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

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

Abstract—A fundamental goal of data center networking is to efficiently interconnect a large number of servers with the low equipment cost. Several server-centric network structures for data centers have been proposed. They, however, are not truly expandable and suffer a low degree of regularity and symmetry. Inspired by the commodity servers in today's data centers that come with dual port, we consider how to build expandable 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 the 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 the low diameter and high bisection width. Additionally, HCN offers a high degree of regularity, scalability, and symmetry, which conform to the modular designs of data centers. BCN is the largest known network structure for data centers with the server degree 2 and network diameter 7. Furthermore, BCN has many attractive features, including the low diameter, high bisection width, large number of node-disjoint paths for the one-to-one traffic, and good fault-tolerant ability. Mathematical analysis and comprehensive simulations show that HCN and BCN possess excellent topological properties and are viable network structures for data centers.

Index Terms—Data center networking, network structures, interconnection networks

1 Introduction

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

A number of novel DCN network structures are proposed recently and can be roughly divided into two categories. One is switch centric, which organizes switches into structures other than tree and puts the interconnection intelligence on switches. Fat-Tree [3], VL2 [8] fall into such a category. The other is server centric, which puts the interconnection intelligence on servers and uses switches only as cross bars. DCell [4], BCube [7], FiConn [5], [6], MDCube [9], and uFix [10] fall into the second category. Among others, a 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 the fault-tolerant ability.

For DCell and BCube, their nice topological properties and efficient algorithms have been derived at the cost as follows: They use more than two ports per server, typically four, and large number of switches and links, so as to scale to a large server population. If they use servers with only two ports, the server population is very limited and cannot be enlarged because they are at most two levels. When network structures are expanded to one higher level, DCell and BCube add one NIC and link for each existing server, and BCube has to be 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.

- *D. Guo is with the Key Laboratory for Information System Engineering, School of Information System and Management, National University of Defense Technology, Changsha 410073, P.R. China. E-mail: guodeke@gmail.com.*
- *T. Chen is with the Key Laboratory for Information System Engineering, School of Information System and Management, National University of Defense Technology, Changsha 410073, P.R. China. E-mail: emilchenn@gmail.com.*
- *D. Li is with the Department of Computer Science, Tsinghua University, 9-402, East Main Building, Beijing 100084, P.R. China. E-mail: tolidan@tsinghua.edu.cn.*
- *M. Li is with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: limo@ntu.edu.sg.*
- *Y. Liu is with the TNLIST and School of Software, TNLIST, Tsinghua University, Beijing 100084, P.R. China, and the Computer Science Department, Hong Kong University of Science and Technology, Hong Kong, P.R. China. E-mail: liu@cse.ust.hk.*
- *G. Chen is with the Shanghai Key Lab of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, P.R. China. E-mail: gchen@nju.edu.cn, gchen@cs.sjtu.edu.cn.*

Manuscript received 31 Jan. 2011; revised 9 Apr. 2012; accepted 12 Apr. 2012; published online 19 Apr. 2012.
Recommended for acceptance by S. Dolev.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-01-0072.
Digital Object Identifier no. 10.1109/TC.2012.90.

Such topologies, however, are not truly expandable. A network is expandable if no changes with respect to the node's configuration and link connections are necessary when it is expanded. This may cause negative influence on applications running on all of the existing servers during the process of topology expansion. We, thus, need to design an expandable and cost-effective network structure that works for commodity servers with constant NIC ports and low-end switches. Other benefits by solving the problem are multifaceted. First, we do not use expensive high-end switches, which are widely used today. Second, it can offer an easy-to-build testbed at a university or institution because those data center infrastructures may only be afforded by a few cash-rich companies.

1.1 Motivation and Contributions

Without loss of generality, we focus on the interconnection of a large number of commodity dual-port servers because such servers are already available in current practice. It is challenging to interconnect a large population of such servers in data centers, because we should also guarantee the low diameter and the high bisection width. FiConn is one of such topologies but suffers a low degree of regularity and symmetry.

In this paper, we first propose a hierarchical irregular compound network, denoted as HCN, which can be expanded by only adding one link to a few number of servers. Moreover, HCN offers a high degree of regularity, scalability, and symmetry, which conform to the modular designs of data centers. Inspired by the smaller network size of HCN compared to FiConn, we further study the degree/diameter problem [11], [12] in the scenario of building a scalable network structure for data centers using dual-port servers.

Given the maximum node degree and network diameter, the degree/diameter problem aims to determine the largest graphs. Specifically, the degree/diameter problem here is to determine desirable DCNs that 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 [13], [14] have been made to study the degree/diameter problem in graph theory, it is still open in the field of DCN.

We then propose BCN, a Bidimensional Compound Network for data centers, which inherits 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. BCN of level one in each dimension is the largest known network structure for data centers, with the server degree 2 and the network diameter 7. In this case, the order of BCN is significantly larger than that of FiConn$(n, 2)$, irrespective of the value of n. For example, if 48-port switches are used, BCN of level one in each dimension offers 787,968 servers, while a level-2 FiConn only supports 361,200 servers. Besides such advantages, BCN has other attractive properties, including the low diameter and cost, high bisection width, high path diversity for the one-to-one traffic, good fault-tolerant 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 for HCN and BCN by exploiting the compound graph. They possess the good regularity and expandability that help reduce the cost of further expansions and are especially suitable for large-scale data centers. Second, BCN of level one in each dimension offers the largest known network structure for data centers with the server degree 2 and the network diameter 7. Third, HCN and BCN use distributed fault-tolerant routing protocols to handle those representative failures in data centers. Moreover, HCN and BCN can be used as the intracontainer and intercontainer network structures for designing a mega data center in a modular way like MDCube and uFix.

1.2 Organization of Paper

The rest of this paper is organized as follows: Section 2 introduces the related work. Section 3 describes the structures of HCN and BCN. Section 4 presents the general and fault-tolerant routing algorithms for HCN and BCN. Section 5 evaluates the topological properties and routing protocols of HCN and BCN through analysis and simulations. Section 6 discusses other design issues in HCN and BCN. Finally, Section 7 concludes this paper and discusses our future work..

2 Related Work

2.1 Constructing Large Interconnection Networks

Hierarchical network is a natural way to construct large interconnection 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 communications, while higher level networks support remote communications. The compound graph is suitable for large-scale systems due to its good regularity and expandability [15].

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 that connects two corresponding 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 node 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 ease of explanation, we

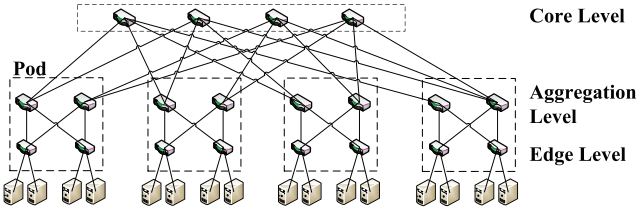


Fig. 1. A Fat-Tree network structure with $n = 4$.

consider the case that the regular G 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, $G^0(G_1) = G_1$.

2.2 Interconnection Structures for Data Centers

We discuss four representative network structures, including Fat-Tree [3], DCell [4], FiConn [6], and BCube [7].

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

HCN and BCN use only the 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 levels of switches [6]. HCN and BCN do not suffer such a limitation and can be extended to accommodate large number of servers, although each server 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 each other. $DCell_0$ is the basic building block in which n servers are connected to a n-port commodity switch. Additionally,

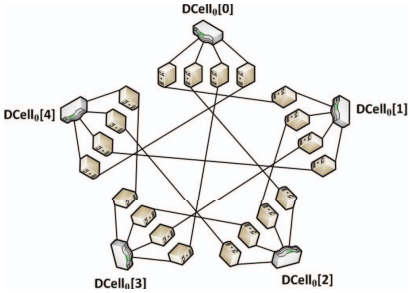


Fig. 2. A DCell_1 network structure with $n = 4$.

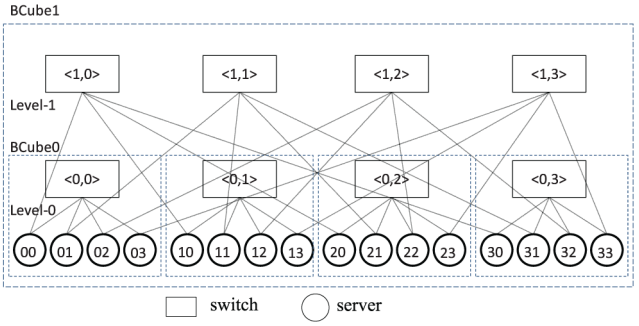


Fig. 3. A BCube(4,1) network structure.

$DCell_i$ is a level-i regular compound graph $G^i(DCell_0)$ constituted recursively for any $i \geq 1$. Fig. 2 illustrates an example of $DCell_1$ with $n = 4$.

HCN and BCN are also server-centric structures like DCell, but differ in several aspects. First, the server degree of 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 because each server uses only two ports. Second, no other hardware cost is introduced on a server in HCN and BCN because they use existing backup port on each server for interconnection. If DCell uses servers with only two ports, the server population is very limited since DCell is at most two levels. 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 expandable.

FiConn shares the similar design principle with HCN and BCN to interconnect large number of commodity dual-port servers, but differs in several aspects. First, the topology of FiConn suffers a low degree of regularity and symmetry. Second, FiConn must append one wiring link to more and more servers when it was expanded to higher level topologies. HCN, however, only appends one wiring link to a constant number of servers during its expansion process.

BCube is proposed for container-based data centers, as shown in Fig. 3. BCube_0 is simply n servers connecting to a n-port switch. 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 connected to multiple levels of miniswitches, but such switches are not directly connected. The server degrees of HCN and BCN are constantly 2, while BCube must allocate each server more NIC ports. In addition, given the same number of servers, BCube uses much more miniswitches and links than HCN and BCN. Actually, BCube is an emulation of the generalized Hypercube [16].

Guo et al. proposed a malfunction detection scheme that can detect improperly connected cables and pinpoint their locations, in BCube data centers [17]. The key insight is to analyze the topology properties of the data center. Such a scheme can also be applied to our network structures, HCN and BCN. More detailed discussion on the miswiring problem, however, we leave as one of our future work.

TABLE 1
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
N	number of servers in a data center
h	level of 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(α,β,h)
HCN(α,h)	level-h HCN
$s_\gamma = \alpha^\gamma\beta$	number of slave servers in BCN(α,β,γ)
BCN$(\alpha,\beta,0)$	level-0 BCN, i.e., the smallest building block
BCN(α,β,h)	level-h BCN in the first dimension
$G(BCN(\alpha,\beta,h))$	a compound graph uses BCN(α,β,h) as G_1 and a complete graph as G
BCN(α,β,h,γ)	a general BCN that always expands in the first dimension while only expands in the second dimension when $h\geq\gamma$
u	order of BCN(α,β,h) in BCN(α,β,h,γ) from the viewpoint of the second dimension
v	order of a $G(BCN(\alpha,\beta,\gamma))$ in BCN(α,β,h,γ) from the viewpoint of the first dimension

3 The BCN Network Structure

We propose two expandable network structures, 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 1 lists the notations used in the rest of this paper.

3.1 Hierarchical Irregular Compound Networks

For any given $h \geq 0$, we denote a level-h irregular compound network as $\mathrm{HCN}(n,h)$. HCN is a recursively defined structure. A high-level $\mathrm{HCN}(n,h)$ employs a low level $\mathrm{HCN}(n,h-1)$ as a unit cluster and connects many such clusters by means of a complete graph. $\mathrm{HCN}(n,0)$ is the smallest module (basic construction unit) that consists of n dual-port servers and a n-port miniswitch. For each server, its first port is used to connect with the miniswitch while the 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.

$\mathrm{HCN}(n,1)$ is constructed using n basic modules $\mathrm{HCN}(n,0)$. In $\mathrm{HCN}(n,1)$, there is only one link between any two basic modules by connecting two *available* servers that belong to different basic modules. Consequently, for each $\mathrm{HCN}(n,0)$ inside $\mathrm{HCN}(n,1)$ all of the servers are associated with a level-1 link except one server that is reserved for the construction of $\mathrm{HCN}(n,2)$. Thus, there are n available servers in $\mathrm{HCN}(n,1)$ for further expansion at a higher level. Similarly, $\mathrm{HCN}(n,2)$ is formed by n level-1 $\mathrm{HCN}(n,1)$s, and has n available servers for interconnection at a higher level.

In general, $\mathrm{HCN}(n,i)$ for $i \geq 0$ is formed by

$$n\ \mathrm{HCN}(n,i-1)\mathrm{s},$$

and has n available servers each in one $\mathrm{HCN}(n,i-1)$ for further expansion. According to Definition 1, $\mathrm{HCN}(n,i)$ acts as G_1 and a complete graph of n nodes acts as G. Here, $G(G_1)$ produces an irregular compound graph because the number of available servers in $\mathrm{HCN}(n,i)$ is n while the node degree of G is $n-1$. To facilitate the construction of any level-h HCN, we introduce Definition 2 as follows:

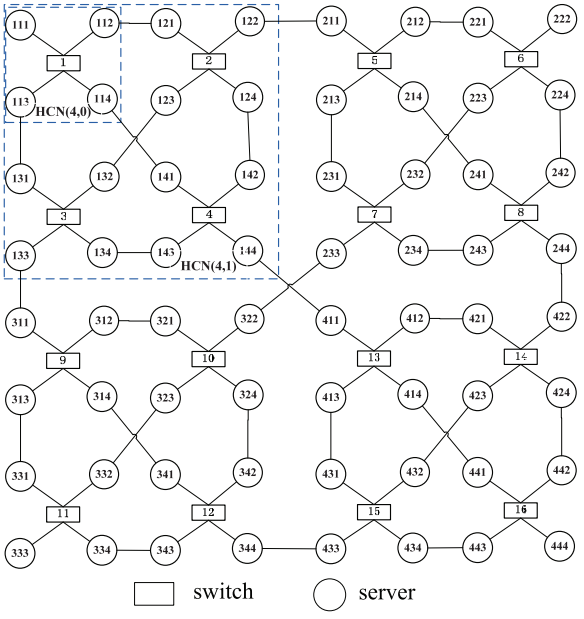


Fig. 4. An example of $\mathrm{HCN}(n,h)$, where $n=4$ and $h=2$.

Definition 2. *Each server in* $\mathrm{HCN}(n,h)$ *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_j*s. 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 $\mathrm{HCN}(4,2)$ constructed according to Definition 2. $\mathrm{HCN}(4,2)$ consists of four $\mathrm{HCN}(4,1)$s and a HCN (4, 1) has four HCN(4, 0)s. The second port of four servers, 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 such a property, let x_i indicate the order of $\mathrm{HCN}(n,i-1)$, containing a server $x_h\cdots x_1x_0$, among all of the level-$(i-1)$ HCNs of $\mathrm{HCN}(n,i)$ 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 $\mathrm{HCN}(n,i-1)$ that contains such a server in $\mathrm{HCN}(n,h)$. We use the server 423 as an example. Here, $x_1 = 2$ indicates the second $\mathrm{HCN}(4,0)$ in $\mathrm{HCN}(4,1)$ that contains such a server. Such a $\mathrm{HCN}(4,0)$ contains the servers 421, 422, 423, and 444. Here, $x_2 = 4$ indicates the fourth level-1 HCN in a level-2 HCN that contains such a server. Thus, $x_2x_1 = 42$ indicates the level-0 HCN that contains the server 423 in a level-2 HCN.

In summary, HCN owns two topological advantages, i.e., expandability and equal server degree, with the benefits of easy implementation and low cost. Additionally, HCN offers a high degree of regularity, scalability, and symmetry. Its network order, however, is less than that of FiConn in the same setting and we, thus, study the degree/diameter problem of DCN.

3.2 BCN Physical Structure

BCN is a multilevel 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 such clusters by means of a complete graph.

Let $\mathrm{BCN}(\alpha,\beta,0)$ denote the basic building block, where $\alpha+\beta=n$. It has n servers and one n-port miniswitch. All of the servers are connected to the miniswitch 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 such a 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.

3.2.1 Hierarchical BCN in the First Dimension

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

3.2.2 Hierarchical BCN in the Second Dimension

There are β available slave servers in the smallest module $\mathrm{BCN}(\alpha,\beta,0)$. In general, there are $s_h=\alpha^h\cdot\beta$ available slave servers in any given $\mathrm{BCN}(\alpha,\beta,h)$ for $h \geq 0$. We study how to utilize those available slave servers to expand $\mathrm{BCN}(\alpha,\beta,h)$ from the second dimension. A level-1 regular compound graph, $G(\mathrm{BCN}(\alpha,\beta,h))$, is a natural way to realize such a goal. It uses $\mathrm{BCN}(\alpha,\beta,h)$ as a unit cluster and connects s_h+1 copies of $\mathrm{BCN}(\alpha,\beta,h)$ by means of a complete graph using the second ports of all of available slave servers. The resultant $G(\mathrm{BCN}(\alpha,\beta,h))$ cannot be further expanded in the second dimension because 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* $\mathrm{BCN}(\alpha,\beta,h)$ *is*

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

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

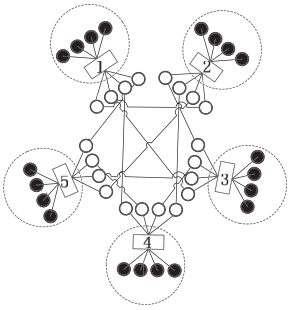


Fig. 5. A $G(BCN(4,4,0))$ structure that consists of slave servers in five $\mathrm{BCN}(4,4,0)$s in the second dimension.

Fig. 5 plots an example of $G(\mathrm{BCN}(4,4,0))$. The four slave servers connected with a miniswitch in $\mathrm{BCN}(4,4,0)$ is the unit cluster. A complete graph is used to connect five copies of $\mathrm{BCN}(4,4,0)$. Consequently, only one remote link is associated with each slave server in a unit cluster. Thus, the node degree is two for each slave server in the resultant network.

3.2.3 Bidimensional Hierarchical BCN

After designing $\mathrm{BCN}(\alpha,\beta,h)$ and $G(BCN(\alpha,\beta,h))$, we design a scalable bidimensional BCN formed by both master and slave servers. Let $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ denote a bidimensional BCN, where h denotes the level of BCN in the first dimension, and γ denotes the level of BCN that is selected as the unit cluster in the second dimension.

In this case, $\mathrm{BCN}(\alpha,\beta,0)$ consists of α master servers, β slave servers and one miniswitch, i.e., it 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 $\mathrm{BCN}(\alpha,\beta,h)$ from the first or second dimension without destroying the existing structure. A bidimensional BCN is always $\mathrm{BCN}(\alpha,\beta,h)$ as h increases when $h<\gamma$. In such a scenario, the unit cluster for expansion in the second dimension has not been formed. When h increases to γ, we achieve $\mathrm{BCN}(\alpha,\beta,\gamma)$ in the first dimension and then expand it from the second dimension using the construction method of $G(\mathrm{BCN}(\alpha,\beta,\gamma))$ in Section 3.2.2. In the resultant $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$, there are $\alpha^\gamma\cdot\beta+1$ copies of $\mathrm{BCN}(\alpha,\beta,\gamma)$ and α available master servers in each $\mathrm{BCN}(\alpha,\beta,\gamma)$. A sequential number u is employed to identify $\mathrm{BCN}(\alpha,\beta,\gamma)$ among $\alpha^\gamma\cdot\beta+1$ ones in the second dimension, where u ranges from 1 to $\alpha^\gamma\cdot\beta+1$. Fig. 5 plots an example of $\mathrm{BCN}(4,4,0,0)$ consisting of five $\mathrm{BCN}(4,4,0)$s, where $h=r=0$. It is worth noticing that $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$ 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 in the following way.

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

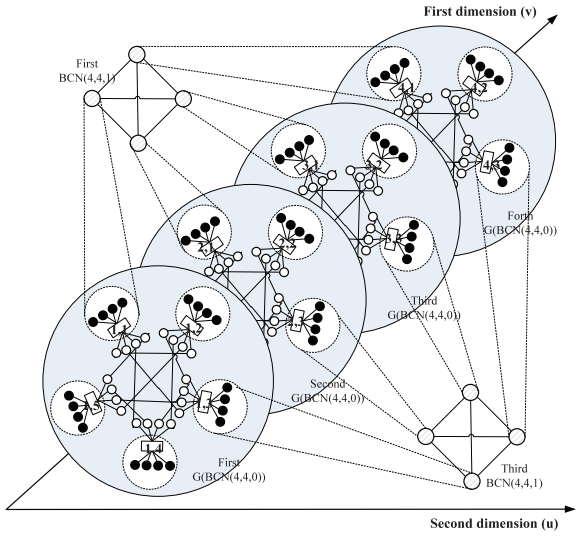


Fig. 6. An illustrative example of $\mathrm{BCN}(4,4,1,0)$.

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

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

Although the wiring is relatively easy because only constant ports per server are used for interconnection, the wiring complexity is still nontrivial in practice. Fortunately, the packaging and wiring technologies in DCube, FiConn, and MDCube can help tackle the wiring problem of our proposals.

3.3 The Construction Methodology of BCN

A higher level BCN network can be built by an incremental expansion using one lower level BCN as a unit cluster and connecting many such clusters by means of a complete graph.

3.3.1 In the Case of $h<\gamma$

In such a case, $\mathrm{BCN}(\alpha,\beta,h)$ can be achieved by the construction methodology of $\mathrm{HCN}(\alpha,h)$ in Section 3.1.

3.3.2 In the Case of $h=\gamma$

As mentioned in Section 3.2.2, all of the slave servers in $\mathrm{BCN}(\alpha,\beta,\gamma)$ are utilized for expansion in the second dimension. Each slave server in $\mathrm{BCN}(\alpha,\beta,\gamma)$ 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 $id(x)$ that denotes its order among all of the slave servers in $\mathrm{BCN}(\alpha,\beta,\gamma)$ and ranges from 1 to s_{γ}. For each slave server, the mapping between a unique id and its label is bijection, as defined in Theorem 2. Meanwhile, the label can be derived from its unique id in the reverse 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 $\mathrm{BCN}(\alpha,\beta,i-1,\gamma)$ that contains the slave server x in a higher level $\mathrm{BCN}(\alpha,\beta,i,\gamma)$ for $1\le i\le\gamma$. In addition, the total number of slave servers in any $\mathrm{BCN}(\alpha,\beta,i-1,\gamma)$ 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 $\mathrm{BCN}(\alpha,\beta,0)$ 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 3.2.2, the resultant BCN network when $h=\gamma$ is a $G(\mathrm{BCN}(\alpha,\beta,\gamma))$ consisting of $s_{\gamma}+1$ copies of a unit cluster $\mathrm{BCN}(\alpha,\beta,\gamma)$. In such a case, $BCN_u(\alpha,\beta,\gamma)$ denotes the uth 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-tuple $[v(x)=1,u,x]$, where $v(x)$ is defined in Theorem 3. In $BCN_u(\alpha,\beta,\gamma)$, all of the 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 of the slave servers in $s_{\gamma}+1$ homogeneous $\mathrm{BCN}(\alpha,\beta,\gamma)$s to constitute a $G(\mathrm{BCN}(\alpha,\beta,\gamma))$. For any two slave servers $[1,u_s,x_s]$ and $[1,u_d,x_d]$, as mentioned in literature [18] 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 (2). In literature [4], 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 for all of the slave servers because the above

two and other permutation methods are suitable to constitute $G(\mathrm{BCN}(\alpha,\beta,\gamma))$. For more information about the two methods, we suggest readers to refer literatures [4], [18].

3.3.3 In the Case of $h > \gamma$

After achieving $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$, the resultant network can be incrementally expanded in the first dimension without destroying the existing structure. As discussed in Section 3.2.3, $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ $(h>\gamma)$ consists of $s_\gamma+1$ copies of a unit cluster $\mathrm{BCN}(\alpha,\beta,h)$ in the second dimension. Each server in $\mathrm{BCN}_u(\alpha,\beta,h)$, the uth unit cluster of $\mathrm{BCN}(\alpha,\beta,h,\gamma)$, is assigned a unique label $x=x_h\cdots x_1x_0$ for $1\le u\le s_\gamma+1$. In addition, $\mathrm{BCN}_u(\alpha,\beta,h)$ has $\alpha^{h-\gamma}\mathrm{BCN}(\alpha,\beta,\gamma)$s in the first dimension. Recall that a sequential number v is employed to rank those $\mathrm{BCN}(\alpha,\beta,\gamma)$s in $\mathrm{BCN}_u(\alpha,\beta,h)$.

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

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

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

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

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

Algorithm 1. Construction of $\mathrm{BCN}(\alpha,\beta,h,\gamma)$
Require: $h>\gamma$

1: Connects all of the 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 $\mathrm{BCN}(\alpha,\beta,0)$}
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 $\mathrm{BCN}(\alpha,\beta,h)$}
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 $v^{th}G(\mathrm{BCN}(\alpha,\beta,\gamma))$ in $\mathrm{BCN}(\alpha,\beta,h,\gamma)$}
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.

4 Routing for One-to-One Traffic

The one-to-one traffic is the basic traffic model and the good one-to-one support also results in the good several-to-one and all-to-one support [7]. In this section, we start with the single-path routing for the one-to-one traffic in BCN without failures of switches, servers, and links. We then study the parallel multipaths for the one-to-one traffic in BCN. Finally, we propose fault-tolerant routing schemes to address those representative failures by employing the benefits of multipaths between any two servers.

4.1 Single Path for the One-to-One Traffic without Failures

4.1.1 In the Case of $h < \gamma$

For any $\mathrm{BCN}(\alpha,\beta,h)$ $(1\le h)$ 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 $\mathrm{BCN}(\alpha,\beta,h)$ but different $\mathrm{BCN}(\alpha,\beta,h-1)$s. The source and destination can be of the master server or the slave server. The routing scheme first determines the link $(dst1,src1)$ that interconnects the two $\mathrm{BCN}(\alpha,\beta,h-1)$s that src and dst are located at. It then derives two subpaths from src to $dst1$ and from $src1$ to dst. The path from src to dst is the combination of the two subpaths and the link $(dst1,src1)$. Each of the two subpaths can be obtained by recursively invoking Algorithm 2.

In Algorithm 2, the labels of a pair of servers are retrieved from the two inputs that can be of 1-tuple or 3-tuple. A 3-tuple indicates that such a $\mathrm{BCN}(\alpha,\beta,h)$ is a component of the entire $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ 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 $\mathrm{BCN}(\alpha,\beta,h)$. Note 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 $O(1)$.

Algorithm 2. FdimRouting(src, dst)
Require: src and dst are two servers in $\mathrm{BCN}(\alpha,\beta,h)$ ($h<\gamma$). The labels of the two servers are retrieved from the inputs, and are $src=s_hs_{h-1}\cdots s_1s_0$ and $dst=d_hd_{h-1}\cdots d_1d_0$, 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 $(dst1,src1)$

From the $FdimRouting$, we obtain the following theorem. Note that the length of the path between two servers connecting to the same switch is one. Such an assumption was widely used in the designs of server-centric network structures for data centers, such as DCell, FiConn, BCube, and MDCube.

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

Proof. For any two servers, src and dst, in $\mathrm{BCN}(\alpha,\beta,h)$ but in different $\mathrm{BCN}(\alpha,\beta,h-1)$s, let D_h denote the length of the single path resulted from Algorithm 2. The entire path consists of two subpaths in two different $\mathrm{BCN}(\alpha,\beta,h-1)$s 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. □

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 usually need to calculate one subpath that contains that next hop..

4.1.2 In the Case of $h\geq\gamma$

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

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

From the $BdimRouting$, we obtain the following theorem.

Theorem 5. *The shortest path length among all of the server pairs in* $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ *($h>\gamma$) 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 interlink between $dst1$ and $src1$, a first subpath from src to $dst1$ and a second subpath from $src1$ to dst. The length of the first subpath is $2^{\gamma+1}-1$ because the two end servers are in the same $\mathrm{BCN}(\alpha,\beta,\gamma)$. Theorem 4 shows that the maximum path length of the second subpath 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 worth noticing that the $GetInterLink$ can directly derive the end servers of the link only based on the three inputs and the constraints in (3). Thus, the time complexity of the $GetInterLink$ 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.

Algorithm 3. BdimRouting(src, dst)
Require: src and dst are denoted as $[v(s_h\cdots s_1s_0),u_s,s_h\cdots s_1s_0]$ and $[v(d_h\cdots d_1d_0),u_d,d_h\cdots d_1d_0]$ in $\mathrm{BCN}(\alpha,\beta,h\geq\gamma,\gamma)$.
1: **if** $u_s==u_d$ **then** {In the same $\mathrm{BCN}(\alpha,\beta,h)$}
2: Return $FdimRouting(src,dst)$
3: $v_c\leftarrow v(s_h\cdots s_1s_0)$ {v_c can also be $v(d_h\cdots d_1d_0)$}
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 $u_s^{th}\mathrm{BCN}(\alpha,\beta,h)$ of $\mathrm{BCN}(\alpha,\beta,h,\gamma)$}
6: $tail\leftarrow FdimRouting(src1,dst)$ {Find a path from $src1$ to dst in the $u_d^{th}\mathrm{BCN}(\alpha,\beta,h)$ of $\mathrm{BCN}(\alpha,\beta,h,\gamma)$}
7: Return $head+(dst1,src1)+tail$

GetInterLink(s,d,v)
1: Infer two slave servers $[s,x=x_h\cdots x_1x_0]$ and $[d,y=y_h\cdots y_1y_0]$ from the s^{th} and $d^{th}\mathrm{BCN}(\alpha,\beta,h)$ in $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ such that (1) $v(x)=v(y)=v$; (2) $[s,x_\gamma\cdots x_1x_0]$ and $[d,y_\gamma\cdots x_1x_0]$ satisfy the constraints defined by Formula 3.
2: Return $([s,x],[d,y])$

4.2 Multipaths 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 can we generate parallel paths between any pair of servers.

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

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 $\mathrm{BCN}(\alpha,\beta,i)$ is the lowest level BCN that contains the two servers src and dst. The FdimRouting determines the link $(dst1, src1)$ that interconnects the two $\mathrm{BCN}(\alpha,\beta,i-1)$s each contains one of the two servers, and then builds the first path passing that link. There are α one lower level $\mathrm{BCN}(\alpha,\beta,i-1)$ in $\mathrm{BCN}(\alpha,\beta,i)$ that contains the dst. The first path does not pass other intermediate $\mathrm{BCN}(\alpha,\beta,i)$s, while each of other $\alpha-2$ parallel paths must traverse one intermediate $\mathrm{BCN}(\alpha,\beta,i-1)$.

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 subpath from the source src to an intermediate server z and a subpath from z to the destination dst. The two subpaths can be produced by the FdimRouting. So far, all of 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, thus, are parallel.

As for $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$ with $\alpha^\gamma\beta+1$ copies of $\mathrm{BCN}(\alpha,\beta,\gamma)$, if src and dst reside in the same $\mathrm{BCN}(\alpha,\beta,\gamma)$, there are $\alpha-1$ parallel paths between src and dst according to Lemma 1. Otherwise, we assume A and B denote two $\mathrm{BCN}(\alpha,\beta,\gamma)$s in which src and dst reside, respectively. In such a case, there exist $\alpha^\gamma\beta$ parallel paths between A and B because $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$ connects $\alpha^\gamma\beta+1$ copies of $\mathrm{BCN}(\alpha,\beta,\gamma)$ by means of a complete graph. In addition, Lemma 1 shows that there are only $\alpha-1$ parallel paths between any two servers in $\mathrm{BCN}(\alpha,\beta,\gamma)$, 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 $\mathrm{BCN}(\alpha,\beta,\gamma,\gamma)$ but not in the same $\mathrm{BCN}(\alpha,\beta,0)$.*

In the case that $h>\gamma$, $\mathrm{BCN}(\alpha,\beta,\gamma)$ is the unit cluster of $\mathrm{BCN}(\alpha,\beta,h,\gamma)$. Assume src and dst are labeled 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 $\mathrm{BCN}(\alpha,\beta,\gamma)$ 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 $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ but not in the same $\mathrm{BCN}(\alpha,\beta,0)$.*

Although BCN has the capability of providing multipaths for the one-to-one traffic, the existing routing schemes, including the FdimRouting and the BdimRouting, only exploit one path. To enhance the transmission reliability for the one-to-one traffic, we adapt the routing path when the transmission meets failures 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 that connects the destination server in the last step. This does not hurt the fault-tolerant ability of those parallel paths except the switch connecting the destination fails. In such a rare case, at most one reachable path exists between two servers.

4.3 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 the 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 the FdimRouting. On the contrary, the *remote reroute* modifies those *remote links* in a path derived by BdimRouting. All of the links that interconnect master servers using the second ports are called the *local links*, while those links that interconnect slave servers using the second ports are called the *remote links*.

4.3.1 Local Reroute

Given any two servers src and dst in $\mathrm{BCN}(\alpha,\beta,h,\gamma)$ $(h<\gamma)$, we can calculate a path from src to dst using the FdimRouting. Consider any failed link $(src1, dst1)$ in such a 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 the *local reroute* to bypass failed links by making local decisions. Here, each server has only local information. That is, it knows only the health state of the other servers on its directly connected switch (the master and slave servers) and the server connected over its second NIC (if any). Each server computes a set of relay servers for a failed next-hop server on demand. The assumption is that if the direct next hop is not reachable at least one of the relay servers can be reachable from the current server.

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

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

Otherwise, we first derive the server $dst2$ that 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 (6), i.e., the $relay$ server of the server $dst1$. In summary, the total number of such relay servers is $\alpha - 2$, where $\alpha \approx (2 \cdot \gamma \cdot n) / (2 \cdot \gamma + 1)$ as shown in Theorem 8. It is unlikely that all of relay servers for a failed one-hop server will be unreachable simultaneously because the switch ports, n, in a data center are typical not small.

In (6), the notation $h - m$ indicates that the two servers $src1$ and $dst1$ are in the same $\mathrm{BCN}(\alpha, \beta, h - m)$ but in two different $\mathrm{BCN}(\alpha, \beta, h - m - 1)$s. There exist $\alpha \mathrm{BCN}(\alpha, \beta, h - m - 1)$ subnets inside such a $\mathrm{BCN}(\alpha, \beta, h - m)$. When $src1$ finds the failure of $dst1$, it chooses one *relay* server from all of $\mathrm{BCN}(\alpha, \beta, h - m - 1)$ subnets in such a $\mathrm{BCN}(\alpha, \beta, h - m)$ except the two subnets that contain $src1$ or $dst1$. If $src1$ selects a relay server for $dst1$ from $\mathrm{BCN}(\alpha, \beta, h - m - 1)$ 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 the server 411 fails, the 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 toward 444, the derived path by Algorithm 2 from 211 to 444 is 211→214→241→244→422→424→442→444. It is worth noticing that if any link in the subpath from 144 to 221 fails, the head of that link must bypass such a 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 subpath from 211 to 444, the *local reroute* is used to address the failed link in the same way.

It is worth 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 a 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 (6). All of the servers each with an identifier starting with 1 from left to right cannot be the relay server because the path from the relay server to the destination will pass the failed link again.

If a server prefers to precompute and store the relay servers, it has to keep a forwarding table for its one-hop neighbors in the first dimension. Such forwarding table for relaying purpose is of size α, with each entry is of size $\alpha - 2$ because there exist $\alpha - 2$ relay servers for a failed one-hop server. Such a method incurs less delay than computing the relay servers on demand, however, consumes additional storage space of $O(\alpha^2)$. Such an overhead can be reduced to α if only a few relay servers are stored in each entry, e.g., three relay servers if such a number is enough for bypassing a failed one-hop server.

4.3.2 Remote Reroute

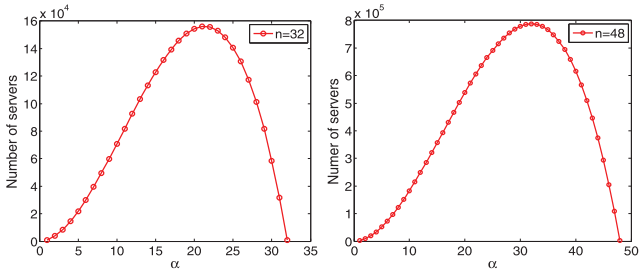
For any two servers, src and dst, in $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ $(h \geq \gamma)$, their 3-tuples are $[v_s, u_s, s_h \cdots s_1 s_0]$ and $[v_d, u_d, d_h \cdots d_1 d_0]$, respectively. The *local reroute* can handle any failed link in the path from src to dst if they are in the same $\mathrm{BCN}(\alpha, \beta, h)$ in $\mathrm{BCN}(\alpha, \beta, h, \gamma)$, i.e., $u_s = u_d$. Otherwise, a pair of servers $dst1$ and $src1$ are derived according to the $GetInterLink$ 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_sth $\mathrm{BCN}(\alpha, \beta, \gamma)$s inside $\mathrm{BCN}_{u_s}(\alpha, \beta, h)$ and $\mathrm{BCN}_{u_d}(\alpha, \beta, h)$, respectively. The link $(dst1, src1)$ is the only one that interconnects the two v_sth $\mathrm{BCN}(\alpha, \beta, \gamma)$s in the two $\mathrm{BCN}(\alpha, \beta, h)$s.

If the packets from src to dst meets failed links in the two subpaths 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 such cases, the packets cannot be forwarded from the u_sth $\mathrm{BCN}(\alpha, \beta, h)$ to the u_dth $\mathrm{BCN}(\alpha, \beta, h)$ inside such a $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ through the desired link $(dst1, src1)$. We propose the *remote reroute* to address such an 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 slave 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 $\mathrm{BCN}_{u_i}(\alpha, \beta, h)$ 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.

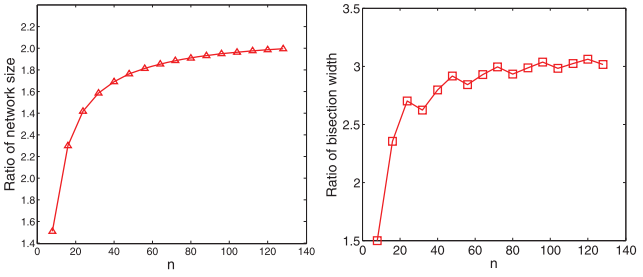
5 Evaluation

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



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

Fig. 7. The network order of $\mathrm{BCN}(\alpha, \beta, 1, 1)$ versus 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 $\mathrm{BCN}(\alpha, \beta, 1, 1)$ to that of FiConn$(n, 2)$, where their network diameters are the same 7.

5.1 Large Network Order

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

Proof. As mentioned in Section 3, any given level BCN consists of α one lower BCNs. There are α^h level-0 BCNs in $\mathrm{BCN}(\alpha, \beta, h)$, where a level-0 BCN consists of α master servers and β slave servers. Thus, proved. □

Lemma 4. *The number of servers in* $G(\mathrm{BCN}(\alpha, \beta, h))$ *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 3, there are $\alpha^h \cdot \beta + 1$ copies of $\mathrm{BCN}(\alpha, \beta, h)$ in $G(\mathrm{BCN}(\alpha, \beta, h))$. In addition, the number of servers in $\mathrm{BCN}(\alpha, \beta, h)$ has been proved by Lemma 3. Thus, proved. □

Theorem 7. *The number of servers in* $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ *is*

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

Proof. Lemma 3 has proved such an issue when $h < r$. In addition, $\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$ is just $G(\mathrm{BCN}(\alpha, \beta, \gamma))$. Thus, there are $\alpha^\gamma \cdot (\alpha + \beta) \cdot (\alpha^\gamma \cdot \beta + 1)$ servers in $\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$. In addition, $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ contains $\alpha^{h-\gamma}\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$s when $h \geq \gamma$. Thus, proved. □

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

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

Proof. The total number of servers in $\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$ 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

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

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 $\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$. Thus, proved. □

Fig. 7 plots the number of servers in $\mathrm{BCN}(\alpha, \beta, 1, 1)$ 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 $\mathrm{BCN}(\alpha, \beta, 1, 1)$ is 787,968 for $n = 48$ and 155,904 for $n = 32$, and can be achieved only if $\alpha = 32$ and 21, respectively. Such experimental results match well with Theorem 8.

Fig. 8a depicts the changing trend of the ratio of the network order of $\mathrm{BCN}(\alpha, \beta, 1, 1)$ to that of FiConn$(n, 2)$ as the number of ports in each miniswitch increases, where α is set to 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 server degree 2 and the network diameter 7, irrespective the value of n. As shown in Table 2, the network size of HCN$(n, 2)$ is less than that of FiConn$(n, 2)$ as expected.

Formula (7) indicates that the network order of 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 because a higher level BCN requires only α one lower level BCNs except $h = \gamma$. 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.

5.2 Low Diameter and Server Degree

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

TABLE 2
Network Orders, Bisection Widths, and Path Diversity of $\mathrm{BCN}(\alpha, \beta, 1, 1)$, FiConn$(n,2)$, and $\mathrm{HCN}(n, 2)$

n	8	16	24	32	40
Network order(BCN)	640	9856	49536	155904	380160
Network order(FiConn)	440	5328	24648	74528	177240
Network order(HCN)	512	4096	13824	32768	64000
Bisection width(BCN)	42	784	4160	12210	30976
Bisection width(FiConn)	28	333	1541	4658	11078
Bisection width(HCN)	16	64	144	256	400
Path diversity(BCN)	5	10	15	21	26
Path diversity(HCN)	7	15	23	31	39

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

BCN of level one in each dimension offers more than 1,000,000 servers if 56-port switches are used, while the server degree and network diameter are only 2 and 7, respectively. This demonstrates the low network diameter and server degree of BCN.

5.3 Connectivity and Path Diversity

The edge connectivity of a single server is one or two in $\mathrm{BCN}(\alpha, \beta, h, \gamma)$. Consider the fact that $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ 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* $\mathrm{BCN}(\alpha, \beta, h, \gamma)$*, the smallest number of remote links or servers that can be deleted to disconnect one* $\mathrm{BCN}(\alpha, \beta, i)$ *from the entire network is*

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

Proof. If $h < \gamma$, consider any subnet $\mathrm{BCN}(\alpha, \beta, i)$ for $0 \leq i < h$ in $\mathrm{BCN}(\alpha, \beta, h, \gamma)$. 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.

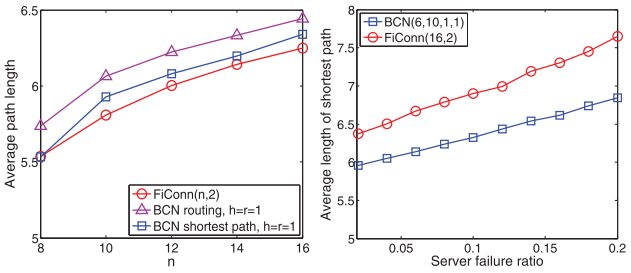
If $h \geq \gamma$, besides the $\alpha - 1$ remote links that connect its master servers the subnet $\mathrm{BCN}(\alpha, \beta, i)$ has $\alpha^{i}{\cdot}\beta$ additional remote links that connect its slave servers. Thus, it can be disconnected only if the corresponding $\alpha - 1 + \alpha^{i}{\cdot}\beta$ 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* $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ *into two parts of about the same size is given by*

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

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 [15]. For $1 \leq h < \gamma$, $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ is a compound graph, where G is a complete graph with α nodes and G_1 is $\mathrm{BCN}(\alpha, \beta, h-1, \gamma)$. 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 $\mathrm{BCN}(\alpha, \beta, h-1, \gamma)$ can be induced in this way and is the same as that of G. Thus, the bisection width of $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ for $1 \leq h < \gamma$ is proved.

$\mathrm{BCN}(\alpha, \beta, h, \gamma)$ 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 $\mathrm{BCN}(\alpha, \beta, h)$. We can see that the bisection width of G is $(\alpha^{\gamma}{\cdot}\beta + 2){\cdot}\alpha^{\gamma}{\cdot}\beta/4$ and 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 $\mathrm{BCN}(\alpha, \beta, \gamma)$, and there is one link between the ith $\mathrm{BCN}(\alpha, \beta, \gamma,)$s in 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; hence, the bisection width of $\mathrm{BCN}(\alpha, \beta, h, \gamma)$ is $\alpha^{h-\gamma}{\cdot}(\alpha^{\gamma}{\cdot}\beta + 2){\cdot}\alpha^{\gamma}{\cdot}\beta/4$. Thus, proved. □



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

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

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

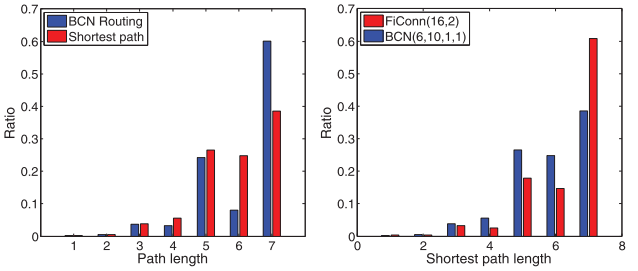
For any $\mathrm{FiConn}(n, k)$, the bisection width is at least $N_k/(4{*}2^k)$, where $N_k = 2^{k+2} * (n/4)^{2k}$ denotes the number of servers in the network [5]. We then evaluate the bisection width of $\mathrm{FiConn}(n, 2)$ and $\mathrm{BCN}(\alpha, \beta, 1, 1)$ under the same server degree, switch degree, and network diameter. In such a setting, the network size of $\mathrm{BCN}(\alpha, \beta, 1, 1)$ outperforms that of $\mathrm{FiConn}(n, 2)$. Fig. 8b shows that $\mathrm{BCN}(\alpha, \beta, 1, 1)$ significantly outperforms $\mathrm{FiConn}(n, 2)$ in terms of the bisection width. We can see from Table 2 that the bisection width of $\mathrm{HCN}(n, 2)$ is less than that of $\mathrm{FiConn}(n, 2)$ as expected. Larger bisection width implies higher network capacity and more resilient against failures.

As proved in Lemma 2, there are $\alpha - 1$ node-disjoint paths between any two servers in $\mathrm{BCN}(\alpha, \beta, \gamma, \gamma)$, 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 such disjoint paths, the transmission rate can be accelerated and the transmission reliability can be enhanced. We see from Table 2 that $\mathrm{BCN}(\alpha, \beta, 1, 1)$ has a high path diversity for an one-to-one traffic and $\mathrm{HCN}(n, 2)$ outperforms $\mathrm{BCN}(\alpha, \beta, 1, 1)$.

5.4 Evaluation of the Path Length

We run simulations on $\mathrm{BCN}(\alpha, \beta, 1, 1)$ and $\mathrm{FiConn}(n, 2)$ in which $n \in \{8, 10, 12, 14, 16\}$ and α is set to 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. 9a 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 because the current routing protocols do not entirely realize the shortest path routing. The 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 FiConn.

Then, we evaluate the fault-tolerant ability of the topology and the routing algorithm of $\mathrm{BCN}(6, 10, 1, 1)$ and a $\mathrm{FiConn}(16, 2)$. The network sizes of $\mathrm{BCN}(6, 10, 1, 1)$ and $\mathrm{FiConn}(16, 2)$ are 5,856 and 5,327, respectively. As shown in



(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.

Fig. 9b, 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 terms of the average shortest path length when the server failure ratio is zero. Such results demonstrate that the topology and routing algorithms of BCN possess better fault-tolerant ability. Note that the network size of HCN is less than that of BCN and FiConn under the same configurations of n and h. Thus, it is clear that the network diameter and average path length are larger than that of BCN and FiConn under the same settings of N and n.

We run simulations on BCN$(\alpha,\beta,1,1)$ and FiConn$(n,2)$, where $n=16$ and $\alpha=6$. The network sizes of BCN$(6,10,1,1)$ and FiConn$(16,2)$ are 5,856 and 5,327, respectively. Thus, the two network structures have the same server degree 2, the same network diameter 7, and the similar network order. Fig. 10a indicates that the routing algorithm of BCN may not discover a few part of shortest paths and replace them with relative long routing paths. Fig. 10b plots the distribution of shortest path for BCN and FiConn. We can see that about 60 percent of the shortest paths in FiConn are of length 7 while only about 40 percent of the shortest paths in BCN are of length 7. On the other hand, the simulation results also match the theoretical values of the network diameters of BCN and FiConn.

5.5 Throughput Comparison

We further compare HCN and BCN with three network structures for data centers, including Fat Tree, DCell, and BCube, in terms of the aggregate capacity, as shown in Table 3. To ensure a fair comparison, we assume that such network structures interconnect the same number of servers with the same type of switches. HCN, BCN, DCell, and BCube are all recursively defined structures, and we denote the levels of them as k_1, k_2, k_3, and k_4, respectively. Typically, there is $k_1 \geq k_2 \geq k_3$ and $k_4 \geq k_3$. Note that N denotes the number of servers in a data center.

The maximum throughput of the one-to-one communication is the number of NIC ports per server. HCN and BCN are twice that of Fat-Tree, but less than that of DCell and BCube whose number of levels is typically larger than 2. In the case of the all-to-all communication, Fat-Tree and BCube perform best because they achieve the nonblock communication between any pair of servers, but HCN and BCN are a little worse than DCell. Such a result is not surprising because HCN and BCN utilize much less switches, links, and ports than the other three structures. We argue that the benefits of HCN and BCN outweigh such a downside because it is unlikely that all servers frequently participate in the all-to-all communication.

TABLE 3
Comparison of HCN, BCN, DCell, Fat-Tree, and BCube

Network structures	HCN	BCN	DCell	Fat Tree	BCube
One-to-one throughput	2	2	k_3	1	k_4
All-to-all throughput	$N/2^{k_1}$	$N/2^{k_2}$	$N/2^{k_3}$ [6]	N	N

6 Discussion

6.1 Extension to More Server Ports

Although we assume that all of the 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. 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 miniswitch. Consider that a set of $m-1$ servers each of which holds two ports and connects with the same miniswitch using its first port. It is clear that the set of $m-1$ servers can totally contribute $m-1$ ports for future higher level interconnections. Intuitively, a server with m ports can be treated as a set of $m-1$ dual-port servers. 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 a constant node degree of more than 2, and we leave such an investigation as our future work.

6.2 Locality-Aware Task Placement

As discussed in Section 5.5, HCN and BCN cannot achieve the same aggregate capacity as Fat-tree, DCell, and BCube. Fortunately, such an issue can be addressed by some techniques at the application layer, due to the observations as follows:

As prior work [6] has shown, a server is likely to communicate with a small subset of other servers when conducting typical applications in common data centers, such as the group communication, and VM migration. 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.

As shown in Table 3, the aggregate bottleneck throughput of HCN and BCN are $N/2^{k_1}$ and $N/2^{k_2}$, respectively. We assume that the bandwidth of each link in a data center is one. Thus, the throughput one flow receives at the bottleneck link is $1/(N\times 2^{k_1})$ and $1/(N\times 2^{k_2})$, respectively. Accordingly, the throughput a flow receives in the all-to-all group communications decreases along with the increase of the group size. A boundary on the group size can, thus, be derived given a constraint on the flow throughput between any pair of servers.

Furthermore, a locality-aware approach can be used for placing those tasks onto servers in HCN. That is, those tasks with intensive data exchange can be placed onto servers, in $\mathrm{HCN}(n,0)$, 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 5.1, HCN is usually sufficient to contain hundreds even thousands of servers, where the number of server hops is at most three. Similarly, we can use the locality-aware mechanism when placing tasks onto data center servers in BCN. Therefore, the locality-aware mechanism can largely save the network bandwidth by avoiding unnecessary remote data communications.

6.3 Impact of Server Routing

In both HCN and BCN, because 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 our HCN and BCN, they usually incur nontrivial CPU overhead. The hardware-based packet forwarding engine like CAFE [19] and ServerSwitch [20] are good candidates for supporting DCN designs. Inspired by the fact that CAFE and ServerSwitch can be easily configured, we can reconfigure them to forward self-defined packets for our HCN or BCN without any hardware redesigning.

7 Conclusion

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

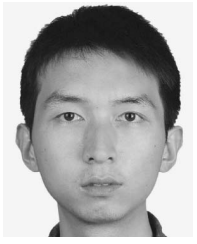
Acknowledgments

The authors would like to thank anonymous reviewers for their constructive comments. The work of Deke Guo is supported in part by the National Science Foundation (NSF) China under Grants No. 61170284 and No. 60903206. The work of Dan Li is supported in part by the NSF China under grant No. 61170291. The work of Yunhao Liu is supported in part by the NSFC Major Program under grant No. 61190110 and National High-Tech R&D Program of China (863) under Grant No. 2011AA010100. The work of Guihai Chen is supported in part by the NSF China under No. 61133006 and the National Basic Research Program of China (973 Program) under No. 2012CB316200.

References

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *Proc. 19th ACM Symp. Operating Systems Principles (SOSP)*, pp. 29-43, 2003.
[2] 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 Trans. Computer Systems*, vol. 26, no. 2, article 4, 2008.
[3] M.A. Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proc. ACM SIGCOMM*, 2008.
[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A Scalable and Fault-Tolerant Network Structure for Data Centers," *Proc. ACM SIGCOMM*, 2008.
[5] D. Li, C. Guo, H. Wu, Y. Zhang, and S. Lu, "Ficonn: Using Backup Port for Server Interconnection in Data Centers," *Proc. IEEE INFOCOM*, 2009.
[6] 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 Trans. Networking*, vol. 19, no. 1, pp. 102-114, Feb. 2011.
[7] 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," *Proc. ACM SIGCOMM*, 2009.
[8] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, and P. Patel, "VL2: A Scalable and Flexible Data Center Network," *Proc. ACM SIGCOMM*, 2009.
[9] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: A High Performance Network Structure for Modular Data Center Interconnection," *Proc. ACM CONEXT*, 2009.
[10] D. Li, M. Xu, H. Zhao, and X. Fu, "Building Mega Data Center from Heterogeneous Containers," *Proc. IEEE 19th Int'l Conf. Network Protocols (ICNP)*, pp. 256-265, 2011.
[11] M. Miller and J. Siran, "Moore Graphs and Beyond: A Survey of the Degree/Diameter Problem," *Electronic J. Combinatorics*, vol. 61, pp. 1-63, Dec. 2005.
[12] N. Alon, S. Hoory, and N. Linial, "The Moore Bound for Irregular Graphs," *Graphs and Combinatorics*, vol. 18, no. 1, pp. 53-57, 2002.
[13] R.M. Damerell, "On Moore Graphs," *Proc. Cambridge Philosophical Soc.*, vol. 74, pp. 227-236, 1973.
[14] M. Imase and M. Itoh, "A Design for Directed Graphs with Minimum Diameter," *IEEE Trans. Computers*, vol. C-32, no. 8, pp. 782-784, Aug. 1983.
[15] D.P. Agrawal, C. Chen, and J.R. Burke, "Hybrid Graph-Based Networks for Multiprocessing," *Telecomm. System*, vol. 10, pp. 107-134, 1998.
[16] L.N. Bhuyan and D.P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Trans. Computers*, vol. C-33, no. 4, pp. 323-333, Apr. 1984.
[17] K. Chen, C. Guo, H. Wu, J. Yuan, Z. Feng, Y. Chen, S. Lu, and W. Wu, "DAC: Generic and Automatic Address Configuration for Data Center Networks," *IEEE/ACM Trans. Networking*, vol. 20, no. 1, pp. 84-99, Feb. 2012.
[18] P.T. Breznay and M.A. Lopez, "A Class of Static and Dynamic Hierarchical Interconnection Networks," *Proc. IEEE Int'l Conf. Parallel Processing (ICPP)*, vol. 1, pp. 59-62, 1994.
[19] G. Lu, Y. Shi, C. Guo, and Y. Zhang, "Cafe: A Configurable Packet Forwarding Engine for Data," *Proc. ACM SIGCOMM Workshop Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, 2009.
[20] G. Lu, C. Guo, Y. Li, and Z. Zhou, "Serverswitch: A Programmable and High Performance Platform for Data Center Networks," *Proc. Eighth USENIX Conf. Networked Systems Design and Implementation (NSDI)*, pp. 15-28, 2011.

Deke Guo received the BS degree in industry engineering from Beijing University of Aeronautic and Astronautic, Beijing, China, in 2001, and the PhD degree in management science and engineering from National University of Defense Technology, Changsha, China, in 2008. He is an associate professor with the College of Information System and Management, National University of Defense Technology, Changsha, China. His research interests include distributed systems, wireless and mobile systems, P2P networks, and interconnection networks. He is a member of the ACM and the IEEE.



Tao Chen received the BS degree in military science, and the MS and PhD degrees in military operational research from the National University of Defense Technology, Changsha, China, in 2004, 2006, and 2011, respectively. He is an assistant professor with the College of Information System and Management, National University of Defense Technology, Changsha, P.R. China. His research interests include wireless sensor networks, peer-to-peer computing, and data center networking. He is a member of the IEEE.

Dan Li received the PhD degree in computer science from Tsinghua University, Beijing, China, in 2007. He is an assistant professor with the Computer Science Department, Tsinghua University. Before joining the faculty of Tsinghua University, he spent two years as an associate researcher with the Wireless and Networking Group, Microsoft Research Asia, Beijing, China. His research interests include Internet architecture and protocols, P2P networks, to cloud computing networks. He is a member of the IEEE.

Mo Li received the BS degree in computer science and technology from Tsinghua University, Beijing, China, in 2004, and the PhD degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2009. He is a Nanyang assistant professor with the Computer Science Division, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, wireless sensor networks, pervasive computing and RFID, and wireless and mobile systems. He is a member of the IEEE.

Yunhao Liu received the BS degree in automation from Tsinghua University, China, in 1995, and the MS and PhD degrees in computer science and engineering from Michigan State University, in 2003 and 2004, respectively. He is currently an EMC chair professor at Tsinghua University, as well as a faculty member with the Hong Kong University of Science and Technology. His research interests include wireless sensor network, peer-to-peer computing, and pervasive computing. He is a senior member of the IEEE Computer Society and the IEEE, and an ACM distinguished speaker.

Guihai Chen received the BS degree from Nanjing University, M. Engineering from Southeast University, and the PhD degree from the University of Hong Kong. He visited Kyushu Institute of Technology, Japan in 1998 as a research fellow, and the University of Queensland, Australia in 2000 as a visiting professor. During September 2001 to August 2003, he was a visiting professor in Wayne State University. He is a distinguished professor and deputy chair with the Department of Computer Science, Shanghai Jiao Tong University. He has published more than 200 papers in peer-reviewed journals and refereed conference proceedings in the areas of wireless sensor networks, high-performance computer architecture, peer-to-peer computing, and performance evaluation. He has also served on technical program committees of numerous international conferences. He is a member of the IEEE Computer Society and a senior member of the IEEE.