ExCCC-DCN: A Highly Scalable, Cost-Effective and Energy-Efficient Data Center Structure

Zhen Zhang, Yuhui Deng, *Member, IEEE,* Geyong Min, *Member, IEEE,* Junjie Xie, and Shuqiang Huang

Abstract—Over the past few years, a lot of data centers have been constructed around the world due to the explosive growth of data. The cost and energy consumption have become the most important challenges of building those data centers. Data centers today use commodity computers and switches instead of high-end servers and interconnections for cost-effectiveness. In this paper, we propose a new type of interconnection networks called *Exchanged Cube-Connected Cycles (ExCCC)*. The *ExCCC* network is an extension of *Exchanged Hypercube (EH)* network by replacing each node with a cycle. The *EH* network is based on link removal from a *Hypercube* network, which makes the *EH* network more cost-effective as it scales up. After analyzing the topological properties of *ExCCC*, we employ commodity switches to construct a new class of data center network models named *ExCCC-DCN* by leveraging the advantages of the *ExCCC* architecture. The analysis and experimental results demonstrate that the proposed *ExCCC-DCN* models significantly outperform four state-of-the-art data center network models in terms of the total cost, power consumption, scalability and other static characteristics. It achieves the goals of low cost, low energy consumption, high network throughput and high scalability simultaneously.

Index Terms—Data center, Interconnection, Topology, Cost-effectiveness, Scalability.

1 Introduction

As a service-oriented platform, data centers form the core of cloud computing over Internet. The data center network entails the design of both the network structure and the associated protocols to interconnect thousands of or even hundreds of thousands of servers within a data center, with low cost, high and balanced network capacity, and strong scalability [1], [2], [3], [4]. Such data centers are essential to offer numerous online applications, such as search, gaming and Web mail. They also provide infrastructure services, such as GFS [5], Map-reduce [6] and Dryad [7].

The data center architectures can be divided into two categories: switch-centric and server-centric [3]. A switch-centric network typically consists of multi-level trees of switches to connect the servers. The switch-centric networks are able to support communications between tens of thousands of servers. In switch-centric designs, the interconnection intelligence depends on switches, while servers do not need to be modified for the interconnection purpose. *Fat-Tree* [8] network belongs to this category. However, the data centers are growing large and the number of servers is increasing at an exponential rate. In order to cope with these challenges, the bandwidth of the switches in the core level is increasing, and the cost of the switches is also getting higher and higher. Then the server-centric data center network models were proposed to reduce the cost and improve the network bandwidth. In server-centric designs, interconnection intelligence is integrated in servers so that they can act as both computing nodes and packet relay nodes. These designs generate overhead of packet relay on the servers. *DCell* [9], *BCube* [10] and *FiConn* [11] fall into this category.

In recent years, a lot of data centers have been constructed around the world due to the explosive growth of data. Many companies, such as Microsoft, Google, Facebook, Yahoo and Amazon, have spent billions of dollars to establish data centers. Huge volumes of hardware, software, and database resources in these large-scale data centers can be allocated dynamically to millions of Internet users simultaneously. Meanwhile, the cost and energy consumption have become the most important challenges of building such data centers [12], [13], [14], [15], [16]. Therefore, data centers today normally use commodity computers and switches instead of high-end servers and interconnections to achieve cost-effectiveness.

In order to construct highly scalable date center network models involving huge quantity of servers, we should reduce the number of switches and network ports as much as possible. By doing so, we not only can save energy by reducing the power consumed by the network devices, but also can decrease the cost of those switches. Based on this idea, this paper attempts to propose a new data center network model which is scalable, cost-effective and power-saving .

Network topology has a significant impact on the performance of data centers. Therefore, the interconnection network designed to construct data centers, must have the properties of low degree, low diameter and high scalability. *Hypercube-type* networks have received much attention over the past few years since they offer a rich interconnection structure with high bandwidth and logarithmic diameter [17], [18], [19], [20]. However, the degree of each processor element in *Hypercube-type* networks grows with its dimension. This makes the *Hypercube-type* networks not well suitable for massively parallel systems. An *Exchanged Hypercube(EH)* is constructed by removing links from a *Hypercube*, which makes the network topology more cost-effective when it scales up [21].

The *Cube-Connected Cycles (CCC)* type network is an extension of *hypercube* network by replacing each node with a *cycle*

- *Z. Zhang, Y. Deng and J. Xie are are with the Department of Computer Science, Jinan University, Guangzhou, Guangdong, China. E-mail:zzhang@jnu.edu.cn, tyhdeng@jnu.edu.cn,xiejunjiejnu@126.com.*
- *G. Min is with the College of Engineering, Mathematicsand Physical Sciences, University of Exeter, Exeter, EX4 4QF, United Kingdom. E-mail: g.min@exeter.ac.uk.*
- *S. Huang is with theNetwork and Educational Technology Center, Jinan University, Guangzhou, Guangdong, China. E-mail:hsq@jnu.edu.cn.*

[22]. The *CCC* architecture is an attractive parallel computation network, because it is suitable for massively parallel systems while preserving all desired features of hypercube. Some variations of *CCC* structures including *Folded cube-connected cycles (FCCC)* [23] and *Extended Cube-Connected Cycles (ECCC)* [24] have been proposed to enhance its properties

Based on *EH*, we propose a new type of interconnection network called *Exchanged Cube-Connected Cycles (ExCCC)* with good topological properties. After analyzing the topological properties of *ExCCC*, we construct a new type of data center network model named *ExCCC-DCN* based on *ExCCC*. The *ExCCC-DCN* models are constructed by using commodity switches. The proposed *ExCCC-DCN* is highly scalable, cost-effective and energy-efficient.

The major contributions of this paper include:

(1) We propose a new type of interconnection network *ExCCC*. The *ExCCC* network is an extension of *EH* network by replacing each node with a cycle. The *EH* network is based on link removal from a *Hypercube* network, which makes the network more cost-effective as it scales up.

(2) We construct a new type of data center network model named *ExCCC-DCN* based on the proposed *ExCCC*. The proposed *ExCCC-DCN* model outperforms four typical data center models including *Fat-Tree*, *BCube*, *DCell* and *FiConn* in terms of a number of performance criteria. For example, the number of ports per switch is a small constant, the servers used in the data center require only two ports, the total number of switches is $O(N/\log N)$, and the diameter of the data center network is $O(\log N)$, where N is the number of the servers.

(3) We analyze the static characteristics including diameter, bisection width, number of switches, number of ports, number of wires of the above five different data center models. Furthermore, the cost and power consumption of the models are also investigated. The analysis results demonstrate that *ExCCC-DCN* is a very good candidate for building large-scale data centers.

(4) We design a simulator and build five different network structures including *Fat-Tree*, *DCell*, *BCube*, *FiConn* and *ExCCC-DCN* in the simulator to evaluate the throughput. The simulator considers the data center network as a graph. The capacity of each edge is customized. The simulator formalizes the flows with four tuples including source host, destination host, start time and flow size. Moreover, it estimates the delay caused by forwarding, queuing, transmission and processing by assigning a fixed *RTT* to each flow. Experimental results depict that the performance of *ExCCC-DCN* outperforms that of *FiConn* to a certain degree. Although the highest throughput of the *ExCCC-DCN* and *FiConn* are both about 110Gbps, *ExCCC-DCN* takes less time to complete the data transmission.

This paper is organized as follows. Section 2 introduces the related work of data centers. Section 3 presents the definition of *ExCCC* and analyzes the topological properties of *ExCCC*. In Section 4, we propose the *ExCCC-DCN* based on *ExCCC* network, and compare the *ExCCC-DCN* against other four state-of-the-art *DCN* models. Finally, Section 5 concludes the paper.

2 Related Work

This section introduces four important data center network structures including *Fat-Tree* [8], *DCell* [9], *BCube* [10] and *FiConn* [11]. The *Fat-Tree* is a switch-centric structure. It has been widely used in practice by companies like Google [25], Microsoft [26], IBM [27], HP [28], SGI [29], SUN [30], and etc. The other three structures belong to server-centric. Those networks have been widely studied in the existing literature for data centers and have been very popular, as demonstrated by their large number of citations received so far. All the four network structures employ commodity switches to build data centers.

Fat-Tree: *Fat-Tree* [8] normally has three levels of switches. It normally consists of n pods. Each pod contains two levels (i.e., the edge level and the aggregation level) of $n/2$ switches. Each n-port switch at the edge level uses $n/2$ ports to connect $n/2$ servers, while using the remaining $n/2$ ports to connect the $n/2$ aggregation level switches in the pod. There are $n^2/4$ n-port switches at the core level. Each switch at the core level has one port connecting to one pod. Therefore, the total number of servers supported by the *Fat-Tree* network structure is $n^3/4$, and the total number of switches is $5n^2/4$. Fig. 1 illustrates the topology of a *Fat-Tree* structure with $n = 4$.

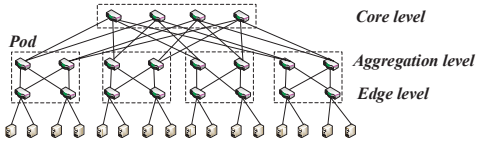

Fig. 1: *Fat-Tree* structure with $n = 4$.

DCell: *DCell* [9] is a level-based and recursively defined network structure. In $DCell_0$, n servers are connected to an n-port commodity switch. Given t servers in a $DCell_k$, $t + 1$ $DCell_k$s are used to build a $DCell_{k+1}$. The t servers in a $DCell_k$ connect to the other t $DCell_k$s, respectively. *DCell* has a high bisection width. Fig. 2 shows the topology of a $DCell_1$.

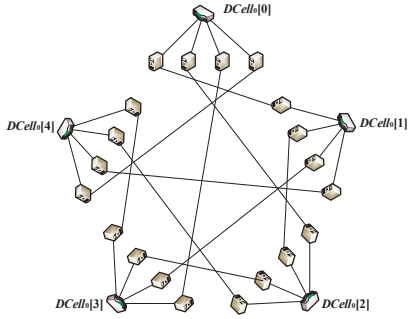

Fig. 2: $DCell_1$ structure with $n = 4$.

FiConn: *FiConn* [11] uses a recursive scheme that is similar to *DCell* to construct a data center network structure. $FiConn_0$ is composed of multiple servers and an n-port commodity switch connecting the servers. Typically, n is an even number such as 8, 16, 32, or 48. Every server in $FiConn_0$ has one port connected to the switch in $FiConn_0$. $FiConn_k$ is constructed by using $b/2 + 1$ $FiConn_{k-1}$, where b is the total number of available backup ports in $FiConn_{k-1}$. In each $FiConn_{k-1}$, $b/2$ servers out of the b servers with available backup ports are selected to connect the $b/2$ other $FiConn_{k-1}$s using their backup ports. Each backup port is connected to one $FiConn_{k-1}$. Fig. 3 depicts a $FiConn_1$ with $n = 4$.

BCube: *BCube* [10] is also a server-centric interconnection topology. However, it is targeted for shipping-container-sized data centers containing $1 - 2K$ servers. It is also a level-based network structure. A $BCube_0$ is composed of n servers connecting to an n-port switch. A $BCube_1$ is constructed by employing n $BCube_0$

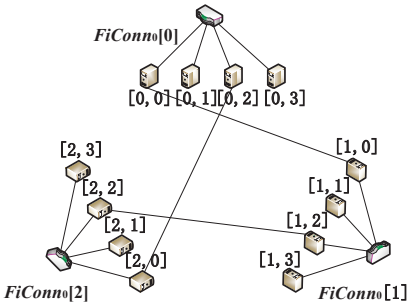

Fig. 3: $FiConn_1$ with $n = 4$.

and n-port switches. By analogy, a $BCube_k$ is constructed by using n $BCube_{k-1}$ and n^k n-port switches. Each server in a $BCube_k$ has $k + 1$ ports. Fig. 4 illustrates a $BCube_1$ with $n = 4$.

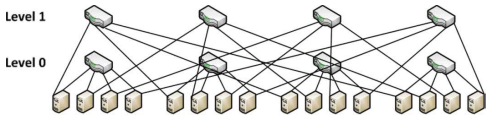

Fig. 4: $BCube_1$ structure with $n = 4$.

3 EXCHANGED CUBE-CONNECTED CYCLES

An interconnection network is usually represented by a graph where the nodes stand for processors and the edges stand for links between the processors. A graph $\Gamma = (V, E)$ is defined by a set V of nodes and a set E of directed edges. The set E is a subset of elements (u, v) of $V \times V$. E is considered symmetric, if $(u, v) \in E$, $(v, u) \in E$, in which case these two opposite arcs (u, v) and (v, u) are denoted as an undirected edge (u, v).

3.1 Definition of ExCCC

Definition 1. The $s+t+1$ dimension *Exchanged Hypercube* is defined as an undirected graph $EH(s, t) = (V, E)(s \geq 1, t \geq 1)$. V is the set of nodes $V = \{a_{s-1}...a_0 b_{t-1}...b_0 c | a_i, b_j, c \in \{0, 1\}\ for\ i \in [0, s), j \in [0, t)\}$, and E is the set of edges $E = \{(v_1, v_2) \in V \times V\}$. E consists of three types of edges (i.e., E_1, E_2 and E_3) which are described as follows:

1) E_1: $v_1 \oplus v_2 = 1$ or,
2) E_2: $v_1[0] = v_2[0] = 1, v_1[s + t : t + 1] = v_2[s + t : t + 1], H(v_1[t : 1], v_2[t : 1]) = 1$ or,
3) E_3: $v_1[0] = v_2[0] = 0, v_1[t : 1] = v_2[t : 1], H(v_1[s + t : t + 1], v_2[s + t : t + 1]) = 1$.

where $\oplus$ denotes the exclusive-OR operator, $v[x : y]$ indicates the bit pattern of v between dimensions y and x inclusive, and $H(x, y)$ implies the Hamming distance between nodes x and y.

Fig. 5 shows an example of $EH(1, 2)$.

The $EH(s, t)$ contains two kinds of hypercubes denoted as $H(s : y_{t-1}...y_0 c)$ with s dimensions and $H(t : x_{s-1}...x_0 c)$ with t dimensions. In $EH(1, 2)$, the node set $\{(0001), (1001)\}$ construct hypercube $H(1 : 001)$ and nodes $\{(0001), (0011), (0101), (0111)\}$ construct hypercube $H(2 : 01)$. The $EH(s, t)$ is constructed by using total 2^t $H(s : yt - 1...y0c)$ and 2^s $H(t : xs - 1...x0c)$.

Definition 2. The $s + t + 1$ dimension $ExCCC(s, t)$ network is an extension of EH by replacing each node in EH with a cycle. The node set of $ExCCC(s, t)$ is:

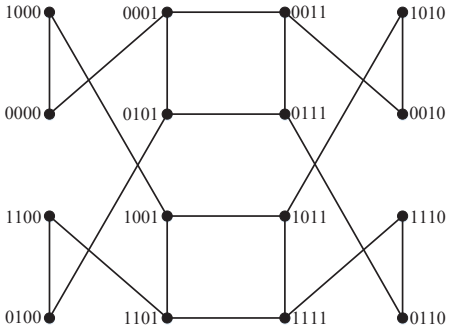

Fig. 5: $EH(1, 2)$.

$V = \{(x_s...x_1 y_t...y_1 0; p) | x_i, y_j \in Z_2, i \in [1, s], j \in [1, t]$ and $p \in [0, s]\} \cup (w_s...w_1 u_t...u_1 1; q) | w_i, u_j \in Z_2, i \in [1, s], j \in [1, t]$ and $q \in [0, t]\}$.

The edges of $ExCCC(s, t)$ are defined as follows:

$e_1 : (x_s...x_1 y_t...y_1 0; p) \rightarrow (x_s...x_{p+1} x_p \oplus 1 x_{p-1}...x_1 y_t...y_1 0; p),\ p \in [1, s]$;

$e_2 : (x_s...x_1 y_t...y_1 0; p) \rightarrow (x_s...x_1 y_t...y_1 0; p \pm 1\ mod\ (s + 1), p \in [0, s]$;

$e_3 : (x_s...x_1 y_t...y_1 0; 0) \rightarrow (x_s...x_1 y_t...y_1 1; 0)$;

$e_4 : (x_s...x_1 y_t...y_1 1; q) \rightarrow (x_s...x_1 y_t...y_{q+1} y_q \oplus 1 y_{q-1}...y_1 1; q), q \in [1, t]$;

$e_5 : (x_s...x_1 y_t...y_1 1; q) \rightarrow (x_s...x_1 y_t...y_1 1; q \pm 1\ mod\ (t + 1), q \in [0, t]$;

$e_6 : (x_s...x_1 y_t...y_1 1; 0) \rightarrow (x_s...x_1 y_t...y_1 0; 0)$.

The edges e_2 and e_5, e_1 and e_4, and e_3 and e_6 in Definition 2 are defined as *cycle-edges*, *cube-edges*, and *exchanged-edges*, respectively.

The $ExCCC(1, 2)$ is shown in Fig. 6.

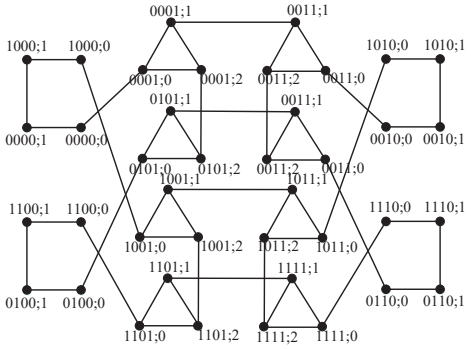

Fig. 6: $ExCCC(1, 2)$.

3.2 Topological Properties

In this section, we investigate different topological properties of *ExCCC*.

Theorem 1. $ExCCC(s, t)$ is isomorphic to $ExCCC(t, s)$.

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic, if given any edge $(u, v) \in E_1$, there exists a bijection $f : V_1 \rightarrow V_2$ such that if $(u, v) \in E_1$ then $(f(u), f(v)) \in E_2$. According to the definition of *ExCCC*, we have $ExCCC(t, s) = (V_1, E_1)$, where

$V_1 = \{(x_s...x_1 y_t...y_1 0; p) | x_i, y_j \in Z_2, i \in [1, s], j \in [1, t]$ and $p \in [0, s]\} \cup (w_s...w_1 u_t...u_1 1; q) | w_i, u_j \in Z_2, i \in [1, s], j \in [1, t]$ and $q \in [0, t]\}$, and $ExCCC(s, t) = (V_2, E_2)$, where

$V_2 = \{(x_t...x_1y_s...y_10; p)|x_i, y_j \in Z_2, i \in [1,t], j \in [1,s]$ and $p \in [0,t]\} \cup (w_t...w_1u_s...u_11; q)|w_i, u_j \in Z_2, i \in [1,t], j \in [1,s]$ and $q \in [0,s]\}$.

Considering an arbitrary node $u = (x_s...x_1y_t...y_1c; p)$ in $ExCCC(t,s)$, we can define a bijection $f : V_1 \to V_2$ as

$$f : (x_s...x_1y_t...y_1c; p) \to (y_t...y_1x_s...x_1c \oplus 1; p)$$

.

In $ExCCC(s,t)$, the node $u = (x_s...x_1y_t...y_1c; p)$ connects two nodes: u' and u''.

- Case 1. If $c = 0$, then $c \oplus 1 = 1$, we have:
  $u' = (x_s...x_1y_t...y_10; p \pm 1 \ mod \ (s+1))$.
  $u'' = \begin{cases}(x_s...x_{p+1}x_p \oplus 1x_{p-1}...x_1y_t...y_10; p) & , \text{if } p \neq 0;\\ (x_s...x_1y_t...y_11; p) & , \text{if } p = 0.\end{cases}$
  In $ExCCC(t,s)$, the node $v = f(u) = (y_t...y_1x_s...x_11; p)$ connects two nodes of v' and v'':
  $v' = (y_t...y_1x_s...x_11; p \pm 1 \ mod \ (s+1))$
  $v'' = \begin{cases}(y_t...y_1x_s...x_{p+1}x_p \oplus 1x_{p-1}...x_10; p) & , \text{if } p \neq 0;\\ (y_t...y_1x_s...x_10; p) & , \text{if } p = 0.\end{cases}$
  It is obvious that $v' = f(u')$ and $v'' = f(u'')$. Then we have $(f(u), f(u')) \in E_2$ and $(f(u), f(u'')) \in E_2$
- Case 2. If $c = 1$, then $c \oplus 1 = 0$, we have:
  $u' = (x_s...x_1y_t1; p \pm 1 \ mod \ (s+1))$;
  $u'' = \begin{cases}(x_s...y_1y_t...y_{p+1}y_p \oplus 1y_{p-1}...y_11; p) & , \text{if } p \neq 0;\\ (x_s...x_1y_t...y_10; p) & , \text{if } p = 0.\end{cases}$

In $ExCCC(t,s)$, the node $v = f(u) = (y_t...y_1x_s...x_10; p)$ connects two nodes v' and v'':

$v' = (y_t...y_1x_s...x_10; p \pm 1 \ mod \ (s+1))$;
$v'' = \begin{cases}(y_t...y_{p+1}y_p \oplus 1y_{p-1}...y_1x_s...x_10; p) & , \text{if } p \neq 0;\\ (y_t...y_1x_s...x_11; p) & , \text{if } p = 0.\end{cases}$

It is obvious that $v' = f(u')$ and $v'' = f(u'')$. Then we have $(f(u), f(u')) \in E_2$ and $(f(u), f(u'')) \in E_2$.

Thus, the networks $ExCCC(s,t)$ and $ExCCC(t,s)$ are isomorphic.■

For simplicity, we only analyze the $ExCCC(s,t)$ network, where $t \geq s$.

Theorem 2. The $ExCCC(s,t)$ network contains 2^{s+t} cycles with length of $s+1$, and contains 2^{s+t} cycles with length of $t+1$.

Proof. As aforementioned, $EH(s,t)$ contains 2^t $H(s : y_{t-1}...y_0c)$ and 2^s $H(t : x_{s-1}...x_0c)$. The number of nodes in $H(s : y_{t-1}...y_0c)$ is 2^s and the number nodes in $H(t : x_{s-1}...x_0c)$ is 2^t. According to the definition of $ExCCC$, the $ExCCC(s,t)$ can be obtained by replacing (1) each node in $H(s : y_{t-1}...y_0c)$ with a circle of length $t+1$; (2) each node in $H(t : x_{s-1}...x_0c)$ with a circle of length $s+1$.■

Since each circle contains only one node as $(x_s...x_1y_t...y_1c; 0)$, the $x_s...x_1y_t...y_1c$ is called *circle-identifier*. We use the *circle-identifier* to denote each cycle in $ExCCC(s,t)$. Two circles $a_s...a_1b_t...b_1c_1$ and $c_s...c_1d_t...d_1c_2$ are called connected, if there exist two nodes $(a_s...a_1b_t...b_1c_1; p) \in a_s...a_1b_t...b_1c_1$ and $(c_s...c_1d_t...d_1c_2; q) \in c_s...c_1d_t...d_1c_2$, and these two nodes are connected by *cube-edge* or *exchanged-edge*. If we condense each cycle into a single node and connect them through *cube-edges* and *exchanged-edges*, we get an *Exchanged hypercube* network.

The degree of a node in a graph is defined as the total number of edges connected to the node. The degree of a network is defined as the largest degree of all the nodes in its graph representation. For any graph Γ, let $V(\Gamma)$ and $E(\Gamma)$ denote the node set and edge set of Γ, respectively, and $d(\Gamma)$ denote the degree of Γ.

Theorem 3. For the $\Gamma = ExCCC(s,t)$, we have:

1) $d(\Gamma) = 2$, if $s = 1$ and $t = 1$;
2) $d(\Gamma) = 3$, if $t \geq 2$.

Proof. According to the definition of $ExCCC$, we have

- Case 1. $s = 1$ and $t = 1$.

  The degrees of node $(x_1y_10; p)$ and node $(x_1y_11; p)$ in $ExCCC(1,1)$ are both 2. Thus, $ExCCC(1,1)$ is regular and $d(ExCCC(1,1)) = 2$.
- Case 2. $t \geq 2$.
  - Case 2.1. $s = 1$ and $t \geq 2$.
    The degree of the node $(x_1y_t...y_10; p)$ in $ExCCC(1,t)$ is 2, and the degree of the node $(x_1y_t...y_11; p)$ is 3. Thus, $ExCCC(1,t)$ is irregular and $d(ExCCC(1,t)) = 3$.
  - Case 2.2. $s \geq 2$ and $t \geq 2$.
    The degree of the node $(x_s...x_1y_t...y_10; p)$ in $ExCCC(s,t)$ is 3, and the degree of the node $(x_s...x_1y_t...y_11; p)$ is 3. Thus, $ExCCC(s,t)$ is regular and $d(ExCCC(s,t)) = 3$.■

Theorem 4. For the graph $\Gamma = ExCCC(s,t)$, we have:

$|V(\Gamma)| = 2^{t+1}(t+3)$ and $|E(\Gamma)| = 2t(3t+7)$, when $s = 1$ and $t \geq 1$;

$|V(\Gamma)| = 2^{s+t}(s+t+2)$ and $|E(\Gamma)| = 3 * 2^{s+t-1}(s+t+2)$, when $s \geq 2$ and $t \geq s$.

Let ΔT represent the smallest change in the number of the network components (nodes or edges) needed to increase the existing number of components T in a network while retaining its topological characteristics. We use $IE = \Delta T/T$ to measure the incremental expand ability of the interconnection network. Let IE_{node} and IE_{edge} denote the node expand ability and the edge expand ability, respectively.

Theorem 5. For $ExCCC(s,t)$, we have $IE_{node} \to 1$ and $IE_{edge} \to 1$ asymptotically.

Proof. According to Theorem 4, we have

1) Case 1. $s = 1$.
   Since $ExCCC(1,t)$ has $2^{t+1}(t+3)$ nodes and $2^t(3t+7)$ edges, and $ExCCC(s,t)$ has $2^{s+t}(s+t+2)$ nodes and $3 * 2^{s+t-1}(s+t+2)$ edges, we have

$$\begin{aligned} IE_{vertex}(1,t) &= \frac{\Delta T_{\text{vertex}}(1,t)}{T_{\text{vertex}}(1,t)} \\ &= \frac{2^{t+2}(t+4) - 2^{t+1}(t+3)}{2^{t+1}(t+3)} \\ &= 1 + \frac{2}{t+3}, \end{aligned}$$

   and

$$\begin{aligned} IE_{edge}(1,t) &= \frac{\Delta T_{\text{edge}}(1,t)}{T_{\text{edge}}(1,t)} \\ &= \frac{2^{t+1}(3(t+1)+7) - 2^t(st+7)}{2^t(3t+7)} \\ &= 1 + \frac{6}{3t+7}, \end{aligned}$$

or

$$IE_{edge}(1,t) = \frac{\Delta T_{edge}(1,t)}{T_{edge}(1,t)} = \frac{3*2^{t+1}(t+4)-2^t(3t+7)}{2^t(3t+7)} = 1+\frac{10}{3t+7}.$$

Thus, $IE_{node}(1,t)$ and $IE_{edge}(1,t)$ are both approaching 1 as $t \to +\infty$.

2) Case 2. $t \geq s \geq 2$. Since $ExCCC(s,t)$ has $2^{s+t}(s+t+2)$ nodes and $3*2^{s+t-1}(s+t+2)$ edges, we have

$$IE_{vertex}(s,t) = \frac{\Delta T_{vertex}(s,t)}{T_{vertex}(s,t)} = \frac{2^{s+t+1}(s+t+3)-2^{s+t}(s+t+2)}{2^{s+t}(s+t+2)} = 1+\frac{2}{s+t+2},$$

and

$$IE_{edge}(s,t) = \frac{\Delta T_{edge}(s,t)}{T_{edge}(s,t)} = \frac{3*2^{s+t}(s+t+3)-3*2^{s+t-1}(s+t+2)}{s*2^{s+t-1}(s+t+2)} = 1+\frac{2}{s+t+2}.$$

Thus, $IE_{node}(s,t)$ and $IE_{edge}(s,t)$ are both approaching 1 as $s \to +\infty$ and/or $t \to +\infty$.

Theorem 5 shows that a $ExCCC(s,t)$ has very small hardware cost when it is expanded into $ExCCC(s+1,t)$ or $ExCCC(s,t+1)$.

Bisection width denotes the minimal number of edges to be removed to partition a network into two parts which are equal in size. A large bisection width implies a high network capacity and a more resilient structure against failures.

Theorem 6. The bisection width of $ExCCC(s,t)$ is 2^{s+t-1}.

Proof.

- Case 1. $s = t = 1$ Obviously, the $ExCCC(1,1)$ contains 16 nodes and its bisection is 2.
- Case 2. $t \geq 2$.
  The node set $\{(x_s...x_1y_t...y_11;p)|p \in [0,t]\}$ can be divided into two node sets V_1 and V_2, where
  $V_1 = \{(x_s...x_1y_t...y_201;p)|p \in [0,t]\}$ and
  $V_2 = \{(x_s...x_1y_t...y_211;p)|p \in [0,t]\}$.
  It is obvious that $|V_1| = |V_2|$.
  The nodes $(x_s...x_1y_t...y_201;0)$ in V_1 connect the nodes $(x_s...x_1y_t...y_200;0)$ which are in node set $V_3 = \{(x_s...x_1y_t...y_200;q)|q \in [0,s]\}$.
  The nodes $(x_s...x_1y_t...y_211;0)$ in V_2 connect the nodes $(x_s...x_1y_t...y_210;0)$ which are in node set $V_4 = \{(x_s...x_1y_t...y_210;q)|q \in [0,s]\}$.
  It is obvious that $|V_3| = |V_4|$.
  The nodes set of $ExCCC(s,t)$ is constructed by V_1, V_2, V_3 and V_4, and $|V_1 \cup V_2| = |V_3 \cup V_4|$. The number of edges between $V_1 \cup V_2$ and $V_3 \cup V_4$ is 2^{s+t-1}.■

3.3 One-to-One routing algorithm in *ExCCC*

An optimal one-to-one routing algorithm is to find the shortest path between a source and destination pair, where the source sends a message to the destination. Suppose the source node is $u = (x_s...x_1y_t...y_1c_1;p)$ and the destination node is $v = (m_s...m_1n_t...n_1c_2;q)$ in $ExCCC(s,t)$. Here, we define a binary string $w_s...w_1z_t...z_1$, where

$w_i = x_i \oplus m_i$, if $1 \leq i \leq s$;

$z_j = y_j \oplus n_j$, if $1 \leq j \leq t$.

Thus, every bit in $w_s...w_1z_t...z_1$ is 0, if the corresponding bits in $x_s...x_1y_t...y_1$ and $m_s...m_1n_t...n_1$ are identical. Otherwise, every bit in $w_s...w_1z_t...z_1$ is 1.

***Example* 1**. For two nodes $u = (101001010110;2)$ and $v = (100111101011;5)$ in $ExCCC(5,6)$, we have $(w_5w_4w_3w_2w_1u_6u_5u_4u_3u_2u_10;2) = (001110111100;2)$.

In $ExCCC(s,t)$, the shortest path from $(x_s...x_1y_t...y_1c_1;p)$ to $(m_s...m_1n_t...n_1c_2;q)$ can be transformed to the path from $(w_s...w_1u_t...u_1c_1,p)$ to $(0^{s+t}c_2,q)$. Here, 0^n denotes 0's subsequence of length n. For example, the path between $u = (101001010110;2)$ and $v = (100111101011;5)$ in $ExCCC(5,6)$ can be transformed to the path from $(001110111100;2)$ to $(000000000001,5)$. The shortest path can be constructed as follows:

$(001110111100;2) \Rightarrow (001010111100;2) \rightarrow$
$(001010111100;3) \Rightarrow (000010111100;3) \rightarrow$
$(000010111100;2) \rightarrow (000010111100;1) \Rightarrow$
$(000000111100;1) \rightarrow (000000111100;0) \rightsquigarrow$
$(000000111101;0) \rightarrow (000000111101;1) \Rightarrow$
$(000000111101;2) \Rightarrow (000000111001;2) \Rightarrow$
$(000000111001;3) \Rightarrow (000000110001;3) \Rightarrow$
$(000000110001;4) \Rightarrow (000000100001;4) \Rightarrow$
$(000000100001;5) \Rightarrow (000000000001;5)$,

where each $\Rightarrow$, $\rightarrow$ and $\rightsquigarrow$ represents a *hypercube-edge*, a *cycle-edge* and an *exchanged edge*, respectively.

To construct such a path from $(w_s...w_1u_t...u_1c_1,p)$ to $(0^{s+t}c_2,q)$, all bits with value 1 in sequence $w_s...w_1u_t...u_1$ must be transformed to 0 through *cube-edges*. Simultaneously, c_1 must be transformed to c_2 through *exchanged-edges* and p must be transformed to q through *circle-edges*. According to the definition of *ExCCC*, the bit sequence $w_s...w_1$ can be transformed to 0^s through *cube-edge* when $c_1 = 0$, while the bit sequence $u_t...u_1$ remains fixed. Similarly, the bit sequence $u_t...u_1$ can be transformed to 0^t through *cube-edge* when $c_1 = 1$, while the bit sequence $w_s...w_1$ remains fixed. It is obvious that the number of *cube-edges* in the shortest path is equal to the number of 1 bit in $w_s...w_1u_t...u_1$. The number of *exchanged-edge* in the shortest path is 0 or 2 when $c_1 = c_2$. The number will become 1 when $c1 \neq c2$.

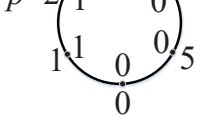

Fig. 7: *Cycle Representation (CR).*

Since the numbers of *cube-edges* and *exchanged-edges* in the shortest path are fixed, we only need to minimize the number of *cycle-edges*. In order to simplify the analysis, we only consider the bit sequence $w_s...w_1c_1$ and $c_1 = 0$. Then, we propose a *Circle Representation (CR)* for the bit sequence $w_s...w_10$. In Fig. 7, we show the *CR* for $(w_5w_4w_3w_2w_10;2) = (001110;2)$ in Example 1.

Now, we define a procedure named $path(w_s...w_10,p,q)$. This procedure can construct the shortest path from p to q on the *CR* of

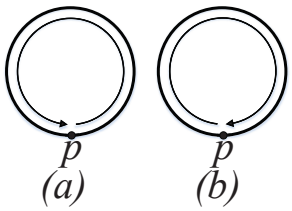

Fig. 8: The path in Case of $p = q$.

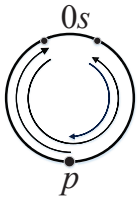

Fig. 9: The path in Case of $p = q$ with 0's subsequence in D.

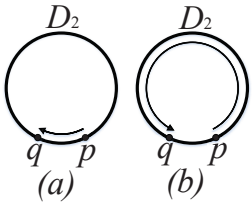

Fig. 10: The path in Case of $p = q - 1$.

$w_s...w_1 0$, and the path must pass through each 1 bit at least once. The length of this shortest path is d. Then, we can construct the shortest path according to the following three cases.

- Case 1. $p = q$, the node set (except p) on CR are defined as D.
  - Case 1.1. If there exists 1's bit in D, two paths can be constructed according to the methods as shown in Fig. 8. The p can be increased or decreased to itself, while these two paths can pass all bits in $w_s...w_1 0$. The lengths of the two paths are both $s+1$.
  - Case 1.2. If there exists 0's subsequence in D, a path can be constructed according to the method as shown in Fig. 9. Here, we only consider the longest 0's subsequence in D. If there are more than one 0's subsequence with the longest length, we can choose anyone of them. The choice may be arbitrary, and it does not affect the result. Let Z be the length of this 0's subsequence, then the length of this shortest path is $2s - 2Z$, where $Z \geq 0$.

  According to Case 1.1 and Case 1.2, we can get $d = min\{s + 1, 2s - 2Z\}$, where $Z \geq 0$.
- Case 2. $p = q \pm 1$ $(mod\ s + 1)$, the nodes p and q divide the CR into two parts: D_1 and D_2, where $|D_1| = 0$ and $|D_2| = s - 1$. Since the analysis of case $p = q + 1$ is similar to that of case $p = q - 1$, we only analyze the case of $p = q - 1$.
  - Case 2.1. A path can be constructed according to the method shown in Fig. 10a when all bits in D_2 are 0s. The length of this path is 1. Otherwise, the path can be constructed as shown in Fig. 10b. The path length is s.

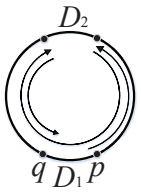

Fig. 11: The path in Case of $p = q - 1$ with 0's subsequence in D_2.

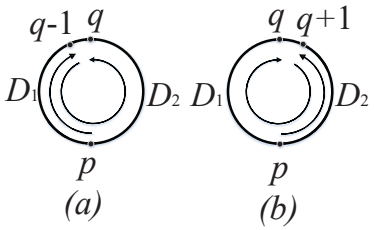

Fig. 12: The paths in Case of $p = q + \tau$.

  - Case 2.2. If there exists 0's subsequence in D_2, a path can be constructed according to the method shown in Fig. 11. Here, we only consider the longest 0's subsequence in D_2. If there is more than one 0's subsequence with longest length, we can choose anyone of them, and it does not affect the result. Let Z be the length of this 0's subsequence, and the length of this path is $2s - 2Z - 1$, where $Z \geq 0$.

  According to Case 2.1 and Case 2.2, we can get $d = min\{s, 2s - 2Z - 1\}$, where $Z \geq 0$.
- Case 3. $p = q \pm \tau$ $(mod\ s+1)$ where $\tau \geq 2$, then the CR can be divided into two parts: D_1 and D_2, where $|D_1| = \tau - 1$, and $|D_2| = s - \tau$. Since the analysis of case $p = q + \tau$ is similar to that of case $p = q - \tau$, we only analyze the case of $p = q - \tau$.
  - Case 3.1. Two paths can be constructed as shown in Fig. 12. The length of the path in Fig. 12a is $s+\tau-1$ and the other in Fig. 12b is $2s - \tau$.
  - Case 3.2. If there exists 0's subsequence in D_1, another path can be constructed according to the method shown in Fig. 13. Here, we only consider the longest 0's subsequence in D_1. If there is more than one 0's subsequence with the longest length, we can choose anyone of them, and it does not affect the result. Let Z_1 be the length of this 0's subsequence, and the length of this shortest path is $s + \tau - 2Z_1 - 1$, where $Z_1 \geq 1$.
  - Case 3.3. If there exists 0's subsequence in D_2, another path can be constructed according to the method shown in Fig. 14. Similar to Case 3.2, we only consider the longest 0's subsequence in D_2. Let Z_2 be the length of this 0's subsequence, and the length of this shortest path is $2s-\tau-2Z_2$, where $Z_2 \geq 1$.

  According to the analysis, we can get $d = min\{s+\tau-2Z_1-1, 2s - \tau - 2Z_2\}$, where $Z_1 \geq 0$ and $Z_2 \geq 0$.

Now, we construct the shortest path from node $(w_s...w_1 u_t...u_1 c_1, p)$ to node $(0^s 0^t c_2, q)$.

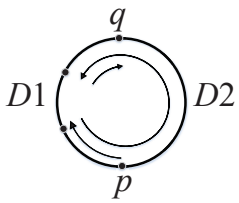

Fig. 13: The path in Case of $p = q - \tau$ with 0s subsequence in D_1.

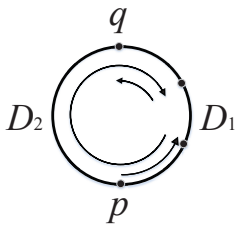

Fig. 14: The path in Case of $p = q - \tau$ with 0s subsequence in D_2

- Case 1. $c_1 = c_2$.
  - Case 1.1. $c_1 = c_2 = 0$ and $u_t...u_1 = 0^t$, the shortest path can be constructed as following:
    $(w_s...w_1 0^{t+1}; p) \xrightarrow{path(w_s...w_1 0,p,q)} (0^{s+t+1}, q)$.
  - Case 1.2. $c_1 = c_2 = 0$ and $u_t...u_1 \neq 0^t$, the shortest path can be constructed as following:
    $(w_s...w_1 u_t...u_1 0; p) \xrightarrow{path(w_s...w_1,p,0)} (0^s u_t...u_1 1; 0)$
    $\xrightarrow{path(u_t...u_1 1,0,0)} (0^{s+t+1}; 0) \xrightarrow{path(0^{s+1},0,1)} (0^{s+t+1}; q)$.
  - Case 1.3. $c_1 = c_2 = 1$ and $w_s...w_1 = 0^s$, the shortest path can be constructed as following:
    $(0^s u_t...u_1 1; p) \xrightarrow{path(u_t...u_1 0,p,q)} (0^{s+t}1, q)$.
  - Case 1.4. $c_1 = c_2 = 1$ and $w_s...w_1 \neq 0^s$, the shortest path can be constructed as following:
    $(w_s...w_1 u_t...u_1 1; p) \xrightarrow{path(u_t...u_1 1,q,0)} (w_s...w_1 0^{t+1}; 0)$
    $\xrightarrow{path(w_s...w_1 1,0,0)} (0^{s+t}1; 0) \xrightarrow{path(0^{t-1},0,q)} (0^{s+t}1; q)$
- Case 2. $c_1 \neq c_2$.
  - Case 2.1 $c_1 = 0$ and $c_2 = 1$, the shortest path can be constructed as following:
    $(w_s...w_1 u_t...u_1 0; p) \xrightarrow{path(w_s...w_1 1,p,0)} (0^s u_t...u_1 1; 0)$
    $\xrightarrow{path(u_t...u_1 0,0,q)} (0^{s+t}1; q)$.
  - Case 2.2 $c_1 = 1$ and $c_2 = 0$, the shortest path can be constructed as following:
    $(w_s...w_1 u_t...u_1 1; p) \xrightarrow{path(u_t...u_1 1,p,0)} (w_s...w_1 0^{t+1}; 0)$
    $\xrightarrow{path(w_s...w_1 0,0,q)} (0^{s+t+1}; q)$.

***Example* 2**. We give some examples in $ExCCC(4, 5)$ which conform to the cases in the routing algorithm.

- Case 1.1: $u = (1001000000; 1)$ and $v = (0000000000; 4)$, the shortest path between u and v can be constructed as follows:
  $(1001000000; 1) \Rightarrow (1000000000; 1) \rightarrow (1000000000; 0)$
  $\rightarrow (1000000000; 4) \Rightarrow (0000000000; 4)$.
- Case 1.2: $u = (1001110010; 1)$ and $v = (0000000000; 4)$, the shortest path between u and v can be constructed as follows:
  $(1001110010; 1) \Rightarrow (1000110010; 1)$
  $\rightarrow (1000110010; 0) \rightarrow (1000110010; 4)$
  $\Rightarrow (0000110010; 4) \rightarrow (0000110010; 0)$
  $\rightsquigarrow (0000110011; 0) \rightarrow (0000110011; 1)$
  $\Rightarrow (0000110001; 1) \rightarrow (0000110001; 0)$
  $\rightarrow (0000110001; 5) \Rightarrow (0000010001; 5)$
  $\rightarrow (0000010001; 4) \Rightarrow (0000000001; 4)$
  $\rightarrow (0000000001; 5) \rightarrow (0000000001; 0)$
  $\rightsquigarrow (0000000000; 0) \rightarrow (0000000000; 4)$.
- Case 1.3: $u = (0000100101; 1)$ and $v = (0000000001; 4)$, the shortest path between u and v can be constructed as follows:
  $(0000100101; 1) \rightarrow (0000100101; 2)$
  $\Rightarrow (0000100001; 2) \rightarrow (0000100001; 1)$
  $\rightarrow (0000100001; 0) \rightarrow (0000100001; 5)$
  $\Rightarrow (0000000001; 5) \rightarrow (0000000001; 4)$.
- Case 1.4: $u = (1001100101; 1)$ and $v = (0000000001; 4)$, the shortest path between u and v can be constructed as follows:
  $(1001100101; 1) \rightarrow (1001100101; 2)$
  $\Rightarrow (1001100001; 2) \rightarrow (1001100001; 1)$
  $\rightarrow (1001100001; 0) \rightarrow (1001100001; 5)$
  $\Rightarrow (1001000001; 5) \rightarrow (1001000001; 0)$
  $\rightsquigarrow (1001000000; 0) \rightarrow (1001000000; 1)$
  $\Rightarrow (1000000000; 1) \rightarrow (1000000000; 0)$
  $\rightarrow (1000000000; 4) \Rightarrow (0000000000; 4)$
  $\rightarrow (0000000000; 0) \rightsquigarrow (0000000001; 0)$
  $\rightarrow (0000000001; 5) \rightarrow (0000000001; 4)$.
- Case 2.1: $u = (1001100100; 1)$ and $v = (0000000001; 4)$, the shortest path between u and v can be constructed as follows:
  $(1001110010; 1) \Rightarrow (1000110010; 1)$
  $\rightarrow (1000110010; 0) \rightarrow (1000110010; 4)$
  $\Rightarrow (0000110010; 4) \rightarrow (0000110010; 0)$
  $\rightsquigarrow (0000110011; 0) \rightarrow (0000110011; 1)$
  $\Rightarrow (0000110001; 1) \rightarrow (0000110001; 0)$
  $\rightarrow (0000110001; 5) \Rightarrow (0000010001; 5)$
  $\rightarrow (0000010001; 4) \Rightarrow (0000000001; 4)$.
- Case 2.2: $u = (1001100101; 1)$ and $v = (0000000000; 4)$, the shortest path between u and v can be constructed as follows:
  $(1001100101; 1) \rightarrow (1001100101; 2)$
  $\Rightarrow (1001100001; 2) \rightarrow (1001100001; 1)$
  $\rightarrow (1001100001; 0) \rightarrow (1001100001; 5)$
  $\Rightarrow (1001000001; 5) \rightarrow (1001000001; 0)$
  $\rightsquigarrow (1001000000; 0) \rightarrow (1001000000; 1)$
  $\Rightarrow (1000000000; 1) \rightarrow (1000000000; 0)$
  $\rightarrow (1000000000; 4) \Rightarrow (0000000000; 4)$.

The diameter of graph Γ, denoted by $diam(\Gamma)$, is defined as the maximum distance for all pairs of distinct nodes in Γ. Diameter is an important topological property of an interconnection network, because it can be used to estimate the communication latency of the network. Low diameter is one of the desirable proper-ties of an interconnection network.

Theorem 7. For $ExCCC(s, t)$, we have:

$$diam(ExCCC(s,t)) = 2(s + t + 1) + \lfloor (s + 1)/2 \rfloor + \lceil (t/2) \rceil.$$

Proof. According to the shortest path procedure, the longest path can be constructed between $(1^{s+t}0; \lceil s/2 \rceil)$ and $(0^{s+t+1}; \lceil t/2 \rceil)$. This path can be constructed as follows:

$(1^{s+t}0; \lceil s/2 \rceil) \xrightarrow{path(1^{s+1},\lceil s/2 \rceil,0)} (0^s 1^{1+t}; 0) \xrightarrow{path(1^{1+t},0,0)}$
$(0^{s+t+1}; 0) \xrightarrow{path(0^{s+1},0,\lceil t/2 \rceil)} (0^{s+t+1}; \lceil t/2 \rceil)$.

The length of this path is:

$$\begin{aligned} d =& 2s + 1 + \lfloor (s-1)/2 \rfloor + 2(t+1) + \lceil t/2 \rceil \\ =& 2(s+t+1) + \lfloor (s-1)/2 \rfloor + \lceil t/2 \rceil + 1. \end{aligned}$$

It is obvious that $\lfloor (s-1)/2 \rfloor + 1 = \lfloor (s+1)/2 \rfloor$, then we have

$$d = 2(s+t+1) + \lfloor (s+1)/2 \rfloor + \lceil t/2 \rceil. \blacksquare$$

Since the number of nodes in $ExCCC(s,t)$ is $2^{s+t}(s+t+2)$, then we have the diameter of $ExCCC$ is $O(\log N)$, where N is the number of nodes in $ExCCC$.

***Example* 3**. According to the Theorem 6, we can get $diam(ExCCC(4,5)) = 25$, which equals to the length of the shortest path between nodes $(1111111110;2)$ and $(0000000000;3)$. The shortest path is:

$(1111111110;2) \Rightarrow (1101111110;2) \rightarrow (1101111110;1) \Rightarrow$
$(1100111110;1) \rightarrow (1100111110;2) \rightarrow (1100111110;3) \Rightarrow$
$(1000111110;3) \rightarrow (1000111110;4) \Rightarrow (0000111110;4) \rightarrow$
$(0000111110;0) \rightsquigarrow (0000111111;0) \rightarrow (0000111111;1) \Rightarrow$
$(0000111101;1) \rightarrow (0000111101;2) \Rightarrow (0000111001;2) \rightarrow$
$(0000111001;3) \Rightarrow (0000110001;3) \rightarrow (0000110001;4) \Rightarrow$
$(0000100001;4) \rightarrow (0000100001;5) \Rightarrow (0000000001;5) \rightarrow$
$(0000000001;0) \rightsquigarrow (0000000000;0) \rightarrow (0000000000;1) \rightarrow$
$(0000000000;2) \rightarrow (0000000000;3)$.

3.4 Comparison with other *CCC* type networks

In this section, we give some comparisons of $ExCCC$ and other CCC-type networks.

Theorem 8. The number of nodes and edges in $ExCCC(s,t)$ will be about half of that in CCC_{s+t+1} as s and t are increased.

Proof. An $(s+t+1)$-dimension *Cube-Connected Cycles*, denoted CCC_{s+t+1}, has $(s+t+1)2^{s+t+1}$ nodes and $3*(s+t+1)2^{s+t}$ edges.

- Case 1. $s = 1$.
  The $ExCCC(1,t)$ has $2^{t+1}(t+3)$ nodes and $2^t(3t+7)$ edges, then we have
  $$\frac{2^{t+1}(t+3)}{2^{t+2}(t+2)} = \frac{t+3}{2(t+2)} = \frac{1}{2} + \frac{1}{2(t+2)}$$
  and
  $$\frac{2^t(3t+7)}{3*2^{t+1}(t+2)} = \frac{3t+7}{6(t+2)} = \frac{1}{2} + \frac{1}{6(t+2)}$$
  Both the two equations approach $1/2$ as $t \rightarrow +\infty$.
- Case 2. $t \geq s \geq 2$.
  The $ExCCC(s,t)$ has $2^{s+t}(s+t+2)$ nodes and $3*2^{s+t-1}(s+t+2)$ edges, then we have
  $$\frac{(s+t+2)2^{s+t}}{(s+t+1)2^{s+t+1}} = \frac{s+t+2}{2*(s+t+1)} = \frac{1}{2} + \frac{1}{2(s+t+1)}$$

It approaches $1/2$ as $s \rightarrow +\infty$ and/or $t \rightarrow +\infty$.

Fig. 15 shows the comparison of the topological properties of the CCC, $FCCC$ and $ExCCC$ in terms of the number of nodes and the number of edges. As shown in Fig. 15, the number of nodes and edges in CCC and $FCCC$ are almost equal, and the numbers of nodes and edges in $ExCCC$ are about half of that in CCC and $FCCC$.

In [31], I. Friš et al. calculated the diameter of CCC_n, and they deduced the following lemma.

Lemma 1.
$diam(CCC_3) = 6$
$diam(CCC_n) = 2n + \lfloor n/2 \rfloor - 2 \ \ if\ n \geq 4$

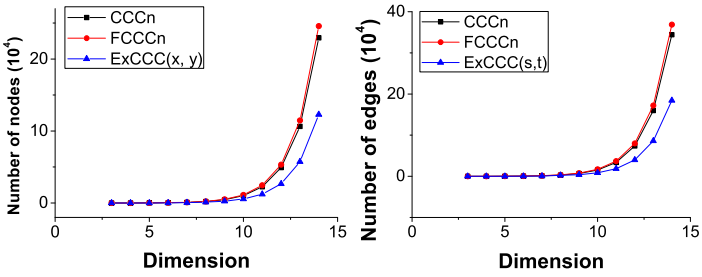

Fig. 15: Comparing the numbers of nodes and edges of the CCC_n, $FCCC_n$ and $ExCCC(s,t)$.

Theorem 9. For the $s+t+1$ dimension $ExCCC(s,t)$ and CCC_{s+t+1}, we have:

$$diam(ExCCC(s,t)) = diam(CCC_{s+t+1}) + 2 \ or\ 3.$$

Proof.

- Case 1. $s = t = 1$.
  We have $diam(ExCCC(1,1)) = 8$ and $diam(CCC_3) = 6$. That is $diam(ExCCC(1,1)) = diam(CCC_3) + 2$.
- Case 2. $s + t > 2$.
  We analyze the diameters of $ExCCC(s,t)$ and CCC_{s+t+1} according to the following four cases.
  - Case 2.1. If s is odd, t is odd, then $s+t+1$ is odd, we can get
    $$\begin{aligned} diam(ExCCC(s,t)) =& 2(s+t+1) + \lfloor (s+1)/2 \rfloor \\ &+ \lceil t/2 \rceil \\ =& 2(s+t+1) + (s+1)/2 \\ &+ (t+1)/2 \\ =& 2(s+t+1) + (s+t)/2 + 1; \end{aligned}$$
    $$\begin{aligned} diam(CCC_{s+t+1}) =& 2(s+t+1) + \lfloor (s+t+1)/2 \rfloor - 2 \\ =& 2(s+t+1) + (s+t)/2 - 2. \end{aligned}$$
    Thus, $diam(ExCCC(s,t)) = diam(CCC_{s+t+1}) + 3$.
  - Case 2.2. If s is odd, t is even, then $s+t+1$ is even, we can get
    $$\begin{aligned} diam(ExCCC(s,t)) =& 2(s+t+1) + \lfloor (s+1)/2 \rfloor \\ &+ \lceil t/2 \rceil \\ =& 2(s+t+1) + (s+1)/2 + t/2 \\ =& 2(s+t+1) + (s+t+1)/2; \end{aligned}$$
    $$\begin{aligned} diam(CCC_{s+t+1}) =& 2(s+t+1) + \lfloor (s+t+1)/2 \rfloor - 2 \\ =& 2(s+t+1) + (s+t+1)/2 - 2. \end{aligned}$$
    Thus, $diam(ExCCC(s,t)) = diam(CCC_{s+t+1}) + 2$.
  - Case 2.3. If s is even, t is odd, then s+t+1 is odd, we can get
    $$\begin{aligned} diam(ExCCC(s,t)) =& 2(s+t+1) + \lfloor (s+1)/2 \rfloor \\ &+ \lceil t/2 \rceil \\ =& 2(s+t+1) + s/2 + (t+1)/2 \\ =& 2(s+t+1) + (s+t+1)/2; \end{aligned}$$
    $$\begin{aligned} diam(CCC_{s+t+1}) =& 2(s+t+1) + \lfloor (s+t+1)/2 \rfloor - 2 \\ =& 2(s+t+1) + (s+t+1)/2 - 2. \end{aligned}$$

TABLE 1: Comparison of *CCC-type* networks.

Topology	Number of Nodes	Number of Edges	Degree	Diameter
CCC_n	$n2^n$	$3n2^{n-1}$	3	6, if $n = 3$. $2n + \lfloor n/2 \rfloor - 2$, if $n \geq 4$
$k\text{-}ECCC_r$	2^k	$2^{k-r}(2^r + k - r)$	4	$2^r + k - r - 2$
$FCCC_n$	$(n+1)2^n$	$3(n+1)2^{n-1}$	3	$2n - 3 + n \bmod 2$
$ExCCC(1,t)$	$(t+3)2^{t+1}$	$(3t+7)2^t$	2 or 3	$2t + \lceil t/2 \rceil + 5$
$ExCCC(s,t)\ t \geq s > 1$	$(s+t+2)2^{s+t}$	$3(s+t+2)2^{s+t-1}$	3	$2(s+t+1) + \lfloor (s+1)/2 \rfloor + \lceil t/2 \rceil$

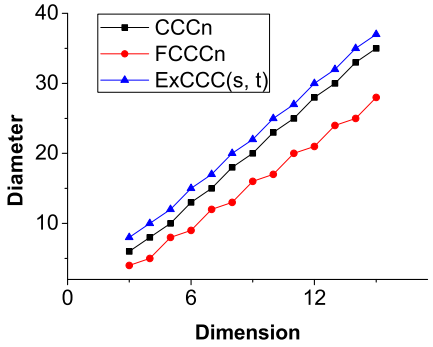

Fig. 16: Comparing the diameters of the CCC_n, $FCCC_n$ and $ExCCC(s,t)$.

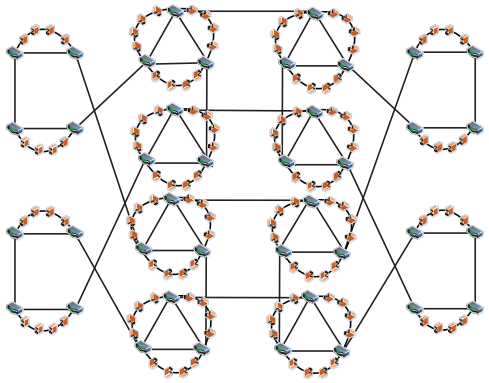

Fig. 17: $ExCCC\text{-}DCN(1,2)$.

Thus, $diam(ExCCC(s,t)) = diam(CCC_{s+t+1}) + 2$.

– Case 2.4. If s is even, t is even, then $s+t+1$ is odd, we can get

$$\begin{aligned} diam(ExCCC(s,t)) &= 2(s+t+1) + \lfloor (s+1)/2 \rfloor + \lceil t/2 \rceil \\ &= 2(s+t+1) + s/2 + t/2 \\ &= 2(s+t+1) + (s+t)/2; \end{aligned}$$

$$\begin{aligned} diam(CCC_{s+t+1}) &= 2(s+t+1) + \lfloor (s+t+1)/2 \rfloor - 2 \\ &= 2(s+t+1) + (s+t)/2 - 2. \end{aligned}$$

Thus, $diam(ExCCC(s,t)) = diam(CCC_{s+t+1}) + 2$.■

Fig. 16 shows the comparison of the *CCC*, *FCCC* and *ExCCC* in terms of diameter. As shown in Fig. 16, the diameter of *ExCCC* is slightly larger than that of the other two kinds of networks.

Table 1 presents a comparison among *CCC-type* networks in terms of the total number of nodes and edges, node degree and diameter, which have significant impacts on the performance of a parallel computing system.

4 DATA CENTER NETWORK *EXCCC-DCN*

Based on the *ExCCC* architecture, we build a new class of data center network models named *ExCCC-DCN*. Our models are suitable for building large-scale data center networks. The number of ports per switch is a small constant, and the servers need only two ports. The total number of switches is $O(N/\log N)$, and the diameter is $O(\log N)$, where N is the number of servers deployed in the data center. Furthermore, it also tends to produce better scalability, low cost and low power consumption.

4.1 Definiton of *ExCCC-DCN*

Let M and N denote the numbers of switches and servers in the data center network model, respectively. The *ExCCC-DCN* model can be constructed as follows.

- Case 1. $ExCCC - DCN(1,t),\ t \geq 2$.
  The $ExCCC(1,t)$ constructs a subnetwork of switches. Then, we add a parallel edge for every *circle-edge*, and $t+2$ servers are linked by each parallel edge.
  Thus, we have:
  $$M = 2^{t+1}(t+3),$$
  $$N = 2^{t+1}(t+2)^2.$$
- Case 2. $ExCCC - DCN(s,t),\ t \geq s \geq 2$
  Same as Case 1, $ExCCC(s,t)$ constructs the subnetwork of switches. We add a parallel edge for every *circle-edge*. Then, $s+t+2$ servers are linked by each parallel edge. Thus, we have:
  $$M = 2^{s+t}(s+t+2),$$
  $$N = 2^{s+t}(s+t+2)^2.$$

According to the definition of *ExCCC-DCN*, we can calculate the number of edges in *ExCCC-DCN*. Then we have:

$$|E(ExCCC-DCN(s,t)| = 2^{t+1}(t+2)(t+3)+2^t(3t+7),\ when\ s=1;$$

$$|E(ExCCC-DCN(s,t)| = 2^{s+t}(s+t+2)(s+t+3)+3*2^{s+t-1}(s+t+2),$$

when $s > 1$.

In the *ExCCC-DCN*, the number of ports per switch is a small constant, the servers need only two ports, and the total number of switches is linear with the server size. In order to achieve low cost and low energy consumption, we connect the servers by a chain. If we replace the chain of servers by using a network of switches and connect each server to only one switch, this will increase the cost and power consumption due to the added network devices. Furthermore, this will make the architecture more complex, and the routing algorithm is difficult to be established. The overall cost and power consumption would be higher than that of other four

TABLE 2: The characteristics of *Fat-Tree*, *DCell*, *BCube*, *FiConn* and *ExCCC-DCN*.

	Fat-Tree	*DCell*	*BCube*	*FiConn*	*ExCCC-DCN*
Diameter	$2\log_2 N$	$\leq 2^{k+1}-1$	$k+1$	$\leq 2^{k+1}-1$	$< 3\log_2 N$
Bisection Width	$N/2$	$> N/(4\log_n N)$	$N/2$	$> N/(4\times 2^k)$	$> N/2(\log_2 N)^2$
Scalability limited by server ports (number of ports)	No(≤ 2)	Yes($k+1$)	Yes($k+1$)	No(≤ 2)	No(2)
Scalability limited by switch ports (number of ports)	Yes (n)	No(n)	No(n)	No(n)	No(5)
Number of switches	$5N/n$	N/n	$(k+1)N/n$	N/n	$N/\log N$

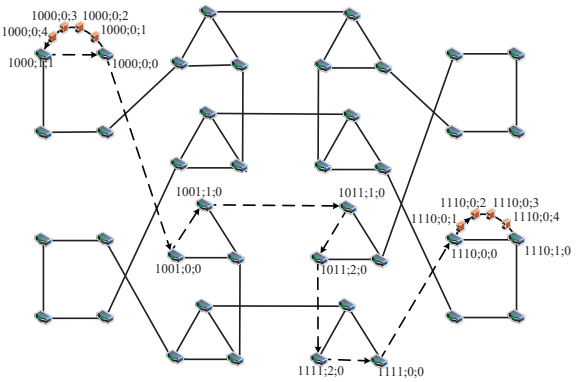

Fig. 18: Addressing and routing in *ExCCC-DCN*(1, 2).

typical data center network structures. Fig. 17 shows the *ExCCC*−*DCN*(1, 2).

Node addressing in *ExCCC-DCN* is as follows.

1) The switch node addressing is the same to that in *ExCCC*, but it includes an extra dimension that is assigned a value of 0. For example, $(1000;1)$ in *ExCCC*$(1,2)$ becomes $(1000;1;0)$ in *ExCCC-DCN*$(1,2)$.
2) For each parallel edge, we give a direction from the switch $(x_s...x_1y_t...y_1c;p;0)$ to the switch $(x_s...x_1y_t...y_1c;p+1;0)$. Then, the server addressing between two switches is determined by the address of its preceding switch node P. The preceding two dimensions of address are the same as those of P. The final dimension begins from 1 and increases by 1 for each additional server.

For example, the addresses of servers between $(1000;0;0)$ and $(1000;1;0)$ are $(1000;0;1)$, $(1000;0;2)$, $(1000;0;3)$ and $(1000;0;4)$ as shown in Fig. 18.

Intuitively, routing scheme is done as follows. Assume that the source node is server A and the destination node is server B. Server A sends a message to its nearest switch C, then the message finds its way to the switch D which is the nearest one to the destination B. Finally, D forwards the message to the server B along the parallel edge.

***Example* 4**. In Fig. 18, routing from $(1000;0;3)$ to $(1110;0;2)$ is denoted by dashed arrows:

$(1000;0;3)\rightarrow(1000;0;4)\rightarrow(1000;1;0)\rightarrow(1000;0;0)\rightarrow(1001;0;0)\rightarrow((1001;1;0)\rightarrow(1011;1;0)\rightarrow(1011;2;0)\rightarrow(1111;2;0)\rightarrow(1111;0;0)\rightarrow(1110;0;0)\rightarrow(1110;0;1)\rightarrow(1110;0;2)$.

Based on the routing scheme, we can calculate the diameter of the *ExCCC-DCN*. Let *diam* denote the diameter of *ExCCC-DCN*(s,t) network and d denote the diameter of the *ExCCC*(s,t) network. Then we have:

- Case 1. *ExCCC-DCN*$(1,t)$, $t\geq 2$.

$$\begin{aligned} diam =& d+t+1+(t+1)\ mod\ 2\\ =& 2(t+2)+1+\lceil\frac{t}{2}\rceil+t+(t+1)\ mod\ 2\\ =& 3(t+2)+\lceil\frac{t}{2}\rceil+(t+1)\ mod\ 2. \end{aligned}$$

- Case 2. *ExCCC-DCN*(s,t), $t\geq s\geq 2$.

$$\begin{aligned} diam =& d+s+t+1+(s+t+1)\ mod\ 2\\ =& 2(s+t+1)+\lfloor\frac{s+1}{2}\rfloor+\lceil\frac{t}{2}\rceil+s\\ &+t+1+(s+t+1)\ mod\ 2\\ =& 3(s+t+1)+\lfloor\frac{s+1}{2}\rfloor+\lceil\frac{t}{2}\rceil\\ &+(s+t+1)\ mod\ 2. \end{aligned}$$

It is clear that the diameter of *ExCCC-DCN*is $O(\log N)$, where N is the number of servers in the network.

4.2 Comparision with other DCN models

For simplicity, we only consider *ExCCC-DCN*(s,t), where $t\geq s\geq 2$. Table 2 compares *ExCCC-DCN* against other four important data center networks including *Fat-Tree*, *DCell*, *BCube* and *FiConn* in terms of diameter, bisection width, scalability, and number of switches, where parameter N represents the number of servers in a data center network, n indicates the number of switch ports. Because *FiConn*, *DCell* and *BCube* are all recursively defined structures, the levels of the network structure are defined as k. All the characteristics have significant impacts on the performance of a data center network structure.

4.2.1 Diameter

Diameter is normally used to evaluate the communication latency of a network. Theoretically, communication delay grows with the increase of the network diameter. Therefore, low diameter is one of the desirable properties of an interconnection network structure. The diameter of *Fat-Tree* is $2\log_2 N$. The exact diameters of *DCell* and *FiConn* are still unknown. However, the upper bounds of the diameters of the two structures are both $2^{k+1}-1$. The diameter of *BCube* is $k+1$. As analyzed above, the diameter of *ExCCC-DCN* is only $O(\log N)$. Through further calculations, we can determine that the diameter of *ExCCC-DCN* is less than $3\log_2 N$.

TABLE 3: Price and Power Consumption of Switch and NICs.

	Product	Ports	Price($)	Power(W)
Switch	D-Link DGS-1008D	8	40	3
NIC	Intel EXPI9400PT	1	96	5
	Intel EXPI9402PT	2	148	7
	Intel EXPI9404PT	4	427	10

4.2.2 Scalability

A typical way to expand a data center is adding more new components including servers and switches to the system rather than replacing old ones. Therefore, the scalability of the data center network is crucial to the overall system performance.

All the five data center network models shown in Table 2 can be expanded to a very large scale. However, the scalability of *DCell* and *BCube* are limited by the server ports, and the scalability of *Fat-Tree* is limited by the switch ports. This means, if some new switches and servers are added into the three data center network models, some additional ports have to be added into the original switches or servers to establish the link connections. In contrast to *DCell*, *BCube* and *Fat-Tree*, *ExCCC-DCN* and *FiConn* can be expanded without adding additional server ports or switch ports.

FiConn is a kind of recursively defined network structure, which means that the k-level structure is constructed by busing several (k-1)-level structures. This feature results in a rapidly increase of the system scale when the level is increased from level k to level (k+1). For example, when we use 8-port switches to construct a *FiConn*, the number of servers in $FiConn_2$ is 440, and the number of servers in $FiConn_3$ will be increased to 24640. In order to solve this problem, incomplete *FiConn* was proposed. An incomplete $FiConn_k$ is constructed by employing a small number of complete $FiConn_{k-1}$, and the $FiConn_{k-1}$ are fully connected. However, the bisection widths of some Incomplete *FiConn*s are extremely low. The reason behind this will be explained in the next subsection.

The scalability of *ExCCC-DCN* is much better than *FiConn*. An incomplete *ExCCC-DCN* network can be constructed firstly. This means we do not add parallel edges which contain servers for some circle edges. When the *ExCCC-DCN* is required to expand, we can dynamically add those parallel edges into the incomplete *ExCCC-DCN* network to increase the system scale. Additionally, the routing algorithm of the incomplete *ExCCC-DCN* network is the same to the routing algorithm developed in section 4.1.

4.2.3 Bisection width

Large bisection width implies a high network capacity and a more resilient structure against failures. As shown in Table 2, the bisection widths of *Fat-Tree* and *BCube* are both $N/2$. The exact bisection widths of *DCell* and *FiConn* are unknown. However, the lower bounds of *DCell* and *FiConn* are $N/(4\log_n N)$ and $N/(4\times 2^k)$, respectively. The bisection of $ExCCC\text{-}DCN(s,t)$ is the same as that of $ExCCC(s,t)$. It is 2^{s+t-1}. Since the number of servers in *ExCCC-DCN* is $N = 2^{s+t}(s+t+2)^2$, then the bisection of *ExCCC-DCN* is $O(N/(\log N)^2)$. Through further calculations, we can determine that the bisection width of *ExCCC-DCN* is larger than $N/2(\log_2 N)^2$.

As shown in Table 2, the bisection widths of *FiConn* and *ExCCC-DCN* are lower than that of other three data center network models. This is mainly because *FatTree*, *BCube* and *DCell* use more switches and links to construct the data center structures. Unfortunately, this approach will increase the cost and energy consumption of building the corresponding data center networks.

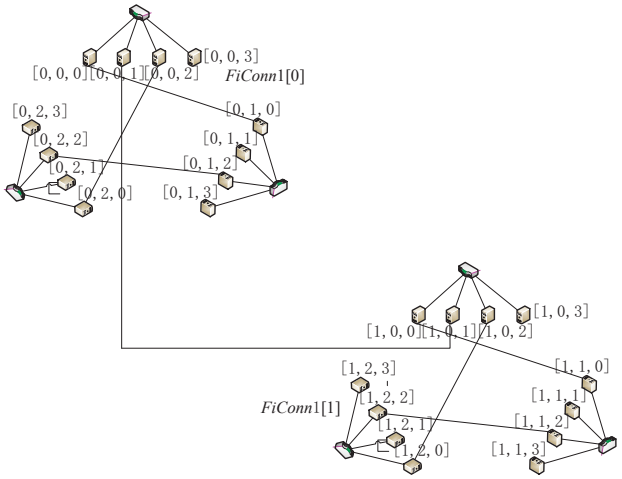

Fig. 19: Incomplete $FiConn_2$

As aforementioned, both *ExCCC-DCN* and *FiConn* can scale without adding additional server ports or switch ports. However, the bisection widths of some incomplete *FiConn* models are extremely low. For example, if there are only two $FiConn_{k-1}$s in an incomplete $FiConn_k$, the two $FiConn_{k-1}$s will be connected through a single link. The bisection width of this structure is 1. When we use 4-port switches to construct a *FiConn*, an incomplete $FiConn_2$ is shown in Fig. 19. The bisection width of this structure is only 1. In order to solve this problem, some additional links have to be added into the incomplete *FiConn* to obtain a higher bisection width. This implies that the connection model of the incomplete *FiConn* is different to that of complete one. Then, the routing algorithm for the incomplete *FiConn* must be changed.

4.2.4 Cost and energy consumption

This section compares the cost and energy consumption of *ExCCC-DCN* against *Fat-Tree*, *DCell*, *BCube* and *FiConn*. In order to explore the impacts of different system scales on the cost and energy consumption, we construct two different scales of data centers containing 2048 servers and 24640 servers respectively, by using the five different network models. Since the number of servers in the five different data center networks is identical (2048 servers or 24640 servers), the total cost and energy consumption of the servers are the same. Therefore, we only analyze and evaluate the cost and energy consumption incurred by the switches and NICs in the data centers. Table 3 shows the price and power consumption of switch and NICs used in the data centers [10], [32]. Because the data center networks must be laid out with wires and cables, the number of wires used in different networks is also evaluated.

Table 4 summarizes the cost, power consumption and the number of wires incurred by the five different network structures for a data center containing 2048 servers. Table 5 describes the same statistics when the data center is expanded from 2048 servers to 24640 severs. Both the two tables demonstrate that the costs and power consumptions of *DCell* and *BCube* are much higher than that of *Fat-Tree*, *FiConn* and *ExCCC-DCN*. Additionally, the cost of *Fat-Tree* is the lowest one among the above three network structures, and the power consumption of *Fat-Tree* is higher than

TABLE 4: Cost, power and wiring comparison of a data center with 2048 servers.

	Cost(k)			Power(kw)			Number of wires
	switch	NIC	total	switch	NIC	total	
FiConn	10.2	303.1	313.3	0.8	14.3	15.1	2841
Fat-Tree	92.2	196.6	288.8	6.9	10.2	17.1	10240
BCube	51.2	874.5	925.7	3.8	20.5	24.3	8192
DCell	10.2	874.5	884.7	0.8	20.5	20.3	3538
ExCCC-DCN	10.2	303.1	313.3	0.8	14.3	15.1	2672

TABLE 5: Cost, power and wiring comparison of a data center with 24640 servers.

	Cost(k)			Power(kw)			Number of wires
	switch	NIC	total	switch	NIC	total	
FiConn	123.2	3646.7	3769.9	9.2	172.5	181.7	35420
Fat-Tree	843.2	2365.4	3208.6	63.2	123.2	186.4	95136
BCube	642.6	10521.3	11163.9	48.2	246.4	294.6	123072
DCell	123.2	10521.3	10644.5	9.2	246.4	255.6	52570
ExCCC-DCN	98.6	3646.7	3745.3	7.4	172.5	179.9	30800

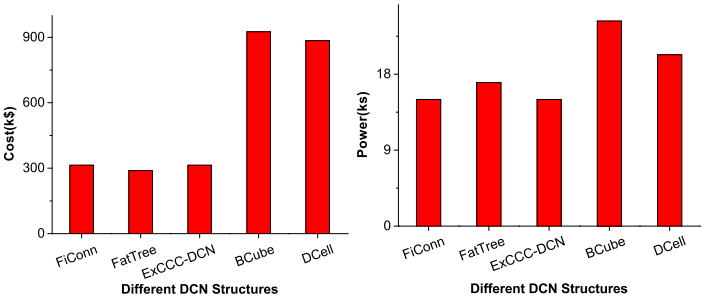

Fig. 20: Cost and power comparison of a data center with 2048 servers.

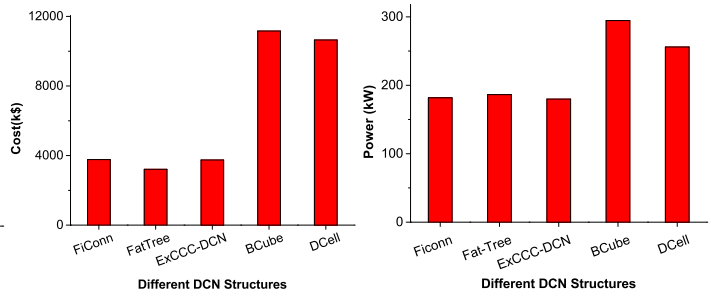

Fig. 21: Cost and power comparison of a data center with 24640 servers.

that of *FiConn* and *ExCCC-DCN*. However, the scalability of *Fat-Tree* is limited by the number of switch ports. Therefore, it is not suitable for constructing large-scale data centers. Neither *BCube* nor *DCell* is suitable for large-scale data centers from the perspective of cost and power consumption in terms of the statistics in Table 4 and Table 5. Fig. 20 and Fig. 21 confirm this conclusion. Although the prices vary in the market, they do not affect the results summarized in Table 4, Table 5, Fig. 20 and Fig. 21. When different models are used to build data center networks, if the number of servers in the data center networks is identified, then the number of switches and the number of NICs are both determined. The prices of the switch and the NICs will affect the total cost of the data center network, but not affect the comparison results in terms of cost. This is because the total costs of the data center networks change along with the changes of the prices of the switch and NICs. The above analysis indicate that both *FiConn* and *ExCCC-DCN* would be suitable for constructing large-scale data centers.

Table 4 and Table 5 also list the number of wires when we use the five different network structures to construct data centers containing different number of servers (2048 servers or 20460 servers). Table 4 shows that the number of wires used by *Fat-Tree* is the smallest one, and *ExCCC-DCN* takes the second place. However, when the data center is expanded from 2048 servers to 20460 servers, *ExCCC-DCN* takes the minimal number of wires. It is only one third of the wires employed by *Fat-Tree*. Additionally, the wires used by *ExCCC-DCN* is still less than that of *FiConn*. From this point of view, *ExCCC-DCN* is a better candidate than *FiConn* due to the decreased complexity, when constructing large-scale data centers.

Now, we consider the number of switches used in *FiConn* and *ExCCC-DCN* when building large-scale data centers containing roughly the same number of servers. Switches with 48 ports are employed to construct the above two network structures. Due to the requirement of *ExCCC-DCN*, a 48-port switch is virtualized to nine 5-port switches to construct *ExCCC-DCN*. For the *ExCCC-DCN*(s,t), the number of servers is $2^{s+t}(s+t+2)^2$, and the total number of switch ports is $5\times 2^{s+t}(s+t+2)$. Then, we need $M_1 = 2^{s+t}(s+t+2)/9$ 48-port switches to construct the network structure. If we use *FiConn* to construct such a data center, the number of 48-port switches is $M_2 = 2^{s+t}(s+t+2)^2/48$. Then we have $M_1/M_2 = 16/(3s+3t+6)$. If we construct large-scale data centers containing tens of thousands of servers, the value of $s+t$ must be big enough. For example, if the number of servers is 25600, then $s+t=8$. Thus, $M_1/M_2 = 8/15$. This means that the number of switches in *ExCCC-DCN* is about half of that in *FiConn*.

4.2.5 Throughput

In order to evaluate the throughput of five different network structures, we design a flow-level simulator based on the approach

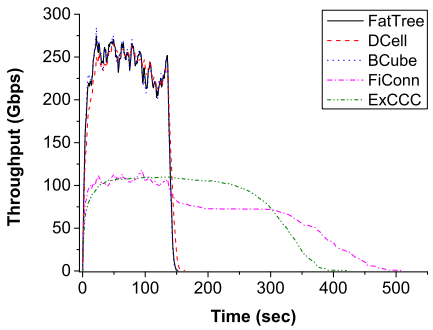

Fig. 22: Throughput Comparison of DCNs.

proposed in [33]. The simulator employs a discrete time model and considers the data center network as a graph. This time model updates the rates of all active flows every microsecond. The capacity of each edge in the graph is customized. The simulator formalizes the flows with four tuples including source host, destination host, start time and flow size. The Maximum Segment Size (MSS) is determined as 1500 Byte. Moreover, the simulator estimates the delay caused by forwarding, queuing, transmission and processing by assigning a fixed Round Trip Time (*RTT*) to each flow. In current data center networks, the intra-rack *RTT* is approximately 100 μs [34]. Therefore, if a specific flow goes through a number of switches, the *RTT* of this flow is the number of switches times 100 μs. Consequently, the simulator performs the flow scheduling for a specific flow every *RTT* time. If a flow is scheduled, it will be transferred to an inactive state during the next *RTT* time. Additionally, the switches in the simulation are assumed to be capable of using all ports at once.

We build five different network structures including *Fat-Tree*, *DCell*, *BCube*, *FiConn* and *ExCCC-DCN* in the simulator to evaluate the throughput. In the simulation, we use 8-port switches to construct *DCell*, *BCube*, *FiConn* and *ExCCC-DCN*, while each switch used in *Fat-Tree* has 26 ports. All the structures contain 4096 servers. The data rates of the links in the experiment are all determined as 1Gbps. Benson et al. [35] conduct an empirical study on the network traffic pattern across ten different data centers. They collect and analyze the data center network topologies, flow-level and packet-level statistics. They report that 80% of flow sizes are smaller than 10KB and the top 10% of large flows cover most of bytes in data centers. Therefore, we generate a synthetic flow workload according to the characteristics summarized in [35]. The workload contains 80000 flows with a total size of 4TB. The maximum flow size is 1GB while the minimum size is 1KB. Furthermore, the source and destination hosts ranging from 0 to 4096 for each flow are chosen randomly.

The experimental results of throughput are demonstrated in Fig. 22. It shows that the performance behaviour of *FatTree*, *BCube* and *DCell* are comparable. They achieve about 250 Gbps throughput and all the data transmissions are completed within 150 seconds. The reason behind this is that the above three network structures employ a large number of switches and abundant links to connect servers, thus achieving very high throughput. Fig. 22 depicts that the performance of *ExCCC-DCN* outperforms that of *FiConn* to a certain degree. Although the highest throughput of the *ExCCC-DCN* and *FiConn* are both about 110Gbps, *ExCCC-DCN* takes less time to complete the data transmission.

5 Conclusion

In this paper, we present a new type of *Exchanged Cube-Connected Cycle (ExCCC)* network. Based on the proposed *ExCCC*, we introduce a data center network structure named *ExCCC-DCN* by leveraging the advantages of *ExCCC*. In order to explore the performance behaviour of *ExCCC-DCN*, we compare *ExCCC-DCN* against other four typical data center network structures including *Fat-Tree*, *DCell*, *BCube* and *FiConn*. According to the analysis in this paper, the scalability of *DCell* and *BCube* are limited by server ports, while the scalability of *Fat-Tree* is limited by switch ports. However, both the *ExCCC-DCN* and *FiConn* can scale without adding additional server ports or switch ports. In contrast to *ExCCC-DCN*, the number of servers in *FiConn* is increased rapidly when the level is increased from k to k+1. This feature brings challenging to designing incrementally scalable data centers when using *FiConn*. The network structure of *ExCCC-DCN* does not have this problem. The additional servers can be incrementally and dynamically added to the system.

The cost, power consumption and number of wires are evaluated when using the five different network structures to build data centers. The experimental results demonstrate that *ExCCC-DCN* is the best candidate for building large-scale data centers. A simulator is also designed to evaluate the throughput of the five network structures. The experimental results demonstrate that the performance of *ExCCC-DCN* outperforms that of *FiConn* to a certain degree. Although the highest throughput of the *ExCCC-DCN* and *FiConn* are both about 110Gbps, *ExCCC-DCN* takes less time to complete the data transmission. Additionally, in contrast to *Fat-Tree*, *DCell*, *BCube* and *FiConn*, *ExCCC-DCN* strikes a very good balance among low cost, low energy consumption, high network throughput and high scalability. It achieves these goals simultaneously. Therefore, we believe that *ExCCC-DCN* is a highly scalable, cost-effective and energy-efficient data center network structure.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions. This work is supported by the National Natural Science Foundation (NSF) of China under Grant (No. 61572232, and No. 61272073), the key program of Natural Science Foundation of Guangdong Province (No.S2013020012865), and the Fundamental Research Funds for the Central Universities.

References

[1] T. Hoff, "Google architecture," *July2007 [Online]. Available: http://highscalability.com/google-architecture*, 2007.
[2] L. Rabbe, "Powering the yahoo! network," *Nov*, 2006.
[3] Y. Zhang and N. Ansari, "On architecture design, congestion notification, tcp incast and power consumption in data centers," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 1, pp. 39–64, 2013.
[4] A. Carter, "Do it green: Media interview with michael manos," *Dec. 2007 [Online]. Available: htt p://edge. technet. com/Media/Doing-ITGreen*, 2007.
[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.
[6] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[7] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.
[9] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.
[10] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.
[11] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "Ficonn: Using backup port for server interconnection in data centers," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2276–2285.
[12] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.
[13] K. Wu, J. Xiao, and L. M. Ni, "Rethinking the architecture design of data center networks," *Frontiers of Computer Science*, vol. 6, no. 5, pp. 596–603, 2012.
[14] H. Xu, C. Feng, and B. Li, "Temperature aware workload managementin geo-distributed data centers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 6, pp. 1743–1753, 2015.
[15] L. Yu, T. Jiang, and Y. Cao, "Energy cost minimization for distributed internet data centers in smart microgrids considering power outages," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 120–130, 2015.
[16] A. M. Al-Qawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *Computers, IEEE Transactions on*, vol. 64, no. 2, pp. 477–491, 2015.
[17] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *Computers, IEEE Transactions on*, vol. 100, no. 4, pp. 323–333, 1984.
[18] A. H. Esfahanian, L. M. Ni, and B. E. Sagan, "The twisted n-cube with application to multiprocessing," *Computers, IEEE Transactions on*, vol. 40, no. 1, pp. 88–93, 1991.
[19] K. Efe, "The crossed cube architecture for parallel computation," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 3, no. 5, pp. 513–524, 1992.
[20] A. El-Amawy and S. Latifi, "Properties and performance of folded hypercubes," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 2, no. 1, pp. 31–42, 1991.
[21] P. K. Loh, W. J. Hsu, and Y. Pan, "The exchanged hypercube," *IEEE Transactions on Parallel & Distributed Systems*, no. 9, pp. 866–874, 2005.
[22] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM*, vol. 24, no. 5, pp. 300–309, 1981.
[23] M. Sebastian, P. N. Rao, and L. Jenkins, "Properties and performance of folded cube-connected cycles," *Journal of systems architecture*, vol. 44, no. 5, pp. 359–374, 1998.
[24] R. A. Ayoubi, Q. M. Malluhi, and M. A. Bayoumi, "The extended cube connected cycles: An efficient interconnection for massively parallel systems," *Computers, IEEE Transactions on*, vol. 45, no. 5, pp. 609–614, 1996.
[25] A. Singh, J. Ong, A. Agarwal *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *ACM SIGCOMM Computer Communication Review*. ACM, 2015, pp. 183–197.
[26] "Inside microsoft's $550 million mega data centers," http://www.informationweek.com/news/hardware/data_centers/showArticle.jhtml?articleID=208403723, 2009.
[27] B. Canney, "Ibm portable modular data center overview," http://www-05.ibm.com/se/news/events/datacenter/pdf/PMDC_Introducion_-_Brian_Canney.pdf, 2009.
[28] HP, "Hp performance optimized datacenter," ftp://ftp.hp.com/pub/c-products/servers/pod/north_america_pod_datasheet_041509.pdf, 2016.
[29] SGI, "Sgi ice cube," http://www.sgi.com/pdfs/4160.pdf, 2016.
[30] O. Sun, "Sun modular datacenter," http://www.sun.com/service/sunmd, 2016.
[31] I. Friš, I. Havel, and P. Liebl, "The diameter of the cube-connected cycles," *Information processing letters*, vol. 61, no. 3, pp. 157–160, 1997.
[32] ZOL, "Online," http://www.zol.com.cn, 2016.
[33] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI*, vol. 10, 2010, pp. 19–19.
[34] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2011.
[35] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 267–280.

Zhen Zhang received the BS and MS degrees in computer science from Jilin University, China, in 1999 and 2003, and PhD degree in College of Computer Science and Engineering from South China University of Technology, China, in 2011. He became a lecturer and an associate professor in 2003 and 2012, respectively, in the department of computer science at Jinan University. His research interests include graph theory, parallel and distributed processing and complex networks.

Yuhui Deng is a professor at the Computer Science Department of Jinan University. Before joining Jinan University, Dr. Yuhui Deng worked at EMC Corporation as a senior research scientist from 2008 to 2009. He worked as a research officer at Cranfield University in the United Kingdom from 2005 to 2008. He received his Ph.D. degree in computer science from Huazhong University of Science and Technology in 2004. His research interests cover green computing, cloud computing, information storage, computer architecture, performance evaluation, etc.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Next Generation Internet, Wireless Communications, Multimedia Systems, Information Security, Ubiquitous Computing, Modelling and Performance Engineering.

Junjie Xie received the B.E. degree in software engineering from Jinan University, Guangzhou, China, in 2013. He is currently a Research Student with the Department of Computer Science, Jinan University. His research interests include network interconnection, data center architecture, and cloud computing.

Shuqiang Huang received the PhD degree in College of Computer Science and Engineering from South China University of Technology, China, in 2009. He is currently an advanced engineer in the Educational Technology Center, Jinan University. His research interests include graph theory, parallel and distributed processing and wireless networks.