The Communication Performance of BCDC Data Center Network

Shuangxiang Kan, Jianxi Fan*, Baolei Cheng
School of Computer Science and Technology,
Soochow University
Suzhou, China
jxfan@suda.edu.cn

Xi Wang
School of Software and Services Outsourcing,
Suzhou Institute of Industrial Technology
Suzhou, China
wangxi0414@suda.edu.cn

***Abstract*—The BCDC network is a new server-centric data center network based on crossed cube. Its decentralized and recursively defined features solve the bandwidth bottleneck problem of the upper switch in the tree structure and make it more scalable. In addition, the degree of BCDC server is constant, which reduces connection cost and technical difficulty relative to DCell and BCube. In this paper, we study and analyze the data communication, fault tolerance, and node-disjoint paths of BCDC through practical experiments. The results show that BCDC is superior to DCell and Fat-Tree in data communication. Moreover, in terms of fault-tolerant routing and node-disjoint paths, the performance of BCDC is no worse than that of DCell and Fat-Tree. We also propose a one-to-one communication algorithm in BCDC under 1-restricted connectivity, analyze time complexity of algorithm, and give an upper bound of the conditional faulty diameter of BCDC under 1-restricted connectivity. The work in this paper provides an important basis for the design and application of the new data center network.**

Keywords-BCDC; data communication; fault tolerance; node-disjoint paths; data center network; 1-restricted connectivity

I. Introduction

The performance of the data center network largely determines the performance of cloud computing. Therefore, we can improve the performance of cloud computing by building a data center network with good performance. As the size and complexity of data center networks grows, so does the number of servers in data center networks. For example, Google had 450,000 servers in 2006, and by the end of 2010, its number of servers had reached more than 900,000. In a data center network with such a large number of servers, how to connect these servers to form a good performance data center network is a challenge to improve cloud computing performance. There are many factors that determine the performance of a data center network, communication performance and fault tolerance are significant aspects of them:

Communication performance: Many applications deployed on data center networks, such as web page retrieval, games, etc., have much more traffic between servers than traffic with external customers. These applications are all implemented through data communication between servers. Therefore, a good data center network should be able to support some typical data transmission methods, such as one-to-one, one-to-many, one-to-all, and all-to-all, etc. On the other hand, improving data transmission bandwidth is also the goal of data center network design.

Fault tolerance: Due to the large number of switches and servers in the data center network, it is difficult to avoid the failure of switches, servers, and links. Therefore, in the application of the data center network, it must be considered how to make the network work normally when some resources (servers, switches, or links) fail, that is, the data center network must consider fault tolerance.

Scalability and cost are also two important aspects of evaluating data center network. With the continuous expansion of data scale, good scalability and low cost will effectively promote the expansion of data center network and the processing of data.

In order to meet the above requirements, various data center networks have been proposed so far [4]–[9]. Such as traditional tree structure, Fat-Tree [4], [11], DCell [6], [10], and BCube [5], etc. However, these factors are mutually influential and mutually restrictive. For example, although the Fat-Tree is simple in structure and easy to implement, its tree structure makes it weak in fault tolerance, uneven bandwidth distribution, insufficient scalability, and high cost. It cannot support one-to-many and many-to-many network communication well; The data center network represented by DCell has better routing performance and high scalability. However, when the switches become faulty, the length of the fault-tolerant path between servers will increase significantly.

A new data center network based on crossed cube was proposed in [13], which is denoted by BCDC. An n-dimensional BCDC can be constructed recursively from two $(n-1)$-dimensional BCDCs. In an n-dimensional BCDC, each server has 2 ports connected to 2 switches, and each switch is equipped with n ports connected to n servers. Because each server is equipped with two ports, the reliability of BCDC will not be greatly affected even if one of the ports becomes faulty.

A lot of theoretical work has been done on BCDC network, so in this paper, we mainly focus on the experimental research of BCDC properties. This will provide an important basis for the deployment possibility of BCDC. The experiments in this paper mainly use the algorithm in [13] to verify the performance of BCDC's data communication, fault tolerance, and node-disjoint paths. So in order to better understand our experiments, we first introduce the algorithms to be used in our experiments. At the same time, we propose one-to-one communication algorithm in BCDC under 1-restricted connectivity, which shows that BCDC can communicate normally when the number of faulty nodes does not exceed $3n-5$ if any node has at least one fault-free neighbor in BCDC. Then we carry out corresponding experiments on data communication, fault tolerance and node-disjoint paths of 3-dimensional BCDC. Besides, we also conducted comparison experiments on two other famous data center networks, DCell and Fat-Tree. The experimental results show that BCDC has superior data communication performance, and its performance in fault tolerance and node-disjoint paths is not inferior to DCell and Fat-Tree.

The organization of the paper is as follows, we will give the definition and construction method of BCDC in section II. Then in section III, we introduce the data communication, fault tolerance, and node-disjoint paths algorithms to be used in the experiments and present one-to-one communication algorithm in BCDC under 1-restricted connectivity. In section IV, experiments and analysis of the BCDC data communication, fault-tolerant routing, and node-disjoint paths are given. Section V concludes the paper.

II. Physical and Logical Structure of BCDC

We use the graph G to represent a data center network. The switch can be considered as a transparent network device [6] in the data center network, so the node can be used to represent the server and the edge represents the link between servers in the data center network. We use $V(G)$ and $E(G)$ to represent the node set and edge set of the graph G, respectively. We use $N_G(u)$ to represent all nodes adjacent to node v. If $U \in V(G)$, then $N_G(U) = \bigcup_{u\in U} N_G(u) - U$. Let $u, v \in V(G)$, the distance between u and v in the graph G, denoted by $dist(G,u,v)$, is defined as the length of the shortest path between u and v in the graph G. The diameter of graph G, denoted by $d(G)$, $= \max\{dist(G,u,v) \mid u,v \in V(G) \text{ and } u \neq v\}$. Let $V^{'} \subseteq V(G)$, we use $G[V^{'}]$ to represent the subgraph in G that are derived from the node subset $V^{'}$. $Connectivity$ is the minimum number of nodes that need to be deleted to make a connected graph G become disconnected, denoted by $\kappa(G)$. However, the probability that all nodes adjacent to a node will fail at the same time is very small in reality. Therefore, in this paper, we propose the algorithm for two faulty-free nodes to communicate under the condition of 1-restricted connectivity. The g-$restricted\ connectivity$, denoted by $\kappa_g(G)$, is a minimum set of faulty nodes F, after deleting F, G will become an unconnected graph or a trivial graph, and the degree of each node is at least g.

A path $P = \langle x_0, x_1, \ldots, x_{n-1}\rangle$ of length $n-1$ is a finite non-empty sequence with different nodes such that $(x_i, x_{i+1}) \in E(G)$ for $0 \leq i \leq n-2$. We also use $\langle x_0, \ldots, x_i, Q, x_j, \ldots, x_{n-1}\rangle$ to denote path P, where Q is the subpath $\langle x_{i+1}, x_{i+2}, \ldots, x_{j-1}\rangle$. The reverse of path P is $\langle x_{n-1}, x_{n-2}, \ldots, x_0\rangle$, denoted by P^{-1}. Moreover, $subpath(P, x_i, x_j)$ is used to represent a subpath of P which starts at node x_i and ends at node x_j. We also use $P[i]$ to denote node x_i in $\langle x_0, x_1, \ldots, x_{n-1}\rangle$ and use $P[-2]$ and $P[-1]$ to denote the penultimate and the last nodes in P, respectively.

As mentioned above, we intend to construct a data center network based on the n-dimensional crossed cube, denoted by CQ_n, which has better performance in many aspects than other networks. See [1]–[3] for the specific definition and properties of crossed cube.

BCDC is a data center network based on the crossed cube. We will deploy switches on the nodes of CQ_n and servers on the edges of CQ_n. For the convenience of description, We do not distinguish between each switch and its address. The same is true for each server and its address.

The address of each switch is denoted by a n-bit binary string $x = x_{n-1}x_{n-2}\ldots x_0$. The address of each server is represented as an ordered pair $[x, y]$, where x and y are two n-bit binary strings $x = x_{n-1}x_{n-2}\ldots x_0$ and $y = y_{n-1}y_{n-2}\ldots y_0$. A server $[x, y]$ is connected to a switch u if and only if $u \in \{x, y\}$ and $(x, y) \in E(CQ_n)$. Thus, we derive the original graph A_n of the n-dimensional BCDC network.

Considering that the switch is transparent in the BCDC network, we can give the definition of the logical graph of the BCDC network as follows:

Definition 1. *[13] 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 $n \geq 3$, 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 $f = d$.*

The original and logical graphs of the 3-dimensional BCDC network are shown in Fig. 1.

Then, we give several properties of BCDC.

Theorem 1. *[13] $\kappa(B_n) = 2n-2$.*

Lemma 1. *[13] The diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$.*

Lemma 2. *[12] For any integer $n \geq 3$, $\kappa_1(B_n) = 3n-4$.*

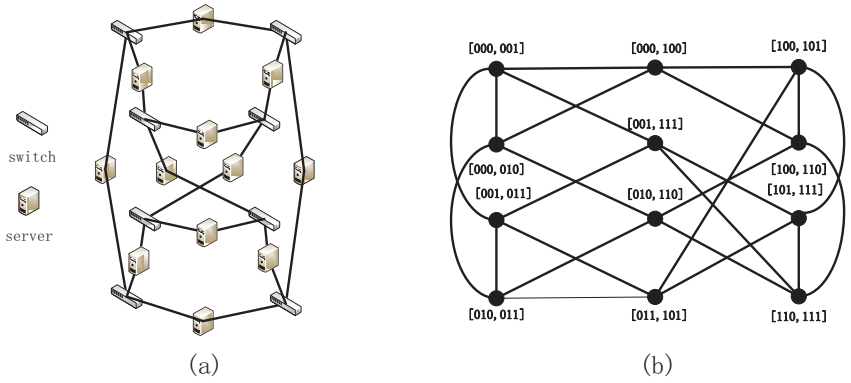


Fig. 1. (a)The original graph of 3-dimensional BCDC A_3, (b)The logical graph of 3-dimensional BCDC B_3.

III. Data Communication, Fault-tolerant Routing, and Node-Disjoint Paths Algorithms of BCDC

A. Fault-Free Routing in BCDC

In this section, we introduce efficient data communication algorithms on BCDC, including one-to-one, one-to-many, one-to-all, and all-to-all algorithms.

1) One-to-One Routing: Efe [3] proposed an algorithm for finding the shortest path between any two nodes in CO_2 with time complexity of $O(n^2)$. Chang et al. [1] improved Efe's algorithm and proposed an algorithm $CSH(CQ_n, u, v)$ with time complexity of $O(n)$. Moreover, they provided a $\rho(u, v)$ function for quickly calculating the distance between any two nodes on CQ_n and $\rho(u, v) = dist(CQ_n, u, v)$. In [13], Wang et al. proposed a algorithm $BRouting$ based on the $CSH(CQ_n, u, v)$ algorithm to find the shortest path of any two nodes in BCDC with a time complexity of $O(n)$. Since any two nodes u, v in BCDC are represented by four nodes $[u_1, u_2]$ and $[v_1, v_2]$ in CQ_n, $BRouting$ algorithm uses the shortest path in $CSH(CQ_n, u_1, v_1)$, $CSH(CQ_n, u_1, v_2)$, $CSH(CQ_n, u_2, v_1)$, and $CSH(CQ_n, u_2, v_2)$ to construct the shortest path between u and v. The length of routing path between u and v constructed by Algorithm $BRouting$ is no more than $\lceil \frac{n+1}{2} \rceil + 1$. The BCDC's one-to-one data communication in our later experiments will be based on the $BRouting$ algorithm.

2) One-to-Many Routing: Wang et al. [13] proposed a algorithm $BMulticast$ with time complexity $O(n|T|^2)$ to realize one to many communication in BCDC, where T is the set of destination nodes. The $BMulticast$ algorithm constructs $|T|$ paths from the source node to the destination nodes by calling BRouting algorithm $|T|$ times, and merges the common nodes and paths to construct a multicast tree. The height of the multicast tree constructed by algorithm $BMulticast$ is no more than $\lceil \frac{n+1}{2} \rceil + 1$.

3) One-to-All(Broadcast) and All-to-All Routing: For the broadcasting algorithm in BCDC, we will use $BMulticast$ algorithm to implement it, using one of the nodes in BCDC as the source node and the remaining nodes as destination nodes, and using $BMulticast$ to construct a broadcast tree to implement data transmission. Since the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$, then the height of broadcast tree constructed by Algorithm $BMulticast$ is no more than $\lceil \frac{n+1}{2} \rceil + 1$. For full exchange communication, we also use the $BMulticast$ algorithm to construct a broadcast tree rooted at each node in B_n. Then, each node in B_n performs data transmission along the broadcast tree rooted at it, thereby implementing all-to-all communication.

B. Fault-Tolerant Routing and One-to-One Communication under 1-Restricted Connectivity in BCDC

Assume that there is a faulty nodes set F in BCDC network, then how to find a shortest fault-free path between any two fault-free nodes in a short period of time is an important issue in BCDC that need to be addressed.

Wang et al. [13] proposed an algorithm $BFRouting$ for finding a one-to-one fault-free path in BCDC when $|F| \leq 2n-3$. This is also the algorithm we used in our experiments. However, we also observe that [13] has the premise that it assumes that all adjacent nodes of a node may fail at the same time, but this has a small probability of appearing in the actual running environment of the data center network. Therefore, in this paper, we propose a one-to-one communication algorithm $BRFTRouting$ for BCDC under 1-restricted connectivity. There are many similarities between the $BFRouting$ algorithm and the $BRFTRouting$ algorithm. The main difference between the two algorithms is the number of faulty nodes and the limitation of node fault conditions. The number of faulty nodes of 1-restricted connectivity is $|F| \leq 3n-5$ and each node has at least one fault-free neighbor.

Theorem 2. *[12] Let $u, v \in V(B_n)$ and $(u, v) \in E(B_n)$, then $N_{B_n}(\{u, v\}) = 3n-4$.*

Given a fault-free node u of B_n, three subgraphs S_1, S_2, and S_3 of B_n and a faulty node set F with $|F| \leq 3n-5$, we give an algorithm, named by $BFinding$, to construct a fault-free path from u to subgraph S_3. Since the time complexity of finding a fault-free neighbor of u in B_n is $O(n)$, we can easily get the time complexity of $BFinding$ is $O(n^3)$.

Algorithm 1 BFinding

Input: a fault-free node $u \in V(B_n)$, three subgraphs S_1, S_2, S_3 in B_n, and a faulty node set $F \subset V(B_n)$ with $|F| \leq 3n-5$, which does not contain all neighboring nodes of any node.
Output: a fault-free path from u to S_3.

function BFINDING1(u, S_3, F)
  for $v \in N_{S_3}(u)$ **do**
    if $v \notin F$ **then**
      return (u, v);
    end if
  end for
end function
function BFINDING2(u, S_1, S_2, S_3, F)
  for $v \in N_{S_2-F}(u)$ **do**
    if $N_{S_3}(v) \not\subseteq F$ **then**
      return $(u,$ BFinding1$(v, S_3, F))$;
    end if
  end for
  for $x \in N_{S_1-F}(u)$ **do**
    for $v \in N_{S_2-F}(x)$ **do**
      if $N_{S_3}(v) \not\subseteq F$ **then**
        return $(u, x,$ BFinding1$(v, S_3, F))$;
      end if
    end for
  end for
  for $x \in N_{S_1-F}(u)$ **do**
    for $v \in N_{S_1-F}(x)$ **do**
      for $w \in N_{S_2-F}(v)$ **do**
        if $N_{S_3}(w) \not\subseteq F$ **then**
          return $(u, x, v,$ BFinding1$(w, S_3, F))$;
        end if
      end for
    end for
  end for
end function

In the following $BRFTRouting$ algorithm, we will call $BFinding$ algorithm and $BFRouting$ algorithm in [13] to construct a fault-free path from u to v in $B_n - F$. In line 9 of the algorithm $BFTRouting$, we use BFS to represent breadth-first search.

Algorithm 2 BRFTRouting

Input: an n-dimensional BCDC, B_n, a faulty node set $F \subset V(B_n)$ with $|F| \leq 3n-5$, which does not contain all neighboring nodes of any node and two fault-free node u, v.
Output: a fault-free path from node u to v in $B_n - F$.

1: **function** BRFTROUTING(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 from u to v in $B_n - F$);
6: **else if** $|F| = 0$ **then**
7: **return** BRouting(B_n, u, v);
8: **else if** $|F| \geq 3n-4$ **then**
9: **return** BFS$(B_n - F, u, v)$;
10: **end if**
11: $F_0 \leftarrow F \cap V(B^0_{n-1})$;
12: $F_1 \leftarrow F \cap V(B^1_{n-1})$;
13: $F_2 \leftarrow F \cap S_n$;
14: $m \leftarrow \min\{|F_0|, |F_1|\}$;
15: **for** $i \in \{0, 1\}$ **do**
16: $B_0 \leftarrow B^i_{n-1}$;
17: $B_1 \leftarrow B^i_{n-1}$;
18: $B_2 \leftarrow B_n[S_n]$;
19: **if** $u, v \in V(B_0)$ **and** $|F_i| = m$ **then**
20: **return** BFRouting(B_0, F_i, u, v);
21: **else if** $u, v \in V(B_0)$ **and** $|F_{\bar{i}}| = m$ **then**
22: $P_1 \leftarrow$BFinding2(u, B_0, B_2, B_1, F);
23: $P_2 \leftarrow$BFinding2(v, B_0, B_2, B_1, F);
24: $P_3 \leftarrow$BFRouting$(B_1, F_{\bar{i}}, P_1[-1], P_2[-1])$;
25: **return** $\langle P_1,$ subpath$(P_3, P_3[1], P_3[-2]),\ P_2^{-1}\rangle$;
26: **else if** $u, v \in S_n$ **then**
27: Choose $j \in \{0, 1\}$ such that $|F_j| = m$;
28: $P_1 \leftarrow$BFinding1(u, B^j_{n-1}, F);
29: $P_2 \leftarrow$BFinding1(v, B^j_{n-1}, F);
30: $P_3 \leftarrow$BFRouting$(B^j_{n-1}, F_j, P_1[-1], P_2[-1])$;
31: **return** $\langle P_1,$ subpath$(P_3, P_3[1], P_3[-2]),\ P_2^{-1}\rangle$;
32: **else if** $u \in V(B_0)$ **and** $v \in V(B_1)$ **and** $|F_i| = m$ **then**
33: $P_1 \leftarrow$BFinding2(v, B_1, B_2, B_0, F);
34: $P_2 \leftarrow$BFRouting$(B_0, F_i, u, P_1[-1])$;
35: **return** $\langle$subpath$(P_2, P_2[0], P_2[-2]), P_1^{-1}\rangle$;
36: **else if** $u \in V(B_0)$ **and** $v \in V(B_1)$ **and** $|F_{\bar{i}}| = m$ **then**
37: $P_1 \leftarrow$BFinding2(u, B_0, B_2, B_1, F);
38: $P_2 \leftarrow$BFRouting$(B_1, F_{\bar{i}}, P_1[-1], v)$;
39: **return** $\langle P_1,$ subpath$(P_2, P_2[1], P_2[-1])\rangle$;
40: **else if** $u \in V(B_0)$ **and** $v \in S_n$ **and** $|F_i| = m$ **then**
41: $P_1 \leftarrow$BFinding1(v, B_0, F);
42: $P_2 \leftarrow$BFRouting$(B_0, F_i, u, P_1[-1])$;
43: **return** $\langle$subpath$(P_2, P_2[0], P_2[-2]), P_1^{-1}\rangle$;
44: **else if** $u \in V(B_0)$ **and** $v \in S_n$ **and** $|F_{\bar{i}}| = m$ **then**
45: $P_1 \leftarrow$BFinding2(u, B_0, B_2, B_1, F);
46: $P_2 \leftarrow$BFinding1(v, B_1, F);
47: $P_3 \leftarrow$BFRouting$(B_1, F_{\bar{i}}, P_1[-1], P_2[-1])$;
48: **return** $\langle P_1,$ subpath$(P_3, P_3[1], P_3[-2]),\ P_2^{-1}\rangle$;
49: **else if** $u \in S_n$ **and** $v \in V(B_0)$ **and** $|F_i| = m$ **then**
50: $P_1 \leftarrow$BFinding1(u, B_0, F);
51: $P_2 \leftarrow$BFRouting$(B_0, F_i, P_1[-1], v)$;
52: **return** $\langle P_1,$ subpath$(P_2, P_2[1], P_2[-1])\rangle$;
53: **else if** $u \in S_n$ **and** $v \in V(B_0)$ **and** $|F_{\bar{i}}| = m$ **then**
54: $P_1 \leftarrow$BFinding1(u, B_1, F);
55: $P_2 \leftarrow$BFinding2(v, B_0, B_2, B_1, F);
56: $P_3 \leftarrow$BFRouting$(B_1, F_{\bar{i}}, P_1[-1], P_2[-1])$;
57: **return** $\langle P_1,$ subpath$(P_3, P_3[1], P_3[-2]),\ P_2^{-1}\rangle$;
58: **end if**
59: **end for**
60: **end function**

Since the time complexity of lines $2 \sim 14$ is $O(n^2)$ and the algorithm $BFRouting$ takes $O(\lceil \log_2 |F| \rceil n^3)$ to construct a fault-free path between nodes u and v with $|F| \leq 2n-3$, the time complexity of $BRFTRouting$ is $O(\lceil \log_2 |F| \rceil n^3)$.

Since the diameter of B_n is $\lceil \frac{n+1}{2} \rceil + 1$, the length of routing path between u and v constructed by Algorithm $BFRouting$ is no more than $4\lceil \log_2 |F| \rceil + \lceil \frac{n - \lceil \log_2 |F| \rceil + 1}{2} \rceil + 1$. As a result, an upper bound of the conditional faulty diameter of BCDC under 1-restricted connectivity is $4\lceil \log_2 (3n-5) \rceil + \lceil \frac{n - \lceil \log_2 (3n-5) \rceil + 1}{2} \rceil + 1$.

For switch (link) fault tolerance, we still use the $CSH(CQ_n, u, v)$ algorithm on CQ_n. During the routing process, when selecting the neighboring node v of node u on the routing path, once the switch or link between u and v is faulty, we will select another neighboring node of u, but the selection is still based on the $CSH(CQ_n, u, v)$ algorithm to ensure that the path constructed is shorter.

In hybrid fault tolerance, the server, switch, or link can become faulty. This situation is more complicated and the fault-tolerant model has not been portrayed in theory. In fact, this situation can lead to a fault-free path between a pair of fault-free servers. Therefore, we intend to use the backtracking method for fault-tolerant routing design in order to make the path obtained shorter. We intend to select fault-free servers, links and switches based on the $CSH(CQ_n, u, v)$ algorithm on CQ_n. Backtracking is performed once there is no suitable path to choose from. If the entire backtracking process does

not select a fault-free path at the end, the algorithm outputs no information about the fault-free path.

C. Disjoint Paths in BCDC

By Theorem 1, the connectivity of B_n is $2n-2$. It is known from Merger's theorem that there are $2n-2$ disjoint paths between any two nodes of B_n. By [1], we have the following method to construct $2n-2$ node-disjoint paths.

Since BCDC is a data center network based on crossed cube, we can construct $2n-2$ disjoint paths of BCDC by means of n disjoint paths between two distinct nodes of CO_n. Let u, v be any two distinct nodes in B_n and $G_0 = B^0_{n-1}$, $G_1 = B^1_{n-1}$, $G_2 = B_n[S_n]$, we set $x_1 = u[0]$, $y_1 = u[1]$, $x_2 = v[0]$, and $y_2 = v[1]$. Thus, x_1, y_1, x_2 and y_2 are the nodes of crossed cube. Therefore, we can construct $2n-2$ disjoint paths by x_1, y_1, x_2 and y_2 according to the following conditions: (1) $u, v \in V(G_2)$. Without loss of generality, we set $x_1, x_2 \in V(CO^0_{n-1})$ and $y_1, y_2 \in V(CO^1_{n-1})$. Thus, we can construct $n-1$ disjoint paths with x_1, x_2 and y_1, y_2, respectively. (2) $u, v \in V(G_0) \cup V(G_2)$ and $|\{u, v\} \cap V(G_0)| = 1$. Without loss of generality, we set $u \in V(G_0)$ and $v \in V(G_2)$. If $(u, v) \in E(B_n)$, we set $y1 = x2$, Thus, we can construct $n-1$ disjoint paths with x_1, x_2 and x_1, y_2, respectively. If $(u, v) \notin E(B_n)$, we can construct $n-1$ disjoint paths with x_1, x_2 and y_1, y_2, respectively. (3) $u, v \in V(G_1) \cup V(G_2)$ and $|\{u, v\} \cap V(G_1)| = 1$. This condition is similar to (2). (4) $u, v \in V(G_0) \cup V(G_1)$ and $|\{u, v\} \cap V(G_0)| = 1$. We can construct $n-1$ disjoint paths with x_1, x_2 and y_1, y_2, respectively.

IV. Experiments and Evaluation

Although the theoretical analysis and simulation experiments of BCDC's data communication, fault tolerance, and disjoint paths performance have been given in [13], if BCDC can still run well in practical experiment, it can be shown that BCDC is an superior network. Therefore, our practical experiment is an important supplement to evaluate the three aspects of BCDC data communication, fault tolerance, and disjoint paths.

To evaluate the data communication, fault tolerance, and node-disjoint paths performance of BCDC, we build a real BCDC network platform A_3. The platform A_3 contains 12 servers. The server is intended to use a normal PC (Operating System: CentOS Linux 7; Memory: 1.8G GiB; Hard Disk: 500G). One Intel Gigabit PCI-E network adapter (Network Card) is deployed for each server, and the switch uses TP-LINK 8-port Gigabit switch (TL-SG1008PE). Each of these servers uses only two ports on the NIC. The network cable connecting the server and the switch is a twisted pair. In addition, In order to better evaluate the performance of BCDC, we also build two other network platforms DCell$_{1,3}$ and Fat-Tree with 12 servers for comparison experiments. DCell$_{1,3}$ and Fat-Tree have the same experimental configuration as A_3. We use modules such as *socket* and *multithreading* in the Python language to implement the corresponding communication algorithms. The important functions and corresponding descriptions in the programs are as follows:

- *totalTransmissionTime()*: Using the *datetime* module to calculate the total program running time.
- *averageCpuUsageRate()*: Using the *psutil* module to calculate the average CPU usage rate.
- *bcdcRouting()*: Building a one-to-one, fault-free transmission path between two nodes in BCDC.
- *bcdcMulticast()*: Using multithreading to construct multiple one-to-one, fault-free paths in BCDC.
- *bcdcDisjointPaths()*: Using multithreading to construct multiple disjoint paths between two nodes in BCDC.

In order to improve the accuracy of experimental data, all experimental data are the average value of 20 runs.

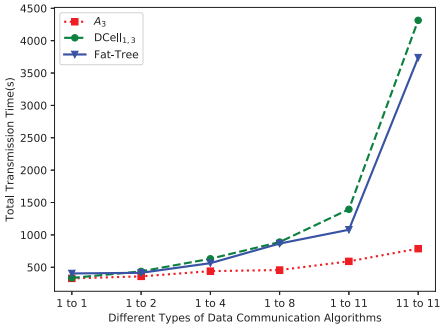
A. Communication performance of A_3, DCell$_{1,3}$, and Fat-Tree

We conduct one-to-one, one-to-two, one-to-four, one-to-eight, one-to-eleven(broadcast), and eleven to eleven(all-to-all) experiments of A_3, DCell$_{1,3}$, and Fat-Tree, respectively. We intend to send 5G data from one source server to all other destination servers and measure the total transmission time and the average CPU usage rate of the server to evaluate the communication performance of these three data center networks. For the specific communication algorithms in BCDC, we refer to the algorithms given in III-A. For DCell, our communication algorithms are based on the literature [6]. The corresponding communication algorithms for Fat-Tree come from [4]. In the experiments, we deploy the transmission program on one of the 12 servers in each data center network, and deploy the receiver programs on the other servers, and collect the experimental results after the data communication. The experimental results are shown in Fig. 2.

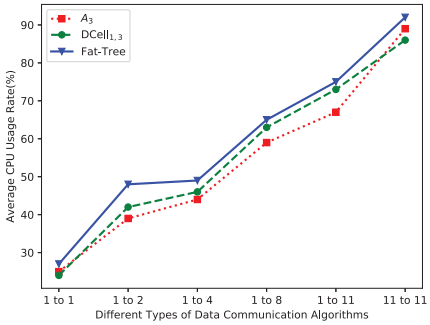
As can be seen from Fig. 2(a), from the perspective of total transmission time, A_3 outperforms DCell$_{1,3}$ and Fat-Tree in data communication performance. Especially for one-on-eight, one-on-eleven, and all to all, the total transmission time of DCell$_{1,3}$ and Fat-Tree is twice or more than the total transmission time of A_3. On the other hand, we can see from Fig. 2(b) that the average CPU usage rate of A_3, DCell$_{1,3}$, and Fat-Tree is almost the same. For example, in the case of all-to-all communication with the highest CPU usage rate, the average CPU usage rates of A_3, DCell$_{1,3}$, and Fat-Tree are 89%, 86%, and 92%, respectively. Therefore, it can still be considered that A_3 has a better routing performance. In summary, A_3 is superior to DCell$_{1,3}$ and Fat-tree in data communication performance.

B. Fault Tolerance of A_3, DCell$_{1,3}$, and Fat-Tree

In this section, we will experiment with fault-tolerant routing for A_3, DCell$_{1,3}$, and Fat-Tree, including server fault tolerance, switch fault tolerance, link fault tolerance, and hybrid fault tolerance. In the fault tolerant experiment of A_3 and DCell$_{1,3}$, the number of faulty servers, faulty switches, faulty links and the number of faulty servers, faulty switches, links in hybrid fault tolerance are 3, 1, 1, (1, 1, 1) and 2, 1, 1, (1, 1, 1). Because the scale of the Fat-Tree we build

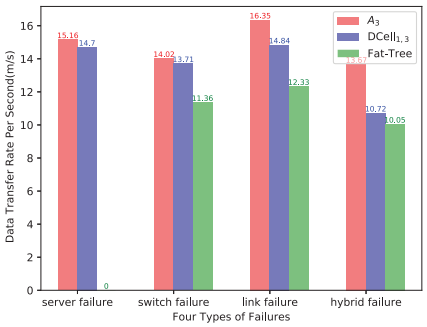


(a) The total transmission time

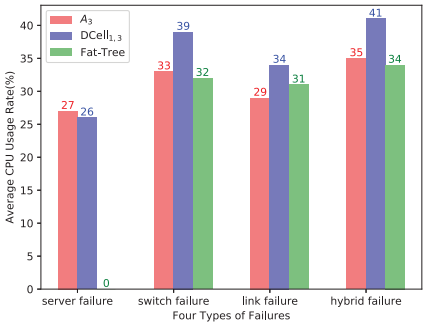


(b) The average CPU usage rate

Fig. 2. The total transmission time and average CPU usage rate of A_3, DCell$_{1,3}$, and Fat-Tree in data communication



(a) The data transfer rate per second



(b) The average CPU usage rate

Fig. 3. The data transfer rate per second and average CPU usage rate of A_3, DCell$_{1,3}$, and Fat-Tree in fault tolerance

is small, we do not distinguish the switches in the core layer, aggregation layer and edge layer in the actual fat tree structure of 12 servers, that is, the switches in the core layer, aggregation layer, and edge layer are TL-SG1008PEs. In addition, because the servers in the Fat-Tree are directly connected to the switches in the edge layer, and the connectivity of the Fat-Tree is 1, we will not carry out fault-tolerant experiments on servers failure like BCDC and DCell. Therefore, In the fault tolerant experiment of Fat-Tree, the number of faulty switches, faulty links and the number of faulty switches, links in hybrid fault tolerance are 1, 1, (1, 1), respectively. We intend to send 5G data from one source server to another destination server and measure the data transfer rate per second and the average CPU usage rate of the server to evaluate the communication performance of these three data center networks. Our fault-tolerant algorithms in BCDC, DCell, and Fat-Tree experiments refer to III-B, [6], and [4], respectively. In the experiment, we deploy the transmission program on one of the 12 servers in each data center network, and deploy the receiver program on another server, and collect the experimental results after data communication. The experimental results are shown in Fig 3.

Since the faulty server in Fat-Tree has no effect on the data communication of other fault-free servers, we have not carried out fault-tolerant experiments of fault server in Fat-Tree. Therefore, we use 0 to replace the experimental results of data transfer rate per second and the average CPU usage rate in the fault-tolerant experiments of fault server in Fat-Tree.

As can be seen from Fig. 3(a), A_3 has faster data transfer rate for four fault-tolerant types than DCell$_{1,3}$ and Fat-Tree. On the other hand, A_3's performance on average CPU usage rate is comparable to that of DCell$_{1,3}$ and Fat-Tree in Fig. 3(b). But on the whole, the experimental results of these three data center networks are not much different. Even in the most complex hybrid fault tolerance, the data transfer rate per second and the average CPU usage rate of A_3, DCell$_{1,3}$, and Fat-Tree are only 13.67m/s, 10.72m/s, 10.05m/s and $35\%, 41\%, 34\%$. This means that A_3 is no worse than DCell$_{1,3}$ and Fat-tree in terms of fault tolerance.

C. Node-Disjoint Paths of A_3 and DCell$_{1,3}$

The third experiment will perform node-disjoint paths experiment on A_3. The connectivity on A_3 is 4. According to Merger's theorem, there are four node-disjoint paths between any two nodes in 3-dimensional A_3. We will arbitrarily select two nodes in the A_3 in the experiment, and transmit 5G data through the four node-disjoint paths between these two nodes, and record the transmission time and the average CPU usage rate(ACUR). Similarly, we will evaluate the node-disjoint paths performance of A_3 through a comparison experiment with DCell$_{1,3}$. The connectivity of DCell$_{1,3}$ is 3, so there are three node-disjoint paths between any two nodes of the DCell. Since the connectivity of Fat-Tree is 1, there is no more than

one node-disjoint paths between any two nodes of Fat-Tree, we don't carry out the node-disjoint paths experiment of Fat-Tree. Our node-disjoint paths algorithms in BCDC and DCell experiments refer to III-C and [15]. In the experiment, we deploy the transmission program on one of the 12 servers in each data center network, and deploy the receiver program on another server, and collect the experimental results after data communication. The experimental results are shown in Table I.

TABLE I
THE TRANSMISSION TIME AND THE AVERAGE CPU USAGE RATE OF NODE-DISJOINT PATHS IN A_3 AND $\text{DCELL}_{1,3}$

Structure	node-disjoint paths	
	time	ACUR
A_3	220s	43%
$\text{DCell}_{1,3}$	245s	46%

It can be seen from Table I, compared with 328s for one-to-one transmission of 5G data between two nodes in Fig. 2(a), it only takes 220s for A_3 to transmit 5G data through four node-disjoint paths between two nodes, which significantly improves the transmission efficiency and effectively improves the data bandwidth. Besides, the performance of node-disjoint paths in A_3 and $\text{DCell}_{1,3}$ is similar in terms of transmission time and average CPU usage rate, which also shows that the data center network BCDC has good performance in terms of node-disjoint paths.

Through the above three experiments, we can know that A_3 has good properties in data communication, fault-tolerant routing and node-disjoint paths. Although the scale of three experiments is not large, it can also reflect that BCDC is a new data center network with great potential.

V. CONCLUSION

In this paper, we introduce the data communication, fault tolerance, and node-disjoint paths algorithms of BCDC and carry out corresponding experiments, and compare them with other two other data center networks, DCell and Fat-Tree. The experimental results show that BCDC is superior to DCell and Fat-Tree in data communication. Moreover, in terms of fault-tolerant routing and node-disjoint paths, the performance of BCDC is no worse than that of DCell and Fat-Tree. At the same time, we propose the algorithm for one-to-one communication under 1-restricted connectivity, analyze the complexity of the algorithm, and give an upper bound of the conditional faulty diameter of BCDC under 1-restricted connectivity, which reflects the good fault tolerance of BCDC. The work in this paper shows that BCDC is a better data center network.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No. 61572337, No. 61972272, No. 61602333, and No. 61702351), the National Natural Science Foundation of China (No. U1905211), the Natural Science Foundation of the Jiangsu Higher Education Institutions of China (No. 18KJA520009), the Priority Academic Program Development of Jiangsu Higher Education Institutions, and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No 17KJB520036.

REFERENCES

[1] C. P. Chang, T. Y. Sung, and L. H. Hsu, "Edge congestion and topological properties of crossed cubes," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 1, pp. 64-80, 2000.
[2] J. Guo, D. Li, and M. Lu, "The g-good-neighbor conditional diagnosability of the crossed cubes under the PMC and MM* model," Theoretical Computer Science, vol. 755, pp. 81-88, 2019.
[3] K. Efe, "A variation on the hypercube with lower diameter," IEEE Transactions on Computers, vol. 40, no. 11, pp. 1312-1316, 1991.
[4] 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.
[5] C. X. Guo, G. H. Lu, D. Li, H. T. Wu, X. Zhang, Y. F. Shi, et al. "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.
[6] C. Guo, H. Wu, K. Tan, L. Shi, Y. G. Zhang, and S. W. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 75-86, 2009.
[7] M. Xu, J. Diakonikolas, E. Modiano, and S. Subramaniam, "A hierarchical WDM-based scalable data center network architecture," arXiv preprint arXiv:1901.06450, 2019.
[8] S. Nasirian, and F. Faghani, "Crystal: A scalable and fault-tolerant Archimedean-based server-centric cloud data center network architecture," Computer Communications, vol. 147, pp. 159-179, 2019.
[9] P. Xie, H. Gu, K. Wang, X. Yu, and S. Ma, "Mesh-of-Torus: a new topology for server-centric data center networks," The Journal of Supercomputing, vol. 75, no. 1, pp. 255-271, 2019.
[10] M. Lv, S. Zhou, X. Sun, G. Lian, and J. Liu, "Reliability evaluation of data center network DCell," Parallel Processing Letters, vol. 28, no. 04, 2018.
[11] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," IEEE transactions on Computers, vol. 100, no. 10, pp. 892-901, 1985.
[12] M. Lv, B. Cheng, J. Fan, X. Wang, J. Zhou, and Y. Wang, "The conditional reliability evaluation of data center network BCDC," unpublished.
[13] X. Wang, J. X. Fan, C.-K. Lin, and J. Y. Zhou, "BCDC: a high-performance, server-centric data center network," Journal of Computer Science and Technology, vol. 33, no. 2, pp. 400-416, 2018.
[14] X. Wang, J. Fan, B. Cheng, J. Zhou, and S. Zhang, "Node-disjoint paths in BCDC networks," Theoretical Computer Science, 2019, unpublished.
[15] X. Wang, J. Fan, C.-K Lin, and X. Jia, "Vertex-disjoint paths in DCell networks," Journal of Parallel and Distributed Computing, vol. 96, pp. 38-44, 2016.