HAL open science

Connectivity and Routing Algorithm of the Data Center Network HSDC

Hui Dong, Jianxi Fan, Baolei Cheng, Yan Wang, Jingya Zhou

To cite this version:

Hui Dong, Jianxi Fan, Baolei Cheng, Yan Wang, Jingya Zhou. Connectivity and Routing Algorithm of the Data Center Network HSDC. 17th IFIP International Conference on Network and Parallel Computing (NPC), Sep 2020, Zhengzhou, China. pp.407-419, 10.1007/978-3-030-79478-1_35 . hal-03768752

HAL Id: hal-03768752

https://inria.hal.science/hal-03768752v1

Submitted on 4 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Connectivity and Routing Algorithm of the Data Center Network HSDC

Hui Dong, Jianxi Fan*, Baolei Cheng, Yan Wang, and Jingya Zhou

School of Computer Science and Technology, Soochow University,
Suzhou 215006, China
jxfan@suda.edu.cn

Abstract. In order to satisfy the rapidly increasing demand for data volume, large data center networks (DCNs) have been proposed. In 2019, Zhang et al. proposed a new highly scalable DCN architecture named HSDC, which can achieve greater incremental scalability. In this paper, we give the definition of the logical graph of HSDC, named H_n, which can be treated as a compound graph of hypercube and complete graph. First, we prove that the connectivity and tightly super connectivity of H_n are both n. Then, we give an $O(n)$ routing algorithm to find a shortest path between any two distinct nodes in H_n, and prove the correctness of this algorithm. In fact, we also prove that the distance constructed by this algorithm is no more than $2d + 1$ if $d < n$ and at most $2d$ if $d = n$, where d is the Hamming distance between any two distinct nodes, and the diameter of H_n is $2n$.

Keywords: Data center network, connectivity, routing algorithm, shortest path

1 Introduction

With the rapid expansion of cloud-based services, large data center networks (DCNs) have been proposed. It is necessary to design a network architecture and related protocols to interconnect thousands or even hundreds of thousands of servers in a single data center. In order to satisfy the rapidly increasing demand for data volume, the performance of DCN has been continuously improved. In many DCN architectures, there are many DCNs inspired by some special interconnected networks. For example, Fat-Tree [1] is based on the Fat-trees interconnection network [2], BCube [3] is based on the generalized hypercube [4], and BCDC [5] is based on the crossed cube. Among them, hypercube is widely used in parallel computers due to its super features. In 2019, Zhang et al. [6] proposed a new DCN architecture named HSDC based on the hypercube, whose highly scalable architecture can achieve greater incremental scalability. The HSDC architecture is constructed by employing 2-port servers and low-cost commodity m-port switches. Zhang et al. compared the static characteristics of

* Corresponding author.

HSDC with other DCNs and the analysis results demonstrate that HSDC is a superior candidate for building large-scale data centers.

In this paper, we focus the logical graph of HSDC. Our main contributions as follows.

1. We give the definition of the logical graph of HSDC, named H_n. Then, we show that the connectivity and tightly super connectivity of H_n are both n.

2. We propose an $O(n)$ node-to-node shortest routing algorithm in H_n, and prove the correctness of this algorithm. We also prove that the length of the shortest path constructed by this algorithm is no more than $2d+1$ if $d < n$ and at most $2d$ if $d = n$, where d is the Hamming distance between any two distinct nodes, and the diameter of H_n is $2n$.

This paper is organized as follows. Section 2 provides the preliminaries used throughout this paper and gives the formal definition of H_n. The proofs about connectivity and tightly super connectivity of H_n are given in Section 3. In Section 4, a node-to-node shortest routing algorithm for H_n is described. Section 5 concludes the paper.

2 Preliminaries

2.1 Terminology and Notation

The basic topology of a data center network can be represented by a graph $G = (V(G), E(G))$, where the switches are regarded as transparent network devices [7, 8] and the remaining processor set is represented by the node set $V(G)$, the links between processors are represented by the edge set $E(G)$. We follow the symbol and definition of the graph proposed by Hsu and Lin. [9]. For any two nodes u and v, if $(u, v) \in E(G)$, then u and v are neighbors, and the neighbor set of u is denoted as $N_G(u) = \{v \mid (u, v) \in E(G)\}$. The degree of node u is expressed by $deg_G(u) = |N_G(u)|$. The minimum degree of node in G is expressed by $\delta(G)$. A path in G is a sequence of edges $P = (a_0, a_1)(a_1, a_2) \cdots (a_{k-1}, a_k)$ where $a_i \in V(G), 0 \leq i \leq k$, $a_i \neq a_j$ and $i \neq j$. The length of a path P is the number of edges in P. For any two distinct nodes a and b, we write the path from a to b by $a \sim b$ and use $a_i \to a_j$ where $i \neq j$ to denote an edge (a_i, a_j) in the path. The distance between a and b is written as $dist(a, b)$, which is the minimum value of all path lengths between a and b. The diameter of G is the maximum distance between any two nodes in G, denoted as $max\{dist(a, b) \mid a, b \in V(G) \ and \ a \neq b\}$. Let G and G' be two graphs. If G is isomorphic to G', we will write $G \cong G'$.

Let F be a subgraph of G, denoted as $F \subseteq G$, if $V(F) \subseteq V(G)$ and $E(F) \subseteq E(G)$. The clique in G refers to a set of nodes such that there exists an edge between any two nodes. It can be seen that the induced subgraph of a clique is a complete subgraph of G. For any subset $F \subset G$, $G - F$ denotes deleting all nodes in F and removing the edges with at least one end-node in F. For any non-empty subset $F \subset G$, if $G - F$ is disconnected, then we called F is a *separating set* of G. The maximal connected subsets of $G - F$ are called *components*. Reliability has always been a concern in interconnected networks.

Connectivity can assess the reliability of an interconnected network. In the following, we make no distinction between interconnected networks and networks. The connectivity of G is denoted by $\kappa(G)$, which is defined as the minimum cardinality of a set of nodes, if any, whose deletion disconnects G or make G be a trivial graph. In the case of node failure, connectivity plays a crucial role in measuring the fault tolerance of network. If $\kappa(G) \geq k$, the graph G is said to be k-connected. A k-regular k-connected graph is super k-connected if any one of its minimum separating set is a set of the neighbors of some node. In addition, if deleting the minimum separation set will cause the graph to contain two components (one of which has only one node), the graph is tightly super k-connected.

2.2 Topological Structure of H_n

By using n-port switches, the $HSDC_n(n)$ architecture is constructed based on n-dimension hypercube. The $HSDC_n(n)$ can be defined as follows.

Definition 1. *[6] In $HSDC_n(n)$, the nodes and edges are defined as follows:*

(1) The switches and servers are identified as $(x_n \cdots x_1; 0)$ and $(x_n \cdots x_1; y)$, respectively;

(2) The edges are defined as $((x_n \cdots x_1; 0), (x_n \cdots x_1; y))$ and $((x_n \cdots x_1; y), (x_n \cdots x_{y+1}\overline{x_y}x_{y-1} \cdots x_1; y))$;

where $x_i \in \{0, 1\}, n \geq 2, 1 \leq i \leq n, 1 \leq y \leq n$ and $\overline{x_y}$ is the complement of x_y.

Fig. 1 shows the structure of $HSDC_4(4)$.

Definition 2. *[10] Given two regular graphs G and K, the compound graph $G(K)$ is obtained by replacing each node of G by a copy of K and replacing each link of G by a link which connects corresponding two copied of K.*

We treat the logical graph H_n of HSDC as a compound graph $G(K)$, where G is a n-dimensional hypercube and K is the same dimensional complete graph. In the following, we will introduce the definition of hypercube and complete graph.

Definition 3. *[11] The node-set V of n-dimensional hypercube Q_n consists of all binary sequence of length n on the set $\{0, 1\}$, i.e., $V = \{x_1x_2 \cdots x_n | x_i \in \{0, 1\}, i = 1, 2, \cdots, n\}$. Two nodes $x = x_1x_2 \cdots x_n$ and $y = y_1y_2 \cdots y_n$ are linked by an edge if and only if x and y differ exactly in one coordinate, i.e., $\sum_{i=1}^{n} |x_i - y_i| = 1$.*

Proposition 1. *[11] The hypercube Q_n has the following properties.*

(1) Q_n is n-regular, has 2^n nodes and $n2^{n-1}$ edges;

(2) Q_n is bipartite;

(3) Q_n is hamiltonian if $n \geq 2$; and eulerian if n is even;

(4) Q_n has the diameter $d(Q_n) = n$;

(5) Q_n has the connectivity $\kappa(Q_n) = \lambda(Q_n) = n$;

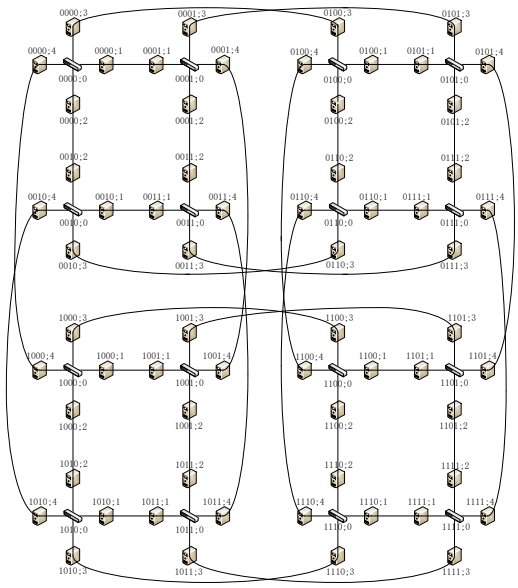

Fig. 1. The structure of $HSDC_4(4)$.

(6) Q_n *is a Cayley graph* $C_\Gamma(S)$ *and, hence, is node-transitive, where* $\Gamma = Z_2 \times \cdots \times Z_2, S = \{100\cdots00, 010\cdots00, \cdots, 00\cdots01\}$*;*

(7) Q_n *is edge-transitive.*

Definition 4. *[9] A complete graph is a simple graph in which every pair of distinct nodes is connected by an edge.*

According to the definition of compound graph and the structure of HSDC. Next, we give the formal definition of H_n.

Definition 5. *The logical graph of HSDC, named as* H_n *where* $n \geq 2$*. The node-set* V *is represented as* $(x_n x_{n-1} \cdots x_1; y)$ *where* $x_n x_{n-1} \cdots x_1$ *is the label of the node in* Q_n *and* y *is the label of the node in* K_n*. Two nodes* $u = (x_n x_{n-1} \cdots x_1; y)$ *and* $v = (x'_n x'_{n-1} \cdots x'_1; y')$ *are adjacent if and only if one of the following conditions is satisfied:*

(1) $x_n x_{n-1} \cdots x_1 = x'_n x'_{n-1} \cdots x'_1$ *and* $y \neq y'$*;*

(2) $y = y'$ *and* $x_n x_{n-1} \cdots x_1, x'_n x'_{n-1} \cdots x'_1$ *differ only in the* y*-th bit.*

We call the edge that satisfies the first condition is a clique edge, and the edge satisfies the second condition is a hypercube edge or cross edge. For each node $u \in V(H_n)$, if $(u, v) \in E(H_n)$ is a clique edge, we say v is an inner-neighbor of u, and v is an external-neighbor of u when (u, v) is a cross edge. Fig. 2 shows the structure of H_4.

In this paper, we let

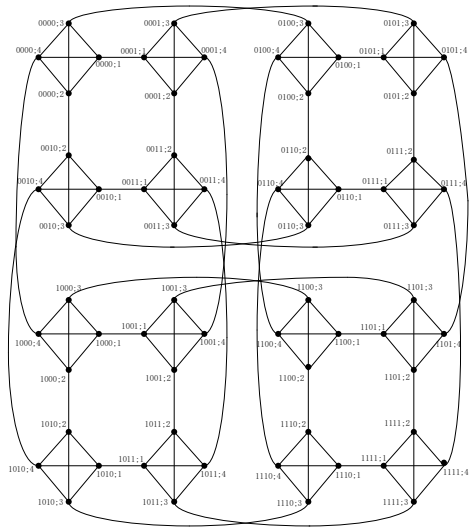

Fig. 2. The structure of H_4.

$$I_{2^n} = \{x_n x_{n-1} \cdots x_1 | x_i \in \{0,1\}, i = 1, 2, \cdots, n\}.$$

We know that H_n consists of 2^n disjoint cliques. For each clique, named as $B_i, i \in I_{2^n}, B_i \cong K_n$.

Proposition 2. *H_n has the following properties.*

(1) H_n is n-regular;

(2) H_n has $n2^n$ nodes and $n^2 2^{n-1}$ edges;

(3) Each node in H_n is associated with a clique and has only one external-neighbor.

3 Connectivity and Tightly Super Connectivity of H_n

As the number of nodes in H_n continues to increase, the failures of the node become the norm. Generally, reliable data transmission in H_n is based on the condition of any set of faulty nodes. In other words, any nodes in H_n may fail. In this case, assuming that the connectivity of H_n is κ, and the number of faulty nodes in the structure is at most $\kappa - 1$, that is, there at least one fault-free path between any two fault-free nodes, which can be used for reliable communication between nodes. Let F be a faulty node-set of H_n, and for any integer $i \in I_{2^n}$, we set $F_i = B_i \cap F$. For convenience, we also let

$$I = \{i \in I_{2^n} \mid |F_i| \geq n-1\}, J = I_{2^n} - I,$$

$$J_0 = \{j \in J \mid |F_j| = 0\}, J_1 = J - J_0 = \{j \in J \mid 1 \leq |F_j| \leq n-2\},$$

and

$$B_J = \cup_{j\in J} B_j, F_J = \cup_{j\in J} F_j.$$

Obviously, $B_j - F_j$ is connected for any $j \in J$.

Lemma 1. *For any subset F of H_n with $|F| \leq n-1$ and $n \geq 2$, we have $|I| \leq 1$ and $B_J - F_J$ is connected.*

Proof. Since $|F| \leq n-1$, we have $|I| \leq 1$ (otherwise, $2n-2 \leq |F_I| \leq |F| \leq n-1$, a contradiction). Let $|J_1| = m$. Then $|J_0| = 2^n - |I| - m$. Note that

$$\begin{aligned} |J_0| &= 2^n - |I| - m \\ &\geq 2^n - |F| \\ &\geq 2^n - (n-1) \\ &\geq 3. \end{aligned}$$

Then we have $J_0 \neq \emptyset$. Now, we shall show that B_{J_0} is connected through induction on $|J_0|$.

(1) When $|J_0| = 3$, we have $n = 2$. It is easy to see that B_{J_0} is connected.

(2) In the induction step, assume that the statement is true for $|J_0| = k$, where $k \geq 3$.

(3) When $|J_0| = k+1$, since for any $i \in J_0$, there are at least $n-(n-1) = 1$ cross edges between B_i and $B_{J_0-\{i\}}$, where $B_{J_0-\{i\}}$ is connected by induction. Hence, B_{J_0} is connected.

For any $j \in J_1$, $|F_j| \leq n-2$, we know $B_j - F_j$ is still connected since $B_j \cong K_n$, and the connectivity of K_n is $n-1$. To prove that $B_J - F_J$ is connected, it suffices to prove that $B_j - F_j$ is still connected to B_{J_0} for any $j \in J_1$. For each B_j with $j \in J_1$, there are at least $n-(m-1)-|I|$ cross edges between B_j and B_{J_0}. Since there exists at least one faulty node in B_j for any $j \in J_1$, we have

$$|F_j| = |F| - |F_I| - |F_{J_1-\{j\}}| \leq n-1-(n-1)|I| - (m-1).$$

Thus, we have

$$\begin{aligned} [n-(m-1)-|I|] - |F_j| &\geq [n-(m-1)-|I|] - [n-1-(n-1)|I| - (m-1)] \\ &= n-(m-1)-|I| - n + 1 + (n-1)|I| + (m-1) \\ &= (n-2)|I| + 1 \\ &\geq 1, \end{aligned}$$

which implies that there exists at least one fault-free cross edge between B_j and B_{J_0}, i.e., $B_j - F_j$ is connected to B_{J_0}. By the arbitrariness of j, $B_J - F_J$ is connected. □

Theorem 1. $\kappa(H_n) = n (n \geq 2)$.

Proof. By Whitney's inequality, we have

$$\kappa(H_n) \leq \lambda(H_n) \leq \delta(H_n) = n.$$

Thus, we just need to show $\kappa(H_n) \geq n$ in the following. That is, we need show that $H_n - F$ is still connected when $|F| = n-1$. We know that $0 \leq |I| \leq 1$. Otherwise, $2n-2 \leq |F| = n-1$, a contradiction.

When $|I| = 0$, by Lemma 1, $H_n - F = B_J - F_J$ is connected.

When $|I| = 1$, say $I = \{i\}$, then $|F_i| = |F| = n-1$, B_i has only one node, say v, and $J_0 = J = I_{2^n} - \{i\}$. Similarly, by Lemma 1, $B_{I_{2^n}-\{i\}}$ is connected. Note that each node in H_n has exactly one external-neighbor by Proposition 2. Then, the node v in $B_i - F_i$ is connected to some node in $B_{I_{2^n}-\{i\}}$.

Thus, $H_n - F$ is connected. □

Theorem 2. *H_n is tightly super n-connected.*

Proof. Let F be a minimum separating set of H_n. Then $|F| = n$ by Theorem 1. According to the definition of tightly super connectivity, we just need to show that $H_n - F$ has exactly two components, one of them is an isolated node.

When $n = 2$, $|F| = 2$, say $F = \{u, v\}$, if exist one node x and $N_G(x) = \{u, v\}$, then $H_n - F$ has exactly an isolated node and a connected component.

Next, we consider the case when $n \geq 3$. It is clear that $|I| \leq 1$ (otherwise, $2n-2 \leq |F_i| \leq |F| = n$, a contradiction).

Case 1. $|I| = 0$.

When $|I| = 0$, we have $H_n - F = B_J - F_J$. First, we know that for any $j \in J_1$, $1 \leq |F_j| \leq n-2$, then $B_j - F_j$ is connected. In the following, we just consider the case that $B_J - F_J$ is not connected. In this case, we know that for any $j \in J_1$, we have $2 \leq B_j - F_j \leq n-1$, therefore, at least two nodes in $B_j - F_j$ are fault-free. $B_J - F_J$ is not connected, if and only if there exist two distinct fault-free nodes in $B_j - F_j$, say v_1, v_2, and their external-neighbors are all faulty. Then, we know that (v_1, v_2) is an isolated edge, a contradiction.

Case 2. $|I| = 1$.

Suppose $I = \{i\}$, we distinguish the following two cases according to the size of $|F_i|$.

Case 2.1. $|F_i| = n$.

Obviously, $V(F_i) = V(B_i)$. We have $B_i - F_i = \emptyset$ and $H_n - F$ is equivalent to remove a clique from H_n, which does not affect the connectivity of H_n. Therefore, $H_n - F$ is connected, a contradiction.

Case 2.2. $|F_i| = n-1$.

Clearly, $|F_J| = |F - F_i| = n - (n-1) = 1$. Let $F_J = \{u\}$. We know $|B_i - F_i| = n - (n-1) = 1$. Thus, $B_i - F_i$ is a node, say v. Since v has only one external-neighbor, say v'. Then $B_i - F_i$ is connected to $B_J - F_J$ if $v' \neq u$. We have that $H_n - F$ is connected, a contradiction.

Now we consider that $B_i - F_i$ is not connected to $B_J - F_J$. This situation is only when $v' = u$, then v become an isolated node in $H_n - F$. Next, we need show that $B_J - F_J$ is connected. We know that $B_J = B_{J_0} + B_{J_1}$ and $F_J = \{u\}$, thus, we know $u \in B_j$ where $j \in J_1$. Since $|V(B_j - \{u\})| = n-1$, $B_j - \{u\}$ is connected. As a result, any nodes in $B_j - \{u\}$ has one external-neighbor in B_{J_0}, which implies $B_j - \{u\}$ is connected to B_{J_0} and $B_J - F_J$ is connected. Thus, v is an isolated node in $H_n - F$ and $H_n - F - \{v\}$ is connected.

Combining two cases above completes the proof. □

4 Shortest Routing Algorithm

In this section, we will propose an efficient node-to-node routing algorithm in H_n, called HRouting, to get a shortest path between any two distinct nodes. Routing is a basic feature to ensure that the network can communicate. A good routing algorithm can reduce the node-to-node transmission delay in the network, and reduce the packet loss rate. As a compound graph of hypercube and complete graph, the shortest path between any two distinct nodes in H_n is inspired by the method of hypercube. The method of hypercube is as follows. Let $x = x_1x_2\cdots x_n$ and $y = y_1y_2\cdots y_n$ be any two distinct nodes, the shortest path between x and y can obtained by this way: start at x and end at y by continuously changing the different bits from left to right. For example, $x = 0101011$, $y = 1010010$, then

$$x = 0101011, \mathbf{1}101011, 1\mathbf{0}01011, 10\mathbf{1}1011, 101\mathbf{0}011, 1010\mathbf{0}1\mathbf{0} = y$$

is a shortest path between x and y, where the boldface bits are obtained by each change. Therefore, the distance between x and y is determined by the number of coordinates of different bits in the two nodes. This number becomes the Hamming distance, denoted by $H(Q_n; x, y)$. We designed an algorithm suitable for H_n by using the method of continuously changing bits. The algorithm is as follows:

```
HRouting
Input: an n-dimensional HSDC, H_n, and two distinct nodes a, b ∈ H_n
       where a = (x_n x_{n-1} ··· x_1; y), b = (u_n u_{n-1} ··· u_1; z);
Output: a shortest path from node a to node b in H_n;
function HRouting(H_n, a, b)
    a_1 ← (x_n x_{n-1} ··· x_1), b_1 ← (u_n u_{n-1} ··· u_1), d ← H(Q_n; a_1, b_1),
    Q ← DIF(a_1, b_1), P ← (a)
    function Find-Path(a, b, d)
        if d = 0 then
            P ← (P, b)
        else if d = 1 then
            if y = z and x_y ≠ u_z then
                P ← (P, b)
            else if y ≠ z and Q[0] = y or z then
                if Q[0] = y then
                    P ← (P, (b_1; Q[0]), b)
                else
                    P ← (P, (a_1; Q[0]), b)
                end if
            else
```

```
16:                 P ← (P, (a_1; Q[0]), (b_1; Q[0]), b)
17:             end if
18:         else if 2 ≤ d ≤ n − 1 then
19:             if y ∈ Q then
20:                 c = (x_n ··· x̄_y ··· x_1; y)
21:             else if y ≠ z and Q[0] = z then
22:                 c = (x_n ··· x̄_{Q[1]} ··· x_1; Q[1])
23:             else
24:                 c = (x_n ··· x̄_{Q[0]} ··· x_1; Q[0])
25:             end if
26:             P ← (Find-Path(a, c, 1) − c, Find-Path(c, b, d − 1))
27:         else if d = n then
28:             c = (x_n ··· x̄_y ··· x_1; y)
29:             P ← (P, Find-Path(c, b, d − 1))
30:         end if
31:     end function
32:     return P
33: end function
34: function DIF(a_1, b_1)
35:     Q ← ()
36:     for i = n to 1 do in parallel
37:         if x_i ≠ u_i then
38:             Q ← (Q, i)
39:         end if
40:     end for
41:     return Q
42: end function
```

Apparently, the length of path constructed by algorithm HRouting is dependent only on the coordinate representation of the source and destination nodes. The core of the algorithm is a sub-algorithm Find-Path, we find that the most time is taken in lines $27-29$, which can be looped at most n times and the time complexity can be computed in $O(n)$ time. The lines $34-42$ construct a list to store the addresses of a_1 and b_1 with different coordinates, which takes $O(n)$ time. The $H(Q_n; a_1, b_1)$ also takes $O(n)$ time. Therefore, the time complexity of constructing the whole routing path in algorithm HRouting is $O(n)$.

Next, we will prove the correctness of the algorithm by Hamming distance . Let

$$a = (x_n x_{n-1} \cdots x_1; y), b = (u_n u_{n-1} \cdots u_1; z), a_1 = (x_n x_{n-1} \cdots x_1), b_1 = (u_n u_{n-1} \cdots u_1),$$

$$d = H(Q_n; a_1, b_1), Q = \mathrm{DIF}(a_1, b_1) = \{i | 1 \le i \le n, x_i \ne u_i\}.$$

When $d = 0$, which means node a and node b are in the same clique, the shortest path is (a, b); When $d = 1$, if $y = z$ and $x_y \ne u_z$, which indicates that (a, b) is a cross edge of H_n, the shortest path is (a, b). Both cases above are $(a, b) \in E(H_n)$. Next, we will consider the case $(a, b) \notin E(H_n)$.

Case 1. $d = 1$.

We use the following method to find a shortest path from a to b.

$$a \sim a^{(0)} \sim b, a^{(0)} = (x_n x_{n-1} \cdots \overline{x_{Q[0]}} \cdots x_1; Q[0]).$$

When $y \neq z$ and $Q[0] = y$, the shortest path is $a \to (b_1; y) \to b$;

When $y \neq z$ and $Q[0] = z$, the shortest path is $a \to (a_1; z) \to b$.

In the remaining cases, the shortest path is $a \to (a_1; Q[0]) \to (b_1; Q[0]) \to b$.

Case 2. $2 \leq d \leq n$.

We find an intermediate node c by reducing the Hamming distance by 1 each time, then the problem is transformed into finding a shortest path between node c and node b. We must ensure that the node c found every time is optimal. We regard the sub-algorithm Find-Path as a cyclic invariant of this algorithm. Next, we need to prove that the cyclic invariant holds at each loop. At the beginning of the algorithm, we first compute the Hamming distance of the two nodes, then input the two nodes and Hamming distance into the sub-algorithm Find-Path. In the sub-algorithm, we give a specific method to determine the intermediate node c to ensure that the cyclic invariant holds. The path construct by this way:

$$a \sim a^{(0)} \sim a^{(1)} \sim \cdots \sim a^{(d-1)} \sim b,$$

among them,

$$a^{(0)} = (x_n x_{n-1} \cdots \overline{x_{Q[0]}} \cdots x_1; Q[0]), a^{(1)} = (x_n x_{n-1} \cdots \overline{x_{Q[1]}} \cdots \overline{x_{Q[0]}} \cdots x_1; Q[1])$$

where $\overline{x_{Q[1]}}$ and $\overline{x_{Q[0]}}$ appear in an undefined order, and so on.

We use the distance from node a to node $a^{(0)}$ as an example to analyze the distance from node $a^{(i)}$ to node $a^{(i+1)}$ in the above steps, and the distance between the remaining two nodes is consistent with it. Through the Case 1, we know that the length from node a to node $a^{(0)}$ has the following cases:

(1) $y = z$ and $x_y \neq u_z$, the length is 1;

(2) $y \neq z$ and $Q[0] = y$ or $Q[0] = z$, the length is 2;

(3) otherwise, the length is 3.

It can be seen from the above three cases that we must perform an accurate conversion to ensure that the obtained node c is optimal. For example, when $y \in Q$, we put $a \sim a^{(y)}$ in the first step, the length of path $a \to (x_n x_{n-1} ... \overline{x_y} ... x_1; y)$ is 1, which is smaller than the other cases. In each conversion, the above analysis is required. Therefore, it can be ensured that the resulting path is the shortest.

Lemma 2. *Let $a = (x_n x_{n-1} \cdots x_1; y)$, $b = (u_n u_{n-1} \cdots u_1; z)$ be any two distinct nodes in H_n. And $a_1 = (x_n x_{n-1} \cdots x_1), b_1 = (u_n u_{n-1} \cdots u_1), d = H(Q_n; a_1, b_1)$, $Q = DIF(a_1, b_1) = \{i | 1 \leq i \leq n, x_i \neq u_i\}$. The distance between a and b is:*

$$dist(a, b) \begin{cases} = 2d + 1 \text{ if } y \notin Q \text{ and } z \notin Q; \\ = 2d - 1 \text{ if } y \neq z \text{ and } y, z \in Q; \\ = 2d \quad\quad \text{if } y = z \text{ and } y, z \in Q \text{ or only one of } y, z \in Q. \end{cases}$$

And the length of the shortest path constructed by this algorithm is no more than $2d + 1$ if $d < n$ and at most $2d$ if $d = n$.

Proof. According to the algorithm HRouting, constructing a path from a to b has the following three cases. Next, we will discuss the length of the path in these cases.

Case 1. $y, z \notin Q$.

We let $a^{(0)} = (x_n x_{n-1} \cdots x_1), a^{(1)} = (x_n x_{n-1} \cdots x_{Q[0]} \cdots x_1)$ and so on. Thus,

$$\begin{aligned} a = (a^{(0)}; y) &\to (a^{(0)}; Q[0]) \to (a^{(1)}; Q[0]) \to \cdots \\ &\to (a^{(i)}; Q[i-1]) \to (a^{(i)}; Q[i]) \to (a^{(i+1)}; Q[i]) \to \cdots \\ &\to (a^{(d-1)}; Q[d-2]) \to (a^{(d-1)}; Q[d-1]) \to (a^{(d)}; Q[d-1]) \to b. \end{aligned}$$

We know that the length from $(a^{(i)}; Q[i-1])$ to $(a^{(i)}; Q[i])$ is 2, and such conversion has been carried out d times, adding the path from $(a^{(d)}; Q[d-1])$ to b, the distance from a to b is $2d+1$.

Case 2. $y \neq z$ and $y, z \in Q$.

Let $y = Q[0]$, $z = Q[d-1]$, which means $(a^{(0)}; y) = (a^{(0)}; Q[0])$ and $(a^{(d)}; Q[d-1]) = b$. After the analysis of Case 1, we can conclude that the distance from a to b is $2d+1-1-1 = 2d-1$.

Case 3. $y = z$ and $y, z \in Q$ or only one of $y, z \in Q$.

Let $y \in Q$, and the case $y = z \in Q$ is equal to only one of $y, z \in Q$. Through the analysis of Case 2, the distance from a to b is $2d+1-1 = 2d$.

After the analysis of the above three cases, we know that when $d = n$, only the latter two cases may occur. Therefore, the length of the shortest path constructed by this algorithm is no more than $2d+1$ if $d < n$ and at most $2d$ if $d = n$. □

Theorem 3. *The diameter of H_n is $2n$.*

Proof. According to the Lemma 2, we know that the distance between any two distinct nodes is no more than $2d+1$ if $d < n$ and at most $2d$ if $d = n$. Thus, the diameter of H_n is $2n$. □

5 Conclusions

In the network, the connectivity $\kappa(G)$ is an important factor determining the reliability and fault tolerance of the network. In this paper, we give the connectivity and tightly super connectivity of H_n, which is the logical graph of the data center network HSDC, and construct a shortest node-to-node routing algorithm HRouting. Finally, we summarize the length of the shortest path between any two distinct nodes and prove that the diameter of the network is $2n$. In the next work, we will study the restricted connectivity of H_n [12]. Furthermore, considering that there are some variants of hypercubes with smaller diameters, such as crossed cubes [13], spined cubes [14], etc., can be used to design DCNs by the method in [6] such that the diameters of the DCNs become smaller. Which deserve our further study.

Acknowledgment

This work was supported by the Joint Fund of the National Natural Science Foundation of China (No. U1905211), the National Natural Science Foundation of China (No. 61972272) and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. M. Al-Fares, A. Loukissas, A. Vahdat, A Scalable, Commodity Data Center Network Architecture, ACM, 38 (4), (2009) 63–74.
2. C. E. Leiserson, Fat-trees: Universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers, 34 (10), (2012) 892–901.
3. C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers, Acm Sigcomm Conference on Data Communication, 2009.
4. L. Bhuyan, D. Agrawal, Generalized Hypercube and Hyperbus Structures for a Computer Network, IEEE Transactions on Computers, 33 (4), (2006) 323–333.
5. X. Wang, J. Fan, C.-K. Lin, J. Zhou, Z. Liu, BCDC: A High-Performance, Server-Centric Data Center Network, Journal of Computer ence and Technology, 33 (2), (2018) 400–416.
6. Z. Zhang, Y. Deng, G. Min, J. Xie, L.T. Yang, Y. Zhou, HSDC: a highly scalable data center network architecture for greater incremental scalability, IEEE Trans. Parallel and Distributed Systems, 30 (5), (2019) 1105–1119.
7. C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, DCell: A scalable and fault-tolerant network structure for data centers, Acm Sigcomm Computer Communication Review, 38 (4), (2008) 75–86.
8. X. Wang, J. Fan, X. Jia, C.-K. Lin, An efficient algorithm to construct disjoint path covers of DCell networks. Theor. Comput. Sci., 609, (2016) 197–210.
9. L.-H. Hsu and C.-K. Lin, Graph theory and interconnection networks, CRC Press, 2009.
10. D. Guo, H. Chen, H. Chen, Y. He, H. Jin, C. Chen, H. Chen, Z. Shu, G. Huang, KCube: a novel architecture for interconnection networks, Inf. Process. Lett., 110 (18–19), (2010) 821–825.
11. J. Xu, Combinatorial theory in networks, Science Press, 2001.
12. X. Wang, J. Fan, J. Zhou, C.-K. Lin, The restricted h-connectivity of the data center network DCell, Discret. Appl. Math, 203, (2016) 144–157.
13. K. Efe, A Variation on the Hypercube with Lower Diameter, IEEE Trans. Computers, 40 (11), (1991) 1312–1316.
14. W. Zhou, J. Fan, X. Jia, S. Zhang, The spined cube: A new hypercube variant with smaller diameter, Inf. Process. Lett., 111 (12), (2011) 561–567.