) $k+\lfloor\frac{n-1}{2}\rfloor$ (if $\lfloor\frac{n}{2}\rfloor+1 \le k \le n-1$) | $m+\lfloor\frac{3(n-1)}{2}\rfloor$ | $m+2k-1$ (if $1 \le k \le \lfloor\frac{n}{2}\rfloor$) $m+k+\lfloor\frac{n-1}{2}\rfloor$ (if $\lfloor\frac{n}{2}\rfloor+1 \le k \le n-1$) |
| Average distance | $\frac{m}{2}$ | $n-4+\frac{n}{2}$ $+\sum_{i=1}^{n}\frac{1}{i}$ | $k-1+\sum_{i=1}^{k}\frac{1}{i}$ $-\frac{2(k-1)}{n}-\frac{k!(n-k)!}{n!}$ | $\frac{m}{2}+n-4+\frac{n}{2}$ $+\sum_{i=1}^{n}\frac{1}{i}$ | $\frac{m}{2}+k-1+\sum_{i=1}^{k}\frac{1}{i}$ $-\frac{2(k-1)}{n}-\frac{k!(n-k)!}{n!}$ |
| Cost | $m^2$ | $(n-1)\lfloor\frac{3(n-1)}{2}\rfloor$ | $(n-1)(2k-1)$ (if $1 \le k \le \lfloor\frac{n}{2}\rfloor$) $(n-1)(k+\lfloor\frac{n-1}{2}\rfloor)$ (if $\lfloor\frac{n}{2}\rfloor+1 \le k \le n-1$) | $(m+n-1)(m+\lfloor\frac{3(n-1)}{2}\rfloor)$ | $(m+n-1)(m+2k-1)$ (if $1 \le k \le \lfloor\frac{n}{2}\rfloor$) $(m+n-1)(m+k+\lfloor\frac{n-1}{2}\rfloor)$ (if $\lfloor\frac{n}{2}\rfloor+1 \le k \le n-1$) |

## 3.4 Comparison on Degree and Diameter

The node degree and diameter are key properties of the interconnection networks. The node degree is the maximum number of the neighbors of a node in the whole network and the diameter is the value of maximum shortest distance of all pairs of the nodes. Node degree represents the port number of a switch module like the Infiniband. Generally, the more degree the network has, the higher hardware cost of the network becomes. Diameter is used for estimating the maximum delay in transmitting a message from one processor to another and influences the message traffic density and fault-tolerance. Thus a network with the lower node degree and shorter diameter is desired. To evaluate such a network it is needed to compare these two properties simultaneously.

Figures 8 and 9 show the comparison of the node degree and diameter against the total number of nodes of the generalized-star cube with that of the hypercube, star graph, star-cube, and $(n,k)$-star graph, respectively. The proposed network can connect a more variety of the number of nodes than others. For example, when we want to make a 100,000-node network, we need to connect at least 131,072, 362,880, 122,880, and 151,200 nodes with the hypercube, star graph, star-cube, and $(n,k)$-star graph, respectively. Contrastingly, the GSC$(n,k,m)$ can connect 107,520 or 110,880 nodes. Note that the node degree of $(11,1)$-star graph is 10. Actually, the $(11,1)$-star graph is a 11-node complete graph, whose diameter is 1. Also, the node degree of $(11,2)$-star graph is 10; its diameter is 3. If the generalized-star cube contains such $(n,k)$-star graphs, the node degree will exceed that of

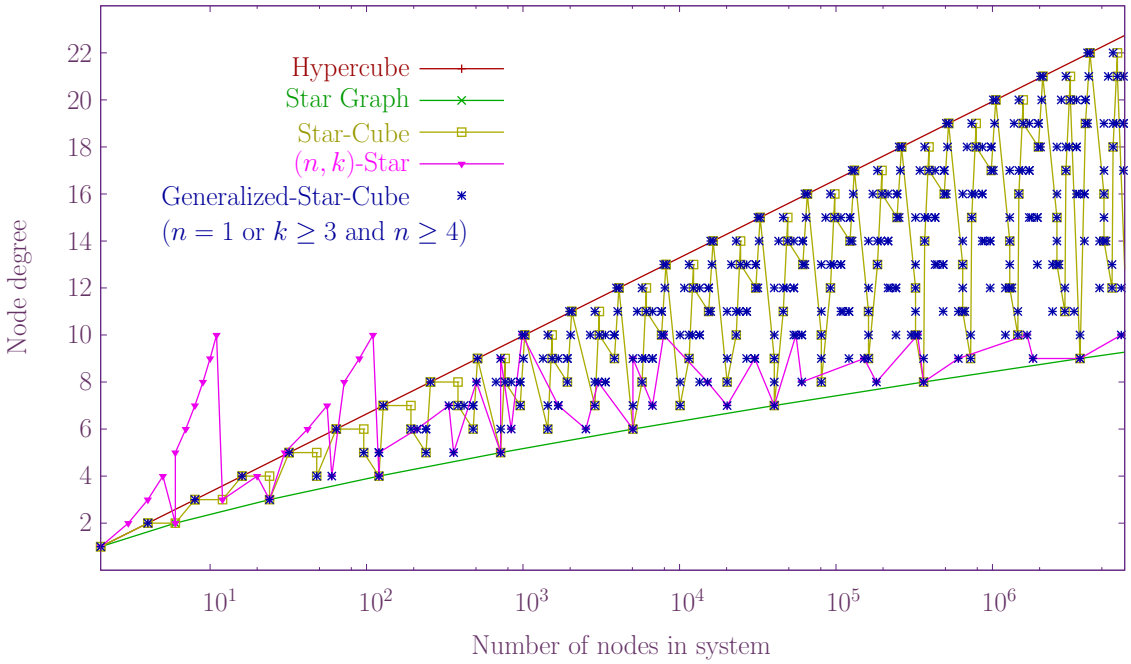



Figure 8: Node degree

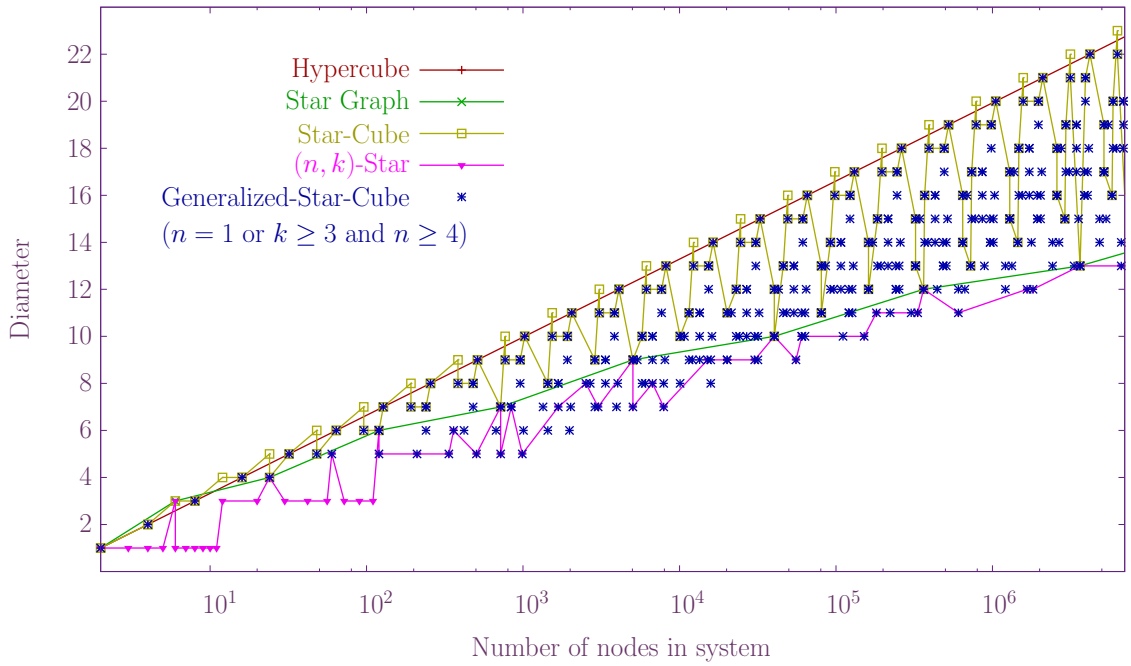



Figure 9: Diameter

hypercube of similar network sizes. Therefore, we recommend $k \geq 3$ (and hence $n \geq 4$) for building a GSC$(n, k, m)$. Such a restriction is reasonable because as $k$ approaches 1, the $(n, k)$-star graph of the GSC$(n, k, m)$ is close to the complete graph of dimension $n$ and such a network is unpractical for constructing a large-scale network.

The diameters of the GSC$(n, k, m)$ with $k \geq 3$ and $n \geq 4$ fall in between the hypercube and $(n, k)$-star graph, as shown in Figure 9. From the figure we can see that the diameter nonlinear variation of the GSC$(n, k, m)$ is resulted from the domination of the cube part and $(n, k)$-star part. When the cube part dominates the network, the diameter is close to the hypercube diameter. Similarly, when the $(n, k)$-star part dominates the network, the diameter is close to the $(n, k)$-star graph diameter. As observed from Figures 8 and 9, the network size of the generalized-star cube changes in smaller steps. Thus we can choose more desirable network size than the hypercube, star graph, star-cube, and $(n, k)$-star graph.

Here, we investigate degree and diameter at the same time, with the comparison on different configurations. For a given approximate sized GSC$(n, k, m)$, selecting a larger number of $m$ will make both the degree and diameter larger. Table 2 gives some GSC$(n, k, m)$ examples with around 100,000 nodes, where #Hypercube $= 2^m$ is the number of nodes in an $m$-cube and $\#(n, k)$-star is the number of nodes in an $(n, k)$-star graph; their product is the total number of nodes in the GSC$(n, k, m)$. We can check that, both GSC$(3, 1, 15)$ and GSC$(4, 3, 12)$ have 98,304 nodes and their diameters are 16. But their degrees are 17 and 15, respectively. Therefore, the GSC$(4, 3, 12)$ which has a smaller $m$ has a better performance than GSC$(3, 1, 15)$.

Table 2: GSC$(n, k, m)$ examples with around 100,000 nodes

| Nodes | $n$ | $k$ | $m$ | Degree | Diameter | Cost | #Hypercube | $\#(n, k)$-star |
|---|---|---|---|---|---|---|---|---|
| 90,112 | 11 | 1 | 13 | 23 | 14 | 322 | 8192 | 11 |
| 92,160 | 6 | 4 | 8 | 13 | 14 | 182 | 256 | 360 |
| 92,160 | 6 | 5 | 7 | 12 | 14 | 168 | 128 | 720 |
| 92,160 | 10 | 2 | 10 | 19 | 13 | 247 | 1024 | 90 |
| 92,160 | 10 | 3 | 7 | 16 | 12 | 192 | 128 | 720 |
| 96,768 | 9 | 4 | 5 | 13 | 12 | 156 | 32 | 3024 |
| **98,304** | **3** | **1** | **15** | **17** | **16** | **272** | 32768 | 3 |
| 98,304 | 3 | 2 | 14 | 16 | 17 | 272 | 16384 | 6 |
| 98,304 | 4 | 2 | 13 | 16 | 16 | 256 | 8192 | 12 |
| **98,304** | **4** | **3** | **12** | **15** | **16** | **240** | 4096 | 24 |
| 98,304 | 6 | 1 | 14 | 19 | 15 | 285 | 16384 | 6 |
| 107,520 | 7 | 3 | 9 | 15 | 14 | 210 | 512 | 210 |
| 107,520 | 7 | 4 | 7 | 13 | 14 | 182 | 128 | 840 |
| 107,520 | 8 | 4 | 6 | 13 | 13 | 169 | 64 | 1680 |
| 107,520 | 8 | 5 | 4 | 11 | 12 | 132 | 16 | 6720 |
| 110,880 | 11 | 5 | 1 | 11 | 10 | 110 | 2 | 55440 |
| 112,640 | 11 | 2 | 10 | 20 | 13 | 260 | 1024 | 110 |
| 114,688 | 7 | 1 | 14 | 20 | 15 | 300 | 16384 | 7 |
| 114,688 | 8 | 2 | 11 | 18 | 14 | 252 | 2048 | 56 |

Finally, we check network cost in Figure 10. Network cost is represented as the product of degree and diameter, so naturally, high-degree and high-diameter topology such as hypercube has high-cost, and low-degree and low-diameter topology such as $(n, k)$-star graph has low-cost. Generalized-star cube is a product graph of hypercube and $(n, k)$-star graph, so its cost also falls in between their network cost as with the cases of the node degree and diameter. Of course, when the cube part dominates the network, the cost is close to the hypercube cost. Similarly, when the $(n, k)$-star part dominates the network, the cost is close to the $(n, k)$-star graph cost.

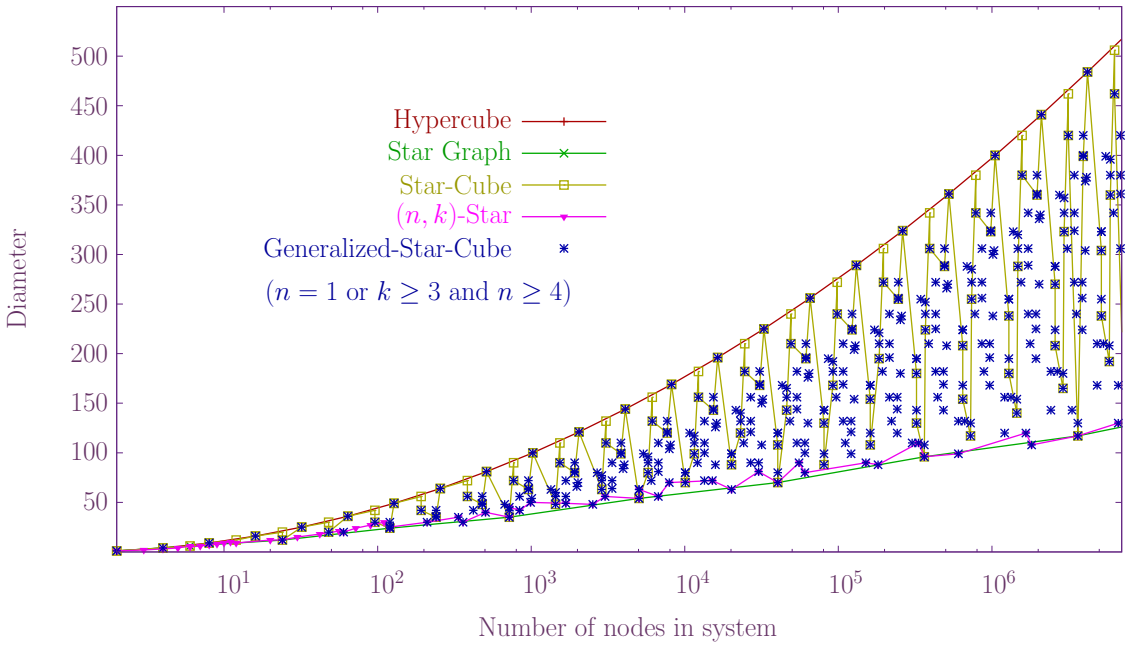



Figure 10: Cost

Consequently, generalized-star cube is a product graph of hypercube and $(n, k)$-star graph, so we can have a great amount of options of the network size. This flexible choice is the very outstanding merit of GSC and its degree and diameter change between hypercube and $(n, k)$-star graph, so we should pick up advantageous selections such as networks comprised of sizable $(n, k)$-stars.

# 4 Shortest-Path Routing Algorithm

In interconnection networks, routing algorithm is an important problem. An optimal routing algorithm is to find a shortest path from a source node to a destination node and to transmit the message through the path. In the generalized-star cube network, routing algorithm is constructed by using the existing routing algorithms for hypercube and $(n, k)$-star graph. Because both hypercube and $(n, k)$-star graph have simple routing algorithms, the routing algorithm of the product graph structure is also simple. Before discussing the details of the routing algorithm for the generalized-star cube, the following notations are defined: $\langle \boldsymbol{s}, \boldsymbol{u} \rangle$ denotes the address of the source node; $\langle \boldsymbol{t}, \boldsymbol{v} \rangle$ denotes the address of the destination node; $symbol \in \langle n \rangle - \langle k \rangle$ denotes an external symbol of $(n, k)$-star graph ($\langle n \rangle = \{1, 2, \ldots, n\}$); and $symbol \in \langle k \rangle$ denotes an internal symbol. The routing algorithm is shown as below.

**Step 1** Transmit message from $\langle \boldsymbol{s}, \boldsymbol{u} \rangle$ to $\langle \boldsymbol{t}, \boldsymbol{u} \rangle$ using a routing algorithm for hypercube.

**Step 2** Transmit message from $\langle \boldsymbol{t}, \boldsymbol{u} \rangle$ to $\langle \boldsymbol{t}, \boldsymbol{v} \rangle$ using a routing algorithm for $(n, k)$-star graph.

In Step 1, the cube part routing algorithm, we need to correct cube-label $\boldsymbol{s}$ of $\langle \boldsymbol{s}, \boldsymbol{u} \rangle$. In the $m$-cube, cube-label is an $m$-bit address and at most $m$ steps are needed to reach from a source node to a destination node. Therefore, the routing computational complexity of the $m$-cube is $\boldsymbol{O}(m)$.

In Step 2, the $(n, k)$-star part routing algorithm, we need to correct $(n, k)$-star-label $\boldsymbol{u}$ of $\langle \boldsymbol{t}, \boldsymbol{u} \rangle$. To clarify our presentation, we consider the destination node label $\boldsymbol{v}$ as the identity node address $\boldsymbol{I}_k = 1\,2 \ldots k$. Similar to the well-known cyclic representation [3], we can represent the cyclic representation for an $(n, k)$-star graph. For each external symbol $u_{m_i}$ ($\notin \langle k \rangle$) in label $\boldsymbol{u}$, we construct an external cycle $C_i = (u_1, u_2, \ldots, u_{m_i})$. In the external cycle, each symbol $u_j$ in $\boldsymbol{u}$ desires the correct

position and it is held by $u_{j+1}$ for $1 \leq j \leq m_i-1$. Additionally, all $u_j$, for $1 \leq j \leq m_i-1$, are internal symbols ($\in \langle k \rangle$). For each external cycle, we define the desired symbol $d_i$ whose desired position is held by the first element $u_1$ of the cycle $C_i$. The rest of the cycles are internal cycles defined as [3]. Therefore, we can find the path from $\boldsymbol{u}$ to $\boldsymbol{I}_k$ ($\boldsymbol{v}$) in the $(n,k)$-star graph by swapping internal symbols and exchanging the external symbols with desired symbols.

The detailed routing algorithm is formally given in Algorithm 1. We assume that $\boldsymbol{s}$ is a source node ($\boldsymbol{s\_s}$ and $\boldsymbol{s\_u}$ are hypercube address and $(n,k)$-star graph address, respectively), $\boldsymbol{t}$ is a destination node ($\boldsymbol{t\_t}$ and $\boldsymbol{t\_v}$ are hypercube address and $(n,k)$-star graph address, respectively), and $\boldsymbol{P}$ is a shortest path from $\boldsymbol{s}$ to $\boldsymbol{t}$. We give an example to find the routing path from $\boldsymbol{u}$ to $\boldsymbol{I}_k$ ($\boldsymbol{v}$). Let $\boldsymbol{u}$ = 3219586 and $\boldsymbol{v}$ = 1234567 in S(9,7). By following the scheme described above, we construct the external cycles $((7),6,8)$ and $((4),9)$ and internal cycle $(1,3)$, where the symbol enclosed in parentheses signifies the desired symbol. First, we need to swap 3 and the symbol in its desired position along internal cycle $(1,3)$ as following ($\rightarrow_\alpha$ means to pass the $\alpha$-edge):

$$\underline{3}2\underline{1}9586 \rightarrow_3 1239586$$

Next, because the first symbol 1 of $\boldsymbol{I}_k$ comes in the first position, we need to swap it and the symbol in not correct position. If there are multiple external symbols in $\boldsymbol{u}$, we choose any one and swap with 1. Then, we now select 8 and swap with 1 as following:

$$\underline{1}239\underline{8}6 \rightarrow_6 8239516$$

In the ordinary way, this step is not included in internal and external cycles. As the result of this step, the external cycle $((7),6,8)$ is modified to $((7),6,1)$. Now, we need to replace the external symbol 8 with one of two candidates 7 and 4. In this case, we select a desired symbol in the external cycle not including 1. Otherwise we will have extra steps to swap 1 with a symbol not in desired position. Thus, we select the candidate 4, replace 8 with 4, and correct the rest of the external symbol as following (replacing 9 with 7):

$$\underline{8}239516 \rightarrow_1 \underline{4}23\underline{9}516 \rightarrow_4 \underline{9}234516 \rightarrow_1 7234516$$

Then to correct along external cycle $((7),6,1)$, we get

$$\underline{7}2345\underline{1}6 \rightarrow_7 \underline{6}2345\underline{1}7 \rightarrow_6 1234567$$

Therefore, the routing computational complexity of $(n,k)$-star graph is $\boldsymbol{O}(k)$. In the routing algorithm of the GSC$(n,k,m)$, we can also find the path in another way, i.e., from $\langle \boldsymbol{s},\boldsymbol{u} \rangle$ to $\langle \boldsymbol{s},\boldsymbol{v} \rangle$ to $\langle \boldsymbol{t},\boldsymbol{v} \rangle$. We can derive length of the path from $\langle \boldsymbol{s},\boldsymbol{u} \rangle$ to $\langle \boldsymbol{t},\boldsymbol{v} \rangle$ by summing up the length of the paths from $\langle \boldsymbol{s},\boldsymbol{u} \rangle$ to $\langle \boldsymbol{t},\boldsymbol{u} \rangle$ and $\langle \boldsymbol{t},\boldsymbol{u} \rangle$ to $\langle \boldsymbol{t},\boldsymbol{v} \rangle$. In addition, we have the following three patterns of routing computational complexity of the GSC$(n,k,m)$. When cube part dominates the network, the routing complexity of the GSC$(n,k,m)$ is $\boldsymbol{O}(m)$. Similarly, when $(n,k)$-star part dominates the network, the complexity is $\boldsymbol{O}(k)$. Otherwise it is $\boldsymbol{O}(m+k)$. Table 3 compares the routing complexities for different networks.

Table 3: Comparison of the routing complexity for different networks

| HC($m$) | $n$-Star | $(n,k)$-Star | SC$(n,m)$ | GSC$(n,k,m)$ |
|---|---|---|---|---|
| $\boldsymbol{O}(m)$ | $\boldsymbol{O}(n)$ | $\boldsymbol{O}(k)$ | $\boldsymbol{O}(m)$ (when cube part $\gg$ star part)<br>$\boldsymbol{O}(n)$ (when cube part $\ll$ star part)<br>$\boldsymbol{O}(m+n)$ (otherwise) | $\boldsymbol{O}(m)$ (when cube part $\gg$ $(n,k)$-star part)<br>$\boldsymbol{O}(k)$ (when cube part $\ll$ $(n,k)$-star part)<br>$\boldsymbol{O}(m+k)$ (otherwise) |

**Algorithm 1** Routing_Algorithm($\boldsymbol{s}$, $\boldsymbol{t}$)

```
1: P := [s];                                  /* (s_1, ..., s_m) and (t_1, ..., t_m) are hypercube address */
2:                                            /* (u_1, ..., u_k) and (v_1, ..., v_k) are (n,k)-star graph address */
3: C := makeExternalCycle(s_u);   /* makeExternalCycle(c) is a function that makes external
4:                                                                          cycles by label c */
5: flag_LoweringPriority := 0;    /* flag_LoweringPriority is a flag to lower external cycles */
6: /* hypercube part */
7: for i = 1 to m do
8:    if s_s_i ≠ t_t_i then
9:       s_s_i := s̃_s_i;                          /* a_b_i is ith address of label b of node a */
10:      P := P ∪ [s];                                          /* ~ is a bit inversion operation */
11:   end if
12: end for
13: /* (n,k)-star graph part */
14: while s_u ≠ t_v do
15:   if s_u_1 ∉ t_v then
16:      if flag_LoweringPriority = 1 then
17:         lowerCyclePriority(C, s_u_1); /* lowerCyclePriority(Cycle, s) is a function that lowers
18:                                                          the priority of Cycle including symbol s */
19:         flag_LoweringPriority := 0;
20:      end if
21:      s_u_1 := getDesiredSymbol(C);        /* getDesiredSymbol(Cycle) is a function that gets
22:                  a desired symbol of first cycle in Cycle then removes its cycle from Cycle */
23:      P := P ∪ [s];
24:   end if
25:   if s_u_1 ≠ t_v_1 then
26:      find i such that t_v_i = s_u_1;
27:   else
28:      if s_u \ t_v ≠ φ then
29:         assign i to the position of any external symbol;
30:         flag_LoweringPriority := 1;
31:      else
32:         find i such that s_u_i ≠ t_v_i;
33:      end if
34:   end if
35:   swap(s_u, 1, i);                          /* swap(a, i, j) is a function that swaps a_i with a_j */
36:   P := P ∪ [s];
37: end while
38: return P
```

# 5 Broadcasting on the Single-Port Model

In this section, we develop the broadcasting algorithm for the single-port model. This algorithm is separated into two parts just like the shortest-path routing algorithm: hypercube part and $(n, k)$-star graph part. First, we consider the broadcasting algorithm for the hypercube part. To implement the algorithm, we use the spanning binomial tree. For the $(n, k)$-star graph part, we adopt an optimal neighborhood broadcasting algorithm for developing an optimal broadcasting algorithm for GSC$(n, k, m)$ on the single-port model.

## 5.1 Spanning Binomial Tree

We can use the spanning binomial tree communication scheme for broadcasting of hypercube, because hypercube's symmetric and binary recursive topology fits perfectly to this communication

scheme. In general, a binomial tree is defined recursively as follows: (1) a binomial tree of order 0 is a single node; (2) a binomial tree of order $m$ has a root node whose children are roots of binomial trees of orders $m-1, m-2, \ldots, 2, 1, 0$ (in this order); a binomial tree of order $m$ has $2^m$ nodes, height $m$. Because the hypercube is both vertex and edge symmetric, we can place the root of a spanning binomial tree in any hypercube node and use the hypercube dimensions in any order [18]. The number of nodes that receive the message in step $i$ is $2^{i-1}$ in the single-port model. Therefore, in $m$ step, $\sum_{i=1}^{m} 2^{i-1} = 2^m - 1$. Hence, this scheme is transmission optimal ($\boldsymbol{O}(m)$).

The detailed broadcast algorithm for hypercube on the single-port model is formally given in Algorithm 2. We use 4 parameters. $d$ is dimensions of the hypercube; ***my_id*** is an ID assigned to each node; $\boldsymbol{s}$ is the ID of a source node; and $\boldsymbol{X}$ is a message. All nodes perform this algorithm in parallel as soon as they received the message $\boldsymbol{X}$, sent by $\boldsymbol{s}$ originally. ***my_virtual_id*** and ***mask*** are used to determine whether a node sending message or not.

**Algorithm 2** ONE_TO_ALL_BC_single-port($d$, ***my_id***, $\boldsymbol{s}$, $\boldsymbol{X}$)

```
1:  my_virtual_id := my_id XOR s;
2:  mask := 2^d − 1;
3:  for i = d − 1 to 0 do
4:    mask := mask XOR 2^i;
5:    if (my_virtual_id AND mask) = 0 then
6:      if (my_virtual_id AND 2^i) = 0 then
7:        virtual_destination := my_virtual_id XOR 2^i;
8:        send(virtual_destination, X);
9:      end if
10:   end if
11: end for
```

Figure 11 shows the case of source node $\boldsymbol{s} = 0000$ in a 4-cube broadcasting a message. It performs broadcasting in 4 steps:

- Step 1:
  0000 → 1000
- Step 2:
  0000 → 0100
  1000 → 1100
- Step 3:
  0000 → 0010
  0100 → 0110
  1000 → 1010
  1100 → 1110
- Step 4:
  0000 → 0001
  0010 → 0011
  0100 → 0101
  0110 → 0111
  1000 → 1001
  1010 → 1011
  1100 → 1101
  1110 → 1111

## 5.2 Neighborhood Broadcasting

The neighborhood broadcasting problem, NBP for short, is a problem that a message of the source node is sent to all its neighbors in the single-port model. This problem for both star graph and $(n, k)$-star graph has been studied well and optimal algorithms were derived [9, 21, 13]. In [9], Fujita developed the algorithm by embedding binomial trees into the star graph. In [21, 13, 14], more simple algorithm was developed with the cycle structures. In this paper, we adopt this neighborhood broadcast algorithm of the cycle structures. For some interconnection topologies with constant node degrees, the time required for neighborhood broadcasting is constant. The lower bound of this NBP on a network with degree $d$ is $\boldsymbol{\Omega}(\log d)$ [14]. For example, the lower bound for NBP in $S_{n,k}$, an $(n, k)$-star graph, is $\boldsymbol{\Omega}(\log n)$ because the degree of $S_{n,k}$ is $n - 1$.

For simplicity, we use the notation $i*$ to represent a node whose first symbol is $i$. Similarly, $*i$ represents a node whose last symbol is $i$. Let $S_{n-1,k-1}(i)$ be a subgraph where all the nodes are of the form $*i$, $1 \leq i \leq n$, then $S_{n-1,k-1}(i)$ is isomorphic to an $(n-1, k-1)$-star graph. This gives us

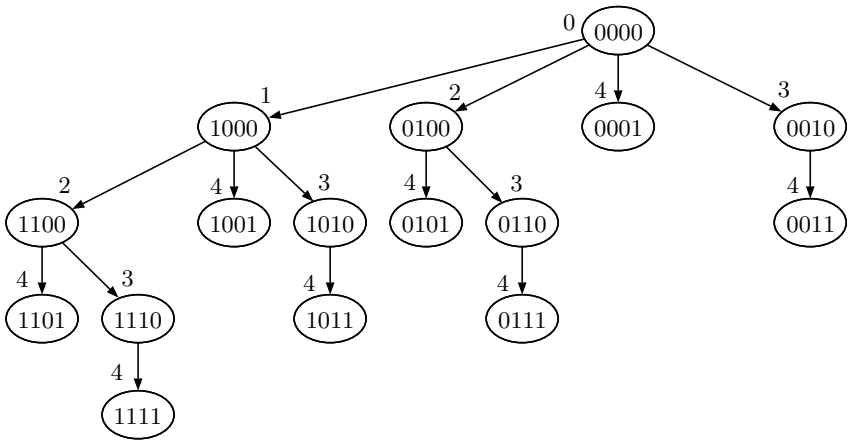



Figure 11: Spanning binomial tree on single-port 4-cube

one way to decompose an $S_{n,k}$ into $n$ $S_{n-1,k-1}(i)$, for $1 \leq i \leq n$ [6, 7]. Unless otherwise stipulate, we will decompose the $(n,k)$-star graph at the last dimension. We efficiently utilize this property for broadcasting on the all-port as well as the single-port model. Because the $(n,k)$-star graph is vertex symmetric, without loss of generality, we assume that the source node is $12\cdots k$. Its $i$-edge neighbors are shown as:

$$21345\cdots k,\ 32145\cdots k,\ 42315\cdots k,\ \ldots,\ k234\cdots 1,$$

and its 1-edge neighbors are shown as:

$$(k+1)234\cdots k,\ (k+2)234\cdots k,\ \ldots,\ n234\cdots k.$$

The neighborhood broadcasting and broadcast algorithms for the two port models are based on the following observations on structural properties of the $(n,k)$-star graph. The proofs for these observations are fairly straightforward and can be found in [14].

**Observation 1** *For any $r \neq 1$, $S_{r,1}$ is a clique $K_r$ (a complete graph of size $r$).*

**Observation 2** *In $S_{n,k}$, for any node $u$, $u$ and all its 1-edge neighbors form a clique $K_{n-k+1}$.*

**Observation 3** *For any $i$-edge neighbor $i*k = i23\cdots(i-1)1(i+1)\cdots k$ and $j$-edge neighbor $j*k = j23\cdots(j-1)1(j+1)\cdots k$ of the node $12\cdots k$ (we assume that $i < j$ without loss of generality), they are on the same cycle of length 6 as follows:*

$$123\cdots i\cdots j\cdots k \rightarrow i23\cdots 1\cdots j\cdots k \rightarrow j23\cdots 1\cdots i\cdots k \rightarrow$$
$$123\cdots j\cdots i\cdots k \rightarrow i23\cdots j\cdots 1\cdots k \rightarrow j23\cdots i\cdots 1\cdots k \rightarrow$$

This cycle involves only $i$-edges. In fact, the above observation also holds true when $k+1 \leq j \leq n$.

**Observation 4** *For any $i$-edge neighbor $i*k = i23\cdots(i-1)1(i+1)\cdots k$ and 1-edge neighbor $j*k = j23\cdots k$ of the node $12\cdots k$, where $k+1 \leq j \leq n$, they are on the same cycle of length 6 as follows:*

$$123\cdots(i-1)i(i+1)\cdots k \rightarrow i23\cdots(i-1)1(i+1)\cdots k \rightarrow j23\cdots(i-1)1(i+1)\cdots k \rightarrow$$
$$123\cdots(i-1)j(i+1)\cdots k \rightarrow i23\cdots(i-1)j(i+1)\cdots k \rightarrow j23\cdots(i-1)i(i+1)\cdots k \rightarrow$$

This cycle involves both $i$-edges and 1-edges.

**Observation 5** *Any two 6-cycles formed as in Observations 3 and 4 with distinct $2 \leq i_1, j_1, i_2, j_2 \leq n$ are disjoint except that they share the source node $12\cdots k$.*

Initially, only the source node has a message. In the first step, the source node sends a message to one of its neighbors through the direct link. In the second step, two nodes have the message. One of them, the source node sends the message like the first step, whereas another one sends the message to a neighbor of the source node through a length-4 path that is part of a 6-cycle. Now the number of neighbors having message including the source node are 4. Next, these four nodes send the message again in the same manner. Thus, three neighbors send the message to other three neighbors of the source node via disjoint paths of length-4 that are parts of three disjoint 6-cycles and the source node forwards directly. This algorithm ends when all neighbors of the source node receive the message.

The detailed neighborhood broadcast algorithm is formally given in Algorithm 3. We use 5 parameters. $\boldsymbol{my\_id}$ is an ID assigned to each node; $flag\_send\_msg$ is a flag specifying whether my node sends a message or not; $\boldsymbol{s}$ is the ID of the source node; $\boldsymbol{Neighbors}$ is a list of neighbors of the source node; and $\boldsymbol{X}$ is a message. The key idea of this algorithm is to design in such a way that: (1) a source node sends a message with direct links; (2) neighbors of the source node send the message in parallel; and (3) if under four neighbors remain, the source node sends the message by direct links; otherwise go back to step (1) and (2). Note that in (1) and (2), all neighbors of the source node which have received the message in each step start the next iteration with other senders. Obviously, after each step, the number of neighbors having the message is doubled (but not done always in the last step). After finished sending the messages to all neighbors, these neighbors' $flag_send_msg$s are assigned to 1.

For example, in an $(8, 4)$-star graph, for the source node $\boldsymbol{s} = 1234$, this neighborhood broadcasting is shown as follows:

- Step 1:
  $1234 \rightarrow 2134$
- Steps 2-5:
  $1234 \rightarrow 3214$
  $2134 \rightarrow 4132 \rightarrow 1432 \rightarrow 2431 \rightarrow 4231$
- Steps 6-9:
  $1234 \rightarrow 5234$
  $2134 \rightarrow 6134 \rightarrow 1634 \rightarrow 2634 \rightarrow 6234$
  $3214 \rightarrow 7214 \rightarrow 1274 \rightarrow 3274 \rightarrow 7234$
  $4231 \rightarrow 8231 \rightarrow 1238 \rightarrow 4238 \rightarrow 8234$

The running time for this algorithm is $\boldsymbol{O}(\log n)$ [14]. Because the lower bound is $\boldsymbol{\Omega}(\log n)$, this algorithm is optimal. Of course, when $n$ is relatively small, it is better to simply forward a message from the source node to its $n-1$ neighbors in $n-1$ steps.

## 5.3 Broadcast Algorithm on the Single-Port Model

The broadcasting problem, BP for short, is a problem a message of the source node is sent to all the nodes in the network. For the single-port model, the BP has a lower bound of $\boldsymbol{\Omega}(\log N)$, where $N$ is the total number of nodes in the network. Therefore, the lower bound for this broadcasting problem on single-port $S_{n,k}$ is $\boldsymbol{\Omega}(\log(n!/(n-k)!)) = \boldsymbol{\Omega}(k \log n)$. Several broadcast algorithms for $(n, k)$-star graph have been studied [5, 16, 14]. Among them, in [14], an optimal time algorithm is proposed by using the neighborhood broadcasting.

The idea of this scheme can be described as follows. Since $S_{n,k}$ can be decomposed as $n$ number of $S_{n-1,k-1}$, the source node sends a message to one node in each of $S_{n-1,k-1}(i)$, where $1 \leq i \leq n$. Now every $S_{n-1,k-1}(i)$ has a node with the message, it recursively carries out the algorithm on each $S_{n-1,k-1}(i)$. Concretely, we assume that the source node is $\boldsymbol{e}_k = 123 \cdots k$ and wants to broadcast a message to all the other nodes in $S_{n,k}$. In the first step, the source node sends the message to its all neighbors using the neighborhood broadcasting. Now all $i$-neighbors of the source node $\boldsymbol{e}_k$ ($2 * k$, $3 * k$, ..., $(k-1) * k$ and $k * 1$) and all 1-neighbors ($(i+1) * k$, $(i+2) * k$, ..., $(n-1) * k$ and $n * k$) have

**Algorithm 3** Neighborhood_Broadcasting(***my_id***, $flag\_send\_msg$, ***s***, ***Neighbors***, ***X***)

```
1: cnt := 1;
2: total_nodes := lengthof(Neighbors);
3: /* At each step, the number of neighbors which have received the message can double */
4: for i = 0 to ⌈log(total_nodes/2)⌉ do
5:   remaining_nodes := total_nodes − cnt;
6:   /* Not need 4-length path for sending messages to the rest of less than 4 neighbors */
7:   if 1 ≤ remaining_nodes ≤ 3 then
8:     for j := 0 to remaining_nodes − 1 do
9:       if my_id = s then
10:        path := Neighbors[total_nodes − j];
11:        send(path, X);      /* send(Path, Message) is a function that sends a Message
12:                                                                  through Path */
13:      end if
14:    end for
15:    stop
16:  else
17:    /* Send messages in parallel */
18:    if my_id = s then
19:      send(Neighbors[2^i], X);
20:    else
21:      /* Neighbors of a source node which have a message */
22:      index := get_index(Neighbors, my_id);      /* get_index(List, element) is a function
23:                                                   that gets index of List where element is stored */
24:      if index + 2^i ≤ total_nodes then
25:        path := make_4_length_path(my_id, Neighbors[index + 2^i]);
26:                                              /* make_4_length_path(Src, Dest) is a function
27:                                                 that makes a length-4 path from Src to Dest */
28:        send(path, X);
29:      else
30:        stop
31:      end if
32:    end if
33:    cnt := 2cnt;
34:    (All neighbors of the source node which have received the message in this step
35:    start the next iteration with other senders)
36:  end if
37: end for
38: if my_id is included in Neighbors then
39:   flag_send_msg := 1;
40: end if
```

the message. Then, these neighbors (except $k*1$) send the message through $k$-dimensional edges in one more time unit. Now $n$ nodes ($*1$, $*2$, ..., $*n$) have the message and these nodes belong to every $S_{n-1,k-1}(i)$, $1 \leq i \leq n$. Therefore, we can recursively broadcast in each $S_{n-1,k-1}(i)$ in parallel. This broadcasting algorithm has $\boldsymbol{O}(k \log n)$ time and is optimal in the view of the $\boldsymbol{\Omega}(k \log n)$ lower bound [14].

The detailed broadcast algorithm for $(n, k)$-star graph on the single-port model is formally given in Algorithm 4. We use 6 parameters. $n$ and $k$ are parameters for $(n, k)$-star graph and the rest of 4 parameters are same as in Algorithm 3. The key idea of this algorithm is to design in such a way that: (1) a source node sends a message to its all neighbors using the neighborhood broadcasting; (2) all neighbors (except $k*1$) send the message through $k$-dimensional edges; (3) these nodes which received the messages in the previous step perform this broadcast algorithm recursively as

**Algorithm 4** Broadcasting_single-port($\boldsymbol{my\_id}$, $flag\_send\_msg$, $n, k$, $\boldsymbol{s}$, $\boldsymbol{X}$)

```
1: if k = 1 then
2:   send_clique(my_id, X);            /* send_clique(Src, Message) is a function that sends
3:                                      Messages centered around Src by a simple broadcasting algorithm */
4: else
5:   if (flag_send_msg = 1) and (my_id = s) then
6:     Neighbors := get_Neighbors(my_id, n − 1);   /* get_Neighbors(Node, n) is a function
7:                                                    that gets n neighbors of Node */
8:     /* Broadcast message to neighbors of a source node */
9:     Neighborhood_Broadcasting(my_id, flag_send_msg, s, Neighbors, X);
10:  end if
11:  /* Send message to all subgraphs of current graph along dimension k */
12:  if flag_send_msg = 1 then
13:    if get_1st_symbol(my_id) ! = get_kth_symbol(s, k) then
14:      if my_id ! = s then
15:        path := swap(my_id, 1, n);     /* swap(a, i, j) is a function that swaps a_i with a_j */
16:        send_to_new_source(path, X);   /* send_to_new_source(Path, X) is a function that
17:                                          change a current source node to destination
18:                                          node of Path (s := destination node &
19:                                                flag_send_msg of destination node := 1) */
20:      end if
21:      if (my_id ! = s) or ((my_id = s) and (n = 2) and (n−k = 1)) then
22:        flag_send_msg := 0;
23:      end if
24:    else
25:      s := my_id;                      /* My node becomes a new source node */
26:      flag_send_msg := 1;
27:    end if
28:  end if
29:  /* Perform broadcasting algorithm in a lower level subgraph */
30:  Broadcasting_single_port(my_id, flag_send_msg, n−1, k−1, s, X);
31: end if
```

new source nodes in their subgraphs; and (4) if these subgraphs form a clique, they simply carry out a standard broadcasting algorithm, otherwise go back to step (1). Note that after a source node gets its neighbors by using a function $Neighborhood\_Broadcasting$, all neighbors (with some exceptions) start to send messages to all subgraphs of current graph along dimension $k$. At this time, these neighbors are checked with their $flag_send_msg$s whether they have the messages or not.

For example, in a $(5,3)$-star graph, for the source node $\boldsymbol{s} = 123$, this broadcast algorithm is shown as follows. Note that in this example, we did not use the neighborhood broadcasting because the number of neighbors are relatively small.

- Step 1-5 ($(5,3)$-star graph):
  $123 \to 213 \to 312$ $123 \to 321$ $123 \to 423 \to 324$ $123 \to 523 \to 325$
- Step 5-9 ($(4,2)$-star graph):
  $123 \to 213$ $123 \to 423 \to 243$ $123 \to 523 \to 253$ $321 \to 231$ $321 \to 421 \to 241$
  $321 \to 521 \to 251$ $312 \to 132$ $312 \to 412 \to 142$ $312 \to 512 \to 152$ $324 \to 234$
  $324 \to 124 \to 214$ $324 \to 524 \to 254$ $325 \to 235$ $325 \to 125 \to 215$
  $325 \to 425 \to 245$
- Step 9-11 ($(3,1)$-star graph):
  $123 \to 423$ $123 \to 523$ $213 \to 413$ $213 \to 513$ $243 \to 143$ $243 \to 543$
  $253 \to 153$ $253 \to 453$ $321 \to 421$ $321 \to 521$ $231 \to 431$ $231 \to 531$

| | | | | | |
|---|---|---|---|---|---|
| $241 \to 341$ | $241 \to 541$ | $251 \to 351$ | $251 \to 451$ | $312 \to 412$ | $312 \to 512$ |
| $132 \to 432$ | $132 \to 532$ | $142 \to 342$ | $142 \to 542$ | $152 \to 352$ | $152 \to 452$ |
| $324 \to 124$ | $324 \to 524$ | $234 \to 134$ | $234 \to 534$ | $214 \to 314$ | $214 \to 514$ |
| $254 \to 154$ | $254 \to 354$ | $325 \to 125$ | $325 \to 425$ | $235 \to 135$ | $235 \to 435$ |
| $215 \to 315$ | $215 \to 415$ | $245 \to 145$ | $245 \to 345$ | | |

Consequently, we have the following three patterns of broadcasting computational complexity of the GSC$(n, k, m)$ on the single-port model. When cube part dominates the network, the lower bound approaches $\boldsymbol{\Omega}(m)(= \boldsymbol{\Omega}(\log 2^m))$ and the broadcasting complexity of the GSC$(n, k, m)$ is $\boldsymbol{O}(m)$. Similarly, when $(n, k)$-star part dominates the network, the lower bound approaches $\boldsymbol{\Omega}(k \log n)(= \boldsymbol{\Omega}(\log(n!/(n-k)!)))$ and the complexity is $\boldsymbol{O}(k \log n)$. Otherwise the lower bound and broadcasting complexity are $\boldsymbol{\Omega}(m + k \log n)$ and $\boldsymbol{O}(m + k \log n)$, respectively. Hence, this algorithm is optimal.

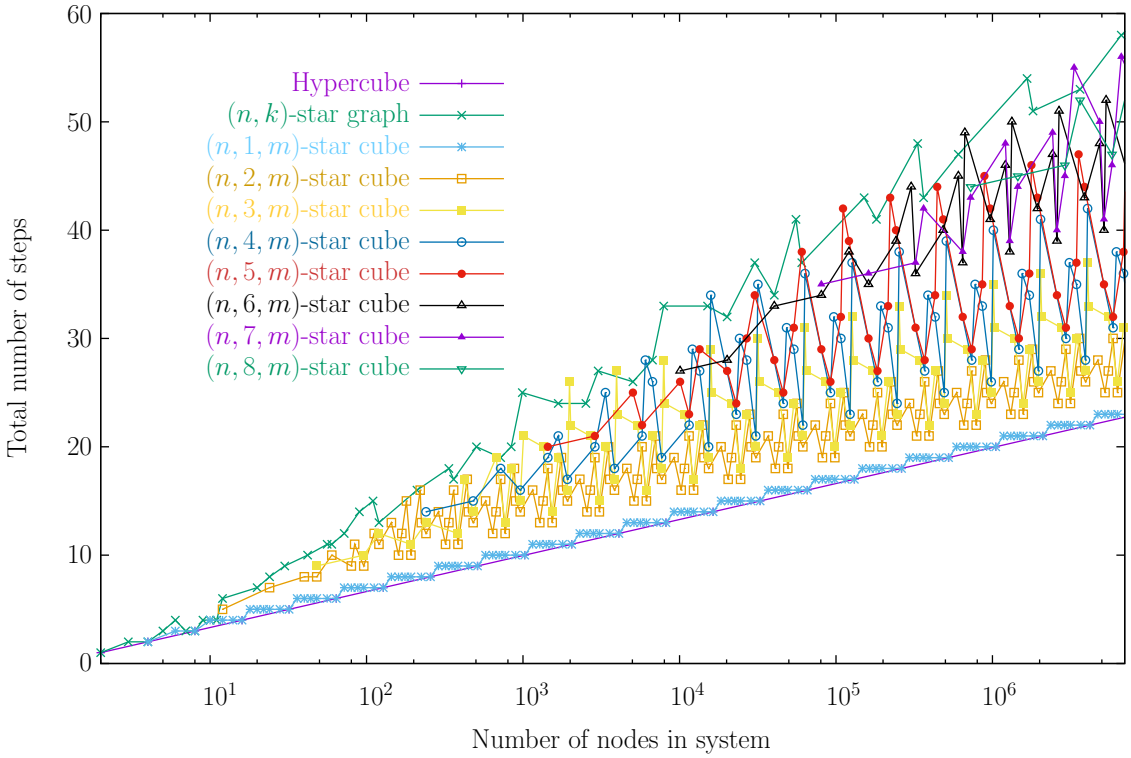



Figure 12: Total number of steps for broadcasting on single-port model

Figure 12 plots the total number of steps required for broadcasting at single-port model on hypercube, $(n, k)$-star graph, and generalized-star cube GSC$(n, k, m)$. We can see that as $k$ in GSC$(n, k, m)$ increases, the number of broadcasting steps also increases. This is because that the communication complexity of $(n, k)$-star graph is $\boldsymbol{O}(k \log n)$, larger than that of hypercubes.

# 6 Broadcasting on the All-Port Model

In this section, we develop the broadcasting algorithm on the all-port model. All-port model means that a node can send/receive messages to/from all its neighbors simultaneously. This algorithm is also separated into two part like the shortest-path routing algorithm and the broadcasting on the single-port model. First, we outline the spanning binomial tree at all-port model for hypercube part. Next, we introduce the minimum dominating set for $(n, k)$-star graph part. Then, we develop an optimal broadcasting algorithm on the all-port model using the minimum dominating set.

## 6.1 Spanning Binomial Tree

Recall from the previous section, we can use the spanning binomial tree communication scheme for broadcasting of hypercube, because hypercube's symmetric and binary recursive topology fits perfectly to this communication scheme. The number of nodes that receive the message in step $i$ is $\binom{m}{i}$ where $m$ is the dimension of the hypercube on the all-port model. Therefore, in $m$ step, $\sum_{i=1}^{m}\binom{m}{i} = 2^m - 1$. Hence, this scheme is transmission optimal ($\boldsymbol{O}(m)$).

The detailed broadcast algorithm for $d$-cube on the all-port model is formally given in Algorithm 5. We use 4 parameters. $d$ is dimensions of the hypercube; ***my_id*** is an ID assigned to each node; $\boldsymbol{s}$ is the ID of the source node; and $\boldsymbol{X}$ is a message. All nodes performs this algorithm in parallel as soon as they have received the message $\boldsymbol{X}$, sent by $\boldsymbol{s}$ originally. In the first step, we convert ***my_id*** to ***my_virtual_id***. Then, each node which has the message sends to the nodes whose addresses differ from ***my_virtual_id*** in a single bit of a right field whose value is zero of ***my_virtual_id***. For example, node 10100 sends a message to node 10110 and 10101; the right field has two bits 00 in node 10100. This operation is realized by using a function *send_binomial_tree*(***Addr***, ***Message***). This function sends ***Messages*** to nodes which have such addresses.

**Algorithm 5** ONE_TO_ALL_BC_all-port_cube($d$, ***my_id***, $\boldsymbol{s}$, $\boldsymbol{X}$)

```
1: my_virtual_id := my_id XOR s;
2: D := [ ];
3: i := 0;
4: while ((my_virtual_id AND 2^i) = 0) and (i < d) do
5:    D := D ∪ [my_virtual_id AND 2^i];
6:    i := i + 1;
7: end while
8: send_all_port(D, X);                    /* send X to neighbors in D simultaneously */
```

For example, in a 4-cube, for the source node $\boldsymbol{s} = 0000$, this broadcast algorithm is shown as follows and Figure 13:

| • Step 1: | • Step 2: | • Step 3: | • Step 4: |
|---|---|---|---|
| $0000 \rightarrow 1000$ | $1000 \rightarrow 1100$ | $1100 \rightarrow 1110$ | $1110 \rightarrow 1111$ |
| $0000 \rightarrow 0100$ | $1000 \rightarrow 1010$ | $1100 \rightarrow 1101$ | |
| $0000 \rightarrow 0010$ | $1000 \rightarrow 1001$ | $1010 \rightarrow 1011$ | |
| $0000 \rightarrow 0001$ | $0100 \rightarrow 0110$ | $0110 \rightarrow 0111$ | |
| | $0100 \rightarrow 0101$ | | |
| | $0010 \rightarrow 0011$ | | |
| (4-bit 0) | (3-, 2-, 1-bit 0) | (2-, 1-, 1-bit 0) | (1-bit 0) |

## 6.2 Minimum Dominating Set of the $(n, k)$-Star Graph

Generally, in graph theory, a dominating set for a graph $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$ is a subset $\boldsymbol{V'} \subseteq \boldsymbol{V}$ such that every vertex not in $\boldsymbol{V'}$ is adjacent to at least one member of $\boldsymbol{V'}$. The domination number is the number of vertices in $\boldsymbol{V'}$, and the minimum dominating set is a dominating set with the smallest domination number. The dominating set problem is to find a minimum dominating set $\boldsymbol{D_G}$ of a graph $\boldsymbol{G}$ with domination number $|\boldsymbol{D_G}|$.

Let $\boldsymbol{D}_{n,k}$ be a minimum dominating set of $\boldsymbol{S}_{n,k}$, then every vertex set $\boldsymbol{D}_{n,k} = \{i*\}$, for $1 \leq i \leq n$, and $|\boldsymbol{D}_{n,k}| = (n-1)!/(n-k)!$ are a minimum dominating set and its domination number, respectively [14]. For example, $S_{4,2}$ has four different minimum dominating sets depending on the value of $i$: (1) $\{12, 13, 14\}$ for $i = 1$, (2) $\{21, 23, 24\}$ for $i = 2$, (3) $\{31, 32, 34\}$ for $i = 3$, and (4) $\{41, 42, 43\}$ for $i = 4$.

$\boldsymbol{D}_{n,k}$ and its neighbors contain all the nodes of $S_{n,k}$, therefore in the all-port model, we can send a message in one time unit by using the $\boldsymbol{D}_{n,k}$. We use this idea and the hierarchical structure of $S_{n,k}$ to develop a broadcasting algorithm for the all-port model in the next subsection.

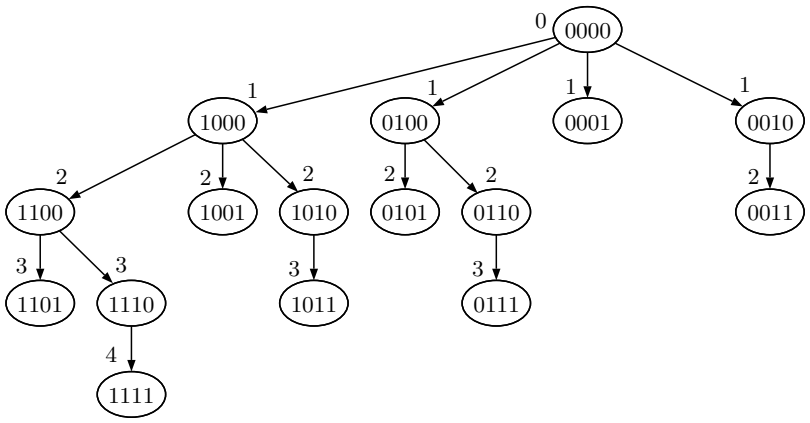



Figure 13: Spanning binomial tree on all-port 4-cube

## 6.3 Broadcasting Algorithm on the All-Port Model

When discussing the BP on interconnection networks of the all-port model, we need to consider the traffic (the total number of messages exchanged) in addition to the time (the number of steps required) [23]. Hence, it is desirable to minimize both the time and traffic. By mitigating the traffic, we can reduce the message redundancy which is a problem that a node receives the same message many times. Broadcast algorithms for $(n, k)$-star on the all-port model have been studied [16, 14]. Among them, in [14], an optimal time algorithm is proposed based on the minimum dominating set.

Now we have the minimum dominating set $\boldsymbol{D}_{n,k}$ (all the nodes forming $i*$) from the previous subsection. Then, a simple broadcasting algorithm on the all-port model for $S_{n,k}$ can be designed by using $\boldsymbol{D}_{n,k}$ as follows: (1) we decompose current subgraph at the last dimension of the source node until forming a clique; (2) when a subgraph forms a clique, the source node sends the message along dimension 1; (3) all nodes with the message send along an upper current dimension; and (4) since the nodes that received the message in the previous step are the minimum dominating set in current dimension, each node in the dominating set sends its message along all dimensions except current dimension (if not finished, go back to step (3)). The key point of this algorithm is that every time sender nodes go to step (3) and the current dimension increases by 1. This means that the available node addresses are deregulated. The detailed broadcast algorithm for $(n, k)$-star graph on the all-port model is formally given in Algorithm 6.

**Algorithm 6** Broadcasting_all-port($\boldsymbol{my\_id}$, $k$, $\boldsymbol{s}$, $\boldsymbol{X}$)

```
1: if k = 1 then
2:    send_1edges(my_id, X);          /* send_1edges(Node, Message) is a function that sends
3:                                       Messages to all 1-dimension neighbors of Node */
4: else
5:    Broadcasting_all-port(my_id, k − 1, s, X);
6:    if Node-my_id has a message X then
7:       path := swap(my_id, 1, k);
8:       send(path, X);
9:    end if
10:   if (Node-my_id has a message X) and (get_1st_symbol(my_id) = get_kth_symbol(s, k))
      then
11:      send_neighbors(my_id, k, X); /* send_neighbors(Node, k, Message) is a function that
12:                                      sends Messages to under k-dimension neighbors of Node */
13:   end if
14: end if
```

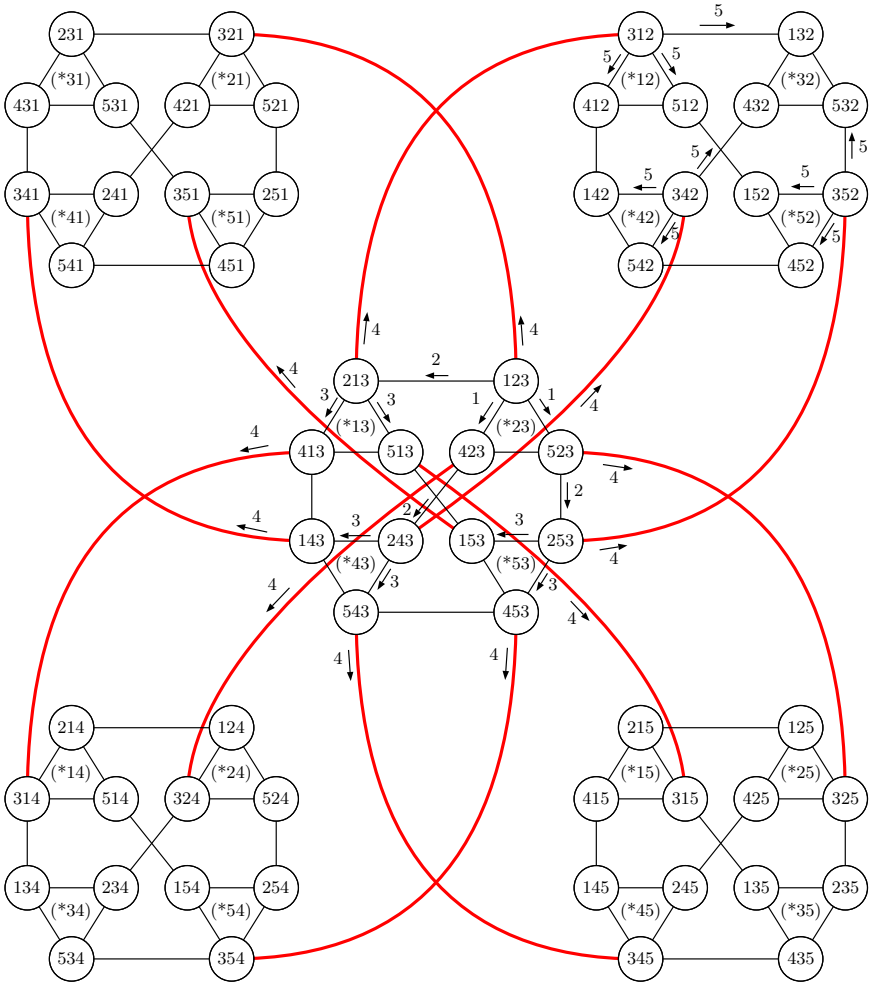



Figure 14: All-port broadcasting on a $(5,3)$-star graph

The optimal running time of this algorithm is proportional to the diameter of the network and $\boldsymbol{O}(k)$ [14]. Furthermore, there is no message redundancy.

Figures 14 and 15 show two examples of the broadcast algorithm in a $(5,3)$-star graph and a $(5,4)$-star graph, for the source node $\boldsymbol{s} = 123$ and $\boldsymbol{s} = 1234$, respectively, where the numbers near by arrowed lines are step numbers. Note that some of arrowed lines are left out because they are the same as the upper right subgraph in each figure.

Consequently, we have the following three patterns of broadcasting computational complexity of the GSC$(n,k,m)$ on the all-port model. When cube part dominates the network, the lower bound of the GSC$(n,k,m)$ approaches $\boldsymbol{\Omega}(m)$ and the broadcasting complexity of the GSC$(n,k,m)$ is $\boldsymbol{O}(m)$. Similarly, when $(n,k)$-star part dominates the network, the lower bound approaches $\boldsymbol{\Omega}(k)$ and the complexity is $\boldsymbol{O}(k)$. Otherwise they are $\boldsymbol{\Omega}(m+k)$ and $\boldsymbol{O}(m+k)$. Because this running time is proportional to the diameter of the GSC$(n,k,m)$, thus it is optimal; meanwhile, there is no message redundancy.

Figure 16 plots the total number of steps required for broadcasting at all-port model on hypercube, $(n,k)$-star graph, and generalized-star cube GSC$(n,k,m)$. We can see that almost the numbers of broadcasting steps of GSC$(n,k,m)$ are less than that of hypercubes at all-port model. And, at a given network size, we can choose different $n$, $k$, and $m$. The figure draws the curves for different $k$. In such a case, at a constant $k$, when $n$ of $(n,k)$-star graph is larger than $m$ of $m$-cube, the number of broadcasting steps decreases. This is because that the diameter of $(n,k)$-star graph is shorter

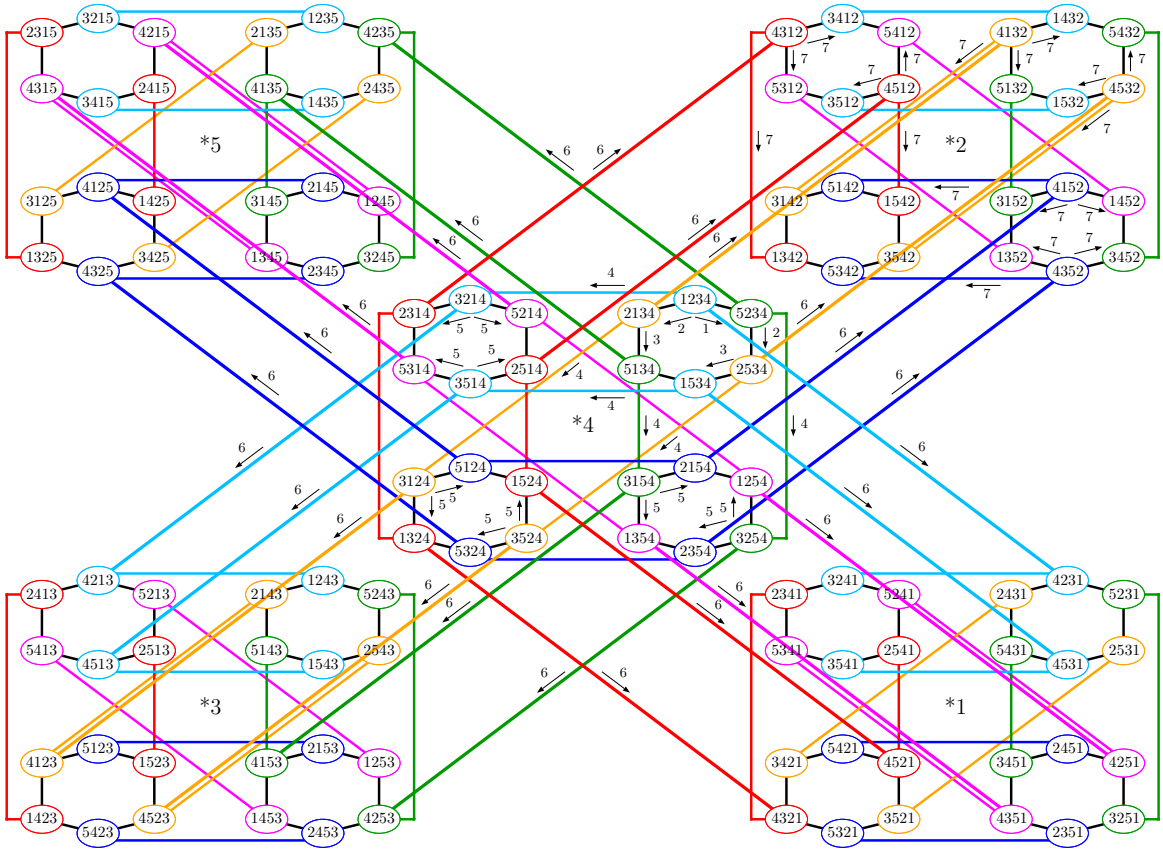



Figure 15: All-port broadcasting on a $(5, 4)$-star graph

than that of hypercubes, and the communication complexity of the all-port broadcasting depends strongly on the diameter. For example, when $k$ is 5, GSC$(6, 5, 7)$ has $92,160$ nodes; GSC$(8, 5, 4)$ has $107,520$ nodes, a closer number to $92,160$. GSC$(6, 5, 7)$ is composed of (6,5)-star graph and 7-cube and its total number of step is 16. In contrast, GSC$(8, 5, 4)$ is composed of (8,5)-star graph and 4-cube and its total number of step is 13.

# 7 Conclusion

In this paper, we proposed a new interconnection network, the generalized-star cube, and described its topological properties. The proposed generalized-star cube retains most of the properties of the hypercube and $(n, k)$-star graph. Then, we shew the comparison on the node degree and diameter. Compared to the hypercube, star graph, $(n, k)$-star graph, and star-cube, GSC can change the network size in smaller steps, therefore we can choose a more desirable network size. Also, the network cost was shown and we found out that GSC's cost lies in between hypercube's and $(n, k)$-star graph's costs, similar to the cases of the node degree and diameter. Next, we gave a shortest-path routing algorithm. We could realize simple shortest-path routing algorithm by separating the proposed topology into hypercube part and $(n, k)$-star graph part and joining existing optimal routing algorithms for them. Afterward, we discussed broadcast algorithms for both of the single-port and all-port models. These algorithms were separated into two parts just like the shortest-path routing algorithm. We utilized efficiently the spanning binomial tree communication scheme for both models of hypercube part. On the other hand, we exploited $(n, k)$-star graph's hierarchical structure for both models of another part. Moreover, in the single-port model, we adopted the neighborhood broadcast algorithm with the cycle structures. Meanwhile, for the all-port model, we

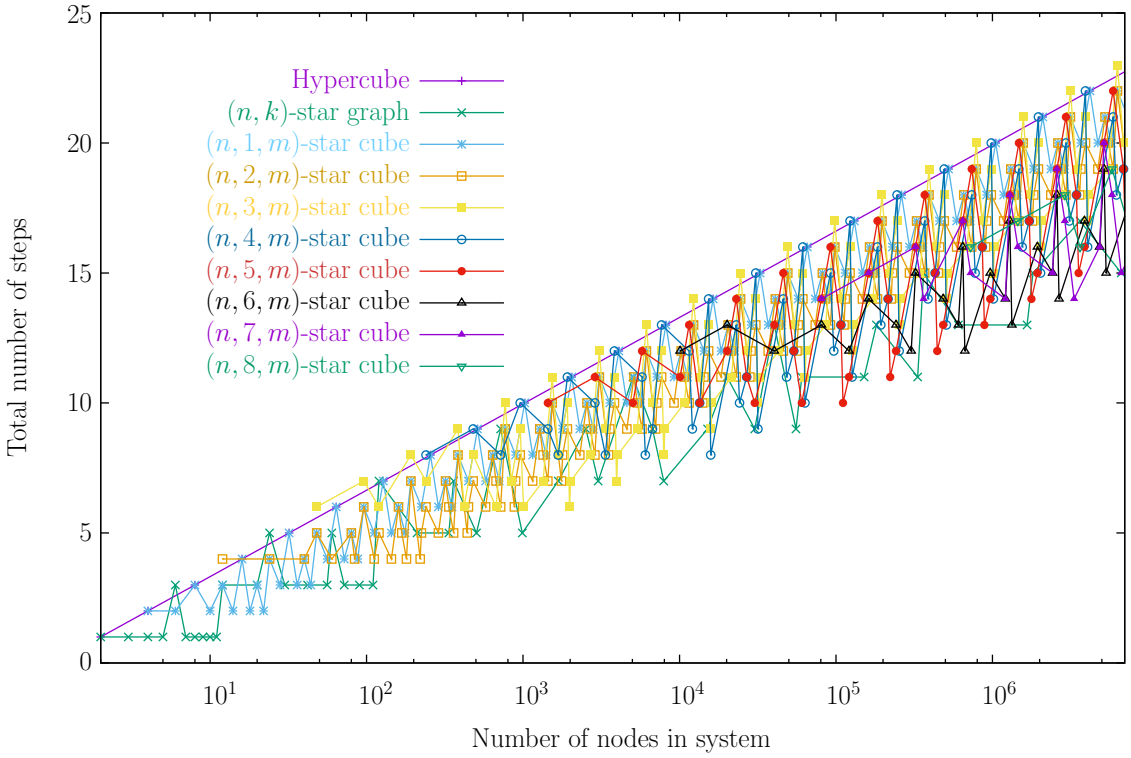



Figure 16: Total number of steps for broadcasting on all-port model

used the minimum dominating set. As a result, we derived optimal broadcasting algorithms for both two models.

In recent research, several product graphs have been proposed based on the star graph and cube-based derivatives [17, 1, 2, 19]. We can also derive new topologies by replacing the star graph of those product graphs with the $(n,k)$-star graph. Meanwhile, a lot of works concerning the generalized-star cube require further research. Some of them are: (1) to find disjoint-path in a generalized-star cube; (2) to develop fault-tolerant routing algorithms for the proposed network with faulty nodes; (3) to develop an efficient all-to-all broadcasting algorithm; and (4) to investigate the embedding of other frequently used topologies into this network.

## References

[1] N. Adhikari and C. R. Tripathy. Mstar: a new two level interconnection network. *Distributed Computing and Internet Technology Lecture Notes in Computer Science, LNCS 7154*, pages 50–61, 2012.

[2] N. Adhikari and C. R. Tripathy. Star-crossed cube: an alternative to star graph. *Turkish Journal of Electrical Engineering & Computer Sciences*, 22(3):719–734, 2014.

[3] S. B. Akers, D. Horel, and B. Krishnamurthy. The star graph: an attractive alternative to the n-cube. In *International Conference on Parallel Processing*, pages 393–400, 1987.

[4] S. B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, 1989.

[5] Y. S. Chen and K. S. Tai. A near-optimal broadcasting in (n,k)-star graphs. *ACIS International Conference on Software Engineering Applied to Networking and Parallel/Distributed Computing*, pages 217–224, 2000.

[6] W. K. Chiang and R. J. Chen. The (n,k)-star graph: a generalized star graph. *Information Processing Letters*, 56(5):259–264, 1995.

[7] W. K. Chiang and R. J. Chen. Topological properties of the (n,k)-star graph. *International Journal of Foundations of Computer Science*, 9(2):235–248, 1998.

[8] K. Day and A. Tripathi. Arrangement graphs: a class of generalized star graphs. *Information Processing Letters*, 42(5):235–241, 1992.

[9] S. Fujita. Optimal neighborhood broadcast in star graphs. *Journal of Interconnection Networks*, 4(4):419–428, 2003.

[10] S. Fujita. A new network topology for p2p overlay based on a contracted star graph. In *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pages 29–33, Dec. 2009.

[11] S. Fujita. P2p dht based on a contracted star graph. In *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 152–155, Oct. 2011.

[12] K. Ghose and K. R. Desai. Hierarchical cubic networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(4):427–435, 1995.

[13] L. He, K. Qiu, and Z. Z. Shen. Neighbourhood broadcasting and broadcasting on the (n,k)-star graph. *Algorithms and Architectures for Parallel Processing*, 5022:70–78, 2008.

[14] Liang He. Properties and algorithms of the (n,k)-star graphs. Master's thesis, Brock University, 2008.

[15] S. Latifi and N. Bagherzadeh. Incomplete star: an incrementally scalable network based on the star graph. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):97–102, 1994.

[16] J. L. Li, M. L. Chen, Y. H. Xiang, and S. W. Yao. Optimum broadcasting algorithms in (n, k)-star graphs using spanning trees. *IFIP International Conference on Network and Parallel Computing (NPC 2007)*, 4672:220–230, 2007.

[17] N. B. Nag, D. Pradhan, and N. K. Swain. Star varietal cube: a new large scale parallel interconnection network. *International Journal of Communication Network and Security*, 1(2):37–44, 2011.

[18] Online-material. Section #11: One-to-all broadcast algorithms. http://pages.cs.wisc.edu/ tvrdik/11/html/Section11.html.

[19] D. Pattanayak, D. Tripathy, and C. R. Tripathy. Star-mobius cube: a new interconnection topology for large scale parallel processing. *International Journal of Emerging Technologies in Computational and Applied Sciences*, 1(7):719–734, 2014.

[20] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, 1981.

[21] K. Qiu. On a unified neighbourhood broadcasting scheme for interconnection networks. *Parallel Processing Letters*, 17(4):425–437, 2007.

[22] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, 1998.

[23] Jang Ping Sheu, Chao Tsung Wu, and Tzung Shi Chen. An optimal broadcasting algorithm without message redundancy in star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 6(6):653–658, 1995.

[24] C. R. Tripathy. Star-cube: a new fault tolerant interconnection topology for massively parallel systems. *Journal of the Institution of Engineers, India, ETE Division*, 84(2):83–92, 2004.

[25] S. G. Ziavras. Rh: a versatile family of reduced hypercube interconnection networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(11):1210–1220, 1994.