electronics

MDPI

Article

Design and Analysis of a Symmetric Log Star Graph with a Smaller Network Cost Than Star Graphs

Jung-Hyun Seo [1] **and Hyeong-Ok Lee** [2,*]

[1] Department of Multimedia Engineering, National University of Chonnam, Yeosu 59626, Korea; jhseo@scnu.ac.kr
[2] Department of Computer Education, National University of Sunchon, Sunchon 315, Korea
* Correspondence: oklee@scnu.ac.kr; Tel.: +82-61-750-3345

Abstract: Graphs are used as models to solve problems in fields such as mathematics, computer science, physics, and chemistry. In particular, torus, hypercube, and star graphs are popular when modeling the connection structure of processors in parallel computing because they are symmetric and have a low network cost. Whereas a hypercube has a substantially smaller diameter than a torus, star graphs have been presented as an alternative to hypercubes because of their lower network cost. We propose a novel log star (LS) that is symmetric and has a lower network cost than a star graph. The LS is an undirected, recursive, and regular graph. In LS_n, the number of nodes is $n!$ while the degree is $2\log_2 n - 1$ and the diameter is $0.5n(\log_2 n)^2 + 0.75n\log_2 n$. In this study, we analyze the basic topological properties of LS. We prove that LS_n is a symmetrical connected graph and analyzed its subgraph characteristics. Then, we propose a routing algorithm and derive the diameter and network cost. Finally, the network costs of the LS and star graph-like networks are compared.

Keywords: star graphs; network cost; routing algorithm; log star; symmetric

Citation: Seo, J.-H.; Lee, H.-O. Design and Analysis of a Symmetric Log Star Graph with a Smaller Network Cost Than Star Graphs. *Electronics* **2021**, *10*, 981. https://doi.org/10.3390/electronics10080981

Academic Editor: Fabio Grandi

Received: 25 March 2021
Accepted: 19 April 2021
Published: 20 April 2021

1. Introduction

A graph consists of nodes and edges. The degree of a node u is the number of edges incident with u. The degree of a graph is the maximum degree over all nodes. The distance between two nodes u and v is the number of edges of the shortest path between u and v. The diameter of a graph is the maximum distance over all pairs of nodes. Graphs are used as models to solve problems in various fields. In computer science, graphs have been applied to the topology of parallel computers, the connection structure of very large-scale integrated (VLSI) internal cores, the connection structure of sink nodes and source nodes in wireless sensor networks (WSNs), and sorting problems. The measures for evaluating graphs vary depending on the field to suit the specific circumstances of each application. Smaller degrees and diameters are better for VLSI and parallel computers, which have a physical connection structure. In contrast, for sorting problems or WSNs with a logical connection structure, the diameter is a more important measure than the degree. In other fields, other evaluation measures may be more important [1]. In the interconnection network for parallel computing, since degree refers to the number of communication ports, the hardware installation cost increases when the degree increases. The diameter represents the length of the message transfer path between two processors, and when the diameter increases, the message transfer time increases, thus affecting the software cost. When the sorting algorithm is modeled as a graph, the diameter of the graph is an important measure because it is the worst time complexity of the sorting algorithm.

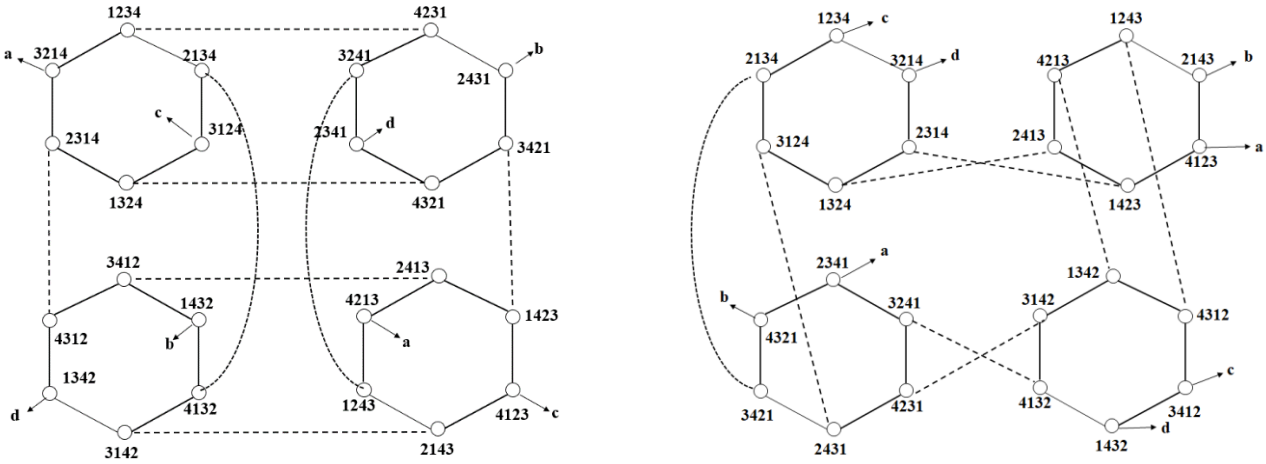
The evaluation measures of a graph include the number of nodes, number of edges, degree, diameter, network cost, bisection bandwidth, connectivity, symmetry, expandability, and fault tolerance. In addition, the answers to the following questions can be evaluated: Is it a regular graph? Is it a planar graph? Is it possible to embed it in another graph? Does it have a Hamiltonian path? Does it have a binary tree as a subgraph? Network cost

and symmetry are also used as evaluation measures [2–4]. In this study, we propose a symmetric graph that has a lower network cost than a star graph.

The network cost of a graph is its degree × diameter. Minimizing the network cost is highly similar to the (*d*,*k*) problem and the packing density problem. These problems involve finding a graph that has a small degree and diameter. Graphs with a low network cost are advantageous in various applications. When a graph is applied, the degree is the initial setup cost, and the diameter is the cost of running the algorithm. When an edge is added to a graph, the degree increases, whereas the diameter decreases. Conversely, if an edge is removed, the degree decreases and the diameter increases. Because there is a trade-off between the degree and diameter, finding a graph with a low network cost is difficult [5,6].

In a symmetric graph, the topological properties of the graph are the same viewed from any point (node). Suppose that for any two nodes u and v in graph G, there is a bjiection that maps u onto v. Consider that G is node-transitive. If for any two edges (u, v) and (x, y) in G there exists a bijection that maps (u, v) onto (x, y), then G is edge-transitive. If G is node and edge-transitive, it is a symmetric graph. When a symmetric graph is applied, it provides a significant advantage: an evaluation measure that is analyzed or an algorithm that is created at an arbitrary node can be used at all other nodes. Examples of algorithms include routing, broadcasting, Hamiltonian cycle (path), and binary graph construction algorithms [4,7,8].

The star graph in Figure 1a was introduced as an alternative to the well-known hypercube because of its excellent network cost. In star graph, each node $S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$ is connected to nodes $S_kS_2S_3 \ldots S_{k-1}S_1S_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$ ($2 \leq k \leq$ n). A hypercube Q_n has a diameter of n, degree of n, and 2^n nodes. The star graph S_n has a diameter of $1.5n - 2$, a degree of $n - 1$, and $n!$ nodes. The network cost of both graphs is $O(n^2)$. Thus, for a network cost on the same order, the number of nodes in the hypercube increases by a factor of two, whereas the number of nodes in the star graph increases by a factorial linear arrangement. The star graph is therefore superior to the hypercube in terms of the network cost per node. Star graph-like networks include stars [9], hyper stars [10], macro stars [11], matrix stars [12], bubble sort [13], and hierarchical stars [14].



(**a**) Star graph S_4 (**b**) log star LS_4

Figure 1. Four-dimensional star S_4 and log star LS_4.

In this study, we propose a symmetric log star graph that has a lower network cost than a star graph. In Section 2, the notation used throughout the paper is listed, and the log star is defined. In Section 3, we analyze the basic topological properties, symmetry, subgraphs, and connectivity of log stars. In Section 4, we present a routing algorithm and

analyze the diameter and network cost, which we compare with the network cost of star graph-like networks. Finally, in Section 5, we present our conclusions.

2. Notation and Definition of the Log Star

In this section, we provide the definitions used in this paper, and we define the log star LS_n. The node address of LS_n uses a permutation of the natural numbers from 1 to n. Let an arbitrary node be $S = S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$. In Definition 2, the symbols s_1 and S_k are exchanged with each other, and in Definition 3, multiple symbols are exchanged.

Definition 1. *S_k is the k-th symbol, s_1 is the most significant bit (MSB), and S_n is the least significant bit (LSB).*

Definition 2. *The exchange operation for S is $EO_k(S) = S_kS_2S_3 \ldots S_{k-1}S_1S_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$, $k = 2^0 + 1, 2^1 + 1, 2^2 + 1, \cdots, 2^{n/2} + 1$.*

Definition 3. *The swap operation for S is $SO_k(S) = S_{k+1} \ldots S_{2k-1}S_{2k}S_1S_2S_3 \ldots S_{k-1}S_ks_{2k+1} \ldots s_{n-2}s_{n-1}s_n$, $k = 2^0, 2^1, 2^2, \cdots, 2^{n/2}$.*

The definition of log star LS_n is as follows.

$LS_n = (V, E) \mid n = 2^m$, where m is a natural number.

$$node\ V = \{\ S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{n-2}S_{n-1}S_n \mid 1 \leq S_k \leq n, 1 \leq k \leq n\ \}.$$

The edges in LS_n are divided into exchange edges (EEs) and swap edges (SEs) and are defined as follows:

Exchange edge $EE_k = (S, EO_k(S))$.

Swap edge $SE_k = (S, SO_k(S))$.

For example, in LS_8, the (connected edge, neighboring node) pairs for node 12345678 are as follows:

Exchange edges: (EE_2, 21345678), (EE_3, 32145678), (EE_5, 52341678).

Swap edges: (SE_1, 21345678), (SE_2, 34125678), (SE_4, 56781234).

Figure 1b shows an LS_4 where the solid lines within the hexagons are EEs, and the dotted lines connecting one hexagon to another are the SEs. Furthermore, SE_1 and EE_2 are the same edges.

A path can be expressed as a sequence of nodes from a start node to a destination node. In this case, the neighboring nodes are connected by edges. A path can be defined as follows:

The notation 1. $\rightarrow$ is used to indicate a path. For example, the path between 1234, 2134, 3421, and 4321 in Figure 1b can be represented as follows:

$1234 \rightarrow 2134 \rightarrow 3421 \rightarrow 4321$ or

$1234 \rightarrow EE_1 \rightarrow 2134 \rightarrow SE_2 \rightarrow 3421 \rightarrow EE_1 \rightarrow 4321$.

3. Topological Properties of the Log Star

Here, we examine the basic topological properties of the LS.

Theorem 1. *LS_n is a regular graph with a degree of $2log_2n - 1$; its number of nodes is n!, and its number of edges is $n!log_2n - 0.5n$.*

Proof. Since the node address of LS_n is a permutation of the natural numbers from 1 to n, the total number of nodes is $n!$. At every node, the number of SEs is $\log_2 n$, and the number of EEs is $\log_2 n$. Since SE_1 and EE_2 are the same, the degree is $2\log_2 n - 1$. Because LS_n has the same degree at each node, it is a regular graph. Further, it is an undirected graph, where the total number of edges is $(n! \times (2\log_2 n - 1))/2 = n!\log_2 n - 0.5n!$. □

Two graphs with the same topology are called isomorphic, regardless of their shape. Consider the graphs G(V, E) and H(V, E) and the mapping functions f: $V(G) \rightarrow V(H)$ and g: $E(G) \rightarrow E(H)$. If bijections f and g exist, then G and H are isomorphic. Here, if f: $V(G) \rightarrow V(G)$ and g: $E(G) \rightarrow E(G)$, then graph G is automorphic and symmetric. This can be explained as follows: for any two nodes u and v of G, if there is a bijection that maps u to v, then G is a node-transitive graph. For any two edges (u, v) and (x, y) of G, if there is a bijection that maps (u, v) to (x, y), G is an edge-transitive graph. If G is both node and edge-transitive, it is a symmetric graph. It is well known that a permutation composed of natural numbers, as in a star graph, is a symmetry group [4] and that symmetry groups are automorphic. Since the nodes of LS_n are permutations from 1 to n, LS_n is a node-transitive graph. The proof of edge transitivity is provided in Theorem 2, and node transitivity is omitted.

Theorem 2. *LS_n is a symmetric graph.*

Proof. Let an arbitrary node be $S = S_1S_2S_3 \ldots S_k \ldots S_{n-1}S_n$. The mapping function is $f = S_1S_2S_3 \ldots S_i \ldots S_{n-1}S_n \rightarrow S_1S_2S_3 \ldots S_i \ldots S_nS_{n-1}$. The mapping function f is a bijection, which can be any function that exchanges symbols. An edge that is connected to S is denoted by (S, T). It is shown that $f(S)$ and $f(T)$ are connected to each other.

First, consider the mapping of an EE:

If $S = S_1S_2S_3 \ldots S_k \ldots S_{n-1}S_n$, then $T = S_KS_2S_3 \ldots S_1 \ldots S_{n-1}S_n$. $f(S) = S_1S_2S_3 \ldots S_k \ldots S_nS_{n-1}$, and $f(T) = S_kS_2S_3 \ldots S_1 \ldots S_nS_{n-1}$. Thus, $f(S)$ and $f(T)$ are exactly connected with the EE_k.

Next, consider the mapping of an SE:

If $S = S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$, then $T = S_{k+1} \ldots S_{2k-1}S_{2k}S_1S_2S_3 \ldots S_{k-1}S_kS_{2k+1} \ldots S_{n-2}S_{n-1}S_n$. $f(S) = S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_nS_{n-1}$, and $f(T) = S_{k+1} \ldots S_{2k-1}S_{2k}S_1S_2S_3 \ldots S_{k-1}S_kS_{2k+1} \ldots S_{n-2}S_nS_{n-1}$. $f(S)$ and $f(T)$ are thus exactly connected to SE_k. Therefore, LS_n is a node and edge-transitive graph and is symmetric. □

If all the nodes of a graph are connected, it is a connected graph; that is, in a connected graph, there is a path from an arbitrary node to every other node in the graph. The node addresses are expressed as permutations. If the MSB can move to any other position, the opposite is also possible. Thus, it has been shown that LS_n is a connected graph.

Theorem 3. *LS_n is a connected graph.*

Proof. We examined LS_4 and LS_8 and generalized LS_n above. Definitions 2 and 3 are used here. In LS_4, $S = S_1S_2S_3S_4$. The movement of the MSB by SO is as follows:

$$S_1S_2S_3S_4 = S.$$
$$S_2S_1S_3S_4 = SO_1(S).$$
$$S_3S_4S_1S_2 = SO_2(S).$$
$$S_3S_4S_2S_1 = SO_2(SO_1(S)).$$

Therefore, LS_4 is a connected graph. In LS_8, $S = s_1s_2s_3s_4s_5s_6s_7s_8$. For LS_4, it has been shown that the MSB can move to positions 2, 3, and 4. If SO_4 is added, the MSB can move to positions 5, 6, 7, and 8. The movement of the MSB by SO is as follows:

$$S_1S_2S_3S_4 = S.$$
$$S_2S_1S_3S_4 = SO_1(S).$$
$$S_3S_4S_1S_2 = SO_2(S).$$
$$S_3S_4S_2S_1 = SO_2(SO_1(S)).$$
$$S_5S_6S_7S_8\ S_1S_2S_3S_4 = SO_4(S).$$
$$S_5S_6S_7S_8S_2\ S_1S_3S_4 = SO_4(SO_1(S)).$$
$$S_5S_6S_7S_8S_3S_4\ S_1S_2 = SO_4(SO_2(S)).$$
$$S_5S_6S_7S_8\ S_3S_4S_2\ S_1 = SO_4(SO_2(SO_1(S))).$$

Therefore, LS_8 is also a connected graph. In LS_n, the MSB can move to positions 2, 3, 4, $\cdots$, $n/2$. If $SO_{n/2}$ is added, the MSB can move to positions $n/2 + 1$, $n/2 + 2$, $\cdots$, n. Therefore, LS_n is a connected graph. □

Let G = (V, E) and H = (V, E). If $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, H is called a subgraph of G. Consider subgraphs H1 and H2 of G. When $V(H1) \cap V(H2) = \emptyset$ and $E(H1) \cap E(H2) = \emptyset$, the two subgraphs are said to be both node and edge-disjoint. Disjoint subgraphs can be used for process allocation, load balancing, broadcasting algorithms, and disjoint paths. The disjoint subgraphs of LS_n are presented below.

Theorem 4. *LS_n consists of n!/(n/2)! subgraphs $LS_{n/2}$. The number of nodes in the subgraphs is (n/2)!, and they are node- and edge-disjoint with each other.*

Proof. The number of nodes in LS_4 is 4!. LS_4 comprises 12 (=4!/2!) LS_2. The number of nodes in LS_2 is 2!. Figure 2 shows the subgraphs $xx12$, $xx13$, $xx14$, $xx23$, $xx24$, $xx21$, $xx34$, $xx31$, $xx32$, $xx41$, $xx42$, and $xx43$. x refers to an arbitrary number.

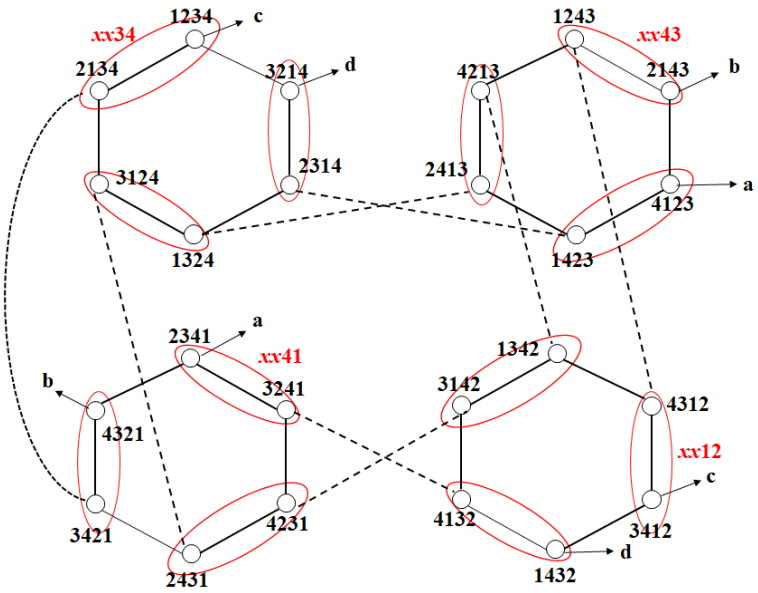


Figure 2. Subgraphs of LS_4.

The number of nodes in LS_8 is 8!. LS_8 consists of 8!/4! LS_4, and the number of nodes in LS_4 is 4!. The number of nodes in LS_n is n!, and LS_n consists of $n!/(n/2)!$ $LS_{n/2}$. The number of nodes in $LS_{n/2}$ is $(n/2)!$. □

Theorem 5. *LS_n comprises $2n!/n^2$ subgraphs. The subgraphs are both node and edge-disjoint, and the number of nodes is $0.5n^2$. All the subgraphs have Hamiltonian cycles and paths.*

Proof. First, LS_4 and LS_8 are examined, and then the result is generalized to LS_n.

We first examine the subgraphs in LS_4:

To define a subgraph, the natural numbers 1 to 4 used in the node addresses of LS_4 are divided into two bundles, regardless of order. For example, they can be divided into bundles 12 and 34. There is no order inside a bundle. The node sets composed of these bundles are thus 1234, 2134, 1243, 2143, 3412, 3421, 4312, and 4321, which are shown as subgraphs (Figure 3). Note that the natural numbers could alternatively be divided into bundles 13 and 24 or 14 and 23. Consider the orange nodes in Figure 3, where the bundles 12 and 34 are not mixed; in the three subgraphs in Figure 3, the representative nodes are 1234, 1324, and 1423. We examine the path of repeating SE_1 and SE_2 four times at these nodes. This path is the Hamiltonian cycle of the subgraph, where the three cycles are represented by green, red, and orange. The number of nodes in a subgraph is 8. Because the numbers of the two bundles do not mix with each other in the SE operation, the three subgraphs are all node and edge-disjoint with each other. If the last SE_2 is omitted, the path is a Hamiltonian path, which is used on LS_8. If the start node of this path is $s_1s_2s_3s_4$, then the destination node is $s_3s_4s_1s_2$. The Hamiltonian sequence is $HS_4 = SE_1 \rightarrow SE_2 \rightarrow SE_1 \rightarrow SE_2 \rightarrow SE_1 \rightarrow SE_2 \rightarrow SE_1$.

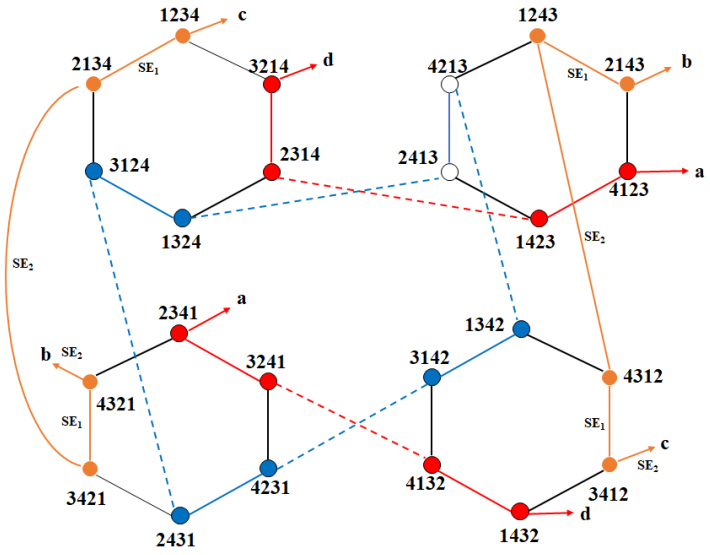


Figure 3. Subgraphs with a Hamiltonian cycle in LS_4.

We now consider the subgraphs in LS_8.

As for LS_4, we examine the division of the natural numbers 1 to 8 into two bundles, where the within-bundle order is irrelevant. For example, suppose the numbers are divided into bundles 1234 and 5678; in this case, n is the number of subgraphs: $(n \times (n/2)) = (1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8) = 1260$. We explain this by taking one subgraph as an example. If HS_4 and SE_4 are repeated four times at node 12345678, a cycle with a path length of 32 ($=n \times (n/2)$) is created. The number of nodes in the subgraph is 32, and $HS_8 = HS_4 \rightarrow SE_4 \rightarrow HS_4 \rightarrow SE_4 \rightarrow HS_4 \rightarrow SE_4 \rightarrow HS_4$:

$$\begin{gathered} 12345678 \rightarrow \\ HS_4 \rightarrow 34125678 \rightarrow SE_4 \rightarrow 56783412 \rightarrow \\ HS_4 \rightarrow 78563412 \rightarrow SE_4 \rightarrow 34127856 \rightarrow \\ HS_4 \rightarrow 12347856 \rightarrow SE_4 \rightarrow 78561234 \rightarrow \\ HS_4 \rightarrow 56781234 \rightarrow SE_4 \rightarrow 12345678. \end{gathered}$$

We now generalize to the subgraphs in LS_n.

For LS_{n-1}, we examine the method of dividing the natural numbers 1–n into two bundles, where the within-bundle order does not matter. In this case, there are $n!(n \times (n/2))$ subgraphs. If an operation is performed by repeating HS_{n-1} and $SE_{n/2}$ four times at node $s = 1234 \cdots n$, a cycle with $n \times (n/2)$ nodes is created. The temporary node used in the operation is denoted as t: $HS_n = HS_{n/2} \rightarrow SE_{n/2} \rightarrow HS_{n/2} \rightarrow SE_{n/2} \rightarrow HS_{n/2} \rightarrow SE_{n/2} \rightarrow HS_{n/2}$.

$$\begin{gathered} \mathrm{s} \rightarrow \\ \mathrm{HS}_{n/2} \rightarrow \mathrm{t} \rightarrow \mathrm{SE}_{n/2} \rightarrow \mathrm{t} \rightarrow \\ \mathrm{HS}_{n/2} \rightarrow \mathrm{t} \rightarrow \mathrm{SE}_{n/2} \rightarrow \mathrm{t} \rightarrow \\ \mathrm{HS}_{n/2} \rightarrow \mathrm{t} \rightarrow \mathrm{SE}_{n/2} \rightarrow \mathrm{t} \rightarrow \\ \mathrm{HS}_{n/2} \rightarrow \mathrm{t} \rightarrow \mathrm{SE}_{n/2} \rightarrow \mathrm{s}. \end{gathered}$$

In the subgraph, the number of nodes and the Hamiltonian path length are the same, as shown below. According to the rule described above for creating subgraphs, the path length of the LS_n subgraph is equal to the path length of LS_{n-1} multiplied by 4. Recall that the path length of LS_4 is 8. The path length can thus be summarized as follows:

$$\begin{gathered} \mathrm{LS}_4 = 8\ (= 4 \times 2). \\ \mathrm{LS}_8 = 32\ (= 8 \times 4). \\ \mathrm{LS}_{16} = 128\ (= 16 \times 8). \\ \mathrm{LS}_{32} = 128\ (= 32 \times 16). \\ \vdots \\ \mathrm{LS}_n = n \times (n/2). \end{gathered}$$

Therefore, LS_n has $n!/(n \times (n/2))$ subgraphs that are node and edge-disjoint with each other and that have a path length of $n \times (n/2)$. □

Lemma 1. *LS_4 has a Hamiltonian cycle (path).*

Proof. LS_4 has a Hamiltonian cycle with a start node of 1234 (Figure 4). Because the log star is a symmetric graph, there is a Hamiltonian cycle at every node. The path for creating the Hamiltonian cycle is as follows, where () × 2 means that the operation in () is repeated twice:

$$\text{Hamiltonian cycle } \mathrm{HC}_4 = ((\mathrm{EE}_3 \rightarrow \mathrm{EE}_1) \times 2 \rightarrow \mathrm{EE}_3 \rightarrow \mathrm{SE}_2) \times 4.$$

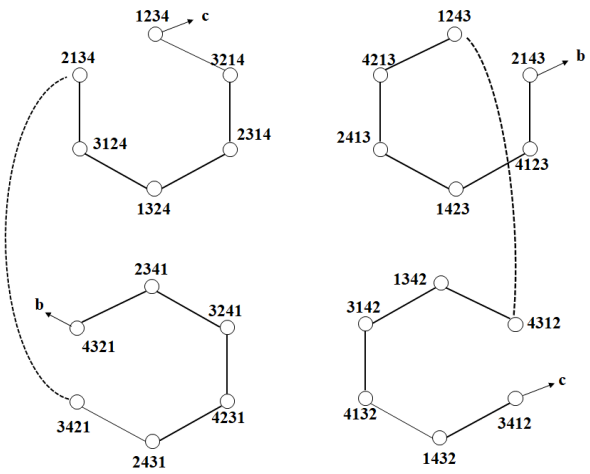


Figure 4. Hamiltonian cycle in LS_4.

If the last operation, SE_2, is removed from HC_4, it becomes the Hamiltonian path. There is a Hamiltonian path with start node s = $s_1s_2s_3s_4$ and destination node $SE_2(s) = s_3s_4s_1s_2$, which is expressed as follows:

Hamiltonian path $HP_4 = ((EE_3 \rightarrow EE_1) \times 2 \rightarrow EE_3 \rightarrow SE_2) \times 3 \rightarrow (EE_3 \rightarrow EE_1) \times 2 \rightarrow EE_3$.

□

4. Routing Algorithm and Comparison of Network Cost

Here, we present a routing algorithm and determine the network cost by analyzing the diameter compared with that of star graph-like networks.

Routing refers to the sending and receiving of messages between two nodes. A routing path is the path through which a message is transmitted and can be represented by a sequence of nodes with edges between the successive nodes in the sequence. Following the routing path is equivalent to changing the address of the start node to that of the destination node by using an edge operator. To describe a routing path, several definitions are first established. This stage is called the "setup," where the symbols to be exchanged are defined. In Figure 5, 12 and 56 are symbols that need to be exchanged. Let the start node be $S = S_1S_2S_3 \ldots S_{k-1}S_kS_{k+1} \ldots S_{2k-1}S_{2k}S_{2k+1} \ldots S_{n-2}S_{n-1}S_n$, and the destination node is $T = t_1t_2t_3 \ldots t_{k-1}t_kt_{k+1} \ldots t_{2k-1}t_{2k}t_{2k+1} \ldots t_{n-2}t_{n-1}t_n$.

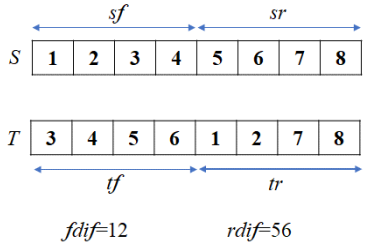


Figure 5. Setup in LS_8.

< setup >

Definition 4. *In the address of node S, there is a set sf = {$S_1S_2S_3 \ldots S_{n/2}$}, and sr = {$S_{n/2+1}, \ldots S_n$}. In the case of node T, tf = {$t_1t_2t_3 \ldots t_{n/2}$}, and tr = {$t_{n/2+1}, \ldots t_n$}.*

Definition 5. *Set fdif = {sf} − {tf}, and set rdif = {sr} − {tr}. In fdif and rdif, there is a sequential order, and the m^{th} rdif element is denoted by rdif[m].*

In Figure 5, for example, suppose $S = 12345678$ and $T = 34561278$. Then $sf = \{1, 2, 3, 4\}$, $tf = \{3, 4, 5, 6\}$, $sr = \{5, 6, 7, 8\}$, and $tr = \{1, 2, 7, 8\}$. In this case, $fdif = \{1, 2\}$, $rdif = \{5, 6\}$, $|fdif| = |rdif| = 2$, $rdif[1] = 5$, and $rdif[2] = 6$.

Routing is performed recursively using a typical separate-and-conquer method. Because it has already been conquered in the separation process, there is no additional conquer work to be performed when the separation is completed. The separation() of Algorithm 2 divides the symbols into two regions: separation() exchanges *fdif* in *sf* with *rdif* in *sr*. Once separation() is executed, there is no additional symbol movement between *sf* and *sr*. Separation() is then executed again on *sf* and *sr*. If $|sf| = |sr| = 1$, the separation() is terminated, and the routing is complete.

We provide an example to clarify the routing process. This explanation should be read with reference to Figure 6 and the routing algorithm. In LS_8, let $S = 32861457$ and $T = 12345678$. $fdif = \{sf\} - \{tf\} = \{8, 6\}$, and $rdif = \{sr\} - \{tr\} = \{1, 4\}$. $|fdif| = |rdif| = 2$. Because $|fdif|$ is not greater than 2 (= $n/4$), swap ① in the routing algorithm is skipped. In separation(), symbols 8 and 6 in *sf* and symbols 1 and 4 in *sr* are exchanged with each other. Figure 6 shows the stages in which the separation() is executed. In stage 1, $sf = \{4, 1, 3, 2\}$,

and *sr* = {6, 8, 5, 7}. In stage 2, the same operation is repeated at *sf* and *sr*. Thus, *S* = *sf* = {4, 1, 3, 2} and *T* = *tf* = {1, 2, 3, 4}, and 4 in *sf* and 2 in *sr* are exchanged. As a result, *sf* = {2, 1} and *sr* = {4, 3}. In the next stage, the results are *sf* = {1, 2} and *sr* = {3, 4}. The operations are the same in the remaining part. If the skipped part is included, the entire routing path is as follows:

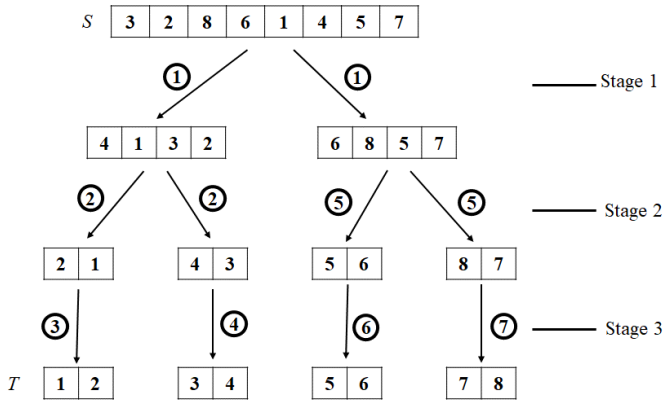


Figure 6. Separation() call steps and sequence in LS_8.

The first execution of separation():

$$S = 32861457 \rightarrow SE_2 \rightarrow 86321457 \rightarrow SE_4 \rightarrow 14578632 \rightarrow EE_5 \rightarrow 84571632 \rightarrow SE_1 \rightarrow 48571632 \rightarrow SE_4 \rightarrow 16324857 \rightarrow SE_1 \rightarrow 61324857 \rightarrow EE_5 \rightarrow 41326857.$$

The second execution of separation():

$$41326857 \rightarrow SE_2 \rightarrow 32416857 \rightarrow SE_1 \rightarrow 23416857 \rightarrow EE_3 \rightarrow 43216857 \rightarrow SE_2 \rightarrow 21436857.$$

The third execution of separation():

$$21436857 \rightarrow SE_1 \rightarrow 12436857.$$

The fourth execution of separation():

$$12436857 \rightarrow SE_2 \rightarrow 43126857 \rightarrow SE_1 \rightarrow 34126857 \rightarrow SE_2 \rightarrow 12346857.$$

The operations on *sr* are performed in the same way as those on *sf*, whereby 6857 becomes 5678. The processing sequence is illustrated in Figure 6.

The routing algorithm is shown in Algorithm 1, where = = is an equality operator and = is an assignment operation:

Algorithm 1. routing(*S*, *T*) {

```
1:  if (|S| = =1) return end-if
2:  <set up >
3:  if (|fdif| > n/4)
4:     S = S → SE_{n/2}                    ———— swap ①
5:     <set up >
6:  end-if
7:  separation(S)
8:  routing(sf, tf)
9:  S = S → SE_{n/2}
10: routing(sf, tr)
11: S = S → SE_{n/2}
12: }
```

Algorithm 2. separation(S) {

```
1: for (m = 1; m ≤ |fdif|; m + +) {
2:     movemsb(sf, fdif[m])
3:     SE n/2
4:     movemsb(sf, rfdif[m])
5:     SE n/2
6: }
7: if (|fdif| is odd) SE n/2 end-if
8: }
```

The separation() exchanges symbols. The exchange method is as follows:

- The first symbol to be exchanged is moved to the MSB through the movemsb() function of Algorithm 3;
- A half is swapped through $SE_{n/2}$;
- The second symbol to be exchanged is moved to the MSB through movemsb();
- MSB and $S_{(n/2)+1}$ are exchanged with each other through $EE_{n/2}$.

Algorithm 3. movemsb(S, s_m) {

```
1:    if (|S| == 1) return end-if
2:    h = the index of s_m in S
3:    if |S|/2 < h SE n/2 end-if
4:    movemsb(sf, s_m)
5: }
```

The diameter is an important measure for evaluating a graph. It is the farthest distance between two nodes in the graph when taking the shortest path. In the worst case, it is the routing path length, which represents the software cost when the graph is implemented as a system.

Theorem 6. *The diameter of* LS_n *is* $0.5n(log_2 n)^2 + 0.75nlog_2 n$.

Proof. The diameter analysis can be replaced by an analysis of the worst-case time complexity of the routing algorithm. In Figure 6, because the size of the problem decreases by half after each iteration and the process ends when the size reaches 1, the number of stages is $a = \log_2 n$. In the worst case, the number of symbols to be exchanged in a single stage is $b = n/2$. The operations required to exchange two symbols are movemsb(), $SE_{n/2}$, movemsb(), and $EE_{n/2}$. When the number of symbols to be exchanged is odd, the EE edge is only required once. In the worst case, movemsb() is required to be $\log_2 n$ for each symbol. The number of operations required to exchange the two symbols is therefore $c = 2\log_2 n + 2+1$. Then, the following would be true:

$$\text{Diameter } k = \text{number of stages } a \times \text{internal time complexity of the stage,}$$

and

$$\text{Internal time complexity of a stage} = (\text{number of symbols to be exchanged b} \times \text{number of operations required to exchange two symbols c})/2.$$

Thus,

$$\begin{aligned} \text{diameter } k &= a \times (bc)/2 \\ &= \log_2 n \times ((n/2 \times (2\log_2 n + 3))/2) \\ &= \log_2 n \times n/4(2\log_2 n + 3) \\ &= \log_2 n \times (0.5n\log_2 n + 0.75n) = 0.5n(\log_2 n)^2 + 0.75n\log_2 n. \end{aligned}$$

□

Network cost is an important measure for the evaluation of a network. Table 1 lists the network cost of LS_n versus various star graph-like networks. To calculate the network cost of other networks, we investigated the diameter and degree of stars [9], hyper stars [10], macro stars [11], matrix stars [12], bubble sort [13], and hierarchical stars [14]. The network cost is shown in ***O*** notation.

Table 1. Network cost comparison of LS_n with other star graph-like networks.

Network	#Node Number	Degree	Diameter	Network Cost
Star graph s(n)	$n!$	$n-1$	$1.5n-2$	$O(1.5n^2)$
Hyper star HS(n,$n/2$)	$n!/(n/2!)^2$	$1.5n$	$n-1$	$O(n^2)$
Macro star MS(2,n)	$(n+1)!$	$1.5n+1$	$2.5n+1$	$O(3.75n^2)$
Matrix star MTS(2,n)	$n!$	$1.5n+1$	$1.75n+2$	$O(2.6n^2)$
Bubble sort BS(2n)	$n!$	$2n-1$	$1.5n-1$	$O(3n^2)$
Hierarchical star HS(n,n)	$(n!)^2$	n	$3n-2$	$O(3n^2)$
Log star LS(n)	$n!$	$2\log_2 n-1$	$0.5n(\log_2 n)^2+0.75n\log_2 n$	$O(n(\log_2 n)^3)$

Since $O(n(\log_2 n)^3) < O(n^2)$, LS is superior to other star graph-like networks in terms of its network cost (Table 1).

5. Discussion

Countless graphs have been published in various fields. Graphs are designed based on the characteristics of the problem to be solved, which also determine the most suitable evaluation measures for a graph. General-purpose evaluation measures that can be used in numerous fields include symmetry and network cost. In this study, we investigated the network cost of the representative star graph-like networks that were published to date and proposed a symmetric log star graph with a low network cost. For the log star LS_n, the number of nodes is n, the degree is $2\log_2 n-1$, and the diameter is $0.5n(\log_2 n)^2+0.75n\log_2 n$. The network cost of LS_n is $O(n(\log_2 n)^3)$, which is smaller than $O(n^2)$—the network cost of star graph-like networks. We analyzed the topological properties, connectedness, symmetry, and subgraph properties of the LS and proposed a routing algorithm. It is expected that in the future, various algorithms, such as broadcasting, processor allocation, parallel path, Hamiltonian path, spanning tree, and embedding, will be studied with respect to this graph.

Author Contributions: Conceptualization, J.-H.S. and H.-O.L.; methodology, J.-H.S.; software, J.-H.S.; validation, J.-H.S.; formal analysis, J.-H.S.; investigation, J.-H.S.; resources, J.-H.S.; data curation, J.-H.S.; writing—original draft preparation, J.-H.S.; writing—review and editing, J.-H.S.; visualization, J.-H.S.; supervision, H.-O.L.; project administration, H.-O.L.; funding acquisition, H.-O.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1A2C1012363).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Seo, J.; Kim, J.; Chang, H.J.; Lee, H. The hierarchical Petersen network: A new interconnection network with fixed degree. *J. Supercomput.* **2018**, *74*, 1636–1654. [CrossRef]
2. Gross, J.L.; Yellen, J. *Graph Theory and Its Applications*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2005; pp. 417–468.
3. Xu, Z.; Huang, X.; Jimenez, F.; Deng, Y. A new record of graph enumeration enabled by parallel processing. *Math* **2019**, *7*, 1214. [CrossRef]
4. Ganesan, A. Cayley graphs and symmetric interconnection networks. *arXiv* **2017**, arXiv:1703.08109.
5. Bermond, J.-C.; Delorme, C.; Quisquater, J.-J. Strategies for interconnection networks: Some methods from graph theory. *J. Parallel Distrib. Comput.* **1986**, *3*, 433–449. [CrossRef]
6. Akers, S.B.; Krishnamurthy, B. A group-theoretic model for symmertric interconnection network. *IEEE Trans. Comput.* **1989**, *38*, 555–565. [CrossRef]

7. Lakshmivarahan, S.; Jwo, J.-S.; Dhall, S.K. Symmetry in interconnection networks based on Cayley graphs of permutation groups: A survey. *Parallel Comput.* **1993**, *19*, 361–407. [CrossRef]
8. Li, J.J.; Ling, B. Symmetric graphs and interconnection networks. *Future Gener. Comput. Syst.* **2018**, *83*, 461–467. [CrossRef]
9. Akers, S.B.; Harel, D.; Krishnamurthy, B. The Star Graph: An Attractive Alternative to the N-Cube. In Proceedings of the International Conference on Parallel Processing, Penn State University, University Park, PA, USA, 17–21 August 1987; The Pennsylvania State University Press: University Park, PA, USA, 1987; pp. 393–400.
10. Bell, M.G.H. Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation. *Transp. Res. Part B Methodol.* **2009**, *43*, 97–107. [CrossRef]
11. Yeh, C.H.; Varvarigos, E. Macro-Star Networks: Efficient Low-Degree Alternatives to Star Graphs for Large-Scale Parallel Architectures. In Proceedings of the Frontiers96 the Sixth Symposium on the Frontiers of Massively Parallel Computation, Annapolis, MD, USA, 27–31 October 1996; IEEE Computer Society Press: Los Alamitos, CA, USA, 1996.
12. Lee, H.O.; Kim, J.S.; Park, K.W.; Seo, J.H. Matrix Star Graphs: A New Interconnection Network Based on Matrix Operations. In Proceedings of the ACSAC: Asia-Pacific Conference on Advances in Computer Systems Architecture, Singapore, 24–26 October 2005; Srikanthan, T., Xue, J., Chang, C.H., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3740, pp. 478–487.
13. Wang, S.; Yang, Y. Fault tolerance in bubble-sort graph networks. *Theor. Comput. Sci.* **2012**, *421*, 62–69. [CrossRef]
14. Song, Z.; Ma, G.; Song, D. Hierarchical Star: An optimal NoC Topology for High-performance SoC Design. In Proceedings of the 2008 International Multi-symposiums on Computer and Computational Sciences, Shanghai, China, 18–20 October 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 158–163.