Wang X, Fan JX, Lin CK *et al.* BCDC: A high-performance, server-centric data center network. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(2): 400–416 Mar. 2018. DOI 10.1007/s11390-018-1826-3

BCDC: A High-Performance, Server-Centric Data Center Network

Xi Wang[1,2], *Member, CCF*, Jian-Xi Fan[1,*], *Member, CCF*, Cheng-Kuan Lin[1], *Member, CCF*
Jing-Ya Zhou[1], *Member, CCF*, and Zhao Liu[1]

[1] *School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2] *School of Software and Services Outsourcing, Suzhou Institute of Industrial Technology, Suzhou 215004, China*

E-mail: {wangxi0414, jxfan, cklin, jy_zhou, liuzhao}@suda.edu.cn

Received May 26, 2017; revised December 28, 2017.

Abstract The capability of the data center network largely decides the performance of cloud computing. However, the number of servers in the data center network becomes increasingly huge, because of the continuous growth of the application requirements. The performance improvement of cloud computing faces great challenges of how to connect a large number of servers in building a data center network with promising performance. Traditional tree-based data center networks have issues of bandwidth bottleneck, failure of single switch, etc. Recently proposed data center networks such as DCell, FiConn, and BCube, have larger bandwidth and better fault-tolerance with respect to traditional tree-based data center networks. Nonetheless, for DCell and FiConn, the fault-tolerant length of path between servers increases in case of failure of switches; BCube requires higher performance in switches when its scale is enlarged. Based on the above considerations, we propose a new server-centric data center network, called BCDC, based on crossed cube with excellent performance. Then, we study the connectivity of BCDC networks. Furthermore, we propose communication algorithms and fault-tolerant routing algorithm of BCDC networks. Moreover, we analyze the performance and time complexities of the proposed algorithms in BCDC networks. Our research will provide the basis for design and implementation of a new family of data center networks.

Keywords data center network, interconnection network, crossed cube, server-centric, fault-tolerant

1 Introduction

With the development of cloud computing such as on-line search, e-commerce, web gaming, on-line video, cloud storage, and infrastructure services, giant data center networks (DCNs) may operate hundreds of thousands of servers. Microsoft doubled the number of servers in its data centers with every 14 months, and this speed outstripped the Moore's law①. In particular, Microsoft ran more than a million servers in 2013[1]. Amazon Web services had 1.3 million servers in 2015 and will operate three million servers by the end of 2020[2]. Thus, we are faced by the challenge of interconnecting such a large number of servers in DCNs, at a low cost, and without compromising performance.

The construction of DCNs should consider many factors, such as scalability, cost, communication performance, and fault-tolerance. In particular, we should pay attention to making a balance among these factors since they are mutually influenced and restricted with one another in the construction of DCNs. Therefore, it has important theoretical and practical significance to design and construct new DCNs with desirable performance.

So far, many kinds of DCNs have been proposed to interconnect hundreds of thousands of or more servers in DCNs[3-10]. In fact, a number of famous DCNs are inspired by some special interconnection networks[9,11-21].

Regular Paper

This paper was supported by the National Natural Science Foundation of China under Grant Nos. 61572337, 61702351, and 61602333, the Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks Foundation under Grant No. WSNL-BKF201701, the China Postdoctoral Science Foundation under Grant No. 172985, the Natural Science Foundation of Jiangsu Higher Education Institutions of China under Grant No. 17KJB520036, the Jiangsu Planned Projects for Postdoctoral Research Funds under Grant No. 1701172B, and the Application Foundation Research of Suzhou of China under Grant No. SYG201653.

*Corresponding Author

①Harris D. Microsoft: Datacenter growth defies Moore's law. 2013. http://www.pcworld.com/article/130921/article.html/, Jan. 2018.

For example, fat-tree[3] was initiated by fat-trees interconnection network[12], BCube[6] was proposed based on generalized hypercube[11], and CamCube[8] uses a direct-connect 3D torus (k-ary 3-cube[13]) topology. Moreover, a lot of interconnection networks have been proposed in decades. Among them, hypercubes are widely used in parallel computers due to their super properties. Nevertheless, by changing links between some nodes in them, various variants of hypercubes were proposed, such as crossed cubes[22], Möbius cubes[23], and twisted cubes[24]. In [25], Fan and He summarized that there exist two characteristics common to the hypercube and its variants: they are all bijectively connected and recursively constructed. With these two properties, a family of bijective connection networks (BC networks for short) can be defined, which includes many well-known networks such as hypercubes, crossed cubes, locally twisted cubes, Möbius cubes, and twisted cubes.

In this paper, we intend to choose a network with good properties from BC networks, and use this structure to build a new type of data center network topology. Compared with n-dimensional Möbius cubes M_n, crossed cubes provide better symmetry since M_n has two non-isomorphic types: one is 0-M_n and the other is 1-M_n; compared with twisted cubes, the dimensions of crossed cubes can support all the positive integers while those of twisted cubes are only limited to odd integers; compared with the n-dimensional hypercube Q_n, crossed cube is superior to Q_n in Hamiltonian-connectivity[26], embeddability[27-28], diameter[22,29], wide diameter[30], etc. Owing to its advantageous properties, the crossed cube, as a class of BC networks, has become one of the attractive interconnection networks and many research achievements on it have been obtained[22,26,28-41].

Therefore, in this paper, we propose BCDC, a high-performance and server-centric DCN, based on a class of BC networks, crossed cube. In BCDC, each server is equipped with two ports for connecting two switches and each switch is used to connect n servers. An n-dimensional BCDC, B_n, can be defined as a recursive network structure. B_n is constructed by two $(n-1)$-dimensional BCDCs and an independent set S_n. In this way, the number of servers in BCDC increases quickly with the dimensional growth of BCDC. For example, if 16-port switches are used, B_{16} can support 524 288 servers (BCDC's detail definition can be referred to Section 2). Although we use the two ports of each server, the server's reliability is not compromised because it still uses the other port when one fails.

In this paper, we have five main contributions as follows.

1) We propose a high-performance and server-centric DCN, based on a class of BC networks, crossed cube.

2) We propose an $O(n)$ one-to-one routing algorithm in B_n. Then, we prove that the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$ for $n \geqslant 3$, which is small and can thus support applications with real-time requirements.

3) We propose high-performance one-to-many and one-to-all routing algorithms in B_n. Then, we prove that the bisection width of B_n is $(n-1)2^{n-1}$ for $n \geqslant 3$, which shows that BCDC has good fault-tolerance with server/switch failures. Moreover, we prove that the aggregate bottleneck throughput of B_n is larger than 2^{n+1} for $n \geqslant 3$, which shows that BCDC has high network capacity for all-to-all routing applications such as MapReduce.

4) We prove that the connectivity of B_n is $2n-2$. Furthermore, we propose an $O(\lceil \log_2 |F| \rceil n^3)$ algorithm for finding a fault-free path between any two distinct fault-free nodes in B_n for any faulty set $F \subset V(B_n)$ with $|F| \leqslant 2n-3$. Then, we prove that the maximal length of the fault-free path constructed by this algorithm is no more than $6m + \lceil \frac{n-m+1}{2} \rceil + 1$ with $m = \lceil \log_2 |F| \rceil$ and $|F| \leqslant 2n-3$.

5) We show simulations and experiments of B_n to evaluate the performance of routing algorithms in BCDC networks.

In summary, our analysis and simulation experiences demonstrate that BCDC is an attractive and practical DCN for mega-data centers, due to its high capacity communications, good fault-tolerance, and manageable cabling complexity.

This paper is organized in this way. The formal definition of the BCDC structure is given in Section 2. We indicate in Section 3 that the connectivity of B_n is $2n-2$. In Section 4, fault-free routing algorithms for B_n are described. Section 5 presents a fault-tolerant routing algorithm for B_n. We show simulations and experiments of B_n in Section 6. Section 7 discusses related work. The concluding remarks are mentioned in Section 8.

2 Preliminaries

A DCN can be represented by a simple graph $G = (V(G), E(G))$, where $V(G)$ represents the node set and each node represents a server, $E(G)$ represents the edge set, and each edge represents a link between servers (switches can be regarded as transparent network devices[4]). An edge with end nodes u and v is denoted by (u, v). For each node $v \in V(G)$, if $(u, v) \in E(G)$, we say u is a neighbor of v or u is adjacent to v. A (u_1, u_n)-path in G is a sequence of nodes $P = (u_1, u_2, \ldots, u_n)$, in which no node is repeated and u_j, u_{j+1} are adjacent for any integer $1 \leqslant j < n$. We also write the path $(u_1, u_2, \ldots, u_n)$ as $(u_1, Q, u_i, \ldots, u_n)$, where Q is the path $(u_2, u_3, \ldots, u_{i-1})$. The reverse of P is $(u_n, u_{n-1}, \ldots, u_1)$, denoted by P^{-1}. The path, starting from u_i and ending with u_j in a path P, can be denoted by $Path(P, u_i, u_j)$. The length of a path P, $l(P)$, is the number of edges in P. We say $P = (u)$ if the path P satisfies $l(P) = 0$. Furthermore, we use $P[i]$ to denote the node u_i for $P = (u_1, u_2, \ldots, u_n)$ for any integer $1 \leqslant i \leqslant n$, and use $P[-1]$ to denote the last node in P. Similarly, $V(P)$ and $E(P)$ are used to represent the node set and the edge set in P, respectively.

Given two distinct nodes u and v of G, the distance between u and v is defined as the length of the shortest path between u and v in G, denoted by $dist(G, u, v)$. The diameter of G is defined as $diam(G) = \max(dist(G, u, v) | u, v \in V(G), u \neq v)$. A graph is connected when there is a path between each pair of nodes. In a connected graph, there are no unreachable nodes. A graph that is not connected is disconnected. The edge-connectivity $\lambda(G)$ of a connected graph G is the smallest number of edges whose removal disconnects G. The connectivity (or node connectivity) $\kappa(G)$ of a connected graph G (other than a complete graph) is the minimum number of nodes whose removal disconnects G.

If $V' \subseteq V(G)$, we use $G[V']$ to denote the subgraph of G induced by the node subset V' in G. Furthermore, we use $G - V'$ to denote $G[V(G) \setminus V']$. We define $N_G(V') = \{x \in V(G) |$ there exists a node $y \in V'$ such that $(x, y) \in E(G)\}$. If $E' \subseteq E(G)$, we use $G[E']$ to denote the subgraph of G induced by the edge subset E' in G. Moreover, we use $G - E'$ to denote $G[E(G) \setminus E']$.

A binary string u with length n can be written as $u_{n-1}u_{n-2} \cdots u_0$, where u_{n-1} is the most significant bit, u_0 is the least significant bit, and $u_i \in \{0, 1\}$ is the i-th bit of u for integer i with $0 \leqslant i \leqslant n-1$. The complement of u_i is denoted by $\bar{u_i}$.

Like the n-dimensional hypercube, Q_n, the n-dimensional crossed cube, CQ_n, has 2^n nodes. Each node of CQ_n is represented by a unique binary string of length n, called the address of the node. For $i \in \{0, 1\}$, let CQ_{n-1}^i denote the graph obtained by prefixing the address of each node of CQ_{n-1} with i. In this paper, we would not distinguish between nodes and their addresses. We adopt the definitions of CQ_n from [22, 29].

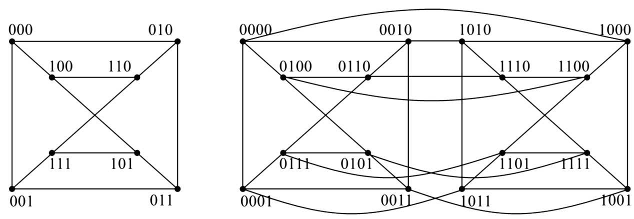
Definition 1. *Two binary strings* $x = x_1x_0$ *and* $y = y_1y_0$ *of length* 2 *are said to be pair-related* (*denoted by* $x \sim y$) *if and only if* $(x, y) \in \{(00, 00), (10, 10), (01, 11), (11, 01)\}$.

Definition 2. *The* n*-dimensional crossed cube,* CQ_n*, is recursively defined as follows.* CQ_1 *is the complete* (*undirected*) *graph on two nodes whose addresses are* 0 *and* 1 *respectively.* CQ_n *consists of* CQ_{n-1}^0 *and* CQ_{n-1}^1*. The most significant bits of the addresses of the nodes in* CQ_{n-1}^0 *and* CQ_{n-1}^1 *are* 0 *and* 1*, respectively. The nodes* $u = u_{n-1}u_{n-2} \cdots u_0 \in V(CQ_{n-1}^0)$ *and* $v = v_{n-1}v_{n-2} \cdots v_0 \in V(CQ_{n-1}^1)$*, where* $u_{n-1} = 0$ *and* $v_{n-1} = 1$*, are joined by an edge in* CQ_n *if and only if*

1) $u_{n-2} = v_{n-2}$ *if* n *is even, and*

2) $u_{2i+1}u_{2i} \sim v_{2i+1}v_{2i}$ (*see Definition* 1)*, for* $\lfloor \frac{n-1}{2} \rfloor > i \geqslant 0$.

Fig.1 demonstrates the 3-dimensional and the 4-dimensional crossed cubes, in which Fig.1(a) shows CQ_3 and Fig.1(b) shows CQ_4.



(a) (b)

Fig.1. (a) 3-dimensional crossed cube CQ_3. (b) 4-dimensional crossed cube CQ_4.

Next, we will propose a new DCN, BCDC, based on the crossed cube. We use nodes (resp. edges) of CQ_n as switches (resp. servers) of the BCDC network. For convenience, we do not distinguish each switch (resp. server) from its address in this paper.

Each switch address of the BCDC network is denoted by a binary string $u = u_{n-1}u_{n-2} \cdots u_0$ with length n. Then, we use $f(u)$ to denote the decimal value of the binary string u. Moreover, given two binary strings $u = u_{n-1}u_{n-2} \cdots u_0$ and $v = v_{n-1}v_{n-2} \cdots v_0$ with $f(u) < f(v)$, each server address of the BCDC network is denoted by an ordered pair $[u, v]$. A server $[u, v]$ connects a switch x if and only if $x \in \{u, v\}$ and $(u, v) \in E(CQ_n)$. Then, we get the original graph of n-dimensional BCDC, denoted by A_n. Clearly, A_n has $n2^{n-1}$ servers, 2^n switches, and $n2^n$ links.

When considering that switches are transparent in BCDC networks, we give the definition of the logical graph of BCDC networks B_n as follows.

Definition 3. *The n-dimensional BCDC network, B_n, is recursively defined as follows. B_2 is a cycle with 4 nodes* $[00, 01]$, $[00, 10]$, $[01, 11]$, *and* $[10, 11]$. *For $n \geqslant 3$, we use B^0_{n-1} (resp. B^1_{n-1}) to denote the graph obtained by B_{n-1} with changing each node $[x, y]$ of B_{n-1} to $[0x, 0y]$ (resp. $[1x, 1y]$). B_n consists of B^0_{n-1}, B^1_{n-1}, and a node set $S_n = \{[a, b] | a \in V(CQ^0_{n-1}), b \in V(CQ^1_{n-1})$, and $(a, b) \in E(CQ_n)\}$ according to the following rules. For nodes $u = [a, b] \in V(B^0_{n-1})$, $v = [c, d] \in S_n$, and $w = [e, f] \in V(B^1_{n-1})$:*

1) *$(u, v) \in E(B_n)$ if and only if $a = c$ or $b = c$;*

2) *$(v, w) \in E(B_n)$ if and only if $e = d$ or $f = d$.*

Fig.2 demonstrates the original graph and logi-

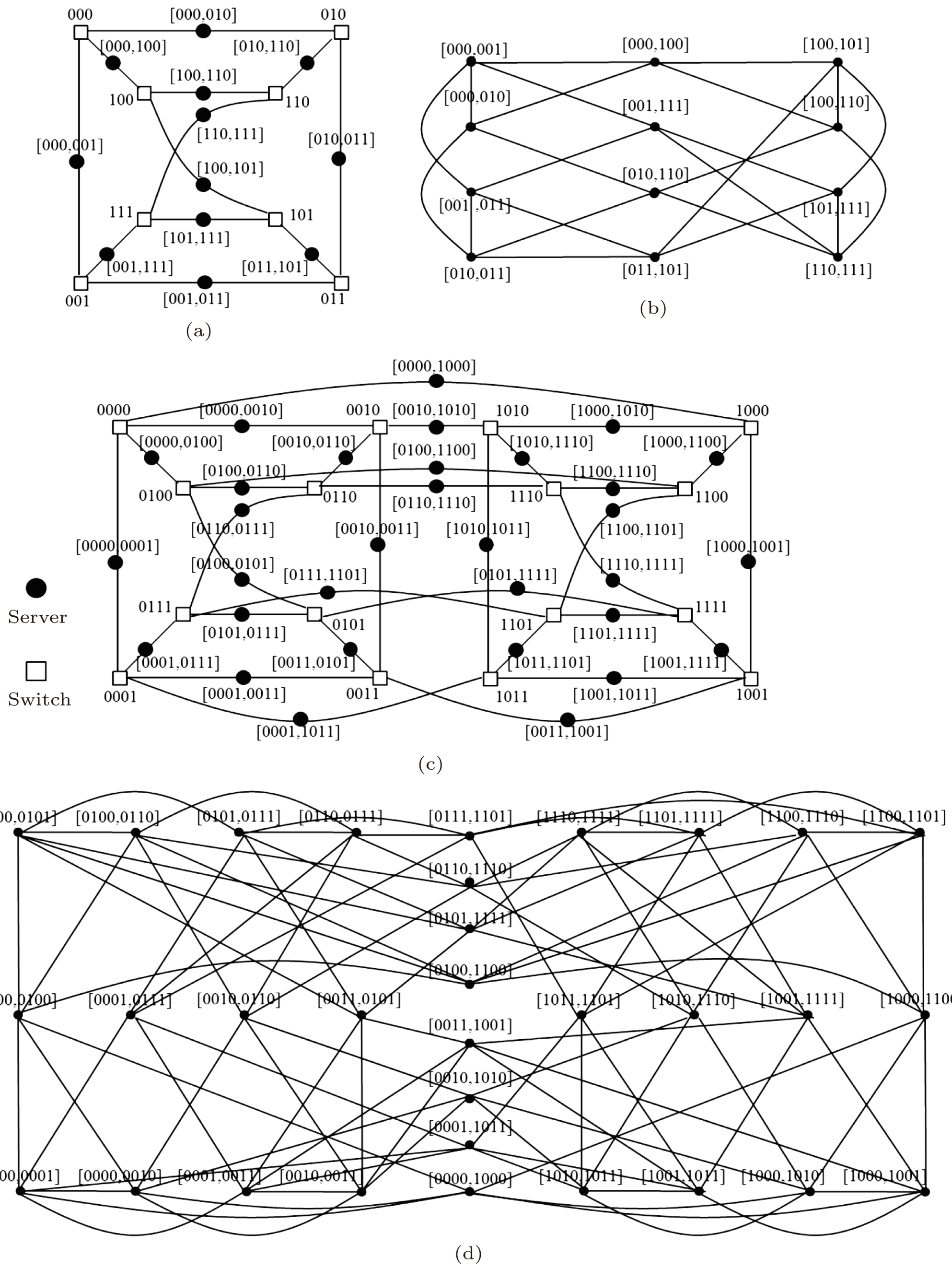


Fig.2. (a) Original graph of 3-dimensional BCDC A_3. (b) Logical graph of 3-dimensional BCDC B_3. (c) Original graph of 4-dimensional BCDC A_4. (d) Logical graph of 4-dimensional BCDC B_4.

cal graph of 3-dimensional BCDC and 4-dimensional BCDC, in which Fig.2(a) shows A_3, Fig.2(b) shows B_3, Fig.2(c) shows A_4, and Fig.2(d) shows B_4, respectively.

B_n is obviously a $(2n-2)$-regular graph. Furthermore, the i-th element of a node u in B_n is denoted by $u[i]$ with $i \in \{0,1\}$. Clearly, B_n has $n2^{n-1}$ nodes and $n(n-1)2^{n-1}$ edges.

The BCDC's construction shows that switches are only adjacent to servers and never adjacent to other switches directly. Thus, we can treat the switches as transparent network devices that connect several neighboring servers adjacent to one another. With 16-port switches, we can support up to 524 288 servers in a B_{16}. Therefore, BCDC meets our goal of using only low-end commodity switches by putting routing computation into servers purely.

3 Connectivity and Edge-Connectivity of BCDC

As the number of nodes in a BCDC increases, node failures may become the norm rather than exception. Usually, reliable data transmission in a BCDC is based on the condition of the set of arbitrary faulty nodes. That is, any nodes in a BCDC can become faulty. Under this condition, supposing that the connectivity of a BCDC is κ and the number of faulty nodes in it is at most $\kappa-1$, then there exists at least one fault-free path (i.e., a path that does not contain any faulty nodes) between any two fault-free nodes, which can be used to reliably communicate between them.

It is well-known that $\kappa(G) \leqslant \lambda(G) \leqslant \delta(G)$ for any graph G[42], and $\kappa(CQ_n) = \lambda(CQ_n) = n$[29]. As a result, CQ_n achieves the maximum connectivity (resp. edge-connectivity) possible for its resources. In what follows, we prove that B_n also achieves the maximum connectivity (resp. edge-connectivity) possible for its resources.

Obviously, Definition 2 and Definition 3 imply the following lemma.

Lemma 1. *For any integer $n \geqslant 3$ and any two nodes $u=[a,b], v=[c,d] \in S_n$, we have*

$$|N_{B^0_{n-1}}(\{u,v\})| = \begin{cases} 2n-2, & \text{if } (a,c) \notin E(CQ_n), \\ 2n-3, & \text{if } (a,c) \in E(CQ_n), \end{cases}$$

and

$$|N_{B^1_{n-1}}(\{u,v\})| = \begin{cases} 2n-2, & \text{if } (b,d) \notin E(CQ_n), \\ 2n-3, & \text{if } (b,d) \in E(CQ_n). \end{cases}$$

Theorem 1. $\kappa(B_n) = \lambda(B_n) = 2n-2$.

Proof. As B_n is $(2n-2)$-regular, we have $\kappa(B_n) \leqslant \lambda(B_n) \leqslant 2n-2$. Thus, it suffices to prove that $\kappa(B_n) \geqslant 2n-2$. We only need to prove the following claim.

Claim. *For any $F \subset V(B_n)$ with $|F| \leqslant 2n-3$, then $B_n - F$ is connected.*

We prove the claim by induction on n. The claim clearly holds for $n \leqslant 2$. Supposing that the claim holds for $n = \tau - 1$ with $\tau \geqslant 3$, we will prove that the theorem holds for $n = \tau$. Let $F_0 = F \cap V(B^0_{\tau-1})$, $F_1 = F \cap V(B^1_{\tau-1})$, $F_2 = F \setminus (F_0 \cup F_1)$, $G_0 = B^0_{\tau-1} - F_0$, $G_1 = B^1_{\tau-1} - F_1$, $G_2 = B_\tau[S_\tau \setminus F_2]$, $G_3 = B_\tau[V(B^0_{\tau-1}) \cup S_\tau] - F_0 \cup F_2$, and $G_4 = B_\tau[V(B^1_{\tau-1}) \cup S_\tau] - F_1 \cup F_2$. We deal with the following five cases for $\tau \geqslant 3$.

Case 1. $F_0 = F$ or $F_1 = F$ or $F_2 = F$. Then, we have the following three subcases.

Subcase 1.1. $F_0 = F$. Then, $F_1 = F_2 = \emptyset$. We can verify that G_4 is connected for $F_1 \cup F_2 = \emptyset$. For any integer $\tau \geqslant 3$, we have $|V(B^0_{\tau-1})| = (\tau-1)2^{\tau-2} > 2\tau - 3 \geqslant |F| = |F_0|$ and thus $V(G_0) \neq \emptyset$. By Definition 3, we can verify that each node of G_0 is adjacent to two nodes in $G_2 \subset G_4$ for $F_2 = \emptyset$. Therefore, $B_\tau - F$ is connected.

Subcase 1.2. $F_1 = F$. The argument is similar to that for subcase 1.1.

Subcase 1.3. $F_2 = F$. We can verify that G_0 (resp. G_1) is connected for $F_0 = \emptyset$ (resp. $F_1 = \emptyset$). For any integer $\tau \geqslant 3$, we have $|S_\tau| = 2^{\tau-1} > 2\tau - 3 \geqslant |F| = |F_2|$ and thus $S_\tau \setminus F_2 \neq \emptyset$. Then, we can verify that G_3 is connected for each node of $B_\tau[S_\tau \setminus F_2]$ is adjacent to $\tau - 1$ nodes in G_0. Furthermore, we can verify that each node of $G_2 \subset G_3$ is adjacent to $\tau - 1$ nodes in G_1. Thus, $B_\tau - F$ is connected.

Case 2. $F_2 = \emptyset$, $F_0 \neq \emptyset$, and $F_1 \neq \emptyset$. Without loss of generality, suppose that $|F_0| \leqslant |F_1|$. For any integer $\tau \geqslant 3$, we have $|F_0| \leqslant \lfloor \frac{|F_0|+|F_1|}{2} \rfloor = \lfloor \frac{|F|}{2} \rfloor \leqslant \lfloor \frac{2\tau-3}{2} \rfloor \leqslant \tau - 2 \leqslant 2\tau - 5$. Thus, by the induction hypothesis, G_0 is connected for $\tau \geqslant 3$. By Definition 3, each node of S_τ has $\tau - 1 > \tau - 2 \geqslant |F_0|$ neighbors in $B^0_{\tau-1}$ which implies that G_3 is connected. Furthermore, each node of G_1 is adjacent to two nodes in $G_2 \subset G_3$ by Definition 3 with $F_2 = \emptyset$. Therefore, $B_\tau - F$ is connected.

Case 3. $F_2 \neq \emptyset$, $F_0 = \emptyset$, and $F_1 \neq \emptyset$. For any integer $\tau \geqslant 3$, we have $|S_\tau| = 2^{\tau-1} > 2\tau - 3$ and thus $|V(G_2)| > 0$. By Definition 3, each node of G_2 has $\tau - 1 > |F_0| = 0$ neighbors in $B^0_{\tau-1}$ which implies that G_3 is connected. We further have the following two subcases.

Subcase 3.1. $|F_2| \leqslant |F_1|$.

1) If $|F_1| \leqslant 2\tau - 5$. By the induction hypothesis, G_1 is connected. For any integer $\tau \geqslant 3$, we have $|F_2| \leqslant \lfloor \frac{|F_2|+|F_1|}{2} \rfloor = \lfloor \frac{|F|}{2} \rfloor \leqslant \lfloor \frac{2\tau-3}{2} \rfloor \leqslant \tau - 2$. Thus, $|S_\tau \setminus F_2| = 2^{\tau-1} - |F_2| \geqslant 2^{\tau-1} - (\tau - 2) \geqslant 2$ for $\tau \geqslant 3$. Then, two distinct nodes u and v in $S_\tau \setminus F_2$ are chosen. By Lemma 1, we have $|N_{B^1_{\tau-1}}(\{u, v\})| \geqslant 2\tau - 3 \geqslant |F_2|+|F_1| > |F_1|$. Thus, there exists at least one node of $G_2 \subset G_3$, which is adjacent to a node in G_1. Therefore, $B_\tau - F$ is connected.

2) If $|F_1| > 2\tau - 5$. It is easy to verify that $|F_1| = 2\tau - 4$ and $|F_2| = 1$. By Definition 3, each node of G_1 is adjacent to two nodes in S_τ. Thus, we can verify that each node of G_1 is adjacent to at least one node in $G_2 \subset G_3$ for $|F_2| = 1 < 2$. Therefore, $B_\tau - F$ is connected.

Subcase 3.2. $|F_2| > |F_1|$. For any integer $\tau \geqslant 3$, we have $|F_1| \leqslant \lfloor \frac{|F_2|+|F_1|}{2} \rfloor = \lfloor \frac{|F|}{2} \rfloor \leqslant \lfloor \frac{2\tau-3}{2} \rfloor \leqslant \tau - 2 \leqslant 2\tau - 5$. Thus, by the induction hypothesis, G_1 is connected for $\tau \geqslant 3$. For any integer $\tau \geqslant 3$, we have $|S_\tau| = 2^{\tau-1} > 2\tau - 3$ and thus $|V(G_2)| > 0$. By Definition 3, each node of G_2 has $\tau - 1 > \tau - 2 \geqslant |F_1|$ neighbors in $B^1_{\tau-1}$. Thus, we can verify that each node of $G_2 \subset G_3$ is adjacent to at least one node in G_1. Therefore, $B_\tau - F$ is connected.

Case 4. $F_2 \neq \emptyset$, $F_0 \neq \emptyset$, and $F_1 = \emptyset$. The argument is similar to that for case 3.

Case 5. $F_0 \neq \emptyset$, $F_1 \neq \emptyset$, and $F_2 \neq \emptyset$. Suppose that $|F_i| > 2\tau - 5$ with $i \in \{0, 1, 2\}$, and we have $|F| = |F_0|+|F_1|+|F_2| > 2\tau-5+1+1 = 2\tau-3$, a contradiction. Thus, we have $|F_i| \leqslant 2\tau-5$ for $i \in \{0, 1, 2\}$, which implies that G_0 and G_1 are connected. We further have the following two subcases.

Subcase 5.1. $|F_2| \leqslant \tau - 2$.

1) If $|F_0| \leqslant |F_1|$. For any integer $\tau \geqslant 3$, we have $|F_0| \leqslant \lfloor \frac{|F_0|+|F_1|+|F_2|}{2} \rfloor = \lfloor \frac{|F|}{2} \rfloor \leqslant \lfloor \frac{2\tau-3}{2} \rfloor \leqslant \tau - 2$. By Definition 3, each node of G_2 has $\tau - 1$ neighbors in $B^0_{\tau-1}$. Thus, we can verify that each node of G_2 is adjacent to at least one node in G_0 for $|F_0| \leqslant \tau-2$, which implies that G_3 is connected. By Definition 3, we have $|S_\tau \setminus F_2| = 2^{\tau-1} - |F_2| \geqslant 2^{\tau-1} - (\tau - 2) \geqslant 2$ for $\tau \geqslant 3$. Then, two distinct nodes u and v in $S_\tau \setminus F_2$ are chosen. By Lemma 1, we have $|N_{B^1_{n-1}}(\{u, v\})| \geqslant 2\tau - 3 \geqslant |F_0| + |F_2| + |F_1| > |F_1|$. Thus, there exists at least one node of $G_2 \subset G_3$, which is adjacent to a node in G_1. Therefore, $B_\tau - F$ is connected.

2) If $|F_0| > |F_1|$. The argument is similar to that of $|F_0| \leqslant |F_1|$ in subcase 5.1.

Subcase 5.2. $|F_2| \geqslant \tau-1$. Then, we have $|F_0| \leqslant \tau-2$ and $|F_1| \leqslant \tau - 2$. For any integer $\tau \geqslant 3$, we have $|S_\tau| = 2^{\tau-1} > 2\tau - 3$ and thus $|V(G_2)| > 0$. By Definition 3, each node of G_2 has $\tau - 1 > \tau - 2 \geqslant |F_0|$ neighbors in $B^0_{\tau-1}$, which implies that each node of G_2 is adjacent to at least one node in G_0, and thus G_3 is connected. By Definition 3, we can verify that each node of G_2 has $\tau - 1$ neighbors in $B^1_{\tau-1}$, which implies that each node of $G_2 \subset G_3$ is adjacent to at least one node in G_1 for $|F_1| \leqslant \tau - 2$. Therefore, $B_\tau - F$ is connected.

In summary, the claim holds for $n = \tau$. □

Theorem 1 shows that B_n achieves the maximum connectivity possible for its resources. When the BCDC network is used to model the topological structure of a large-scale DCN, our result can provide an accurate measure for the fault tolerance of the network.

4 Fault-Free Routings in BCDC

In this section, we will study one-to-one, one-to-many, one-to-all, and all-to-all routings in BCDC without any fault elements. Particularly, we prove that the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$, the bisection width of B_n is $(n - 1)2^{n-1}$, and the aggregate bottleneck throughput of B_n is larger than 2^{n+1}.

4.1 One-to-One Routing in BCDC

In [22], Efe proposed an $O(n^2)$ algorithm to find a shortest path between any two distinct nodes in CQ_n. In [30], Chang *et al*. improved this algorithm. They gave an $O(n)$ algorithm, which we call $CSH(CQ_n, u, v)$ algorithm, to get a shortest path between any two distinct nodes u and v in CQ_n. Furthermore, they gave the distance function between any two distinct nodes u and v in CQ_n, denoted by $\rho(u, v)$, and proved an important result as follows: $dist(CQ_n, u, v) = \rho(u, v)$. For details on the CSH algorithm, please refer to [30]. The following lemma indicates the diameter of CQ_n.

Lemma 2. *If* $n \geqslant 3$*, then* $diam(CQ_n) = \lceil \frac{n+1}{2} \rceil$[22,29].

The global link state routing scheme is not suitable for one-to-one routing in BCDC-based DCNs since BCDC's goal is to interconnect up to tens of thousands of servers. Furthermore, one-to-one routing in BCDC cannot use the hierarchical OSPF②, since it needs a backbone area to interconnect all the other areas. This results in both single point failure and bandwidth bottleneck.

B_n uses a simple and efficient one-to-one (unicast) routing algorithm, called `BRouting`, to get

②Moy J. RFC 2328: OSPF version 2. 1998. https://datatracker.ietf.org/doc/rfc2328/, Jan. 2018.

the shortest path between any two distinct nodes in B_n, as shown in Algorithm 1. `BRouting` is a shortest-path routing scheme. It can be shown by the following example. For a B_8, the path using `BRouting` between nodes [01100110, 01100111] and [01011110, 01011111] is ([01100110, 01100111], [01100110, 01111110], [01010110, 01111110] [01010110, 01011110], [01011110, 01011111]) with length 4, which is a shortest path.

Algorithm 1. `BRouting`

Input: an n-dimensional BCDC, B_n, and two distinct nodes $u, v \in V(B_n)$
Output: a shortest path from node u to node v in B_n

function BRouting(B_n, u, v)
  $u_1 \leftarrow u[0]$, $u_2 \leftarrow u[1]$, $v_1 \leftarrow v[0]$, $v_2 \leftarrow v[1]$;
  $d_1 \leftarrow \rho(u_1, v_1)$, $d_2 \leftarrow \rho(u_1, v_2)$, $d_3 \leftarrow \rho(u_2, v_1)$;
  $d_4 \leftarrow \rho(u_2, v_2)$, $d \leftarrow \min\{d_1, d_2, d_3, d_4\}$;
  if $d_1 = d$ **then**
    return $(u, \mathtt{CPath}(CQ_n, u_1, v_1), v)$;
  else if $d_2 = d$ **then**
    return $(u, \mathtt{CPath}(CQ_n, u_1, v_2), v)$;
  else if $d_3 = d$ **then**
    return $(u, \mathtt{CPath}(CQ_n, u_2, v_1), v)$;
  else
    return $(u, \mathtt{CPath}(CQ_n, u_2, v_2), v)$;
  end if
end function
function CPath(CQ_n, u, v)
  $P \leftarrow (u)$, $Q \leftarrow \mathtt{CSH}(CQ_n, u, v)$;
  for $i = 1$; $i < l(Q)$; $i++$ **do**
    if $Q[i] < Q[i+1]$ **then**
      $P \leftarrow (P, [Q[i], Q[i+1]])$;
    else
      $P \leftarrow (P, [Q[i+1], Q[i]])$;
    end if
  end for
  return P;
end function

Obliviously, the path construction of `BRouting` is dependent only on the addresses of the source and destination nodes. Furthermore, `BRouting` can be performed quickly when building large networks in practice. Chang *et al*. proved that $CSH(CQ_n, u, v)$ and $\rho(u, v)$ can be computed in $O(n)$ time[30]. Therefore, the time complexity of constructing the whole routing path in algorithm `BRouting` is $O(n)$, which is within the minimum possible order of magnitude.

The following theorem gives the diameter of the BCDC network.

Theorem 2. *The diameter of* B_n *is* $\lceil \frac{n+1}{2} \rceil + 1$.

Proof. We firstly prove that the upper bound on the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$. Two distinct nodes u and v in B_n are chosen. Then, let $u = [a, b]$ and $v = [c, d]$. By Definition 3 and Lemma 2, we have $dist(B_n, u, v) \leqslant \min\{dist(CQ_n, a, c),\ dist(CQ_n, a, d),\ dist(CQ_n, b, c),\ dist(CQ_n, b, d)\} + 1 \leqslant diam(CQ_n) + 1 = \lceil \frac{n+1}{2} \rceil + 1$.

Thus, the upper bound on the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$.

Furthermore, we will prove that the lower bound on the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$. Then, let $u = [0^n, 0^{n-2}10]$ and $v = [1^{n-3}001,\ 1^{n-3}011]$. By Definition 3 in [22], we have $dist(B_n, u, v) = \min\{\rho(0^n,\ 1^{n-3}001),\ \rho(0^n,\ 1^{n-3}011),\ \rho(0^{n-2}10,\ 1^{n-3}001),\ \rho(0^{n-2}10,\ 1^{n-3}011)\} + 1 = \min\{\lceil \frac{n+1}{2} \rceil, \lceil \frac{n+1}{2} \rceil, \lceil \frac{n+1}{2} \rceil, \lceil \frac{n+1}{2} \rceil\} + 1 = \lceil \frac{n+1}{2} \rceil + 1$.

Therefore, the lower bound on the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$.

In summary, the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$. □

Obviously, the diameter of BCDC is small when considering the total number of servers, which benefits applications with real-time requirement.

4.2 One-to-Many Routing in BCDC

In one-to-many routing, a BCDC server can use its multiple links to perform high throughput. This property is useful for services such as GFS[43], which can use multiple links at a server to speed up file replication and recovery in DCN.

We introduce a simple and efficient one-to-many (multicast) routing algorithm for B_n, called `BMulticast`, showed in Algorithm 2. In order to express the algorithm compactly, we need some notations. We use mT to denote a multicast tree rooted at node s to a node set T on B_n such that $s \in V(B_n)$, $T \subset V(B_n)$, and $\{s\} \cap T = \emptyset$. Similarly, $V(mT)$ and $E(mT)$ are used to represent the node set and the edge set in mT, respectively. `BMulticast` could construct a multicast tree from a source node s to a destination node set $T = \{t_1, t_2, \ldots, t_m\}$ in B_n with $s \notin T$. It can be shown by the following example. For a B_8, a multicast tree using `BMulticast` from node [01100110, 01100111] to nodes {[01010110, 01010111], [01011110, 01011111], [01100010, 01110010]} with the height 4 is shown in Fig.3.

Obliviously, the construction of a multicast tree in Algorithm 2 is dependent only on the addresses of the source node and destination nodes. Furthermore, Algorithm 2 can be carried out quickly when building large networks in practice. We can find that the most time is taken in lines 6~11 of Algorithm 2, which can be computed in $O(n|T|)$ time. Since algorithm `BRouting` and $\rho(u, v)$ can be computed in $O(n)$ time, the time complexity to construct the whole multicast tree in Algorithm 2 is $O(n|T|^2)$. Since the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$, the height of the multicast tree constructed by algorithm `BMulticast` is no more than $\lceil \frac{n+1}{2} \rceil + 1$.

Algorithm 2. `BMulticast`

Input: an n-dimensional BCDC, B_n, and $\{s\}, T \subset V(B_n)$ with $\{s\} \cap T = \emptyset$
Output: a multicast tree mT from a node s to a node set T on B_n

function BMULTICAST(B_n, s, T)
  $V(mT) \leftarrow \{s\}$, $E(mT) \leftarrow \emptyset$;
  while $|T| > 0$ **do**
    Find a node $t \in T$, $P \leftarrow$`BRouting`(B_n, s, t);
    if $E(mT) \neq \emptyset$ **then**
      $U \leftarrow V(mT) \cap V(P) \setminus \{s\}$;
      if $|U| > 0$ **then**
        $d = \max(\{\rho(s, u) | u \in U\})$;
        Find a node $v \in U$ with $d = \rho(s, v)$;
        $P \leftarrow$ `Path`$(P, v, P[-1])$;
      end if
    end if
    $E(mT) \leftarrow E(mT) \cup E(P)$;
    $V(mT) \leftarrow V(mT) \cup V(P)$;
    $T \leftarrow T \setminus V(mT)$;
  end while
  return mT;
end function

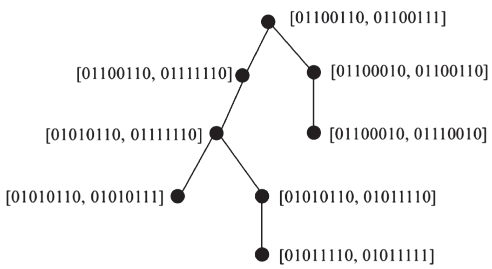


Fig.3. Multicast tree from a source node [01100110, 01100111] to destination nodes {[01010110, 01010111], [01011110, 01011111], [01100010, 01110010]} in B_8.

4.3 One-to-All Routing in BCDC

We demonstrate that BCDC can accelerate one-to-all routing (broadcast) significantly. In broadcast, one node delivers messages to all the other nodes. A simple approach of constructing a spanning tree from one node to all the other nodes, and then broadcasting messages along the tree, is not fault-tolerant. When one node is broken, the subtree in the spanning tree under that node will not receive the broadcast message anymore.

To resolve the issue mentioned above, we introduce, `BBroadcast`, a simple and robust broadcast scheme. In `BBroadcast`, one node delivers the broadcast packet to all its $2n-2$ neighbors when broadcasting a packet. If a node receives a broadcast packet, it first checks whether this packet has been received before. Then, this node drops a duplicate packet but broadcasts a new packet to its other $2n-1$ neighbors. `BBroadcast` is fault-tolerant in that a broadcast packet can reach all the nodes while the network is connected.

In `BBroadcast`, we limit the broadcast scope by encoding a scope value n into each broadcast message. The message is broadcasted only within the whole BCDC network that contains the source node. Since the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$, a broadcast message needs $\lceil \frac{n+1}{2} \rceil + 1$ steps to reach all the nodes in B_n.

4.4 All-to-All Routing in BCDC

In computer networking, if the network is bisected into two partitions, the bisection bandwidth of a network topology is the bandwidth available between the two partitions[42]. The bisection width of a network is significant in the performance measurement of all-to-all routing in a network. Then, we have the following theorem.

Theorem 3. *The bisection width of B_n, denoted by $\omega(B_n)$, is $(n-1)2^{n-1}$ for $n \geqslant 3$.*

Proof. Note that B_n is constructed from two identical $(n-1)$-dimensional BCDCs, B^0_{n-1} and B^0_{n-1}, and a node set S_n. Choose $W_0, W_1 \subset S_n$ such that $W_0 \cap W_1 = \emptyset$ and $|W_0| = |W_1| = \frac{|S_n|}{2} = 2^{n-2}$. Let $E_i = \{(a, b) | a \in V(B^i_{n-1}), b \in W_i$, and $(a, b) \in E(B_n)\}$ with $i \in \{0, 1\}$. By Definition 3, we have $E_0 \cap E_1 = \emptyset$ and $|E_0| = |E_1| = (n-1)2^{n-2}$. Then, $B_n - E_0 \cup E_1$ is disconnected, which follows $\omega(B_n) \leqslant 2(n-1)2^{n-2} = (n-1)2^{n-1}$.

We define an embedding of a complete graph of $n2^{n-1}$ nodes, denoted by K, into B_n, where each edge in K is embedded into B_n. Suppose that $\omega(B_n) < (n-1)2^{n-1}$. It follows that B_n can be partitioned into two subgraphs of equal size by removing a cut of $\omega(B_n)$ edges. This cut of B_n also induces a bisection of K. Since each edge of B_n is contained in no more than $\frac{n^2}{n-1}2^{n-3}$ shortest paths for $n \geqslant 3$, denoted by C, it follows that $\omega(K) = \omega(B_n)C < (n-1)2^{n-1}(\frac{n^2}{n-1}2^{n-3}) = n^2 2^{2n-4}$, which is contradictory to $\omega(K) = \frac{(n2^{n-1})^2}{4} = n^2 2^{2n-4}$. Thus, $\omega(B_n) \geqslant (n-1)2^{n-1}$.

Therefore, we have $\omega(B_n) = (n-1)2^{n-1}$. □

The large bisection width of BCDC implies that there are numerous possible paths between any pair of nodes. Therefore, BCDC is intrinsically fault-tolerant. Theorem 3 also shows that BCDC can well support MapReduce[44].

Under the all-to-all model, every server establishes a flow with all other servers. Among all the flows, the flows that receive the smallest throughput are called the bottleneck flows[6].

To evaluate the capacity of BCDC, we use the metric ABT (aggregate bottleneck throughput), defined in

[6] as the number of flows times the throughput of the bottleneck flows, in the all-to-all traffic model. The larger ABT, the shorter finish time of all-to-all job in a network.

Theorem 4. *The aggregate bottleneck throughput for a BCDC network under the all-to-all routing is larger than* 2^{n+1} *for* $n \geqslant 3$.

Proof. Let $ABT(B_n)$ be the aggregate bottleneck of B_n. Then, let $N = n2^n$ (resp. $M = n2^{n+1}$) denote the number of nodes (resp. links) in BCDC. Moreover, we use L to denote the average path length from one node to the rest nodes using `BRouting`. For $n \geqslant 3$, we have $L < diam(B_n) = \lceil \frac{n+1}{2} \rceil + 1 \leqslant n$. Based on [6], we have $ABT(B_n) = \frac{N(N-1)M}{N(N-1)L} = \frac{M}{L} > \frac{n2^{n+1}}{n} = 2^{n+1}$. □

An advantage of BCDC is that it does not have performance bottlenecks in the all-to-all routing since all the links are used equally. As a result, the ABT of BCDC increases linearly as the number of nodes increases.

5 Fault-Tolerant Routing in BCDC

In this section, we first give algorithm `BFRouting`, to construct a fault-free path between any two distinct fault-free nodes in B_n with a faulty node set $F \subset V(B_n)$ and $|F| \leqslant 2n-3$. Then, we analyze the time complexity of the algorithm `BFRouting`. Furthermore, we analyze the maximal length of paths constructed by algorithm `BFRouting`.

Theorem 5. *For any integer* $n \geqslant 3$*, any faulty node set* $F \subset V(B_n)$ *with* $|F| \leqslant 2n-3$*, and any* $u \in V(B^i_{n-1}) - F$ *with* $i \in \{0,1\}$*, there exists at least one fault-free path* $P = (\alpha_0 = u, \alpha_1, \ldots, \alpha_l)$ *of length* l *with* $2 \leqslant l \leqslant 3$*, from* u *into* B^{1-i}_{n-1}*, such that* $\alpha_0, \alpha_1, \ldots, \alpha_{l-2} \in V(B^i_{n-1})$*,* $\alpha_{l-1} \in S_n$*, and* $\alpha_l \in V(B^{\bar{i}}_{n-1})$.

Proof. Without loss of generality, suppose that $i = 0$. Let $\beta_1, \beta_2, \alpha_3, \alpha_4, \ldots$, and α_{2n-2} be all the $2n-2$ neighbors of u in B_n with $\beta_1, \beta_2 \in S_n$ and $\alpha_3, \alpha_4, \ldots, \alpha_{2n-2} \in V(B^0_{n-1})$. $2n-4$ nodes $\beta_3, \beta_4, \ldots, \beta_{2n-2}$ in S_n are chosen such that $(\alpha_i, \beta_i) \in E(B_n)$ for $3 \leqslant i \leqslant 2n-2$ with $|\{\beta_1, \beta_2, \ldots, \beta_{2n-2}\}| = 2n-2$. Then, $2n-2$ nodes $\gamma_1, \gamma_2, \ldots, \gamma_{2n-2}$ in B^1_{n-1} are chosen such that $(\beta_i, \gamma_i) \in E(B_n)$ for $1 \leqslant i \leqslant 2n-2$ with $|\{\gamma_1, \gamma_2, \ldots, \gamma_{2n-2}\}| = 2n-2$.

As a sequence, we construct $2n-2$ disjoint paths from B^0_{n-1} into B^1_{n-1} as follows (see Fig.4):

$$P_1 = (u, \beta_1, \gamma_1),$$
$$P_2 = (u, \beta_2, \gamma_2),$$
$$P_3 = (u, \alpha_3, \beta_3, \gamma_3),$$
$$P_4 = (u, \alpha_3, \beta_4, \gamma_4),$$
$$\vdots$$
$$P_{2n-3} = (u, \alpha_{2n-3}, \beta_{2n-3}, \gamma_{2n-3}), \text{ and,}$$
$$P_{2n-2} = (u, \alpha_{2n-2}, \beta_{2n-2}, \gamma_{2n-2}).$$

Since $|F| \leqslant 2n-3 < 2n-2$, there exists at least one fault-free path P_j among the $2n-2$ paths $P_1, P_2, \ldots, P_{2n-2}$, where $1 \leqslant j \leqslant 2n-2$.

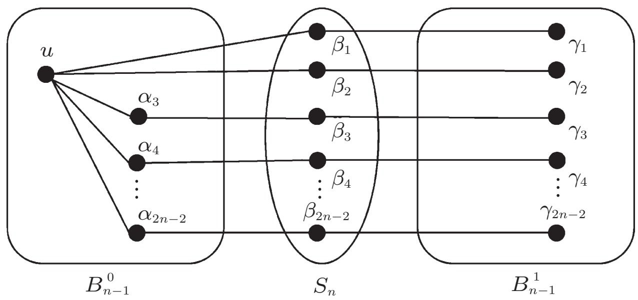


Fig.4. Illustration of constructing $2n-2$ disjoint paths from B^0_{n-1} into B^1_{n-1} in Theorem 5.

If $1 \leqslant j \leqslant 2$, then $P_j = (u, \beta_j, \gamma_j)$ is a fault-free path of length 2 from $u \in V(B^0_{n-1})$ into B^1_{n-1}, where $\beta_j \in S_n$ and $\gamma_j \in V(B^1_{n-1})$; otherwise, there exists a fault-free path $P_j = (u, \alpha_j, \beta_j, \gamma_j)$ of length 3 from $u \in V(B^0_{n-1})$ into B^1_{n-1}, where $j \geqslant 2$, $\alpha_j \in V(B^0_{n-1})$, $\beta_j \in S_n$, and $\gamma_j \in V(B^1_{n-1})$. □

Theorem 6. *For any integer* $n \geqslant 3$*, any faulty node set* $F \subset V(B_n)$ *with* $|F| \leqslant 2n-3$*, and any two distinct nodes* $u, v \in V(B^i_{n-1} - F)$ *with* $i \in \{0,1\}$ *and* $(u,v) \notin E(B_n)$*, let* P *(resp.* Q*) be a fault-free path from* u *(resp.* v*) into* $B^{\bar{i}}_{n-1}$*. If* $V(P) \cap V(Q) \neq \emptyset$*, then the length of* $H = (\mathtt{Path}(P,u,x), \mathtt{Path}(Q,x,v))$ *satisfies* $2 \leqslant l(H) \leqslant 6$*, where* x *be the first common node of the two paths* P *and* Q*; otherwise,* $h = l(P) + l(Q)$ *satisfies* $4 \leqslant h \leqslant 6$.

Proof. Without loss of generality, suppose that $i = 0$. We use $\{x_1, x_2\}$ to denote $N_{B_n[S_n]}(u)$ and $\{y_1, y_2\}$ to denote $N_{B_n[S_n]}(v)$, respectively. Then, let $W_1 = N_{B^1_{n-1}}(x_1)$, $W_2 = N_{B^1_{n-1}}(x_2)$, $W_3 = N_{B^1_{n-1}}(y_1)$, and $W_4 = N_{B^1_{n-1}}(y_2)$. According to Theorem 5, we let

$$P = \begin{cases} (u, \beta_1, \gamma_1), & \text{if } (x_1 \notin F \text{ and } W_1 \setminus F \neq \emptyset) \text{ or} \\ & (x_2 \notin F \text{ and } W_2 \setminus F \neq \emptyset), \\ (u, \alpha_2, \beta_2, \gamma_2), & \text{otherwise,} \end{cases}$$

(resp.

$$Q = \begin{cases} (v, \beta_3, \gamma_3), & \text{if } (y_1 \notin F \text{ and } W_3 \setminus F \neq \emptyset) \text{ or} \\ & (y_2 \notin F \text{ and } W_3 \setminus F \neq \emptyset), \\ (v, \alpha_4, \beta_4, \gamma_4), & \text{otherwise,} \end{cases}$$

) be a fault-free path from u (resp. v) into B^1_{n-1} with $\alpha_2 \in V(B^0_{n-1})$, $\beta_1, \beta_2 \in S_n$, and $\gamma_1, \gamma_2 \in V(B^1_{n-1})$ (resp. $\alpha_4 \in V(B^0_{n-1})$, $\beta_3, \beta_4 \in S_n$, and $\gamma_3, \gamma_4 \in V(B^1_{n-1})$). We address the following two cases with respect to P and Q.

Case 1. $V(P) \cap V(Q) \neq \emptyset$. If $\beta_1 = \beta_3$, $l(H) = 2$; if $\beta_1 = \beta_4$ or $\beta_2 = \beta_3$, $l(H) = 3$; if $\beta_2 = \beta_4$ or $\gamma_1 = \gamma_3$, $l(H) = 4$; if $\gamma_1 = \gamma_4$ or $\gamma_2 = \gamma_3$, $l(H) = 5$; otherwise, $l(H) = 6$ (see Fig.5(a)).

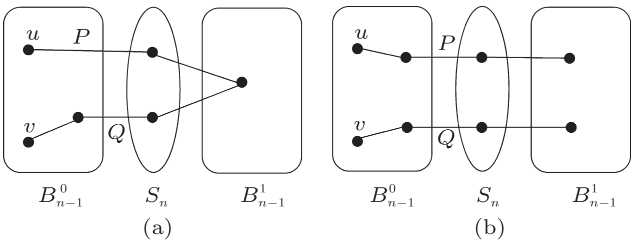


Fig.5. Illustration for (a) case 1 in Theorem 6 and (b) case 2 in Theorem 6.

Case 2. $V(P) \cap V(Q) = \emptyset$. If $\gamma_1 \in V(P)$ and $\gamma_3 \in V(Q)$, $h = 4$; if ($\gamma_1 \in V(P)$ and $\gamma_4 \in V(Q)$) or ($\gamma_2 \in V(P)$ and $\gamma_3 \in V(Q)$), $h = 5$; otherwise, $h = 6$ (see Fig.5(b)). □

Theorem 7. *There exists an $O(\lceil \log_2 |F| \rceil n^3)$ algorithm for finding a fault-free path P between any two distinct fault-free nodes in B_n with a faulty node set $F \subset V(B_n)$ and $|F| \leqslant 2n-3$.*

Proof. Considering fault-tolerant one-to-one routing for the given two distinct nodes u and v in $B_n - F$ with a faulty node set $F \subset V(B_n)$ and $|F| \leqslant 2n-3$, we propose an efficient algorithm, `BFRouting`. To simplify the presentation of the proposed routing algorithm, we first introduce two algorithms, namely `BMapping` and `BBinding`, that will be the two core components of the proposed algorithm.

Based on Theorem 5, we provide Algorithm 3, `BMapping`. In line 2, it takes $O(n)$ time to find the node v in $N_G(u) \setminus F$ by using the connection rules given in Definition 3. Thus, we can verify that the time complexity of function `BMapping1` in algorithm `BMapping` is $O(n)$. In lines 6∼10, it takes $O(n^2)$ time to construct a required fault-free path $P = (u, x, v)$ from u into $B^{\bar{i}}_{n-1}$ with $i \in \{0, 1\}$, such that $u \in V(B^i_{n-1})$, $x \in S_n$, and $v \in V(B^{\bar{i}}_{n-1})$. In lines 11∼17, it takes $O(n^3)$ time to construct a required fault-free path $P = (u, x, y, v)$ from u into $B^{\bar{i}}_{n-1}$ with $i \in \{0, 1\}$, such that $u, x \in V(B^i_{n-1})$, $y \in S_n$, and $v \in V(B^{\bar{i}}_{n-1})$. Thus, we can verify that the time complexity of function `BMapping2` in algorithm `BMapping` is $O(n^3)$.

In addition, we propose Algorithm 4, `BBinding` based on Theorem 6. Given two distinct fault-free nodes $u, v \in V(B_n)$, a subgraph $G \subset B_n$, two paths P and Q with $u = P[1]$ and $v = Q[1]$, and a faulty node set $F \subset V(B_n)$ with $|F| \leqslant 2n-3$, we construct a fault-free path from u to v in $B_n - F$, which will be used in algorithm `BFRouting`. In lines 2∼5, 13, and 19 of algorithm `BBinding`, we will analyze the time complexity of algorithm `BBinding` with algorithm `BFRouting` in the next block since `BFRouting` is called in lines 3, 13, and 19 of algorithm `BBinding`. In lines 6 and 7 of algorithm `BBinding`, it takes $O(n)$ time to choose the first common node from two paths P and Q and takes $O(1)$ time to join two sub-paths constructed by P and Q. In lines 10∼12 and 16∼18 of algorithm `BBinding`,

Algorithm 3. `BMapping`

Input: a node $u \in V(B_n)$, three subgraphs H, S, G in B_n, and a faulty node set $F \subset V(B_n)$
Output: a fault-free path from u into G

1: **function** BMapping1(u, G, F)
2: Choose a node $v \in N_G(u)$ such that $u \notin F$;
3: **return** (u, v);
4: **end function**
5: **function** BMapping2(u, H, S, G, F)
6: **for** $v \in N_{S-F}(u)$ **do**
7: **if** $N_G(v) \not\subseteq F$ **then**
8: **return** (u, `BMapping1`(v, G, F));
9: **end if**
10: **end for**
11: **for** $v \in N_{H-F}(u)$ **do**
12: **for** $x \in N_{S-F}(v)$ **do**
13: **if** $N_G(x) \not\subseteq F$ **then**
14: **return** (u, v, `BMapping1`(x, G, F));
15: **end if**
16: **end for**
17: **end for**
18: **end function**

Algorithm 4. `BBinding`

Input: two nodes $u, v \in V(B_n)$, a subgraph $G \subset B_n$, two paths P and Q with $u = P[1]$ and $v = Q[1]$, and a faulty node set $F \subset V(B_n)$
Output: a fault-free path from u to v in $B_n - F$

1: **function** BBinding1(G, F, u, v, P, Q)
2: **if** $V(P) \cap V(Q) = \emptyset$ **then**
3: $S \leftarrow$ `BFRouting`($G, F, P[-1], Q[-1]$);
4: **return** (P, S, Q^{-1});
5: **end if**
6: Find the first common node x from P and Q;
7: **return** (`Path`(P, u, x), `Path`(Q^{-1}, x, v));
8: **end function**
9: **function** BBinding2(G, F, u, v, Q)
10: **if** $u \in V(Q)$ **then**
11: **return** `Path`(Q, u, v);
12: **end if**
13: **return** (`BFRouting`($G, F, u, Q[-1]$), Q^{-1});
14: **end function**
15: **function** BBinding3(G, F, u, v, P)
16: **if** $v \in V(P)$ **then**
17: **return** `Path`(P, u, v);
18: **end if**
19: **return** (P, `BFRouting`($G, F, P[-1], v$));
20: **end function**

it takes $O(1)$ time to return a sub-path constructed by P (resp. Q) directly.

Accordingly, we propose our main algorithm, Algorithm 5, `BFRouting`. Given two fault-free distinct nodes u and v in B_n and a faulty node set $F \subset V(B_n)$ with $|F| \leqslant 2n-3$, we construct a fault-free path from node u to node v in $B_n - F$. Suppose that a path in algorithm `BFRouting` is saved by a doubly linked circular list whose head u and tail v are pointed by two pointers. Furthermore, each node is stored by a tuple.

Algorithm 5. `BFRouting`

Input: an n-dimensional BCDC, B_n, a faulty node set $F \subset V(B_n)$ with $|F| \leqslant 2n-3$, and two nodes $u, v \in V(B_n - F)$
Output: a fault-free path from node u to node v in $B_n - F$

1: **function** BFRouting(B_n, F, u, v)
2: **if** $(u, v) \in E(B_n)$ **then**
3: **return** (u, v);
4: **else if** $n = 2$ **then**
5: **return** (a fault-free path between u and v in $B_n - F$);
6: **else if** $|F| = 0$ **then**
7: **return** `BRouting`(B_n, u, v);
8: **else if** $|F| \geqslant 2n-2$ **then**
9: **return** `BFS`($B_n - F, u, v$);
10: **end if**
11: $F_0 \leftarrow F \cap V(B^0_{n-1})$, $F_1 \leftarrow F \cap V(B^1_{n-1})$;
12: $F_2 \leftarrow F \cap S_n$, $m \leftarrow \min\{|F_0|, |F_1|\}$
13: **for** $i \in \{0, 1\}$ **do**
14: $B_0 \leftarrow B^i_{n-1}$, $B_1 \leftarrow B^{\bar{i}}_{n-1}$, and $B_2 \leftarrow B_n[S_n]$;
15: **if** $u, v \in V(B_0)$ **and** $|F_i| = m$ **then**
16: **return** `BFRouting`(B_0, F_i, u, v);
17: **else if** $u, v \in V(B_0)$ **and** $|F_{\bar{i}}| = m$ **then**
18: $P \leftarrow$`BMapping2`(u, B_0, B_2, B_1, F);
19: $Q \leftarrow$`BMapping2`(v, B_0, B_2, B_1, F);
20: **return** `BBinding1`($B_1, F_{\bar{i}}, u, v, P, Q$);
21: **else if** $u, v \in S_n$ **then**
22: Choose $j \in \{0, 1\}$ such that $|F_j| = m$;
23: $P \leftarrow$`BMapping1`(u, B^j_{n-1}, F);
24: $Q \leftarrow$`BMapping1`(v, B^j_{n-1}, F);
25: **return** `BBinding1`($B^j_{n-1}, F_j, u, v, P, Q$);
26: **else if** $u \in V(B_0)$ **and** $v \in V(B_1)$ **and** $|F_{1-i}| = m$ **then**
27: $P \leftarrow$`BMapping2`(u, B_0, B_2, B_1, F);
28: **return** `BBinding3`(B_1, F_{1-i}, u, v, P);
29: **else if** $u \in V(B_0)$ **and** $v \in V(B_1)$ **and** $|F_i| = m$ **then**
30: $P \leftarrow$`BMapping2`(v, B_1, B_2, B_0, F);
31: **return** `BBinding2`(B_0, F_i, u, v, P);
32: **else if** $u \in V(B_0)$ **and** $v \in S_n$ **and** $|F_i| = m$ **then**
33: $P \leftarrow$`BMapping1`(v, B_1, F);
34: **return** `BBinding2`(B_0, F_i, u, v, P);
35: **else if** $u \in V(B_0)$ **and** $v \in S_n$ **and** $|F_{\bar{i}}| = m$ **then**
36: $P \leftarrow$`BMapping2`(u, B_0, S_n, B_1, F);
37: $Q \leftarrow$`BMapping1`(v, B_1, F);
38: **return** `BBinding1`(B_1, F, u, v, P, Q);
39: **else if** $u \in S_n$ **and** $v \in V(B_0)$ **and** $|F_i| = m$ **then**
40: $P \leftarrow$`BMapping1`(u, B_0, F);
41: **return** `BBinding3`(B_0, F_i, u, v, P);
42: **else if** $u \in S_n$ **and** $v \in V(B_0)$ **and** $|F_{\bar{i}}| = m$ **then**
43: $P \leftarrow$`BMapping1`(u, B_1, F);
44: $Q \leftarrow$`BMapping2`(v, B_0, B_2, B_1, F);
45: **return** `BBinding1`($B_1, F_{\bar{i}}, u, v, P, Q$);
46: **end if**
47: **end for**
48: **end function**

In what follows, we will analyze the time complexity of two algorithms `BFRouting` and `BBinding` as follows. In lines 2~5 of Algorithm `BFRouting`, it takes constant time to construct the required fault-free path. In lines 6 and 7 of algorithm `BFRouting`, it takes $O(n)$ time to construct the required path in fault-free B_n. In lines 8~9 of algorithm `BFRouting`, we construct a fault-free path from node u to node v in $B_n - F$ using the famous BSF function. In lines 11, 12, and 14 of algorithm `BFRouting`, it takes $O(1)$ time to compute F_0 (resp. F_1, F_2, m, B_0, B_1, and B_2).

We use $U(u, v, n)$ to denote the time of finding a fault-free path between u and v in $B_n - F$. Furthermore, we assume that n is sufficiently large. Let

$$T(n) = \max\{U(u, v, n) | u, v \in V(B_n) \setminus F \text{ and } u \neq v\}. \tag{1}$$

Accordingly, we have $T(2) = O(1)$. We can claim the following discussions with respect to n and a faulty node set F for $n \geqslant 3$ and $|F| \leqslant 2n-3$. In lines 15 and 16 of algorithm `BFRouting`, we have

$$T(n) \leqslant T(n-1) + O(1). \tag{2}$$

In lines 17~31, 35~38, and 39~41 of algorithm `BFRouting`, we have

$$T(n) \leqslant T(n-1) + O(n^3). \tag{3}$$

In lines 32~34 and 42~45 of algorithm `BFRouting`, we have

$$T(n) \leqslant T(n-1) + O(n). \tag{4}$$

Thus, based on (1)~(4) and Definition 3, we have

$$\begin{aligned} T(n) &\leqslant \max\{ \sum_{i=1}^{\lceil \log_2 |F| \rceil} O((n-i+1)^3) + \\ &\qquad O(n - \lceil \log_2 |F| \rceil), O(n^2), O(n)\} \\ &\leqslant \max\{O(\lceil \log_2 |F| \rceil n^3), O(n^2), O(n)\} \\ &\leqslant O(\lceil \log_2 |F| \rceil n^3). \end{aligned} \tag{5}$$

Therefore, according to (5), under the worst case, the time complexity of algorithm `BFRouting` is $T(n) \leqslant \lceil \log_2 |F| \rceil n^3$ when $|F| \leqslant 2n-3$. □

In order to analyze the maximal length of the fault-free path constructed by algorithm `BFRouting`, we give the following theorem.

Theorem 8. *The maximal length of the fault-free path constructed by algorithm `BFRouting` is no more than* $6m + \lceil \frac{n-m+1}{2} \rceil + 1$ *if a faulty node set* $|F| \leqslant 2n-3$ *satisfies* $|F| \leqslant 2n-3$ *in the worst case with* $m = \lceil \log_2 |F| \rceil$.

Proof. We use $M(n)$ to denote the length of path P constructed by algorithm `BFRouting` between u and v in $B_n - F$. Clearly, when $|F| = 0$, we have $M(n) \leqslant \lceil \frac{n+1}{2} \rceil + 1$. We can claim the following discussions with respect to n and F for $n \geqslant 3$ and $|F| \leqslant 2n-3$. In lines 15 and 16 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)$. In lines 17∼20 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)+6$. In lines 21∼25 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)+2$. In lines 26∼29 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)+3$. In lines 32∼34 and 39∼41 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)+1$. In lines 35∼38 and 42∼45 of algorithm `BFRouting`, we have $M(n) \leqslant M(n-1)+4$.

Thus, let $m = \lceil \log_2 |F| \rceil$, we have

$$\begin{aligned} M(n) &\leqslant \max\{\sum_{i=1}^{m} 6 + \lceil \frac{n-m+1}{2} \rceil + 1, \lceil \frac{n+1}{2} \rceil + 1\} \\ &\leqslant \max\{6m + \lceil \frac{n-m+1}{2} \rceil + 1, \lceil \frac{n+1}{2} \rceil + 1\} \\ &\leqslant 6m + \lceil \frac{n-m+1}{2} \rceil + 1. \end{aligned} \tag{6}$$

Therefore, according to (6), under the worst case, the maximal length of the fault-free path constructed by algorithm `BFRouting` is $M(n) \leqslant 6m + \lceil \frac{n-m+1}{2} \rceil + 1$ when $|F| \leqslant 2n-3$ and $m = \lceil \log_2 |F| \rceil$. □

6 Simulations

In this section, we focus on simulations of BCDC related to routing. BCDC is conjectured to have high degrees of regularity, high bandwidth, good fault-tolerance, and other nice properties for DCNs.

6.1 Evaluation of BRouting

In Section 4, we discuss the performance of `BRouting`. We use N to denote the number of supported servers and n to denote the number of switch ports. Table 1 computes the average path lengths (*mean*) and the standard deviations (*stdev*) under `BRouting` and the shortest-path routing for BCDCs with different n, respectively. In the experiment, we can find that the expected path length gotten by `BRouting` is equal to the value computed by the shortest-path routing, while `BRouting` is much simpler than the shortest-path routing.

Fig.6 demonstrates the simulation results of diameters for different sized BCDCs and BCubes, where the two structures utilize the dual-port configuration of servers. It can be seen from Fig.6 that BCDC owns a smaller diameter compared with BCube regardless of the network size.

Table 1. Mean Value and Standard Deviation of Path Length in Shortest-Path Routing and `BRouting`

n	N	Shortest-Path		`BRouting`	
		mean	*stdev*	*mean*	*stdev*
3	12	1.67	0.54	1.67	0.54
4	32	2.11	0.72	2.11	0.72
5	80	2.49	0.77	2.49	0.77
6	192	2.90	0.86	2.90	0.86
7	448	3.26	0.88	3.26	0.88
8	1 024	3.66	0.96	3.66	0.96
9	2 304	4.00	0.97	4.00	0.97
10	5 120	4.40	1.05	4.40	1.05

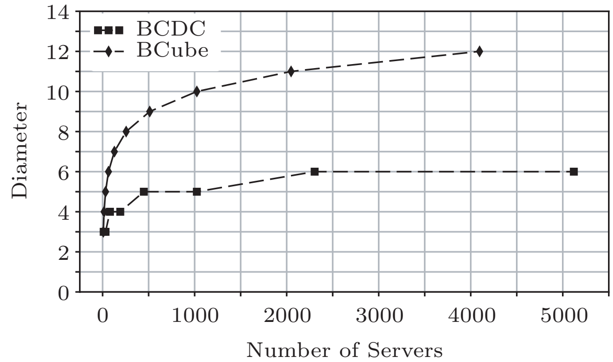


Fig.6. Diameter difference between BCDC and BCube when utilizing the dual-port configuration of servers.

6.2 Path Failure Ration Under Server/Switch Failures

In this subsection, we use simulations to evaluate the performance of fault-tolerant one-to-one routing in BCDC networks. In our simulations, different types of failures are randomly generated. The results are obtained by averaging over 20 simulation runs.

We consider the path failure ration of fault-tolerant routing under various server/switch failure ratios. This is to emulate the performance of fault tolerant one-to-one routing in BCDC. In a large data center, both servers and switches are facing failures that cannot be fixed immediately. We are interested in BCDC's performance to see whether our fault-tolerant one-to-one routing works well under high server/switch failure ratios.

In our simulations, we use a 9-dimensional BCDC with 2 304 servers and 512 switches. The normal link rate is 1 Gb/s for links between servers and switches, while the high-speed link rate is 10 Gb/s for switches.

Fig.7 plots the path failure ration under various server failure ratios versus that under various switch failure ratios in B_8. Then, Fig.7 shows the path failure ratio as the server/switch failure ratios vary from 0% to 20% in B_8. The result demonstrates that BCDC has good fault-tolerance even when the server/switch failure ratio is as high as 20%. Particularly, the path failure ratio achieves 9.7% in B_8 while the bound of server failure ratios is 10%. Moreover, Fig.7 shows that the path failure ratio of B_8 under the server failures performs even better as dimension n gets larger.

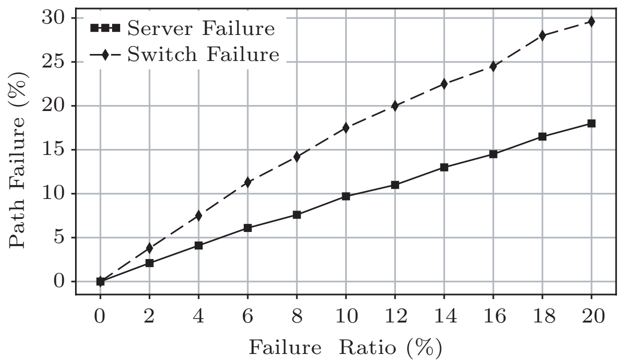


Fig.7. Path failure ration under various server/switch failure ratios in B_8.

We see that the path failure ration increases with the server/switch failure ratios. However, the path failure ratio is nearly 0% when the failure ratio is lower than 1%. This is because very few nodes are disconnected from the graph (indicating the robustness of our BCDC structure). Furthermore, the path failure ratio under server/switch failures cannot achieve such performance since it is not globally optimal when the failure ratio is higher than 20%. Besides, the path failure ratios under the server failures are smaller than those under the switch failure ratios, and smaller than 18.2% (resp. 29.6%) under the server (resp. switch) failure ratio 20%. From the information above, our result demonstrates that the performance of robustness is excellent in our BCDC structure while the server/switch failure ratios are as high as 20%.

6.3 ABT Under Failures

In this subsection, we use simulations to compare the aggregate bottleneck throughput (ABT) of BCDC, BCube[6], and fat-tree[3], under random server and switch failures. In our simulations, different types of failures are randomly generated. The results are obtained by averaging over 20 simulation runs.

For all the three structures, we use 9-port switches to build the network structures. In our simulations, we use a 9-dimensional BCDC with 2 304 servers and 512 switches. The normal link rate is 1 Gb/s for links between servers and switches, while the high-speed link rate is 10 Gb/s for switches. Furthermore, we use 9-port switches to construct the network structures of BCube of DCell. The BCube network we use is a partial $\text{BCube}_{3,9}$ with $n = 9$ that uses three full $\text{BCube}_{2,9}$. The DCell structure is a partial $\text{DCell}_{2,9}$ which contains 25 full $\text{DCell}_{1,9}$ and six full $\text{DCell}_{0,9}$. We use BSR routing for BCube[6] and DFR for DCell[4].

In Section 4, ABT without failures of BCDC is studied. In this simulation, we focus on the case of all-to-all routing in B_9 with 2 304 servers and 512 switches, and evaluate it by randomly choosing servers or switches from the whole BCDC as the failed ones in Fig.8. In BCDC, graceful performance degradation states that when the server or switch failure ratio increases, ABT decreases slowly and there are no dramatic performance falls. For server failures, resulted from either server crash or hardware failure, we find that ABT degrades smoothly for a reasonable failure ratio: for the server failure ratio (resp. switch failure ratios) of 2%, ABT drops by 4% (resp. 5.8%), from 1 152 to 1 106 (resp. from 1 152 to 1 086). ABT under server failure ratio (resp. switch failure ratios) drops by 36.2% (resp. 48.9%) at a high failure ratio of 20%.

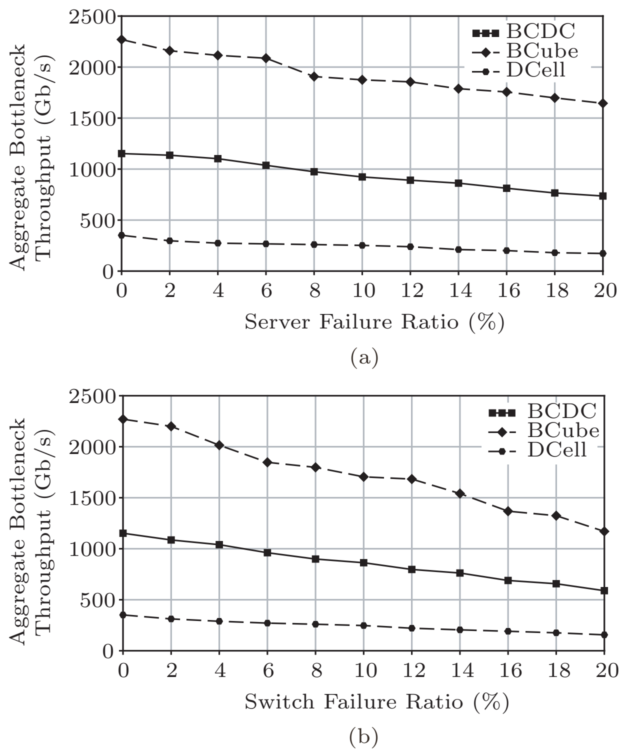


Fig.8. Aggregate bottleneck throughput in BCDC, BCube, and DCell under various (a) server failure ratios and (b) switch failure ratios, respectively.

Switch failure has higher impact on ABT than server failure, and similar phenomena are also observed in [6] for fat-tree, DCell, and BCube networks. In BCDC, a failed switch breaks not only all links for servers connected to it, but also all links using it. Note that in our simulations, the maximum failure ratio of 20% rarely happens in a well managed data center. Therefore, BCDC performs well under server/switch failures.

Compared with DCell, BCDC performs well under both server and switch failures. The result is due to two main reasons. Firstly, the traffic is imbalanced at different levels of links in DCell, and the low-level links of DCell always carry much more traffic flow than high-level links. Secondly, partial DCell makes traffic imbalanced even for links at the same level[4]. Compared with BCube, BCDC performs worse under both server and switch failures. In BCube, servers have more live links under the server/switch failure. Thus, BCube has more balanced traffic than BCDC. Actually, BCube has larger ABT under the server/switch failure model than BCDC and DCell[4,6].

7 Related Work

Data center networks have been extensively studied in cloud computing in recent years[3-6]. In this section, we compare BCDC with several representative DCN architectures. Our comparisons show that BCDC is a significant structure for data centers, due to its high network capacity, good fault-tolerance, and manageable cabling complexity.

Table 2 shows the comparison results. We use N to denote the number of supported servers and n to denote the number of switch ports. The metrics used are: 1) server node degree (degree): small server degree means few links, which leads to small deployment cost; 2) connectivity: high connectivity typically results in high fault-tolerant capacity; 3) network diameter: a small diameter benefits routing applications with real-time requirement; 4) bisection width (BiW): a large BiW shows good fault-tolerance property and high network capacity; 5) aggregate bottleneck throughput (ABT): a large ABT means short finish time in all-to-all jobs.

Switch-centric DCNs use servers to connect a switching fabric, such as tree and fat-tree[3,12]. However, they do not support all-to-all traffic well with existing Ethernet switches[4]. As we show in Table 2, BCDC provides much better support for ABT (all-to-all) than tree. In detail, tree provides the lowest aggregate bottleneck throughput since the throughput is only the capacity of the root switch[6]. Furthermore, compared with fat-tree[3], BCDC provides better one-to-many and one-to-all support and can be directly built using commodity switches without any switch upgrade (see Section 4).

DCell builds complete graphs at each level and scales up with doubly exponential growth. As a result, DCell targets for huge data centers rather than BCDC[4]. However, the traffic in DCell is imbalanced: the level-0 links carry much higher traffic than other links[4]. Therefore, the bisection width of DCell is smaller than that of BCDC. BCDC does not have performance bottlenecks and provides much higher network capacity.

BCube forms a server-centric architecture, supports various bandwidth-intensive applications by speeding up one-to-x and all-to-all traffic patterns, and exhibits graceful performance degradation as the server and/or switch failure rate increases[6]. BCube provides high network capacity for all-to-all traffic rather than BCDC[6]. However, BCDC has a smaller diameter and utilizes the dual-port configuration existing in most commodity DCN servers.

FiConn[5] is a recursively defined DCN architecture. $FiConn(n, 0)$ consists of n servers and an n-port switch connecting these servers, which is the ba-

Table 2. Comparison of Data Center Network Structures

Structure	Degree	Connectivity	Diameter	BiW	ABT
Tree	1	–	$2\log_{n-1}N$	1	n
Fat-tree	1	–	$2\log_2 N$	$\frac{N}{2}$	N
DCell	$k+1$	$n+k+1$	$< 2\log_n N - 1$	$\frac{N}{4\log_n N}$	$\frac{N}{2^{k'}}^+$
BCube	$k+1$	$(k+1)(n-1)$	$k+1$	$\frac{N}{2}$	$\frac{n(N-1)}{n-1}$
FiConn	$2-\frac{1}{2^k}{}^*$	$n-1$	$\leq 4\log_{\frac{n}{4}} N - 1$	$\geqslant \frac{N}{2^{k+2}}$	$> \frac{N}{2*3^k-1}$
BCDC	2	$2n-2$	$\lceil\frac{\log_2 \frac{N}{n}}{2}\rceil+1$	$\frac{(n-1)N}{n}$	$> \frac{4N}{n}$

Note: *: $FiConn(n,k)$ is an irregular graph, and thus we show the average server node degree; $^+$: k' is smaller than k.

sic construction unit. Let N denote the server number of $FiConn(n,k)$ for $k > 0$, and the number of $FiConn(n,k-1)$'s in an $FiConn(n,k)$ is equal to $\frac{N}{2}+1$. In each $FiConn(n,k-1)$, $\frac{N}{2}$ servers out of the N servers with one port remaining are selected to connect the other $\frac{N}{2}$ $FiConn(n,k-1)$'s using their second ports, each for one $FiConn(n,k-1)$. Compared with FiConn[5], BCDC is a regular graph and can significantly reduce the network complexity.

8 Conclusions

In this paper, we introduced BCDC, a high-performance and server-centric data center network, based on crossed cube. An n-dimensional BCDC defines a recursive network structure. We pointed out that a high-dimensional BCDC is constructed by two low-dimensional BCDCs and a node set. Thus, the number of servers in BCDC grows quickly with BCDC's dimension. The diameter of BCDC is $\lceil\frac{n+1}{2}\rceil+1$, which is small. Thus, BCDC can support applications with real-time requirements. The bisection width of BCDC is $(n-1)2^{n-1}$ for $n \geqslant 3$, showing that BCDC may well tolerate server/link faults.

We showed that BCDC significantly accelerates one-to-one, one-to-many, and one-to-all routing and provides high network capacity for all-to-all routing. BCDC also runs its fault-tolerant routing algorithm, `BFRouting`. `BFRouting` performs distributed, fault-tolerant routing without using global states and has good performance. Moreover, BCDC offers high network capacity under server/switch failures. In our analysis and simulations, BCDC is an attractive and practical data center network for mega-data centers, due to its high network capacity, good fault-tolerance, and manageable cabling complexity.

Acknowledgment We thank the anonymous reviewers and editors for their valuable suggestions that help to improve the presentation of the paper.

References

[1] Harris D. Ballmer's millionserver claim doesn't seem so crazy. https://gigaom.com/2013/07/17/ballmers-million-server-claim-doesnt-seem-so-crazy/#comments, July 2013.

[2] Dignan L. AWS financials on deck: The road to 3 million servers in operation. http://www.zdnet.com/article/aws-financials-on-deck-the-road-to-3-million-servers-in-operation/, April 2015.

[3] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In *Proc. the ACM SIGCOMM Conf. Data Communication*, August 2008, pp.63-74.

[4] Guo C X, Wu H T, Tan K, Shi L, Zhang Y G, Lu S W. DCell: A scalable and fault-tolerant network structure for data centers. In *Proc. the ACM SIGCOMM Conf. Data Communication*, August 2008, pp.75-86.

[5] Li D, Guo C X, Wu H T, Tan K, Zhang Y G, Lu S W. FiConn: Using backup port for server interconnection in data centers. In *Proc. IEEE INFOCOM*, April 2009, pp.2276-2285.

[6] Guo C X, Lu G H, Li D, Wu H T, Zhang X, Shi Y F, Tian C, Zhang Y G, Lu S W. BCube: A high performance, server-centric network architecture for modular data centers. In *Proc. the ACM SIGCOMM Conf. Data Communication*, August 2009, pp.63-74.

[7] Greenberg A, Hamilton J R, Jain N, Kandula S, Kim C, Lahiri P, Maltz D A, Patel P, Sengupta S. VL2: A scalable and flexible data center network. In *Proc. the ACM SIGCOMM Conf. Data Communication*, August 2009, pp.51-62.

[8] Abu-Libdeh H, Costa P, Rowstron A, O'Shea G, Donnelly A. Symbiotic routing in future data centers. In *Proc. ACM SIGCOMM*, Aug.30-Sept.3, 2010, pp.51-62.

[9] Yu Y, Qian C. Space shuffle: A scalable, flexible, and high-performance data center network. *IEEE Trans. Parallel and Distributed Systems*, 2016, 27(11): 3351-3365.

[10] Zheng K, Wang L, Yang B H, Sun Y, Uhlig S. LazyCtrl: A scalable hybrid network control plane design for cloud data centers. *IEEE Trans. Parallel and Distributed Systems*, 2017, 28(1): 115-127.

[11] Bhuyan L N, Agrawal D P. Generalized hypercube and hyperbus structures for a computer network. *IEEE Trans. Computers*, 1984, C-33(4): 323-333.

[12] Leiserson C E. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, 1985, 34(10): 892-901.

[13] Dally W J. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Computers*, 1990, 39(6): 775-785.

[14] Xiang D, Zhang Y L, Pan Y. Practical deadlock-free fault-tolerant routing in meshes based on the planar network fault model. *IEEE Trans. Computers*, 2009, 58(5): 620-633.

[15] Xiang D. Deadlock-free adaptive routing in meshes with fault-tolerance ability based on channel overlapping. *IEEE Trans. Dependable and Secure Computing*, 2011, 8(1): 74-88.

[16] Lin D, Liu Y, Hamdi M, Muppala J. FlatNet: Towards a flatter data center network. In *Proc. IEEE Global Communications Conf.*, December 2012, pp.2499-2504.

[17] Wang T, Su Z Y, Xia Y, Qin B, Hamdi M. *NovaCube*: A low latency Torus-based network architecture for data centers. In *Proc. IEEE Global Communications Conf.*, December 2014, pp.2252-2257.

[18] Wang T, Su Z Y, Xia Y, Liu Y, Muppala J, Hamdi M. SprintNet: A high performance servercentric network architecture for data centers. In *Proc. IEEE Int. Conf. Communications*, June 2014, pp.4005-4010.

[19] Wang T, Su Z Y, Xia Y, Muppala J, Hamdi M. Designing efficient high performance server-centric data center network architecture. *Computer Networks*, 2015, 79: 283-296.

[20] Wang T, Su Z Y, Xia Y, Hamdi M. CLOT: A cost-effective low-latency overlaid Torus-based network architecture for data centers. In *Proc. IEEE Int. Conf. Communications*, June 2015, pp.5479-5484.

[21] Li D W, Wu J, Liu Z Y, Zhang F. Towards the tradeoffs in designing data center network architectures. *IEEE Trans. Parallel and Distributed Systems*, 2017, 28(1): 260-273.

[22] Efe K. A variation on the hypercube with lower diameter. *IEEE Trans. Computers*, 1991, 40(11): 1312-1316.

[23] Cull P, Larson S M. The Möbius cubes. *IEEE Trans. Computers*, 1995, 44(5): 647-659.

[24] Abraham S, Padmanabhan K. The twisted cube topology for multiprocessors: A study in network asymmetry. *Journal of Parallel and Distributed Computing*, 1991, 13(1): 104-110.

[25] Fan J X, He L Q. BC interconnection networks and their properties. *Chinese Journal of Computers*, 2003, 26(1): 84-90. (in Chinese)

[26] Wang D J. Hamiltonian embedding in crossed cubes with failed links. *IEEE Trans. Parallel and Distributed Systems*, 2012, 23(11): 2117-2124.

[27] Kulasinghe P, Bettayeb S. Embedding binary trees into crossed cubes. *IEEE Trans. Computers*, 1995, 44(7): 923-929.

[28] Fan J, Lin X, Jia X. Optimal path embedding in crossed cubes. *IEEE Trans. Parallel and Distributed Systems*, 2005, 16(12): 1190-1200.

[29] Efe K. The crossed cube architecture for parallel computation. *IEEE Trans. Parallel and Distributed Systems*, 1992, 3(5): 513-524.

[30] Chang C P, Sung T Y, Hsu L H. Edge congestion and topological properties of crossed cubes. *IEEE Trans. Parallel and Distributed Systems*, 2000, 11(1): 64-80.

[31] Efe K, Blackwell P K, Slough W, Shiau T. Topological properties of the crossed cube architecture. *Parallel Computing*, 1994, 20(12): 1763-1775.

[32] Kulasinghe P D. Connectivity of the crossed cube. *Information Processing Letters*, 1997, 61(4): 221-226.

[33] Fan J X, Jia X H. Edge-pancyclicity and path-embeddability of bijective connection graphs. *Information Sciences*, 2008, 178(2): 340-351.

[34] Yang X F, Dong Q, Tang Y Y. Embedding meshes/tori in faulty crossed cubes. *Information Processing Letters*, 2010, 110(14/15): 559-564.

[35] Zhou S M. The conditional diagnosability of crossed cubes under the comparison model. *International Journal of Computer Mathematics*, 2010, 87(15): 3387-3396.

[36] Dong Q, Zhou J L, Fu Y, Yang X F. Embedding a mesh of trees in the crossed cube. *Information Processing Letters*, 2012, 112(14/15): 599-603.

[37] Cheng B L, Fan J X, Jia X H, Zhang S K. Independent spanning trees in crossed cubes. *Information Sciences*, 2013, 233: 276-289.

[38] Cheng B L, Fan J X, Jia X H, Wang J. Dimension-adjacent trees and parallel construction of independent spanning trees on crossed cubes. *Journal of Parallel and Distributed Computing*, 2013, 73(5): 641-652.

[39] Chen H C, Kung T L, Hsu L Y. 2-disjoint-path-coverable panconnectedness of crossed cubes. *The Journal of Supercomputing*, 2015, 71(7): 2767-2782.

[40] Chen H C, Zou Y H, Wang Y L, Pai K J. A note on path embedding in crossed cubes with faulty vertices. *Information Processing Letters*, 2017, 121: 34-38.

[41] Cheng B L, Wang D J, Fan J X. Constructing completely independent spanning trees in crossed cubes. *Discrete Applied Mathematics*, 2017, 219: 100-109.

[42] Diestel R. Graph Theory (4th edition). Springer, 2010.

[43] Ghemawat S, Gobioff H, Leung S T. The Google file system. In *Proc. the 19th ACM Symp. Operating Systems Principles*, October 2003, pp.29-43.

[44] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 2008, 51(1): 107-113.

Xi Wang received his B.S. degree in management science from Jiangsu University, Suzhou, in 2008. He received his M.S. and Ph.D. degrees in computer science from Soochow University, Suzhou, in 2011 and 2015, respectively. He is currently working as a post-doctor in the School of Computer Science and Technology at Soochow University, Suzhou. His research interests include data center networks, parallel and distributed systems, and interconnection architectures.

Jian-Xi Fan received his B.S., M.S. and Ph.D. degrees in computer science from Shandong Normal University, Jinan, Shandong University, Jinan, and City University of Hong Kong, Hong Kong, in 1988, 1991, and 2006, respectively. He is currently a professor in the School of Computer Science and Technology at Soochow University, Suzhou. He was a visiting scholar in the Department of Computer Science at Montclair State University (May 2017~August 2017) and a senior research fellow in the Department of Computer Science at City University of Hong Kong (May 2012~August 2012). His research interests include parallel and distributed systems, interconnection architectures, data center networks, algorithms, and graph theory.

Cheng-Kuan Lin received his B.S. degree in science applied mathematics from the Chinese Culture University, Taipei, in 2000, and received his M.S. degree in mathematics from "National" Central University, Taipei, in 2002. He obtained his Ph.D. degree in computer science from "National" Chiao Tung University, Hsinchu, in 2011. He is currently an associate professor of computer science at School of Computer Science and Technology at the Soochow University, Suzhou. His research interests include graph theory, design and analysis of algorithms, discrete mathematics, wireless sensor networks, mobile computing, and parallel and distributed computing.

Jing-Ya Zhou received his B.S. and Ph.D. degrees in computer science from Anhui Normal University, Wuhu, and Southeast University, Nanjing, in 2005 and 2013 respectively. He is currently a lecturer with the School of Computer Science and Technology, Soochow University, Suzhou. His research interests include cloud computing, parallel and distributed systems, online social networks, and data center networking.

Zhao Liu received his B.S. degree from Zhengzhou University of Light Industry, Zhengzhou, in 2003, and his M.S. and Ph.D. degrees from Soochow University, Suzhou, in 2006 and 2016, respectively, all in computer science. He is currently an engineer of computer science with the School of Computer Science and Technology at Soochow University, Suzhou. His research interests include parallel and distributed systems, algorithms, and interconnection architectures.