Expansible and Cost-Effective Network Structures for Data Centers Using Dual-Port Servers

Deke Guo, *Member, IEEE,* Tao Chen, *Member, IEEE,* Dan Li, *Member, IEEE,* Yunhao Liu, *Senior Member, IEEE*
Guihai Chen, *Senior Member, IEEE*

***Abstract*— A fundamental goal of data-center networking is to efficiently interconnect a large number of servers with low equipment cost. Several server-centric network structures for data centers have been proposed. They, however, are not truly expansible and suffer low degree of regularity and symmetry. Inspired that the commodity servers in today's data centers come with two built-in NIC ports, we consider how to build expansible and cost-effective structures without expensive high-end switches and additional hardware on servers except the two NIC ports. In this paper, two such network structures called HCN and BCN are designed, both of which are of server degree 2. We also develop low-overhead and robust routing mechanisms for HCN and BCN. Although the server degree is only 2, HCN can be expanded very easily to encompass hundreds of thousands servers with low diameter and high bisection width. Additionally, HCN offers high degree of regularity, scalability and symmetry, which very well conform to a modular design of data centers. BCN is the largest known network structure for data centers with server degree 2 and network diameter 7. Furthermore, BCN has many attractive features, including low diameter, high bisection width, large number of node-disjoint paths for one-to-one traffic, and good fault-tolerance ability. Mathematical analysis and comprehensive simulations show that HCN and BCN possess excellent topology properties and are viable network structures for data centers.**

I. Introduction

Mega data centers have emerged as infrastructures for building online applications, such as web search, email and on-line gaming, as well as infrastructural services, such as GFS [1], HDFS [2], BigTable [3], and CloudStore [4]. Inside a data center, large number of servers are interconnected using a specific data center networking (DCN) [5], [6], [7], [8], [9] structure with design goals. They include low equipment cost, high network capacity, support of incremental expansion, and robustness to link/server/switch failures.

The tree-based structures are increasingly difficult to meet the design goals of DCN [5], [6], [7], [8]. Consequently, a number of novel DCN topologies are proposed recently. These topologies can be roughly divided into two categories. One is switch-centric, which organizes switches into structures other than tree and puts interconnection intelligence on switches. Fat-Tree [5], VL2 [10] fall into this category. The other is server-centric, which puts interconnection intelligence on servers and uses switches only as cross-bars. DCell [6], BCube [9], FiConn [7], [8], and MDCube [11] fall into the second category. Among others, server-centric topology has the following advantages. First, in current practice, servers are more programmable than switches, so the deployment of new DCN topology is more feasible. Second, multiple NIC ports in servers can be used to improve the end-to-end throughput as well as fault tolerance.

For DCell and BCube, many nice topological properties and efficient algorithms have been derived at the following cost. They use more than 2 ports per server, typically 4, and large number of switches and links, so as to scale to a large server population. If they use servers with only 2 ports, the server population is very limited and cannot be enlarged since they are at most two layers. When network structures are expanded to one higher level, DCell as well as BCube add one NIC and link for each existing server, and BCube is appended large number of additional switches. Note that although upgrading servers like installing additional NICs is cheap in terms of the equipment cost, the time and human power needed to upgrade tens or hundreds of thousands servers are very expensive.

Hence, a major drawback of these topologies is that they are not truly expansible. A network is expansible if no changes with respect to node configuration and link connections are necessary when it is expanded. This might cause negative influence on applications running on all existing servers during the process of topology expansion. The fundamental problem is that, we need to design an expansible and cost-effective network structure that works for commodity servers with constant ports and low-end, multi-port commodity switches. Other potential benefits by solving the problem are multifaceted. First, we do not use expensive high-end switches, which are widely used today. Second, the wiring becomes relatively easy since only constant server ports are used for interconnection. Third, it offers an easy-to-build test bed at a university or institution since those data-center infrastructures may only be afforded by a few cash-rich companies [7], [8].

A. Motivation and Contributions

Without loss of generality, we focus on the interconnecting of a large number of commodity servers with only two ports since such servers are readily available in current data centers. It is challenging to interconnect a large population of such servers in data centers, because we should also guarantee the low diameter and high bisection width. FiConn is one of such kind of topologies, however, suffers low degree of regularity and symmetry, which are desirable to the modular design of distributed systems, involving a large number of computing elements. Note that we also extend our designs to more server ports in Section VI.

In this paper, we first propose a hierarchical irregular compound network, denoted as HCN, which can be expanded independent of the server degree by only adding one link to a few number of servers. Moreover, HCN offers high degree of regularity, scalability and symmetry, which very well conform to a modular design of data centers. Inspired by the smaller network order of HCN than FiConn, we further study the degree/diameter problem [12], [13], [14] to determine desirable structures for data centers, which satisfy the aforementioned design goals and accommodate the largest number of dual-port servers.

Inspired by the smaller network size of HCN than FiConn, we further study the degree/diameter problem [12], [13] in the

scenario of building a scalable server-centric DCN topology using dual ports on a server. The degree/diameter problem determines the largest graphs of given maximum degree and diameter. Specifically, the degree/diameter problem here is to determine desirable DCNs which satisfy the aforementioned design goals and support the largest number of servers under the two constraints as follows. First, the basic building block is n servers that are connected to a n port commodity switch. Second, two basic building blocks are interconnected by a link between two servers each in one building block without connecting any two switches directly. Although many efforts [15], [14] have been given to the degree/diameter problem, the degree/diameter problem of DCN is an open problem.

We propose BCN, a class of Bidimensional Compound Networks for data centers which inherit the advantages of HCN. BCN is a level-i irregular compound graph recursively defined in the first dimension for $i \geq 0$, and a level one regular compound graph in the second dimension. In each dimension, a high-level BCN employs a one lower level BCN as a unit cluster and connects many such clusters by means of a complete graph. A BCN of level one in each dimension is the largest known DCN with server degree 2 and network diameter 7. In this case, the order of DCN is significantly larger than that of FiConn$(n,2)$ with the same server degree and network diameter, irrespective of the value of n. For example, if 48-port switches are used, a BCN of level one in each dimension offers 787,968 servers, while a level-2 FiConn only supports 361,200 servers. The thirty data centers of Google supports more than 450,000 servers [16]. Besides these advantages, BCN has other attractive properties, including low diameter and cost, high bisection width, high path diversity for one-to-one traffic, good fault-tolerance ability, and relative shorter fault tolerant path than FiConn.

The major contributions of this paper are summarized as follows. First, we propose two novel design methodologies of HCN and DCN by exploiting the compound graph. They possess good regularity and expansibility that help reduce the cost of further expansions, and are especially suitable for large-scale data centers. Second, a BCN of level one in each dimension offers the largest known network structure for data centers with server degree 2 and diameter 7. Third, HCN and BCN use distributed fault-tolerant routing protocols to handle those representative failures in data centers.

B. Organization of Paper

The rest of this paper is organized as follows. Section II introduces the related work. Section III describes the structures of HCN and BCN. Section IV presents the general and fault-tolerant routing of HCN and BCN. Section V evaluates the topology properties and routing protocols in HCN and BCN through analysis and simulations. Section VI discusses other three important design issues in HCN and BCN. Finally, section VII concludes this paper and future work.

II. Related work

A. Ways of Constructing Large Interconnection Networks

Hierarchical network is a natural way to construct large networks, where many small basic networks in the lower level are interconnected with higher level constructs. In a hierarchical network, lower level networks support local communication, while higher level networks support remote communication. Many schemes have been proposed to construct large networks, including overlay, join, product, composition, compound, and complete bipartite graphs [17]. Among such schemes, The compound graph is observed to be suitable for large-scale systems due to good regularity and expansibility [17].

Definition 1: Given two regular graphs G and G_1, a level-1 regular compound graph $G(G_1)$ is obtained by replacing each node of G by a copy of G_1 and replacing each link of G by a link which connects corresponding two copies of G_1.

A level-1 regular compound graph $G(G_1)$ employs G_1 as a unit cluster and connects many such clusters by means of a regular graph G. In the resultant graph, the topology of G is preserved and only one link is inserted to connect two copies of G_1. An additional remote link is associated to each node in a cluster. For each node in the resultant network, the degree is identical. A *constraint* must be satisfied for the two graphs to constitute a regular compound graph. The node degree of G must be equal to the number of nodes in G_1. An *irregular compound graph* is obtained while the order of G_1 is not necessarily equal to the node degree of G.

A level-1 regular compound graph can be extended to level-i ($i \geq 2$) recursively. For easy of explanation, we consider that case that the regular G_2 is a complete graph. A level-2 regular compound graph $G^2(G_1)$ employs $G(G_1)$ as a unit cluster and connects many such clusters using a complete graph. More generically, a level-i ($i>0$) regular graph $G^i(G_1)$ adopts a level-$(i-1)$ regular graph $G^{i-1}(G_1)$ as a unit cluster and connects many such clusters by a complete graph. Consequently, the node degree of a level-i regular compound graph increases by i than the node degree of G_1. In addition, $G^0(G_1)=G_1$.

B. Interconnection Structures for Data Centers

We now discuss representative interconnection structures for data centers, including the tree-based structure, and four recent proposals of Fat-Tree [5], DCell [6], FiConn [8], and BCube [9].

For the tree-based structure, servers are connected by commodity switches at the first level. At the second level, the commodity switches are connected using core switches. At the next higher level, more expensive and higher-speed switches are employed to connect the next lower level switches. It is well-known that this kind of tree structure does not scale well due to many limitations. For example, a high level switch is the bandwidth bottleneck and suffers single point of failure for a subtree branch. Although redundant switches try to address this challenging problem at the some extent by incurring even higher cost, it does not fundamentally solve the problem.

Fig.1 illustrates an example of a Fat-Tree structure with n=4 and three levels of switches. At the core level, there are $(n/2)^2$ n-port switches each of which has one port connecting to one of

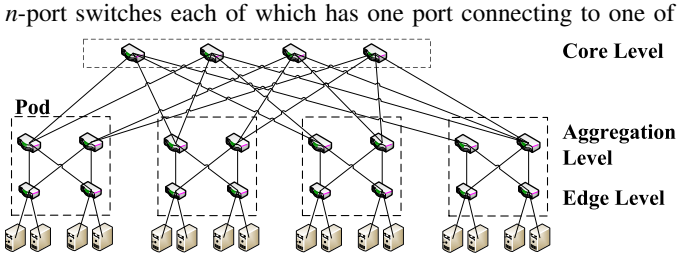


Fig. 1. Fat-Tree structure with $n = 4$. It has three levels of switches.

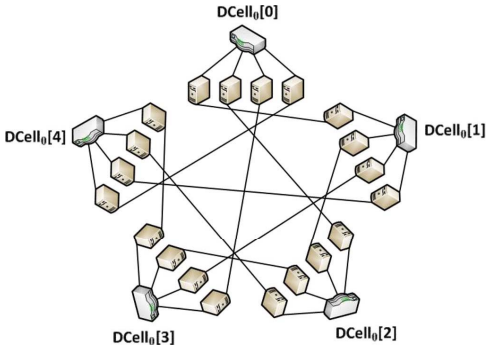


Fig. 2. DCell_1 structure with $n=4$. It is composed of five DCell_0's.

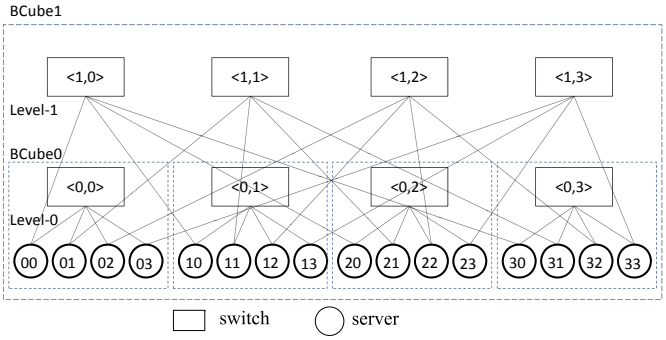


Fig. 3. BCube structure with $n=4$. It is composed of four BCube_0's and another level of four switches.

n pods, each containing two level of $n/2$ switches, i.e., the edge level and the aggregation level. Each n-port switch at the edge level uses its half ports to connect $n/2$ servers and another half ports to connect the $n/2$ aggregation level switches in the pod. Thus, the Fat-Tree structure can support $n^3/4$ servers.

HCN and BCN use only one lowest level of switches by putting the interconnection intelligence on servers, hence the number of used switches is much smaller than Fat-Tree. Therefore, HCN and BCN significantly reduce the cost on switches. In addition, the number of servers Fat-Tree accommodates is limited by the number of switch ports, given the three layers of switches [8]. HCN and BCN do not suffer this limitation and can extend to large number of servers, although each has only two ports.

DCell is a new structure that has many desirable features for data centers. Any high-level DCell is constituted by connecting given number of the next lower level Dcells. Dcells at the same level are fully connected with one another. $DCell_0$ is the basic building block (module) in which n servers are connected to a n ports commodity switch. $DCell_i$ is a level-i regular compound graph $G^i(DCell_0)$ constituted recursively for any $i \geq 1$. Fig.2 illustrate an example of DCell_1 with n=4.

HCN and BCN are also server-centric structures like DCell, but they are different in several aspects. First, the server node degree in a DCell_k is $k+1$, but that of HCN and BCN are always 2. Consequently, the wiring cost is less than that of DCell since each server uses only two ports. Second, no other hardware cost is introduced on a server in HCN and BCN since they use existing backup port on each server for interconnection. If DCell uses servers with only 2 ports, the server population is very limited since DCell is at most two layers. Third, when network structures are expanded to one higher level, DCell adds one NIC and wiring link for each existing server, while HCN and BCN only append one wiring link to a constant number of servers. That is, DCell is not truly expansible while HCN and BCN are.

FiConn shares the similar design principle as HCN and BCN to interconnect large number of commodity servers with only two ports, but they are different in several aspects. First, the topology of FiConn suffers low degree of regularity and symmetry, which are desirable to the modular design of large scale distributed systems. Second, FiConn must append one wiring link to more and more servers when it was expanded to higher level topologies, but HCN only appends one wiring link to a constant number of servers during its expansion process. Third, the order of BCN is significantly larger than that of FiConn$(n,2)$ with the server degree 2 and network diameter 7, irrespective of the value of n.

Recently, BCube is proposed as a new server-centric interconnection structure for shipping-container-sized data centers, as shown in Fig.3. A BCube_0 is simply n servers connecting to an n-port switch. A BCube_k ($k \leq 1$) is constructed from n BCube_{k-1}'s and n^k n-port switches. Each server in a BCube_k has $k+1$ ports. Servers with multiple NIC ports are connect to multiple layers of mini-switches, but those switches are not directly connected.

Unlike BCube, HCN and BCN are designed for mega data centers. The server node degree of HCN and BCN is constantly 2, while BCube must allocate each server more NIC ports. In addition, given the same number of servers, BCube uses much more mini-switches and links than HCN and BCN. Actually, BCube is an emulation of the generalized Hypercube [18].

Although HCN and BCN possess many advantages, they may not be able to achieve higher aggregate networking capacity compared to Fat-tree, Dcell, and BCube. In fact, this results from the lesser number of links and switches, which is a tradeoff for low cost and easy wiring. However, this issue can be addressed by the locality-aware mechanism as discussed in Section VI. In our future work, we will better utilize the link capacity by designing traffic-aware routing algorithms to balance the usages of different levels of links.

III. The BCN Network Structure

We propose two expansible network structures, i.e., HCN and BCN, which build scalable and low-cost data centers using dual port servers. For each structure, we start with the physical structure, and then propose the construction methodology. Table I lists the symbols and notations used in the rest of this paper.

A. Hierarchical Irregular Compound Networks

For any given $h \geq 0$, we denote a level-h irregular compound network as $HCN(n,h)$. HCN is a recursively defined structure. A high-level $HCN(n,h)$ employs a low level $HCN(n,h-1)$ as a unit cluster and connects many such clusters by means of a complete graph. $HCN(n,0)$ is the smallest module (basic construction unit), which consists of n servers each with dual-ports and a n-port mini-switch. For each server, its first port is used to connect with the mini-switch while its second port is employed to interconnect with another server in different smallest modules for constituting larger networks. A server is *available* if its second port has not been connected.

$HCN(n,1)$ is constructed using n basic modules $HCN(n,0)$. In $HCN(n,1)$, there is only one link between any two basic modules by connecting two *available* servers that belong to different basic modules. Consequently, for each $HCN(n,0)$ of $HCN(n,1)$ all its servers are associated with a level-1 link except one server which is reserved for the construction of $HCN(n,2)$. Thus, there are n

TABLE I
SUMMARY OF MAIN NOTATIONS

Term	Definition
α	number of master servers in the level-0 BCN
β	number of slave servers in the level-0 BCN
n	$n=\alpha+\beta$ is the number of ports of a mini-switch
h	level of a BCN in the first dimension
γ	level of the unit BCN in the second dimension
$s_h=\alpha^h\beta$	number of slave servers in any given $BCN(\alpha,\beta,h)$
$s_\gamma=\alpha^\gamma\beta$	number of slave servers in $BCN(\alpha,\beta,\gamma)$
$BCN(\alpha,\beta,0)$	level-0 BCN, i.e., the smallest building block
$BCN(\alpha,\beta,h)$	level-h BCN in the first dimension
$G(BCN(\alpha,\beta,h))$	a compound graph uses $BCN(\alpha,\beta,h)$ as G_1 and a complete graph as G
$BCN(\alpha,\beta,h,\gamma)$	a general BCN that always expands in the first dimension while only expands in the second dimension when $h\geq\gamma$
u	order of a $BCN(\alpha,\beta,h)$ in $BCN(\alpha,\beta,h,\gamma)$ from the viewpoint of the second dimension
v	order of a $G(BCN(\alpha,\beta,\gamma))$ in $BCN(\alpha,\beta,h,\gamma)$ from the viewpoint of the first dimension

available servers in $HCN(n,1)$ for further expansion at a higher level. Similarly, $HCN(n,2)$ is formed by n level-1 $HCN(n,1)$, and has n available servers for interconnection at a higher level.

In general, $HCN(n,i)$ for $i\geq 0$ is formed by n $HCN(n,i-1)$, and has n available servers each in a $HCN(n,i-1)$ for further expansion of network. According to Definition 1, $H(n,i)$ acts as G_1 and a complete graph acts as G. Here, $G(G_1)$ produces an irregular compound graph since the number of available servers in $H(n,i)$ is n while the node degree of G is $n-1$. To facilitate the construction of any level-h HCN, we define Definition 2 as follows.

Definition 2: Each server in $HCN(n,h)$ is assigned a label $x_h\cdots x_1x_0$, where $1\leq x_i\leq n$ for $0\leq i\leq h$. Two servers $x_h\cdots x_1x_0$ and $x_h\cdots x_{j+1}x_{j-1}x_j^j$ are connected only if $x_j\neq x_{j-1}$, $x_{j-1}=x_{j-2}=\cdots=x_1=x_0$ for some $1\leq j\leq h$, where $1\leq x_0\leq\alpha$ and x_j^j represents j consecutive x_js. Here, n servers are reserved for further expansion only if $x_h=x_{h-1}=\cdots=x_0$ for any $1\leq x_0\leq n$.

In any level-h HCN, each server achieves a unique label produced by Definition 2 and is appended a link to its second port. Fig.4 plots an example of $HCN(4,2)$ constructed according to Definition 2. $HCN(4,2)$ consists of four $H(4,1)$s representing by the blue area, while $H(4,1)$ has four $H(4,0)$s representing by the red area. The second port of four servers labeled 111, 222, 333, and 444 are reserved for further expansion.

In a level-h HCN, each server recursively belongs to level-0, level-1,level-2,...,level-h HCNs, respectively. Similarly, any lower level HCN belongs to many higher level HCNs. To characterize this property, x_i indicates the order of a $HCN(n,i-1)$, containing a server $x_h\cdots x_1x_0$, among all level-$(i-1)$ HCNs of $HCN(n,i)$ for $1\leq i\leq h$. We further use $x_hx_{h-1}\cdots x_i$ $(1\leq i\leq h)$ as a prefix to indicate the $HCN(n,i-1)$ that contains this server in $HCN(n,h)$. We use server 423 as an example. $x_1=2$ indicates the 2^{th} $HCN(4,0)$ in a $HCN(4,1)$ this server is located at. This $HCN(4,0)$ contains the servers 421, 422, 423, and 444. $x_2=4$ indicates the 4^{th} level-1 HCN in a level-2 HCN that contains this server. Thus, $x_2x_1=42$ indicates the level-0 HCN that contains the server 423 in a level-2 HCN.

We have emphasized two topological advantages, i.e., expansibility and equal degree, with the consideration of easy implementation and low cost. HCN owns the two properties, and can be expanded very easily to any order independent of the node degree. Moreover, HCN offers high degree of regularity,

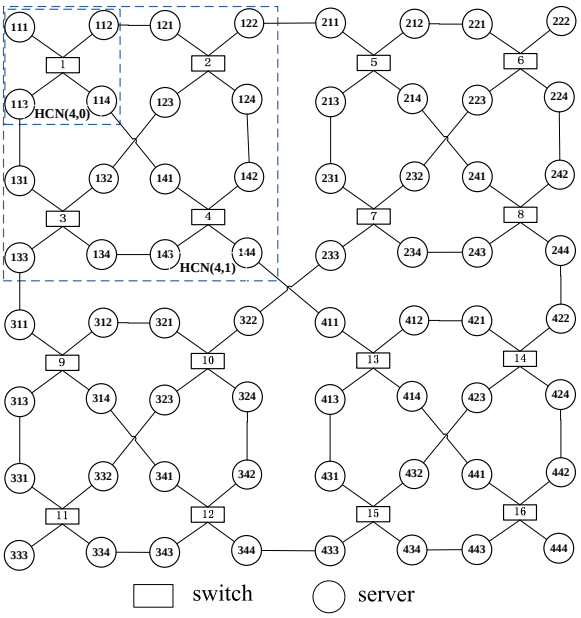


Fig. 4. An illustrative example of $HCN(n,h)$, where $n=4$ and $h=2$.

scalability and symmetry which very well conform to a modular design and implementation of data centers. Inspired by the fact that the order of HCN is less than that of FiConn under the same configurations, we further study the degree/diameter problem of DCN on the basis of HCN.

B. BCN Physical Structure

BCN is a multi-level irregular compound graph recursively defined in the first dimension, and a level one regular compound graph in the second dimension. In each dimension, a high-level BCN employs a one low level BCN as a unit cluster and connects many clusters by means of a complete graph.

Let $BCN(\alpha,\beta,0)$ denote the basic building block, where $\alpha+\beta=n$. It has n servers and a n-port mini-switch. All servers are connected to the mini-switch using their first ports, and are partitioned into two disjoint groups, referred to as the *master* and *slave* servers. Here, servers really do not have master/slave relationship in functionality. The motivation of this partition is just to ease the presentation. Let α and β be the number of master servers and slave servers, respectively. As discussed later, the second port of master servers and slave servers are used to constitute larger BCNs in the first and second dimensions, respectively.

1) Hierarchical BCN in the first dimension: For any given $h\geq 0$, we use $BCN(\alpha,\beta,h)$ to denote a level-h BCN formed by all *master* servers in the first dimension. For any $h>1$, $BCN(\alpha,\beta,h)$ is an irregular compound graph, where G is a complete graph with α nodes while G_1 is $BCN(\alpha,\beta,h-1)$ with α available master servers. It is worth noticing that, for any $h\geq 0$, $BCN(\alpha,\beta,h)$ still has α available master servers for further expansion, and is equivalent to $HCN(\alpha,h)$. The only difference is that each mini-switch also connects β slave servers besides α master servers in $BCN(\alpha,\beta,h)$.

2) Hierarchical BCN in the second dimension: There are β available slave servers in the smallest module $BCN(\alpha,\beta,0)$. In general, there are $s_h=\alpha^h\cdot\beta$ available slave servers in any given $BCN(\alpha,\beta,h)$ for $h\geq 0$. We study how to utilize those available slave servers to expand $BCN(\alpha,\beta,h)$ from the second dimension.

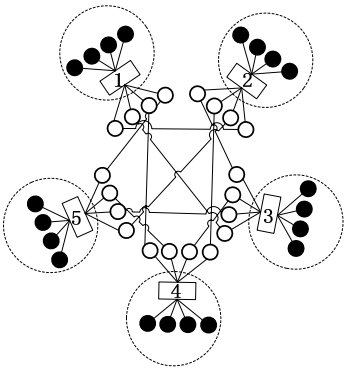


Fig. 5. A $G(BCN(4,4,0))$ structure. It is composed of slave servers in five $BCN(4,4,0)$ in the second dimension.

A level-1 regular compound graph $G(BCN(\alpha,\beta,h))$ is a natural way to realize this goal. It uses $BCN(\alpha,\beta,h)$ as a unit cluster and connects s_h+1 copies of $BCN(\alpha,\beta,h)$ by means of a complete graph using the second port of all available slave servers. The resulting $G(BCN(\alpha,\beta,h))$ cannot be further expanded in the second dimension since it has no available slave servers. It, however, still can be expanded in the first dimension without destroying the existing network.

Theorem 1: The total number of slave servers in any given $BCN(\alpha,\beta,h)$ is

$$s_h = \alpha^h \cdot \beta. \tag{1}$$

Proof: We know that any given $BCN(\alpha,\beta,i)$ is built with α copies of a lower-level $BCN(\alpha,\beta,i-1)$ for $1 \le i$. Thus, it is reasonable that a $BCN(\alpha,\beta,h)$ has α^h smallest module $BCN(\alpha,\beta,0)$. In addition, each smallest module has β slave servers. Consequently, the total number of slave servers in $BCN(\alpha,\beta,h)$ is $s_h = \beta \cdot \alpha^h$. Thus proved. ■

Fig.5 plots an example of $G(BCN(4,4,0))$. All four slave servers connected with a mini-switch in $BCN(4,4,0)$ is the unit cluster. A complete graph is employed to connect five copies of $BCN(4,4,0)$. Consequently, only one remote link is associated with each slave server in a unit cluster. Thus, the degree is two for each slave server in the resultant network.

3) Bidimensional hierarchical BCN: After separately designing $BCN(\alpha,\beta,h)$ and $G(BCN(\alpha,\beta,h))$, we design a scalable Bidimensional BCN formed by both master and slave servers. Let $BCN(\alpha,\beta,h,\gamma)$ denote a Bidimensional BCN, where h denote the level of BCN in the first dimension, and γ denotes the level of a BCN which is selected as the unit cluster in the second dimension.

In this case, $BCN(\alpha,\beta,0)$ consisting of α master servers, β slave servers and a mini-switch is still the smallest module of any level Bidimensional BCN.

To increase servers in data centers on-demand, it is required to expand an initial lower-level $BCN(\alpha,\beta,h)$ from the first or second dimension without destroying an existing structure. A Bidimensional BCN is always $BCN(\alpha,\beta,h)$ as h increases when $h<\gamma$. In this scenario, the unit cluster for expansion in the second dimension has not been formed. When h increases to γ, we achieve $BCN(\alpha,\beta,\gamma)$ in the first dimension and then expand it from the second dimension using the construction method of $G(BCN(\alpha,\beta,\gamma))$ in Section III-B.2. In the resulting $BCN(\alpha,\beta,\gamma,\gamma)$, there are $\alpha^\gamma \cdot \beta$+1 copies of $BCN(\alpha,\beta,\gamma)$ and α available master servers in each $BCN(\alpha,\beta,\gamma)$. A sequential number u is employed to identify a $BCN(\alpha,\beta,\gamma)$ among $\alpha^\gamma \cdot \beta$+1 ones in the second dimension, where u ranges from 1 to $\alpha^\gamma \cdot \beta$+1. Fig.5

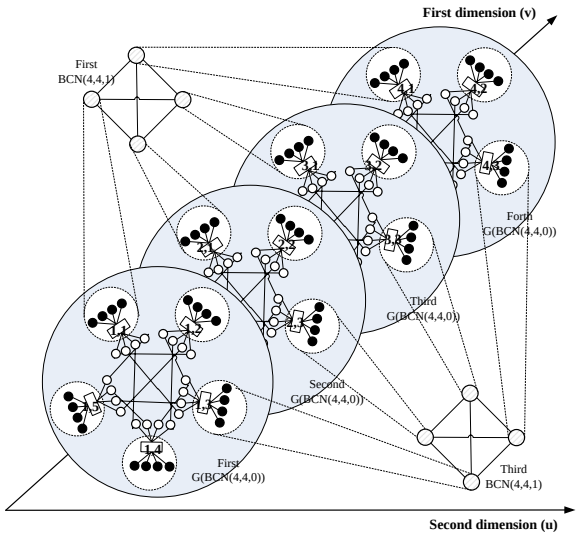


Fig. 6. An illustrative example of $BCN(4,4,1,0)$.

plots an example of $BCN(4,4,0,0)$ consisting of five $BCN(4,4,0)$, where h=r=0. It is worth noticing that a $BCN(\alpha,\beta,\gamma,\gamma)$ cannot be further expanded in the second dimension since it has no available slave servers. It, however, still can be expanded in the first dimension without destroying the existing network by the following approach.

We further consider the case that h exceeds γ. That is, each $BCN(\alpha,\beta,\gamma)$ in $BCN(\alpha,\beta,\gamma,\gamma)$ becomes $BCN(\alpha,\beta,h)$ in the first dimension once h exceeds γ. There are $\alpha^{h-\gamma}$ homogeneous $BCN(\alpha,\beta,\gamma)$ inside each $BCN(\alpha,\beta,h)$. Thus, we use a sequential number v to identify a $BCN(\alpha,\beta,\gamma)$ inside each $BCN(\alpha,\beta,h)$ in the first dimension, where v ranges from 1 to $\alpha^{h-\gamma}$. Thus, the coordinate of each $BCN(\alpha,\beta,\gamma)$ in the resulting structure is denoted by a pair of v and u.

It is worth noticing that only those $BCN(\alpha,\beta,\gamma)$ with v=1 in the resulting structure are connected by a complete graph in the second dimension, and form the first $G(BCN(\alpha,\beta,\gamma))$. Consequently, messages between any two servers in different $BCN(\alpha,\beta,\gamma)$ with the same value of v except v=1 must be relayed by related $BCN(\alpha,\beta,\gamma)$ in the first $G(BCN(\alpha,\beta,\gamma))$. Thus, the first $G(BCN(\alpha,\beta,\gamma))$ becomes a bottleneck of the resulting structure. To address this issue, all $BCN(\alpha,\beta,\gamma)$ with v=i are also connected by means of a completed graph and produce the i^{th} $G(BCN(\alpha,\beta,\gamma))$, for other values of v besides 1. By now, we achieve $BCN(\alpha,\beta,h,\gamma)$ in which each $G(BCN(\alpha,\beta,\gamma))$ is a regular compound graph mentioned in Section II, where G is a complete graph with $\alpha^\gamma \cdot \beta$ nodes and G_1 is a $BCN(\alpha,\beta,\gamma)$ with $\alpha^\gamma \cdot \beta$ available slave servers.

Fig.6 plots a $BCN(4,4,1,0)$ formed by all master and slave servers from the first and second dimensions. Note that only the first and third $BCN(4,4,1)$ are plotted, while other three $BCN(4,4,1)$ are not shown due to page limitations. We can see that $BCN(4,4,1,0)$ has five homogeneous $BCN(4,4,1)$ in the second dimension and four homogeneous $G(BCN(4,4,0))$ in the first dimension. In the resulting structure, the degree of each slave server is two while the degree of each master server is at least one and at most two.

C. The Construction Methodology of BCN

A higher level BCN network can be built by incrementally expansion using one lower level BCN as a unit cluster and

connecting many such clusters by means of a complete graph.

1) In the case of $h<\gamma$: In this case, $BCN(\alpha,\beta,h)$ can be achieved by the construction methodology of $HCN(\alpha,h)$ as mentioned in Section III-A.

2) In the case of $h=\gamma$: As mentioned in Section III-B.2, all slave servers in $BCN(\alpha,\beta,\gamma)$ are utilized for expansion in the second dimension. Each slave server in $BCN(\alpha,\beta,\gamma)$ is identified by a unique label $x=x_\gamma\cdots x_1x_0$ where $1\le x_i\le\alpha$ for $1\le i\le\gamma$ and $\alpha+1\le x_0\le n$. Besides the unique label, each slave server can be equivalently identified by a unique $id(x)$ which denotes its order among all slave servers in $BCN(\alpha,\beta,\gamma)$ and ranges from 1 to s_γ. For each slave server, the mapping between a unique id and its label is bijection defined in Theorem 2. Meanwhile, the label can be derived from its unique id in a reversed way.

Theorem 2: For any slave server $x=x_\gamma\cdots x_1x_0$, its unique id is given by

$$id(x_\gamma\cdots x_1x_0)=\sum_{i=1}^{\gamma}(x_i-1)\cdot\alpha^{i-1}\cdot\beta+(x_0-\alpha). \tag{2}$$

Proof: x_i denotes the order of the $BCN(\alpha,\beta,i-1,\gamma)$ that contains the slave server x in a higher level $BCN(\alpha,\beta,i,\gamma)$ for $1\le i\le\gamma$. In addition, the total number of slave servers in any $BCN(\alpha,\beta,i-1,\gamma)$ is $\alpha^{i-1}\cdot\beta$. Thus, there exist $\sum_{i=1}^{\gamma}(x_i-1)\cdot\alpha^{i-1}\cdot\beta$ slave servers in other smallest modules before the smallest module $BCN(\alpha,\beta,0)$ that contains the server x. On the other hand, there are other $x_0-\alpha$ slave servers that reside in the same smallest module with the server x but has a lower x_0 than the server x. Thus proved. ■

As mentioned in Section III-B.2, the resultant BCN network when $h=\gamma$ is $G(BCN(\alpha,\beta,\gamma))$ consisting of $s_\gamma+1$ copies of a unit cluster $BCN(\alpha,\beta,\gamma)$. In this case, $BCN_u(\alpha,\beta,\gamma)$ denotes the u^{th} unit cluster in the second dimension. In $BCN_u(\alpha,\beta,\gamma)$, each server is assigned a unique label $x=x_\gamma\cdots x_1x_0$ and a 3-tuples $[v(x)=1,u,x]$, where $v(x)$ is defined in Theorem 3. In $BCN_u(\alpha,\beta,\gamma)$, all master servers are interconnected according to the rules in Definition 2 for $1\le u\le s_\gamma+1$.

Many different ways can be used to interconnect all slave servers in $s_\gamma+1$ homogeneous $BCN(\alpha,\beta,\gamma)$ to constitute a $G(BCN(\alpha,\beta,\gamma))$. For any two slave servers $[1,u_s,x_s]$ and $[1,u_d,x_d]$, as mentioned in literatures [19], [20] they are interconnected only if

$$\begin{aligned} u_d &= (u_s+id(x_s)) \bmod (s_\gamma+2)\\ id(x_d) &= s_\gamma+1-id(x_s), \end{aligned} \tag{3}$$

where $id(x_s)$ and $id(x_d)$ are calculated by Formula 2. In literature [6], the two slave servers are connected only if

$$\begin{aligned} u_s &> id(x_s)\\ u_d &= id(x_s)\\ id(x_d) &= (u_s-1) \bmod s_\gamma. \end{aligned} \tag{4}$$

This paper does not focus on designing new interconnection methods of all slave servers since the above two and other permutation methods are all suitable to constitute $G(BCN(\alpha,\beta,\gamma))$. For more information about the two methods, we suggest readers to refer literatures [6], [20].

3) In the case of $h>\gamma$: After achieving $BCN(\alpha,\beta,\gamma,\gamma)$, the resulting network can be incrementally expanded in the first dimension without destroying the existing structure. As discussed in Section III-B.3, $BCN(\alpha,\beta,h,\gamma)$ ($h>\gamma$) consits of $s_\gamma+1$ copies of a unit cluster $BCN(\alpha,\beta,h)$ in the second dimension. Each server in $BCN_u(\alpha,\beta,h)$, the u^{th} unit cluster of $BCN(\alpha,\beta,h,\gamma)$, is assigned a unique label $x=x_h\cdots x_1x_0$ for $1\le u\le s_\gamma+1$. In addition, $BCN_u(\alpha,\beta,h)$ has $\alpha^{h-\gamma}$ $BCN(\alpha,\beta,\gamma)$ in the first dimension. Recall that a sequential number v is employed to rank those $BCN(\alpha,\beta,\gamma)$ in $BCN_u(\alpha,\beta,h)$.

Algorithm 1 Construction of $BCN(\alpha,\beta,h,\gamma)$

Require: $h>\gamma$

1: Connects all servers that have the same u and the common length-h prefix of their labels to the same min-switch using their first ports. {Construction of all smallest modules $BCN(\alpha,\beta,0)$}
2: **for** $u=1$ to $\alpha^\gamma\cdot\beta+1$ **do** {Interconnect master servers that hold the same u to form $\alpha^\gamma\cdot\beta+1$ copies of $BCN(\alpha,\beta,h)$}
3: Any master server $[v(x),u,x=x_h\cdots x_1x_0]$ is interconnected with a master server $[v(x'),u,x'=x_h\cdots x_{j+1}x_{j-1}x_j^j]$ using their second ports if $x_j\neq x_{j-1}$, $x_{j-1}=\cdots=x_1=x_0$ for some $1\le j\le h$, where $1\le x_0\le\alpha$ and a_j^j represents j consecutive a_js.
4: **for** $v=1$ to $\alpha^{h-\gamma}$ **do** {Connect slave servers that hold the same v to form the v^{th} $G(BCN(\alpha,\beta,\gamma))$ in $BCN(\alpha,\beta,h,\gamma)$}
5: Interconnect any two slave servers $[v(x),u_x,x=x_h\cdots x_1x_0]$ and $[v(y),u_y,y=y_h\cdots y_1y_0]$ using their second ports only if (1) $v(x)=v(y)$; (2) $[u_x,x_\gamma\cdots x_1x_0]$ and $[u_y,y_\gamma\cdots y_1y_0]$ satisfy the constraints in Formula 3.

In $BCN_u(\alpha,\beta,h)$, each server $x=x_h\cdots x_1x_0$ is assigned a 3-tuples $[v(x),u,x]$, where $v(x)$ is defined in Theorem 3. A pair of u and $v(x)$ is sufficient to identify the unit cluster $BCN(\alpha,\beta,\gamma)$ that contains the server x in $BCN(\alpha,\beta,h,\gamma)$. For a slave server x, we further assign a unique $id(x_\gamma\cdots x_1x_0)$ to indicates the order of x among all slave servers in the same $BCN(\alpha,\beta,\gamma)$.

Theorem 3: For any server labeled $x=x_h\cdots x_1x_0$ for $h\ge\gamma$, the rank of the module $BCN(\alpha,\beta,\gamma)$ in $BCN(\alpha,\beta,h)$ this server resides in is given by

$$v(x) = \begin{cases} 1, \text{ if } h=\gamma\\ x_{\gamma+1}, \text{ if } h=\gamma+1\\ \sum_{i=\gamma+2}^{h}(x_i-1)\cdot\alpha^{i-\gamma-1}+x_{\gamma+1}, \text{ if } h>\gamma+1 \end{cases} \tag{5}$$

Proof: Recall that any $BCN(\alpha,\beta,i)$ is constructed with α copies of $BCN(\alpha,\beta,i-1)$ for $1\le i$. Therefore, the total number of $BCN(\alpha,\beta,\gamma)$ in $BCN(\alpha,\beta,i)$ for $i>\gamma$ is $\alpha^{i-\gamma}$. In addition, x_i indicates the $BCN(\alpha,\beta,i-1)$ in the next higher level $BCN(\alpha,\beta,i)$ this server is located at for $1\le i$. Thus, there are $(x_i-1)\cdot\alpha^{i-\gamma-1}$ $BCN(\alpha,\beta,\gamma)$ in other x_i-1 previous $BCN(\alpha,\beta,i-1)$ inside $BCN(\alpha,\beta,i)$ for $\gamma+2\le i\le h$. In addition, $x_{\gamma+1}$ indicates the sequence of the $BCN(\alpha,\beta,\gamma)$ in $BCN(\alpha,\beta,\gamma+1)$ the server x resides in. Therefore, the rank of the $BCN(\alpha,\beta,\gamma)$ in $BCN(\alpha,\beta,h)$ this server resides in is given by Formula 5. Thus proved. ■

After assigning a 3-tuples to all master and slave servers, we propose a general procedure to constitute a $BCN(\alpha,\beta,h,\gamma)$ ($h>\gamma$), as shown in Algorithm 1. The entire produce includes three parts. The first part groups all servers into the smallest modules $BCN(\alpha,\beta,0)$ for further expansion. The second part constructs $s_\gamma+1$ homogeneous $BCN(\alpha,\beta,h)$ by connecting the second port of those master servers which have the same u and satisfy the constraints mentioned in Definition 2. Furthermore, the third part connects the second port of those slave servers that have the same v and satisfy the constraints defined by Formula 3. Consequently, the construction produce results in $BCN(\alpha,\beta,h,\gamma)$ consisting of $\alpha^{h-\gamma}$ homogeneous $G(BCN(\alpha,\beta,\gamma))$. Note that it is not necessary that the connection rule of all slave servers must be Formula 3. It also can be that defined by Formula 4.

Algorithm 2 FdimRouting(src, dst)

Require: src and dst are two servers in $BCN(\alpha,\beta,h)$ ($h<\gamma$). The labels of the two servers are retrieved from the inputs, and are $src=s_h s_{h-1}\cdots s_1 s_0$ and $dst=d_h d_{h-1}\cdots d_1 d_0$, respectively.

1: $pref \leftarrow CommPrefix(src, dst)$
2: Let m denote the length of $pref$
3: **if** $m==h$ **then**
4: Return (src, dst) {The servers connect to the same switch.}
5: $(dst1, src1) \leftarrow GetIntraLink(pref, s_{h-m}, d_{h-m})$
6: $head \leftarrow FdimRouting(src, dst1)$
7: $tail \leftarrow FdimRouting(src1, dst)$
8: Return $head + (dst1, src1) + tail$

GetIntraLink($pref, s, d$)

1: Let m denote the length of $pref$
2: $dst1 \leftarrow pref + s + d^{h-m}$ {d^{h-m} represents $h-m$ consecutive d}
3: $src1 \leftarrow pref + d + s^{h-m}$ {s^{h-m} represents $h-m$ consecutive s}
4: Return $(dst1, src1)$

IV. Routing for one-to-one traffic in BCN

One-to-one traffic is the basic traffic model and good one-to-one support also results in good several-to-one and all-to-one support [9]. In this section, we start with single-path routing for one-to-one traffic in BCN without failures of switches, servers, and links. We then study the parallel multi-paths for one-to-one traffic in BCN. Finally, we propose fault-tolerant routing schemes to address those representative failures by employing the benefits of multi-paths between any servers.

A. Single-path for One-to-One Traffic in BCN without Failures

1) In the case of $h<\gamma$: For any $BCN(\alpha,\beta,h)$ ($1\le h$) in the first dimension, we propose an efficient routing scheme, denoted as *FdimRouting*, to find a single-path between any pair of servers in a distributed manner. Let src and dst denote the source and destination servers in the same $BCN(\alpha,\beta,h)$ but different $BCN(\alpha,\beta,h-1)$. The source and destination can be of master server or slave server. The routing scheme first determines the link $(dst1, src1)$ that interconnects the two $BCN(\alpha,\beta,h-1)$ that src and dst are located at. It then derives two sub-paths from src to $dst1$ and from $src1$ to dst. The path from src to dst is the combination of the two sub-paths and $(dst1, src1)$. Each of the two sub-paths can be obtained by invoking Algorithm 2 recursively.

In Algorithm 2, the labels of a pair of servers are retrieved from the two inputs which can be of 1-tuple or 3-tuples. A 3-tuples indicates that this $BCN(\alpha,\beta,h)$ is a component of the entire $BCN(\alpha,\beta,h,\gamma)$ when $h\ge\gamma$. The *CommPrefix* calculates the common prefix of src and dst and the *GetIntraLink* identifies the link that connects the two sub-BCNs in $BCN(\alpha,\beta,h)$. Notice that the two ends of the link can be directly derived from the indices of the two sub-BCNs according to Definition 2. Thus, the time complexity of *GetIntraLink* is $O(1)$.

From *FdimRouting*, we obtain the following theorem. Note that the length of the path between two servers connecting to the same switch is one. This assumption was widely used in the designs of server-centric architectures for data centers, such as DCell, FiConn, BCube, and MDCube [11].

Theorem 4: The longest shortest path length among all the server pairs of $BCN(\alpha,\beta,h)$ is at most $2^{h+1}-1$ for $h\ge 0$.

Proof: For any servers src and dst in $BCN(\alpha,\beta,h)$ but in different $BCN(\alpha,\beta,h-1)$, let D_h denote the length of the single path resulted from Algorithm 2. The entire path consists of two sub-paths in two different $BCN(\alpha,\beta,h-1)$ and one link connects the two lower BCNs. It is reasonable to infer that $D_h=2\cdot D_{h-1}+1$ for $h>0$ and $D_0=1$. We can derive that $D_h=\sum_{i=0}^{h}2^i$. Thus proved. ∎

Algorithm 3 BdimRouting(src, dst)

Require: src and dst are denoted as $[v(s_h\cdots s_1 s_0), u_s, s_h\cdots s_1 s_0]$ and $[v(d_h\cdots d_1 d_0), u_d, d_h\cdots d_1 d_0]$ in $BCN(\alpha,\beta,h\ge\gamma,\gamma)$.

1: **if** $u_s==u_d$ **then** {In the same $BCN(\alpha,\beta,h)$}
2: Return $FdimRouting(src, dst)$
3: $v_c \leftarrow v(s_h\cdots s_1 s_0)$ {v_c can also be $v(d_h\cdots d_1 d_0)$}
4: $(dst1, src1) \leftarrow GetInterLink(u_s, u_d, v_c)$
5: $head \leftarrow FdimRouting(src, dst1)$ {Find a path from src to $dst1$ in the u_s^{th} $BCN(\alpha,\beta,h)$ of $BCN(\alpha,\beta,h,\gamma)$}
6: $tail \leftarrow FdimRouting(src1, dst)$ {Find a path from $src1$ to dst in the u_d^{th} $BCN(\alpha,\beta,h)$ of $BCN(\alpha,\beta,h,\gamma)$}
7: Return $head + (dst1, src1) + tail$

GetInterLink(s, d, v)

1: Infer two slave servers $[s, x=x_h\cdots x_1 x_0]$ and $[d, y=y_h\cdots y_1 y_0]$ from the s^{th} and d^{th} $BCN(\alpha,\beta,h)$ in $BCN(\alpha,\beta,h,\gamma)$ such that (1) $v(x)=v(y)=v$; (2) $[s, x_\gamma\cdots x_1 x_0]$ and $[d, y_\gamma\cdots x_1 x_0]$ satisfy the constraints defined by Formula 3.
2: Return $([s,x],[d,y])$

The time complexity of Algorithm 2 is $O(2^h)$ for deriving the entire path, and can be reduced to $O(h)$ for deriving only the next hop since we only need to calculate one sub-path that contains that next hop.

2) In the case of $h\ge\gamma$: Consider the routing scheme in any $BCN(\alpha,\beta,h,\gamma)$ consisting of $\alpha^\gamma\cdot\beta+1$ copies of $BCN(\alpha,\beta,h)$ for $h\ge\gamma$. The *FdimRouting* scheme can discover a path only if the two servers are located at the same $BCN(\alpha,\beta,\gamma)$. In other cases, Algorithm 2 alone cannot guarantee to find a path between any pair of servers. To handle this issue, we propose *BdimRouting* scheme for the cases that $h\ge\gamma$.

For any pair of servers src and dst in $BCN(\alpha,\beta,h,\gamma)$ ($h\ge\gamma$), Algorithm 3 invokes Algorithm 2 to discover the path between the two servers only if they are in the same $BCN(\alpha,\beta,h)$. Otherwise, it first identifies the link $(dst1, src1)$ that interconnects the $v(src)^{th}$ $BCN(\alpha,\beta,\gamma)$ of $BCN_{u_s}(\alpha,\beta,h)$ and $BCN_{u_d}(\alpha,\beta,h)$. Note that the link that connects the $v(dst)^{th}$ instead of the $v(src)^{th}$ $BCN(\alpha,\beta,\gamma)$ of $BCN_{u_s}(\alpha,\beta,h)$ and $BCN_{u_d}(\alpha,\beta,h)$ is an alternative link. Algorithm 3 then derives a sub-path from src to $dst1$ that are in the $v(src)^{th}$ $BCN(\alpha,\beta,\gamma)$ inside $BCN_{u_s}(\alpha,\beta,h)$ and another sub-path from $src1$ to dst that are in $BCN_{u_d}(\alpha,\beta,h)$ by invoking Algorithm 2. Consequently, the path from src to dst is the combination of the two sub-paths and $(dst1, src1)$.

From *BdimRouting*, we obtain the following theorem.

Theorem 5: The longest shortest path length among all the server pairs of $BCN(\alpha,\beta,h,\gamma)$ ($h>\gamma$) is at most $2^{h+1}+2^{\gamma+1}-1$.

Proof: In Algorithm 3, the entire routing path from src to dst might contain an inter-link between $dst1$ and $src1$, a first sub-path from src to $dst1$ and a second sub-path from $src1$ to dst. The length of the first sub-path is $2^{\gamma+1}-1$ since the two end servers are in the same $BCN(\alpha,\beta,\gamma)$. Theorems 4 shows that the maximum path length of the second sub-path is $2^{h+1}-1$. Consequently, the length of the entire path from src to dst is at most $2^{h+1}+2^{\gamma+1}-1$. Thus proved. ∎

It is worthy noticing that *GetInterLink* can directly derive the end servers of the link only based on the three inputs and the constraints in Formula 3. Thus, the time complicity of *GetInterLink* is $O(1)$. The time complexity of Algorithm 3 is $O(2^k)$ for deriving the entire path, and can be reduced to $O(k)$

for deriving only the next hop.

B. Multi-paths for One-to-One Traffic

Two parallel paths between a source server *src* and a destination server *dst* exist if the intermediate servers on one path do not appear on the other. We will show how to generate parallel paths between two servers.

Lemma 1: There are $\alpha-1$ parallel paths between any *src* and *dst* in a $BCN(\alpha,\beta,h)$ but not in the same $BCN(\alpha,\beta,0)$.

We show the correctness of Lemma 1 by constructing such $\alpha-1$ paths. The construction procedure is based on the single-path routing, FdimRouting, in the case of $h<\gamma$. We assume that $BCN(\alpha,\beta,i)$ is the lowest level BCN that contains the two servers *src* and *dst*. FdimRouting determines the link $(dst1,src1)$ that interconnects the two $BCN(\alpha,\beta,i-1)$ each contains one of the two servers, and then built the first path passing that link. There are α one lower level $BCN(\alpha,\beta,i-1)$ in the $BCN(\alpha,\beta,i)$ containing the *dst*. The first path does not pass other intermediate $BCN(\alpha,\beta,i)$, while each of other $\alpha-2$ parallel paths must traverse one intermediate $BCN(\alpha,\beta,i-1)$.

Let $x_h\cdots x_1x_0$ and $y_h\cdots y_1y_0$ denote the labels of $src1$ and $dst1$, respectively. Now we construct the other $\alpha-2$ parallel paths from *src* to *dst*. First, a server labeled $z{=}z_h\cdots z_1z_0$ is identified as a candidate server of $src1$ only if z_{i-1} is different from x_{i-1} and y_{i-1} while other parts of its label is the same as that of the label of $src1$. It is clear that there exist $\alpha-2$ candidate servers of $src1$. Second, we find a parallel path from *src* to *dst* by building a sub-path from the source *src* to an intermediate server z and a sub-path from z to the destination *dst*. The two sub-paths can be produced by FdimRouting. So far, all the $\alpha-1$ parallel paths between any two servers are constructed. Note that each path is built in a fully distributed manner only based on the labels of the source and destination without any overhead of control messages.

We use Fig.4 as an example to show the three parallel paths between any two servers. The first path from 111 to 144 is 111→114→141→144, which is built by Algorithm 2. Other two paths are 111→113→131→134→143→144 and 111→112→121→124→142→144. We can see that the three paths are node-disjointed and hence parallel.

As for $BCN(\alpha,\beta,\gamma,\gamma)$ with $\alpha^\gamma\beta+1$ copies of $BCN(\alpha,\beta,\gamma)$, if *src* and *dst* reside in the same $BCN(\alpha,\beta,\gamma)$, there are $\alpha-1$ parallel paths between *src* and *dst* according to Lemma 1. Otherwise, we assume *A* and *B* denote the $BCN(\alpha,\beta,\gamma)$ in which *src* and *dst* reside, respectively. In this case, there exist $\alpha^\gamma\beta$ parallel path between *A* and *B* since $BCN(\alpha,\beta,\gamma,\gamma)$ connects $\alpha^\gamma\beta+1$ copies of $BCN(\alpha,\beta,\gamma)$ by means of a complete graph. In addition, Lemma 1 shows that there are only $\alpha-1$ parallel paths between any two servers in $BCN(\alpha,\beta,\gamma)$, such as *A* and *B*. Accordingly, it is easy to infer that Lemma 2 holds.

Lemma 2: There are $\alpha-1$ parallel paths between any two servers in a $BCN(\alpha,\beta,\gamma,\gamma)$ but not in the same $BCN(\alpha,\beta,0)$.

In the case of $h>\gamma$, $BCN(\alpha,\beta,\gamma)$ is the unit cluster of $BCN(\alpha,\beta,h,\gamma)$. Assume *src* and *dst* are labelled as $[v(s_h\cdots s_1s_0),u_s,s_h\cdots s_1s_0]$ and $[v(d_h\cdots d_1d_0),u_d,d_h\cdots d_1d_0]$, and reside in two unit clusters with labels $<v(s_h\cdots s_1s_0),u_s>$ and $<v(d_h\cdots d_1d_0),u_d>$, respectively. According to Lemmas 1 and 2, there are $\alpha-1$ parallel paths between *src* and *dst* if $u_s{=}u_d$ or $v(s_h\cdots s_1s_0){=}v(d_h\cdots d_1d_0)$. In other cases, we select a $BCN(\alpha,\beta,\gamma)$ with label $<v(s_h\cdots s_1s_0),u_d>$ as a relay cluster. As aforementioned, there are $\alpha^\gamma\beta$ parallel paths between the unit clusters $<v(s_h\cdots s_1s_0),u_s>$ and $<v(s_h\cdots s_1s_0),u_d>$, while only $\alpha-1$ parallel paths between $<v(s_h\cdots s_1s_0),u_d>$ and $<v(d_h\cdots d_1d_0),u_d>$. In addition, Lemma 1 shows that there are only $\alpha-1$ parallel paths between any two servers in the same unit cluster. Accordingly, $\alpha-1$ parallel paths exist between *src* and *dst*. Actually, the number of parallel paths between *src* and *dst* is also $\alpha-1$ for another relay cluster $<v(d_h\cdots d_1d_0),u_s>$. The two groups of parallel paths only intersect inside the unit clusters $<v(s_h\cdots s_1s_0),u_s>$ and $<v(d_h\cdots d_1d_0),u_d>$. So far, it is easy to derive Theorem 6.

Theorem 6: No matter whether $h{\le}\gamma$, there are $\alpha-1$ parallel paths between any two servers in a $BCN(\alpha,\beta,h,\gamma)$ but not in the same $BCN(\alpha,\beta,0)$.

Although BCN has the capability of multi-paths for one-to-one traffic, the existing routing schemes including FdimRouting and BdimRouting only exploit one path. To enhance the transmission reliability for one-to-one traffic, we adapt the routing path when the transmission meets a failure of a link, a server, and a switch. It is worth noticing that those parallel paths between any pair of servers pass through the common switch which connects the destination server in the last step. This does not hurt the fault-tolerant capacity of those parallel paths except the switch connecting the destination fails. In this rare case, at most one useable path exists between two servers.

C. Fault-tolerant Routing in BCN

We first give the definition of a failed link that can summarize three representative failures in data centers.

Definition 3: A link $(src1,dst1)$ is called failed only if the head $src1$ does not fail, however, cannot communicate with the tail $dst1$ no matter whether they are connected to the same switch or not. The failures of $dst1$, link, and the switch that connects $src1$ and $dst1$ can result in a failed link.

We then improve the *FdimRouting* and *BdimRouting* using two fault-tolerant routing techniques, i.e., the *local-reroute* and *remote-reroute*. The local-reroute *adjusts a routing path that consists of local links* on the basis of *FdimRouting*. On the contrary, the *remote-reroute* modifies those *remote links* in a path derived by *BdimRouting*. All links that interconnect master servers using the second port are called the *local links*, while those links that interconnect slave servers using the second port are called the *remote links*.

1) Local-reroute: Given any two servers *src* and *dst* in $BCN(\alpha,\beta,h,\gamma)$ ($h<\gamma$), we can calculate a path from *src* to *dst* using *FdimRouting*. Consider any failed link $(src1,dst1)$ in this path, where $src1$ and $dst1$ are labeled $x_h\cdots x_1x_0$ and $y_h\cdots y_1y_0$, respectively. The *FdimRouting* does not take failed links into account. We introduce *local-reroute* to bypass failed links by making local decisions.

The basic idea of *local-reroute* is that $src1$ immediately identifies all usable candidate servers of $dst1$ and then selects one such server as an intermediate destination, denoted as *relay*. The server $src1$ first routes packets to *relay* along a path derived by *FdimRouting* and then to the final destination dst_0 along a path from *relay* to dst_0. If any link in the first sub-path from $src1$ to *relay* fails, the packets are routed towards dst_0 along a new *relay* of the tail of the failed link, and then all existing *relay* servers in turn. On the other hand, the *local-reroute* handles any failed link in the second sub-path from *relay* to dst_0 in the same way. Actually, any sub-path from a current server to the newest relay

server, between any two adjacent relay servers, or from the oldest relay server to dst_0 might meet failed links. In these cases, the local-reroute handles those failed links in the same way.

A precondition of the *local-reroute* is that $src1$ can identify a *relay* server for $dst1$ by only local decisions. Let m denote the length of the longest common prefix of $src1$ and $dst1$. Let $x_h \cdots x_{h-m+1}$ denote the longest common prefix of $src1$ and $dst1$ for $m \geq 1$. If $m \neq h$, the two servers $dst1$ and $src1$ are not connected with the same switch, and then the label $z_h \cdots z_1 z_0$ of the *relay* server can be given by

$$\begin{aligned} & z_h \cdots z_{h-m+1} = y_h \cdots y_{h-m+1} \\ & z_{h-m} \in \{\{1,2,\cdots,\alpha\} - \{x_{h-m}, y_{h-m}\}\} \\ & z_{h-m-1} \cdots z_1 z_0 = y_{h-m-1} \cdots y_1 y_0. \end{aligned} \tag{6}$$

Otherwise, we first derive the server $dst2$ which connects with the server $dst1$ using their second ports. The failure of the link $(src1, dst1)$ is equivalent to the failure of the link $(dst1, dst2)$ unless $dst1$ is the destination. Thus, we can derive a *relay* server of the server $dst2$ using Formula 6, i.e., the *relay* server of the server $dst1$.

In Formula 6, the notation $h-m$ indicates that the two servers $src1$ and $dst1$ are in the same $BCN(\alpha,\beta,h-m)$ but in two different $BCN(\alpha,\beta,h-m-1)$. There exist α $BCN(\alpha,\beta,h-m-1)$ subnets inside this $BCN(\alpha,\beta,h-m)$. When $src1$ finds the failure of $dst1$, it chooses one *relay* server from all $BCN(\alpha,\beta,h-m-1)$ subnets in this $BCN(\alpha,\beta,h-m)$ except the two subnets that contain $src1$ or $dst1$. If $src1$ selects a relay server for $dst1$ from the $BCN(\alpha,\beta,h-m-1)$ that contains $src1$, the packets will be routed back to $dst1$ that fails to route those packets.

In Fig.4, 111→114→141→144→411→414→441→444 is the path from 111 to 444 derived by Algorithm 2. Once the link 144→411 and/or server 411 fails, server 144 immediately finds server 211 or 311 as a relay server, and calculates a path from it to the relay server. If the relay server is 211, the path derived by Algorithm 2 is 144→142→124→122→122→211. After receiving packets towards 444, the derived path by Algorithm 2 from 211 to 444 is 211→214→241→244→422→424→442→444. It is worthy noticing that if any link in the sub-path from 144 to 221 fails, the head of that link must bypass this failed link and reaches 221 in the same way. If the link 122→211 fails, the server 311 will replace 211 as the relay server of 411. If there is a failed link in the sub-path from 211 to 444, the *local-reroute* is used to address the failed link in the same way.

It is worthy noticing that the failed link $(141,144)$ will be found if the server 144 in the path from 111 to 444 fails. The failure of link $(141,144)$ is equivalent to the failure of the link $(144,411)$. Hence, the servers 211 and 311 are the *relay* servers derived by the aforementioned rules and Formula 6. All the servers with identifier starting with 1 from left to right cannot be the relay server since the path from the relay server to the destination will pass the failed link again.

2) Remote-reroute: For any two servers *src* and *dst* in $BCN(\alpha,\beta,h,\gamma)$ $(h \geq \gamma)$, their 3-tuples are $[v_s, u_s, s_h \cdots s_1 s_0]$ and $[v_d, u_d, d_h \cdots d_1 d_0]$, respectively. The *local-reroute* can handle any failed link in the path from *src* to *dst* if they are in the same $BCN(\alpha,\beta,h)$ in this $BCN(\alpha,\beta,h,\gamma)$, i.e., $u_s = u_d$. Otherwise, a pair of servers $dst1$ and $src1$ are derived according to *GetInterLink* operation in Algorithm 3, and are denoted as $[u_s, v_s, x = x_h \cdots x_1 x_0]$ and $[u_d, v_s, y = y_h \cdots y_1 y_0]$, respectively. In other words, $dst1$ and $src1$ are in the v_s^{th} $BCN(\alpha,\beta,\gamma)$ inside $BCN_{u_s}(\alpha,\beta,h)$ and $BCN_{u_d}(\alpha,\beta,h)$, respectively. The link $(dst1, src1)$ is the only one that interconnects the two v_s^{th} $BCN(\alpha,\beta,\gamma)$ in the two $BCN(\alpha,\beta,h)$.

If the packets from *src* to *dst* meets failed links in the two sub-paths from *src* to $dst1$ and from $src1$ to *dst*, the *local-reroute* can address those failed links. The *local-reroute*, however, cannot handle the failures of $dst1$, $src1$, and the links between them. In these cases, the packets cannot be forwarded from the u_s^{th} $BCN(\alpha,\beta,h)$ to the u_d^{th} $BCN(\alpha,\beta,h)$ inside this $BCN(\alpha,\beta,h,\gamma)$ through the desired link $(dst1, src1)$. We propose the *remote-reroute* to address this issue.

The basic idea of *remote-reroute* is to transfer the packets to another slave server $dst2$ that is connected with the same switch together with $dst1$ if at least one such slaver server and its associated links are usable. The label of $dst2$ is $x_h \cdots x_1 x_0'$ where x_0' can be any integer ranging from $\alpha+1$ to n except x_0. Assume that the other end of the link that is incident from $dst2$ using its second port is a slave server $src2$ in another $BCN_{u_i}(\alpha,\beta,h)$ inside the entire network. The packets are then forwarded to the slave server $src2$, and are routed to the destination *dst* along a path derived by Algorithm 3. If a link in the path from $src2$ to *dst* fails, the *local-reroute*, *remote-reroute* and Algorithm 3 can handle the failed links.

V. Evaluation

In this section, we analyze several basic topology properties of BCN, including the network order, network diameter, server degree, connectivity, and path diversity. Then we conduct simulations to evaluate the distributions of path length, average path length, and the robustness of routing algorithms.

A. Large Network Order

Lemma 3: The total number of servers in $BCN(\alpha,\beta,h)$ is $\alpha^h \cdot (\alpha+\beta)$, including α^{h+1} master and $\alpha^h \cdot \beta$ slave servers.

Proof: As mentioned in Section III, any given level BCN consists of α one lower BCNs. There are α^h level-0 BCN in $BCN(\alpha,\beta,h)$, where a level-0 BCN consists of α master servers and β slave servers. Thus proved. ■

Lemma 4: The number of servers in $G(BCN(\alpha,\beta,h))$ is $\alpha^h \cdot (\alpha+\beta) \cdot (\alpha^h \cdot \beta + 1)$, including $\alpha^{h+1} \cdot (\alpha^h \cdot \beta + 1)$ and $\alpha^h \cdot \beta \cdot (\alpha^h \cdot \beta + 1)$ master and slave servers, respectively.

Proof: As mentioned in Section III, there are $\alpha^h \cdot \beta + 1$ copies of $BCN(\alpha,\beta,h)$ in $G(BCN(\alpha,\beta,h))$. In addition, the number of servers in one $BCN(\alpha,\beta,h)$ has been proved by Lemma 3. Thus proved. ■

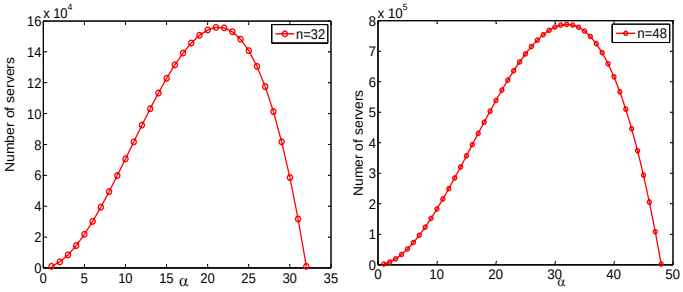
Theorem 7: The number of servers in $BCN(\alpha,\beta,h,\gamma)$ is

$$\begin{cases} \alpha^h \cdot (\alpha+\beta), \text{ if } h < \gamma \\ \alpha^{h-\gamma} \cdot \left(\alpha^\gamma \cdot (\alpha+\beta) \cdot (\alpha^\gamma \cdot \beta + 1)\right), \text{ if } h \geq \gamma \end{cases} \tag{7}$$

Proof: Lemmas 3 has proved this issue when $h < r$. In addition, $BCN(\alpha,\beta,\gamma,\gamma)$ is just $G(BCN(\alpha,\beta,\gamma))$. Thus, there are $\alpha^\gamma \cdot (\alpha+\beta) \cdot (\alpha^\gamma \cdot \beta + 1)$ servers in $BCN(\alpha,\beta,\gamma,\gamma)$. In addtion, $BCN(\alpha,\beta,h,\gamma)$ contains $\alpha^{h-\gamma}$ $BCN(\alpha,\beta,\gamma,\gamma)$ when $h \geq \gamma$. Thus proved. ■

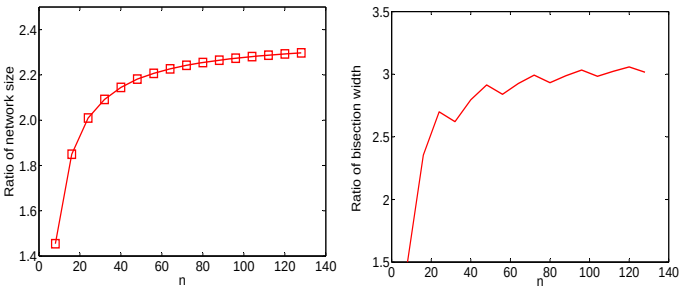
Theorem 8: For any given $n = \alpha+\beta$, the optimal α that maximizes the total number of servers in $BCN(\alpha,\beta,\gamma,\gamma)$ is given by

$$\alpha \approx (2 \cdot \gamma \cdot n)/(2 \cdot \gamma + 1). \tag{8}$$



(a) $n=32$ (b) $n=48$

Fig. 7. The network order of $BCN(\alpha,\beta,1,1)$ vs. α ranging from 0 to n.



(a) The ratio of network order. (b) The ratio of bisection width.

Fig. 8. The ratio of network order and bisection width of $BCN(\alpha,\beta,1,1)$ to that of FiConn$(n,2)$, where the diameter of the two networks is the same 7.

Proof: The total number of servers in $BCN(\alpha,\beta,\gamma,\gamma)$ is denoted as

$$\begin{aligned} f(\alpha) &= \alpha^{\gamma}\cdot(\alpha+\beta)\cdot(\alpha^{\gamma}\cdot\beta+1) \\ &= n\cdot\alpha^{\gamma}+n^2\cdot\alpha^{2\gamma}-n\cdot\alpha^{2\gamma+1}. \end{aligned}$$

Thus, we have

$$\begin{aligned} \frac{\varphi f(\alpha)}{\varphi\alpha} &= n\cdot\alpha^{\gamma-1}\left(\gamma+2\gamma\cdot n\cdot\alpha^{\gamma}-(2\gamma+1)\alpha^{\gamma+1}\right) \\ &\approx n\cdot\alpha^{\gamma-1}\left(2\gamma\cdot n\cdot\alpha^{\gamma}-(2\gamma+1)\alpha^{\gamma+1}\right). \end{aligned}$$

Clearly the derivative is 0 when $\alpha\approx(2\cdot\gamma\cdot n)/(2\cdot\gamma+1)$. At the same time, the second derivative is less than 0. Thus, $\alpha\approx(2\cdot\gamma\cdot n)/(2\cdot\gamma+1)$ maximizes the total number of servers in $BCN(\alpha,\beta,\gamma,\gamma)$. Thus proved. ■

Fig.7 plots the number of servers in $BCN(\alpha,\beta,1,1)$ when n=32 or 48. The network order goes up and then goes down after it reaches the peak point as α increases in the both cases. The largest network order of $BCN(\alpha,\beta,1,1)$ is 787,968 for n=48 and 155904 for n=32, and can be achieved only if α=32 and 21, respectively. This matches well with Theorem 8.

Fig.8(a) depicts the changing trend of the ratio of network order of BCN$(\alpha,\beta,1,1)$ to that of FiConn$(n,2)$ as the number of ports in each mini-switch increases, where α is assigned the optimal value $\alpha\approx(2\cdot\gamma\cdot n)/(2\cdot\gamma+1)$. The results show that the number of servers of BCN is significantly larger than that of FiConn$(n,2)$ with the same server degree 2 and network diameter 7, irrespective the value of n.

Formula 7 indicates that the network order of a BCN grows double-exponentially when h increases from $\gamma-1$ to γ, while grows exponentially with h in other cases. On the contrary, the network order of FiConn always grows double-exponentially with its level. Consequently, it is not easy to incrementally deploy FiConn because a level-k FiConn requires a large number of level-$(k-1)$ FiConns. In the case of BCN, incremental deployment is relative easy since a higher level BCN requires only α one lower level BCN except h=γ. On the other hand, the incomplete BCN can relieve the restriction on the network order for realizing incremental deployment by exploiting the topological properties of BCN in both dimensions.

B. Low Diameter and Server Degree

According to Theorems 4 and 5, we obtain that the diameters of $BCN(\alpha,\beta,h)$ and $BCN(\alpha,\beta,h,\gamma)$ $(h<\gamma)$ are $2^{h+1}-1$ and $2^{\gamma+1}+2^{h+1}-1$, respectively. In practice, h and γ are small integers. Therefore, BCN is a low-diameter network.

After measuring the network order and diameter of BCN, we study the node degree distribution in $BCN(\alpha,\beta,h,\gamma)$. If $h<\gamma$, the node degree of master servers are 2 except the α available master servers for further expansion. The α master servers and all slave servers are of degree 1. Otherwise, there are $\alpha\cdot(\alpha^{\gamma}\cdot\beta+1)$ available master servers that are of degree 1. Other master servers and all slave servers are of degree 2.

A BCN of level one in each dimension offers more than 1000,000 servers if 56-port switches are used, while the server degree and network diameter are only 2 and 7, respectively. This demonstrates the low diameter and server degree of BCN.

C. Connectivity and Path Diversity

The edge connectivity of a single server is one or two in $BCN(\alpha,\beta,h,\gamma)$. Consider the fact that $BCN(\alpha,\beta,h,\gamma)$ is constituted by a given number of low level subnets in the first dimension. We further evaluate the connectivity of BCN at the level of different subnets in Theorem 9.

Theorem 9: In any $BCN(\alpha,\beta,h,\gamma)$, the smallest number of remote links or servers that can be deleted to disconnect one $BCN(\alpha,\beta,i)$ from the entire network is

$$\begin{cases} \alpha-1, \text{ if } h<\gamma \\ \alpha-1+\alpha^{i}\cdot\beta, \text{ if } h\geq\gamma. \end{cases} \quad (9)$$

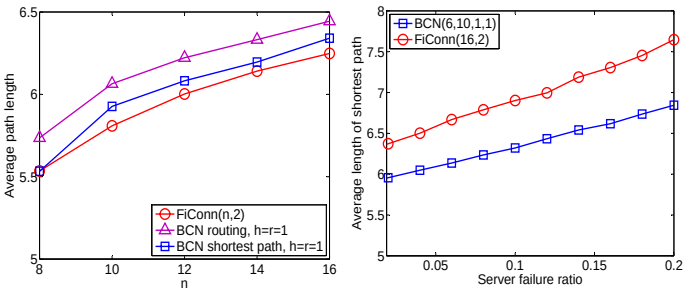
Proof: If $h<\gamma$, consider any subnet $BCN(\alpha,\beta,i)$ for $0\leq i<h$ in $BCN(\alpha,\beta,h,\gamma)$. If it contains one available master server for further expansion, only $\alpha-1$ remote links are used to interconnect with other homogeneous subnets. It is clear that the current subnet is disconnected if the corresponding $\alpha-1$ remote links or servers are removed.

If $h\geq\gamma$, besides the $\alpha-1$ remote links that connect its master servers the subnet $BCN(\alpha,\beta,i)$ has $\alpha^{i}\cdot\beta$ additional remote links that connect its slave servers. Thus, it can be disconnected only if the corresponding $\alpha-1+\alpha^{i}\cdot\beta$ remote links or servers are removed. Thus proved. ■

Theorem 10: **Bisection width:** The minimum number of *remote* links that need to be removed to split a $BCN(\alpha,\beta,h,\gamma)$ into two parts of about the same size is given by

$$\begin{cases} \alpha^2/4, \text{ if } h<\gamma \text{ and } \alpha \text{ is an even integer} \\ (\alpha^2-1)/4, \text{ if } h<\gamma \text{ and } \alpha \text{ is an odd integer} \\ \alpha^{h-\gamma}\cdot\frac{(\alpha^{\gamma}\cdot\beta+2)\cdot\alpha^{\gamma}\cdot\beta}{4}, \text{ if } h\geq\gamma. \end{cases} \quad (10)$$

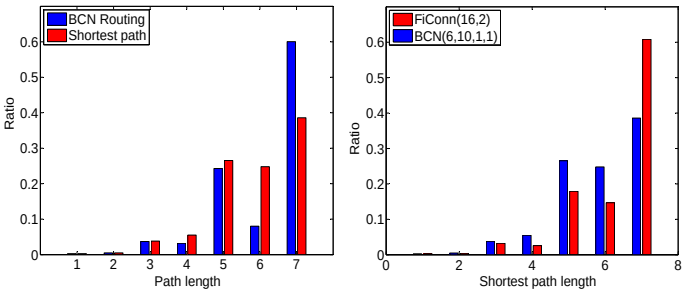
Proof: It is worth noticing that the bisection width of a compound graph $G(G_1)$ is the maximal one between the bisection widths of G and G_1 [17]. For $1\leq h<\gamma$, $BCN(\alpha,\beta,h,\gamma)$ is compound graph, where G is a complete graph with α nodes and G_1 is a $BCN(\alpha,\beta,h-1,\gamma)$. We can see that the bisection width of G is $\alpha^2/4$ if α is an even number and $(\alpha^2-1)/4$ if α is an odd number. The bisection width of $BCN(\alpha,\beta,h-1,\gamma)$ can



(a) The average length of shortest path and routing path vs. the value of n

(b) The average length of routing path BCN and FiConn vs. the server failure ratio.

Fig. 9. The path length of *BCN* and *FiConn* under different configurations



(a) The distribution of shortest path and routing path in $BCN(6,10,1,1)$.

(b) The shortest path length distribution in $BCN(6,10,1,1)$ and $FiConn(16,2)$

Fig. 10. The path length distribution under all to all traffic.

be induced in this way and is the same as that of G. Thus, the bisection width of $BCN(\alpha,\beta,h,\gamma)$ for $1\leq h<\gamma$ is proved.

$BCN(\alpha,\beta,h,\gamma)$ for $h\geq\gamma$ is a compound graph, where G is a complete graph with $\alpha^{\gamma}\cdot\beta+1$ nodes and G_1 is a $BCN(\alpha,\beta,h)$. We can see that the bisection width of G is $(\alpha^{\gamma}\cdot\beta+2)\cdot\alpha^{\gamma}\cdot\beta/4$ andc that of G_1 is $\alpha^2/4$ if α is an even number and $(\alpha^2-1)/4$ if α is an odd number, which is less than that of G. Moreover, G_1 has $\alpha^{h-\gamma}$ copies of $BCN(\alpha,\beta,\gamma)$, and there is one link between the i^{th} $BCN(\alpha,\beta,\gamma,)$ of two copies of G_1 for $1\leq i\leq\alpha^{h-\gamma}$. Thus, there are $\alpha^{h-\gamma}$ links between any two copies of G_1, and hence the bisection width of $BCN(\alpha,\beta,h,\gamma)$ is $\alpha^{h-\gamma}\cdot(\alpha^{\gamma}\cdot\beta+2)\cdot\alpha^{\gamma}\cdot\beta/4$. Thus proved. ■

For any FiConn(n,k), the bisection width is at least $N_k/(4*2^k)$, where $N_k=2^{k+2}*(n/4)^{2^k}$ denote the number of servers in the network [7]. We then evaluate the bisection width of FiConn$(n,2)$ and $BCN(\alpha,\beta,1,1)$ under the same server degree, switch degree, and network diameter. In this setting, the network size of $BCN(\alpha,\beta,1,1)$ outperforms that of FiConn$(n,2)$. Fig.8(b) shows that $BCN(\alpha,\beta,1,1)$ significantly outperforms FiConn$(n,2)$ in term of bisection width. Larger bisection width implies higher network capacity and more resilient against failure.

As proved in Lemma 2, there are $\alpha-1$ node-disjoint paths between any two servers in a $BCN(\alpha,\beta,\gamma,\gamma)$, where the optimal α is $2\cdot\gamma\cdot n/(2\cdot\gamma+1)$. Thus, the path diversity between any two servers is about $\lceil 2n/3\rceil-1$. With these disjoint paths, transmission rate can be accelerated and transmission reliability can be enhanced. Table II summarizes the network order and path diversity of $BCN(\alpha,\beta,1,1)$ under different value of n. This demonstrates the advantage that $BCN(\alpha,\beta,1,1)$ has high path diversity for one-to-one traffic. Note that there is a tradeoff to maximize the order or path diversity of BCN. Actually, the largest path diversity is obtained only if $\alpha=n$.

D. Evaluation of Path Length

We run simulations on $BCN(\alpha,\beta,1,1)$ and FiConn$(n,2)$ in which $n\in\{8,10,12,14,16\}$ and α is assigned its optimal value. The ratio of network order of BCN to that of FiConn varies between 1.4545 and 1.849. For the all to all traffic, Fig.9(a) shows the average length of the shortest path of FiConn, the shortest path of BCN, and the routing path of BCN. For any BCN, the routing path length is a little bit larger than the shortest path length since the current routing protocols do not entirely realize the shortest path routing. FdimRouting can be further improved by exploiting those potential shortest paths due to links in the second dimension. Although the network order of BCN is a lot larger than that of FiConn, the average shortest path length is a little bit larger than that of BCN.

TABLE II

NETWORK ORDER AND PATH DIVERSITY OF $BCN(\alpha,\beta,1,1)$

n	8	16	24	32	40	48
Network order	640	9856	49536	155904	380160	787968
Path diversity	5	10	15	21	26	31

Then we evaluate the fault-tolerance ability of the topology and routing algorithm of BCN$(6,10,1,1)$ and FiConn$(16,2)$. The network sizes of BCN$(6,10,1,1)$ and FiConn$(16,2)$ are 5856 and 5327, respectively. As shown in Fig.9(b), the average routing path length of BCN and FiConn increase with the server failure ratio. The average routing path length of BCN is a lot shorter than that of FiConn under the same server failure ratio although FiConn outperforms BCN in term of the average shortest path length when the server failure ratio is zero. These results demonstrate that the topology and routing algorithms of BCN possess better fault-tolerant ability. It is worthy noticing that $BCN(6,10,1,1)$ in Fig.9(b) supports less servers than $BCN(11,5,1,1)$ in Fig.9(a).

We then run simulations on $BCN(\alpha,\beta,1,1)$ and FiConn$(n,2)$ in which $n=16$ while α is not its optimal value but 6. The network sizes of $BCN(6,10,1,1)$ and FiConn$(16,2)$ are 5856 and 5327, respectively. The simulation results shown in Fig.10 conform the theoretical results about the network diameter of BCN and FiConn. Fig.10(a) further indicates that the routing algorithm of BCN fails to discover some shortest paths and causes some additional long routing paths. Fig.10(b) indicates that BCN and FiConn face the similar but not same distribution of the shortest paths even they have the same server degree 2, same network diameter 7, and similar network order.

VI. DISCUSSION

A. Extension to More Server Ports

Although we assume that all servers are equipped with two built-in NIC ports, the design methodologies of HCN and BCN can be easily extended to involve any constant number, denoted as m, of server ports. In fact, servers with four embedded NIC ports have been available, due to the rapid innovation on server hardware. Given any server with m ports, it can contribute $m-1$ ports for future higher-level interconnection after reserving one port for connecting with a mini-switch. Consider that a set of $m-1$ servers each of which holds two ports and connects with the same mini-switch using its first port. It is clear that the set of $m-1$ servers can totally contribute $m-1$ ports for future higher-level interconnection. Intuitively, a server with m ports can be treated as a set of $m-1$ servers each with two ports. In this way, we can extend HCN and BCN to embrace any constant number

of server ports. In fact, there can be many other specific ways for interconnecting servers with constant degree of more than 2, and we leave the investigation as our future work.

B. Locality-Aware Task Placement

Although the proposed network structures possess many advantages, such as excellent topological properties, easy wiring, and low cost, they may not be able to achieve satisfactory end-to-end throughput when all the nodes are transmitting or receiving packets simultaneously. This results from the lesser number of links and switches used by HCN and BCN. Fortunately, this issue can be addressed by some techniques at the application layer, due to the observation as follows.

As observed in [8], a server is likely to communicate with a small subset of other servers when conducting typical applications in common data centers, such as group communication, VM migration, and file chunk replication. Additionally, data centers with hierarchical network structures, for example HCN and FiConn, hold an inherent benefit. That is, lower level networks support local communications, while higher level networks are designed to realize remote communications.

Therefore, a locality-aware approach can be used to placing those tasks onto servers in HCN. That is, those tasks with intensive data exchange can be placed onto servers, in a $\mathrm{HCN}(n,0)$, which connect to the same switch. If those tasks need some more servers, they may reserve a one higher lever structure $\mathrm{HCN}(n,1)$, and so on. There is only a few even one server hop between those servers. As proved in Section V-A, HCN is usually sufficient to contain hundreds of servers, where the number of server hops is at most three. Similarly, we can use a locality-aware mechanism when placing tasks onto data-center servers in BCN. Therefore, the locality-aware mechanism can largely save network bandwidth by avoiding unnecessary remote data communications.

C. Impact of Server Routing

In HCN as well as BCN, since servers that connect to other modules at a different level have to forward packets, they will need to devote some processing resources for this aspect. Although we can use software based packet forwarding schemes for HCN and BCN, they usually incur non-trivial CPU overhead. Lu et. al implemented a hardware based configurable packet forwarding engine using NetFPGA, CAFE [21], as a good candidate for supporting DCN designs. Inspired by the fact that CAFE can be easily configured and it can forward packets at line-rate, we can easily re-configure CAFE to forward self-defined packets for HCN or BCN without any hardware re-designing.

VII. Conclusion

In this paper, we propose HCN and BCN, two novel server-interconnection network structures that utilize hierarchical compound graphs to interconnect servers with two-ports only and low-cost commodity switches. They own two topological advantages, i.e., the expansibility and equal degree. Moreover, HCN offers high degree of regularity, scalability and symmetry which very well conform to a modular design of data centers. BCN of level one in each dimension is the largest known DCN with server degree 2 and diameter 7. It is highly scalable to support hundreds of thousands of servers with low diameter, low cost, high bisection width, high path diversity for one-to-one traffic, and good fault-tolerance ability. Analysis and simulations show that HCN and BCN are viable structures for data centers.

Following the work in this paper, we plan to study several issues in the future. The first issue is to study the incomplete HCN and BCN networks so as to incrementally deploy HCN and BCN, which is important for building mega data centers. Second, we will further design traffic-aware routing protocols to balance the usages of different levels of links and enlarge the network capacity. The third issue is to accelerate the typical traffic patterns significantly by exploiting the good topological features, such as large number of node-disjoint paths between any two servers.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proc. SOSP*, Bolton Landing, NY, USA, 2003, pp. 29–43.

[2] D. Borthakur. The hadoop distributed file system: Architecture and design. [Online]. Available: http://hadoop.apache.org

[3] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Comput Systems*, vol. 26, no. 2, 2008.

[4] CloudStore. Higher performance scalable storage. [Online]. Available: http://kosmosfs.sourceforge.net/

[5] M. A. Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. SIGCOMM*, Seattle, Washington, USA, 2008.

[6] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. SIGCOMM*, Seattle, Washington, USA, 2008.

[7] D. Li, C. Guo, H. Wu, Y. Zhang, and S. Lu, "Ficonn: Using backup port for server interconnection in data centers," in *Proc. IEEE INFOCOM*, Brazil, 2009.

[8] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, and J. Wu, "Scalable and cost-effective interconnection of data-center servers using dual server ports," *IEEE/ACM Transactions on Networking*, vol. doi:10.1109/TNET.2010.2053718, 2010.

[9] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *Proc. SIGCOMM*, Barcelona, Spain, 2009.

[10] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, and P. P. and, "VL2: A scalable and flexible data center network," in *Proc. SIGCOMM*, Barcelona, Spain, 2009.

[11] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: A high performance network structure for modular data center interconnection," in *Proc. CONEXT*, Rome, Italy, 2009.

[12] M. Miller and J. Siran, "Moore graphs and beyond: A survey of the degree/diameter problem," *Electronic Journal of Combinatorics*, vol. 61, pp. 1–63, Dec. 2005.

[13] N. Alon, S. Hoory, and N. Linial, "The Moore bound for irregular graphs," *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.

[14] M. Imase and M. Itoh, "A design for directed graphs with minimum diameter," *IEEE Trans. Computers*, vol. 32, no. 8, pp. 782–784, Aug. 1983.

[15] R. M. Damerell, "On Moore graphs," in *Proc. Cambridge Philosophical Society*, 1973, pp. 227–236.

[16] T. Holf. (2007, Jul.) Google architecture. [Online]. Available: http://highscalability.com/google-architecture

[17] D. P. Agrawal, C. Chen, and J. R. Burke, "Hybrid graph-based networks for multiprocessing," *Telecommunication system*, vol. 10, pp. 107–134, 1998.

[18] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Transactions on Computers*, vol. 33, no. 4, pp. 323–333, 1984.

[19] P. T. Breznay and M. A. Lopez, "Tightly connected hierarchical interconnection networks for parallel processors," in *Proc. IEEE ICPP*, vol. 1, 1993, pp. 307–310.

[20] ——, "A class of static and dynamic hierarchical interconnection networks," in *Proc. IEEE ICPP*, vol. 1, 1994, pp. 59–62.

[21] G. Lu, Y. Shi, C. Guo, and Y. Zhang, "Cafe: A configurable packet forwarding engine for data," in *Proc. PRESTO*, Barcelona, Spain, 2009.