Front.Comput.Sci.
DOI

RESEARCH ARTICLE

Compound Graph Based Hybrid Data Center Topologies

Lailong LUO [1], Deke GUO (✉)[1], Wenxin LI [2], Tian ZHANG [3], Junjie XIE[1], Xiaolei ZHOU[1]

1 Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, 410073, China
2 Network and Cloud Computing Laboratory, Dalian University of Technology, Dalian, 116024, China
3 School of Information Management, Wuhan University, Wuhan, 430070, China

Abstract In large-scale data centers, many servers are interconnected via a dedicated networking structure, so as to satisfy specific design goals, such as the low equipment cost, the high network capacity, and the incremental expansion. The topological properties of a networking structure are critical factors that dominate the performance of the entire data center. The existing networking structures are either fully random or completely structured. Although such networking structures exhibit advantages on given aspects, they suffer obvious shortcomings in other essential fields. In this paper, we aim to design a hybrid topology, called R3, which is the compound graph of structured and random topology. It employs random regular graph as a unit cluster and connects many such clusters by means of a structured topology,i.e. the generalized hypercube. Consequently, the hybrid topology combines the advantages of structured as well as random topologies seamlessly. Meanwhile, a coloring-based algorithm is proposed for R3 to enable fast and accurate routing. R3 possesses many attractive characteristics, such as the modularity and expansibility at the cost of only increasing the degree of any node by one. Comprehensive evaluation results show that our hybrid topology possesses excellent topology properties and network performance.

Keywords Data center networking, Compound graph, Hybrid topology, Routing design

Received month dd, yyyy; accepted month dd, yyyy

E-mail: *guodeke@gmail.com*

1 Introduction

Data centers are the dominant infrastructure of cloud computing and network applications. Inside a data center, large number of servers and switches are interconnected using a specific data center networking (DCN). That is, all switches are interconnected to form a given networking structure, and those switches use their remaining ports to connect large-scale servers. Recently, some novel networking structures were proposed for future data centers. Such proposals can be roughly divided into two categories. The first is structured topologies, each of which organizes switches into structured networks with strict interconnection rules on components in a data center. Fat-Tree [1], VL2 [2], BCN [3] fall into this category. On the contrary, random topologies break strict interconnection rules by introducing random links among switches, e.g. SMDC [4], Jellyfish [5], Scafida [6]. Such structured topologies exhibit high throughput but are not incremental expansion. Those random topologies are incremental expansion; however, they suffer complex cabling and routing processes. On the other hand, the superiority of structured and random topologies is complementary. In this paper, the following two fundamental problems motivate us to design new networking structures that can tightly integrate the superiority together and abandon the weakness.

Can existing structured topologies of data centers satisfy the requirements of today's applications perfectly? For structured topologies, the strict interconnection rules simplify the construction of DCNs; however, they limit the incremental expansion of deployed data centers. In reality, a production

data center needs to be gradually extended along with the increasing demands of applications and users. But the structured DCNs fail to support the incremental expansion.

Can random topologies truly improve the performance of DCNs? Random DCNs, such as Jellyfish [5] and Scafida [6], support the incremental expansion naturally. Meanwhile, random links decrease the network diameter by connecting remote nodes together. Furthermore, random DCNs can integrate heterogeneous devices during the expansion process [7]. However, beside the cabling cost due to remote random links, routing and maintenance in random DCNs are essential issues. First, the Dijkstra or Floyd algorithm, whose computation complexity is $O(n^2)$ or even $O(n^3)$, is used to search the shortest paths between any pair of nodes in random DCNs. It is clear that routing in random DCNs is time-consuming. Second, disordered and unsystematic link distribution incurs nontrivial maintain cost. That is, running a random DCN is relative expensive.

Fortunately, the superiorities of structured DCNs and random DCNs are complementary. This motivates us to seek new topologies which can integrate their superiorities together and avoid their weakness. For this reason, we propose a family of hybrid topologies, which are compound graphs of given structured and random topologies. That is, given number of random structures are unit clusters, which are then interconnected by means of a structured structure. In this way, a family of hybrid DCNs is built when utilizing different random and structured graph. Consequently, the resulting hybrid DCNs can naturally integrate the characteristics of incremental expansion and fast routing via compound graph theory. Note that any of such hybrid topologies is a structured topology from the global viewpoint, while is a random topology from a local viewpoint.

To fully exploit the benefits of our hybrid topologies, a coloring-based routing algorithm is proposed to derive the routing path between any pair of nodes in the hybrid DCNs. Typically, a unique identifier, which consists of two parts, i.e., the inter-identifier and the inner-identifier, is assigned to each node. The inter-identifier locates the random cluster the nodes reside in and the inner-identifier locates the node inside random clusters. The structured links are colored with different colors, the two endpoints of each colored link are assigned with the same inner-identifier. In this way, coupled with the color of structured links, the inter-identifiers can derived the random cluster level path, then the paths inner each random cluster can be calculated with shortest path algorithm like Dijkstra. Then, we build an integer programming model to minimize the average path length (APL) problem in our hybrid structures with given number of servers and port count per switch. Furthermore, a random cluster level expansion algorithm is designed for realizing the incremental expansion.

The major contributions of this paper are summarized as follows.

- We propose the design methodology of hybrid DCNs, which embeds some random clusters into a given structured topology. We further design a nonlinear integer programming model to derive an optimized hybrid topology.
- We design an edge coloring based routing algorithm, whose routing cost is much lower than traditional algorithms. The associated incremental expansion methods are designed in different levels.
- We conduct extensive experiments. The results demonstrate that, compared with Jellyfish, our hybrid structures bring less routing time and cabling cost. Compared with the generalized hypercube, our hybrid structures process better network order and throughput.

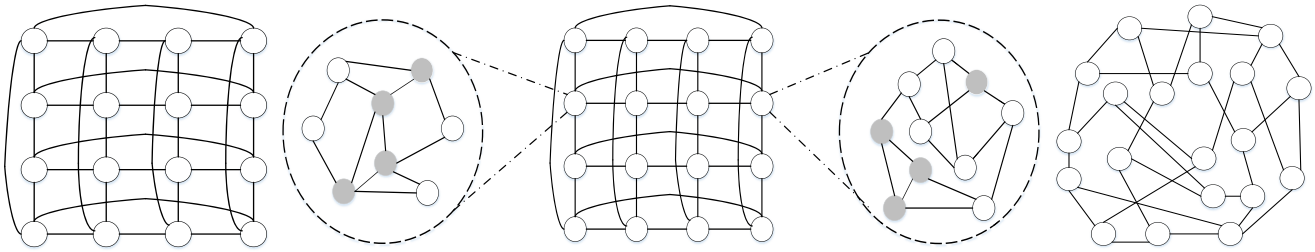
The rest of this paper is organized as follows. Section 2 summarizes the related preliminaries. Section 3 proposes the structures of hybrid DCNs, and Section 4 designs the associated routing algorithm. Section 5 tries to optimize the design of hybrid DCNs. Section 6 discusses the issue of incremental expansion. Section 7 evaluates the performance of our proposal. Section 8 introduces the related work, and Section 9 concludes this paper.

2 Preliminaries

To clearly present our proposal, we first introduce the background and preliminary knowledge. We summarize the definitions and properties of generalized hypercube and random regular graph, which are representing structured and random topologies, respectively.

2.1 Generalized hypercube

Generalized hypercube(GHC) [8] is defined as follows. Let $G(m_s, m_{s-1}, \cdots, m_2, m_1)$ denote a generalized hypercube of N node, where $N{=}m_s{*}m_{s-1}{*}\cdots{*}m_2{*}m_1$ and $m_i{\geq}2$ for all $1{\leq}i{\leq}s$. Each node has a unique r-digit identifier $x_s x_{s-1}\cdots x_2 x_1$, where $x_i{\in}[0, m_i{-}1]$ for all $1{\leq}i{\leq}s$. Two nodes are adjacent if and only if their identifiers are different in one dimension, i.e. the hamming distance between two identifiers is exactly 1.



(a) 2D torus. (b) Our hybrid design. (c) Random graph.

Fig. 1 Structured and random topologies are extremes. Between them, lies an example of our design, which is a compound graph with 2D-Torus as the structured part and different types of random clusters. The solid nodes in each random cluster are chosen to connect with other random clusters.

GHC is more flexible than standard hypercube. Hypercube can only accommodate 2^i nodes, for $i \geq 0$, while GHC can accommodate any number of nodes via careful configuration. For example, we can not construct a hypercube with $N = 24$, but can derive several GHCs with different configurations since 24=2∗2∗2∗3=2∗3∗4=4∗6=3∗8. Among such GHCs, there is a trade-off between the network diameter and the node degree since larger node degree contributes to shorter network diameter.

GHC is a representative structured topology, where the construction and routing rules are deterministic. Furthermore, GHC is very flexible such that designers could configure the network structure based on their demand.

2.2 Random regular graph

A random r-regular graph (r-RRG) is a graph selected from $G_{n,r}$, which denotes the probability space of all r-regular graphs on n vertices, where $3 \leq r < n$ and $n \times r$ is even [9]. Generally, a RRG has excellent topology characteristics. First, all nodes have the same degree. Second, given r and n, a RRG has a lower bound on the network diameter. That is, for the same n, the network diameter of different RRGs is roughly $\log_{r-1} n$.

r-RRGs have excellent properties such as coloring and Hamiltonian cycle. Most importantly, RRGs support the incremental expansion properly by adding racks one by one without changing the existing network too much. Because of these outstanding characteristics, RRGs were introduced into data center topologies.

2.3 Compound graph

Definition 1. *Given two structured graphs G and G_1, a level-1 compound graph $G(G_1)$ is obtained by replacing each node of G with a copy of G_1, and then replacing each link of G by a link, which connects two corresponding copies of G_1 [3].*

A compound graph is usually visualized as a node-link-diagram using the nested box metaphor [10]. If the node degree of G is equal to the number of nodes in G_1, the resultant graph is a *complete compound graph*. Otherwise, it is an *incomplete compound graph*. In fact, higher level compound graphs can be derived from low level ones, recursively. $G(G_1)$ maintains the topological characteristic of G and G_1. In other words, the compound graph combines the advantages of both parts efficiently. For this reason, the use of compound graphs has become more important in recent years. For example, in social network analysis large networks are transformed in a so-called block model in order to cope with the huge amount of data. From a graphtheoretical point of view these block models are compound graphs [10]. Also, for DCNs, several proposals are designed based on the compound graph, e.g. BCN [3], DCell [11], KCube [12], DCube [13] and so on.

3 Hybrid topology design

In this paper, we aim to combine the advantage of both structured topologies and random ones via designing a family of hybrid topologies for data centers. *Compound graphs* (either complete or incomplete) are introduced as the medium between random and structured graphs. More precisely, we combine the generalized hypercube with the random regular graph to derive a hybrid topology called *R3*.

3.1 Overview of topologies

The existing wired DCNs topologies are mostly belong to two categories, i.e. random ones and structured ones. Structured topologies lack scalability and incremental expansion, while the random ones suffer considerable routing overhead. For example, the routing in hypercube is easy to implement since the hamming distance between two node identifiers will judge the existence of a direct link between any pair of nodes.

Table 1 Symbols and notations.

Term	Definition
G	A simple graph
N	The total number of node in a graph
$G(G_1)$	A compound graph based on G and G_1
m_i	Order of the i^{th} dimension in GHC
Δ	Maximum node degree in a simple graph
$x'(G)$	Rdge chromatic number of G
T	Total number of servers in R3
p	Port count of each switch in R3
t	Total switch that used in R3
α	Number of switch ports that link with switches
β	Number of switch ports that link with servers

When a hypercube based data center needs to be extended, the data center has to double the amount of servers. For Scafida [6], the topology can be incrementally expanded by adding any number of servers, but routing is very tough due to large number of random links and the lack of topology information. In this paper, we pursue hybrid topologies for data centers, which can combine the superiorities of both random and structured topologies. The resultant hybrid topologies are both random and structured at different viewpoints.

Available structured topologies: Tree, Hypercube, Generalized Hypercube, Torus, and other structured topologies are permitted to appear at our hybrid topologies.

Available random topologies: Small-word network, Scale-free network and Random regular graph of random topologies are potential choices. Each random cluster plays as a node in the selected structured topology.

Design principle of hybrid topologies: we first choose one structured topology and single or multiple random topologies, and then combine them together with the principle of compound graph. In the resultant topology, the links derived from structured graph is called **structured links**, while the links inside a random clusters is called **random links**. In our design, only one structured topology can be used since it is the container of random clusters, but heterogeneous random clusters can be used. The incremental expansion and routing will be demonstrated later in Sections 4 and 6, respectively.

Without loss of generality, in our hybrid topologies, each switch is viewed as an intelligent node, whose partial ports interconnect with other switches, while the rest ports are used to connect servers inside a rack.

3.2 R3: compound graph theory based hybrid topology

As depicted in Section 2, compound graph is a powerful method to integrate two kinds of topologies while remaining their superiorities. This motivates us to use the compound graph theory to construct our hybrid topologies. Different combinations of structured and random topologies result in different hybrid topologies. This will enlarge the design space and increase the design flexibility. In this way, our hybrid topologies enable designers to construct their data center networks on demand.

Algorithm 1 Building Hybrid Topology, H

Require: Given a structured topology G, r denotes the number of nodes in G. Let $Adjacent[r][r]$ denote the adjacent matrix of G and R_i denote the random clusters.

```
1: Initialize each random cluster;
2: Let Link[x] count all links already connected to the x^th node in G;
3: Let Degree[x] be the degree of the x^th node in G;
4: for i=0 to r do
5:     for j=i to r do
6:         if Adjacent[i][j]==1 and Link[a]<Degree[a], a=i, j then
7:             add a link between i^th and j^th random clusters;
8:             Link[i]++;
9:             Link[j]++;
10: return The hybrid topology H.
```

To construct a hybrid topology correctly based on the compound graph theory, three constraints must be satisfied.

Constraint 1: all random clusters must be interconnected via a structured topology.

Constraint 2: the number of random clusters must equal to the number of chosen nodes in the structured topology G.

Constraint 3: the lower bound on the amount of nodes in each random cluster can't be less than the maximum degree plus one in the structured topology.

We describe the basic process of building a hybrid topology in Algorithm 1. In Algorithm 1, given the number of links that have been linked to each node in G, the adjacent matrix determine whether current nodes need to be linked. If i^{th} and j^{th} random clusters are connected, a link will be added to connect one random node in each cluster. If the number of nodes in a random cluster is less than the degree of node in the structured topology then the degree of some nodes will be increased by more than one. Each proposed hybrid topology is the generalization of the involved random and structured topologies under specific settings. This design methodology has two extreme cases. If the random clusters have only one node, the resultant topology is just the structured graph we used (depicted in Fig.1(a)). If the structured graph is just a single node, as shown in Fig.1(c), then the hybrid topologies degrade into a fully random cluster.

Fig.1(b) depicts an example of hybrid topology constructed with the Torus and random topologies. If each random

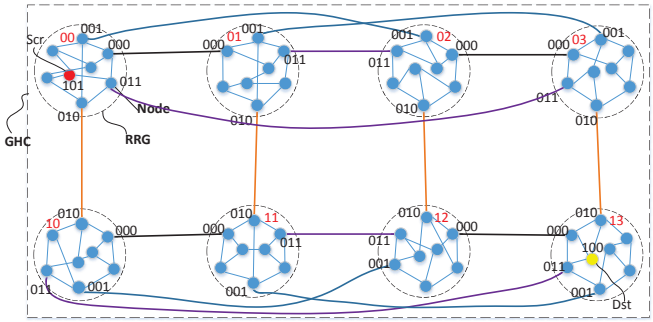


Fig. 2 A 2 × 4 R3 topology.

cluster is viewed as a node in Torus, then the topology is *structured, and hence easy-routing*. While, in each random cluster, the topology is *random and incremental expandable*. In our hybrid topology, different random clusters are allowed to be embedded; meanwhile, it is not necessary that the amount of nodes in such random clusters are the same. That is, our methodology can derive all hybrid topologies that lie between structured and random topologies.

As aforementioned, the vast design space results in variable hybrid topologies with diverse randomness. To simplify the presentation, we focus on a representative hybrid topology, called *R3*. R3 employs the generalized hypercube as its structured part and the random regular graph as the unit of random clusters. Generalized hypercube and random regular graph are two representative topologies, which are widely used for designing network structures for data centers.

Definition 2. *R3*$(G(m_s, m_{s-1}, \cdots, m_1), r\text{-}RRGs)$ *denotes a kind of hybrid topologies, each of which is the compound graph of a random regular graph r-RRG and a GHC, whose dimensions are* $m_s, m_{s-1}, \cdots, m_1$*, respectively.*

Definition 3. *In a R3 topology, the selected nodes from each random cluster for linking with other random clusters are called* Boundary Nodes.

Therefore, according to Definitions 2 and 3, a *R3* topology contains number of $m_s{*}m_{r-1}{*}\cdots{*}m_1$ RRGs and the amount of boundary nodes is $\sum(m_i{-}1)$, where $1{<}i{<}s$. Fig.2 depicts a 2×4 dimension topology, where eight random clusters of 3-RRGs are embedded and each cluster accommodates 8 nodes. These structured links across random clusters form a 2×4 generalized hypercube. Every cluster is assigned a two-dimensional identifier(the two digit red number in Fig.2). Similarly, each node in a random cluster also owns its identifier, the allocation rules will be discussed later in Section 4. All boundary nodes are chosen randomly, while, the amount of nodes in a random cluster can be an arbitrary value, which is not less than the node degree in GHCs. In *R3*, all parameters are adjustable. Due to the flexibility of parameter setting, R3s with different scales can be easily built according to its definition.

3.3 Deployment strategy for hybrid based data center

To put the hybrid topologies into real deployment, we investigate that both top-down wiring strategy and down-top wiring strategy are feasible.

The top-down deployment strategy mainly includes two steps. At the first step, we build the structured topology since it is the main skeleton of our hybrid topology. Having selected an appropriate structured topology, we deploy the boundary nodes of each cluster and interconnect these boundary nodes with structured links. When all structured links have been cabled, the first step will be terminated. At the second step, we fill the random clusters via adding nodes into the clusters in a way the random topologies requires. In this way, the hybrid topology will be constructed successfully.

Unlike the top-down strategy, the down-top deployment strategy establishes the random clusters at the first step so that all random blocks are prepared for later interconnection. However, how to establish the random clusters depends on which topology the designer utilized. Note that the number of random clusters must be equal to the number of nodes in the structured topology we employed according to the compound graph theory. At the second step, we chose the boundary nodes from each random cluster and interconnect them together via structured links in a way the structured topology requires. When all links have been cabled, the construction process will be terminated.

4 Efficient routing methods of R3

Routing in our hybrid topologies needs a dedicated design method because of the coexistence of random links and structured links. In our hybrid topologies, the major challenge of routing comes from the embedded random clusters. In random topologies, like Scafida or Jellyfish, the shortest routing path between any pair of nodes can be decided only by *Dijkstra* like methods, which incur considerable searching cost in large-scale data centers. In structured topologies, the topological characteristics can significantly ease the computation process of the shortest routing path.

Unlike BCube, where regularity and symmetry of the topology supports fast routing under differen flow pattern-

s [14], the routing in R3, however, turns to be complicated in each of our hybrid topologies due to the following two challenges. First, the unstructured topology of each random cluster denies the possibility to improve the efficiency of existing routing algorithms. Second, the associated nodes of each structured links can't be located precisely since they are chosen randomly. In this paper, we focus on addressing the second challenging issue. It is clear that the obstacle of routing results from all of random nodes. We thus regularize those random nodes by coloring all of structured links and make routing just like in a totally structured topology.

Definition 4. *An edge coloring of a graph is an assignment of colors to the edges of the graph so that no two adjacent edges have the same color. The least amount of employed colors is called the* edge chromatic number*, denoted as $x'(G)$ [15].*

Theorem 1. *Let Δ denote the maximum node degree in a simple graph G, where $\Delta \leq x'(G) \leq \Delta+1$. If $x'(G)=\Delta$, the graph is called* class 1 graph*, else,* class 2 graph *[15].*

Theorem 2. *[15] Let K_n be the n-regular graph, then*

$$x'(G) = \begin{cases} n-1 & if\ n\ is\ oven \\ n & if\ n\ is\ odd \end{cases}. \quad (1)$$

Theorems 1 and 2 have been proved in [15]. These theorems bring us an insight to identify boundary nodes in each random cluster.

Observation: Structured topologies, e.g., Ring, Torus, Hypercube, Generalized Hypercube and Cayley Graph are all of structured graphs.

Coupled with Theorem 1 and Theorem 2, the observation manifest that structured topologies that used in today's DCNs are colorable. Fig.2 depicted the color result on $R3(G(2,4,3$-RRGs). Constraint 3 in Section 2 must be meet to ensure a successful edge coloring. Fig.2 depicts the coloring result on $R3(G(2,4,3$-RRGs). To enable the success of edge coloring, the proposed Constraint 3 in Section 2 must be satisfied. After coloring each structured link, we assign each boundary node an inner-identifier according to the color of associated link so that the nodes, which linked by the same link, will have the same inner-identifier. In this novel way, the random cluster level paths will be calculated easily. So, the used inner-identifier of boundary nodes can significantly reduce the routing complexity.

4.1 Edge coloring based identifier allocation

To efficiently enable the routing, an identifier is usually introduced to identify each node in existing DCNs. In our hybrid topologies, the identifier consists of two parts. The inter-identifier contains the construction information of structured topology, while the inner-identifier locates nodes in each random cluster.

The inter-identifier is determined by the behind structured topology. For example, if the structured topology is a Tesseract (4-dimension hypercube) that accommodates 8 nodes, then a three binary digit identifier can identify each node. If the structured topology is a 3∗4∗5 GHC, then a three digit identifier from 000 to 234 will work. Based on the rules how the structured topologies are built, we can always design an identifier system which we can refer to find their neighbors, thus result in convenience in routing. So, the inter-identifier not only identifies the random cluster each node resides in, but also eases the design of routing scheme at the level of structured topology.

The inner-identifier is the most challenging part since boundary nodes are chosen randomly. In the edge coloring theory, whether a graph is class 1 is a typical NP-complete problem, which cannot be solved in polynomial time. We employ DSATUR [16], the best known heuristic algorithm in this area, to approximate the optimal solution. Such an algorithm usually generates multiple coloring strategies, one of which will be randomly selected. Basically, each color identifies a specific inner-identifier. As shown in Fig. 2, the structured links are colored with four colors, i.e. black, purple, orange and blackish green, which represent inner-identifier 000, 011, 010 and 001, respectively. Furthermore, the inner-identifier of each boundary node in a random cluster is assigned with the identifier of the associated structured link. As shown in Fig.2, all boundary nodes are assigned the inner-identifier with 000, 011, 010 and 001, respectively. As for the rest nodes in the random cluster, we calculate the inner-identifier interval according to the binary system. For example, if there are 9 nodes in a random cluster, a 4-digit binary range from 0000 to 1000 will identify all of them. Then the identifiers in the interval, except those used as boundary nodes, are assigned to the non-boundary nodes randomly.

4.2 Identifier-based routing algorithm

According to allocated identifiers of all nodes, especially those boundary nodes, we derive a routing algorithm for our hybrid topologies. Generally, the transmission of data flows between any pair of nodes can be usually divided into a series of inter-cluster and intra-cluster routing. Such two kinds of routing are totally different because of the lack of structured links inside each random cluster. We can distinguish

Algorithm 2 Routing in hybrid topologies

Require: The Hybrid topology, H; The source node *src*, and its identifier *iden-src*; The destination node *dst*, and its identifier *iden-dst*; The path number k; the number of digit in inter-identifier x.

```
1: Coloring the links and allocate identifiers;
2: Let tem be a integer with default value 0;
3: Let iden1 and iden2 be a identifier respectively;
4: if GetInterIden(iden-src,x) == GetInterIden(iden-dst,x) then
5:     path = kStar(iden-src,iden-dst,k);
6: else
7:     Get the inter-identifier of clusters, denoted as inter-iden;
8:     Get structured links needed, denoted as structured;
9:     Get color of links needed;
10:    Get inner-identifier of boundary nodes, denote as iden-color;
11:    while tem < iden-color.size do
12:        iden1 ⟵ iden-src;
13:        iden2 ⟵ inter-iden[tem]+iden-color[tem];
14:        path += Dijkstra(iden1,iden2);
15:        path += inter-iden[tem]);
16:        tem++;
17:    path += Dijkstra(iden2, dst);
18: return The routing path path.
19: function GetInterIden(iden, x)
20:    for i=1 to x do
21:        inter-iden += coor[i];
    return inter-iden.
```

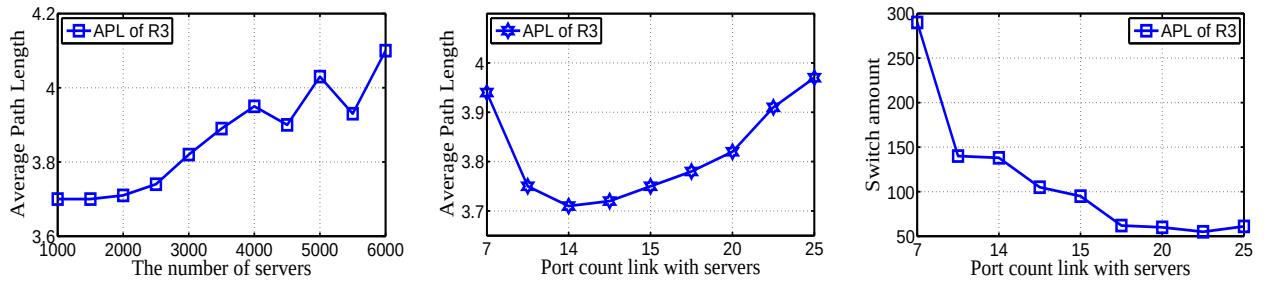
such two kinds of routing just according to the introduced identifier. From the global viewpoint, given a pair of nodes, if their inter-identifiers are the same, they need to involve the intra-cluster routing method, otherwise, the inter-cluster routing method should be employed.

The intra-cluster routing is just the same as routing in a random graph like Jellyfish [5] or Scafida [6]. But in our cases, routing can be simpler since the number of nodes in our random cluster is much less than that of Jellyfish or Scafida. Typically, we employ the shortest k-path algorithm to search the paths between any pair of nodes, furthermore, ECMP protocol can be used to control data transmission and avoid congestion.

The inter-cluster routing aims at finding the shortest path from the source to destination on the random cluster level. To be specific, we need to determine the relay random clusters and all boundary nodes on the way. Inter-cluster routing consists of two steps. First, calculating the relay random clusters from the source cluster, where the source node locates, to the destination cluster, where the destination node locates. Since the inter-identifiers contain the topology information of the structured graph, which the hybrid topology utilizes, the relay random clusters can be easily derived from the inter-identifiers of source node and destination node. Second, determining the boundary nodes of all related random clusters that have been derived from the first step. According to the colors of structured links we have allocated in Section 4.1, the inner-identifiers of each boundary node along the random cluster level path will be gained. This work is straightforward since the inner-identifiers of the two endpoints, which connects with the same structured link, share the same identifier with the colored link. In this special way, the random cluster level path can be derived based on the structured link colors and the identifier system we have established before.

From the global viewpoint, given a pair of source node and destination node, first of all, the routing algorithm judges whether they belong to the same random cluster according to their inter-identifiers. If yes, then the intra-cluster routing algorithm will be employed to find the path. On the contrary, then inter-cluster routing will provide the random cluster level path; hence, all relay random clusters and relay boundary nodes are determined. Then, the path need to be specified at a finer level. That is, each relay random cluster along the random cluster level path will employ intra-cluster routing to find the relay nodes inside them. With the relay nodes and links inside each relay random cluster and the structured links added into the path, the whole routing process is accomplished.

As explained in Algorithm 2, given two nodes, we first judge whether they belong to the same random cluster according to their inter-identifiers. If it is true, K^* Algorithm [17], the most effective heuristic search algorithm so far, is adopted to search k shortest routing paths. If they resides in different random clusters, Algorithm 2 identifies the structured links and their colors, and find those boundary nodes in each relay cluster. We then add links to the path iteratively by invoking the *Dijkstra* algorithm in each cluster. In Fig.2, a source node with the red color in the cluster 00 needs to communicate with a destination node with the yellow color in the cluster 13. Their identifier are given as 00101 and 13100, respectively. Obviously, they belong to different random clusters, and there exist two routing paths between clusters 00 and 13, i.e. 00⟶03⟶13 and 00⟶10⟶13. We use the first path as an example, 03 is a relay cluster on the random cluster level. The purple and orange links associated with inner-identifiers 011 and 010 are two inter-cluster links. Furthermore, the boundary nodes can be located, i.e. 00001 in cluster 00, 03011 and 03010 in cluster 03, and 13010 in 13 cluster. Then, the *Dijkstra* algorithm is utilized to derive a path of added links inside each random cluster, e.g., the shortest path from 00101 to 00011 in cluster 00, from 03011 to 3010 in cluster 03, and from 13010 to 13100 in cluster 13.



(a) The APL of R3 when T varies. (b) The APL of R3 when β varies. (c) The number of switches in R3 when β varies.

Fig. 3 The APL of R3 with different parameter setting.

In this way, any pair of nodes can finally achieve the routing path inside our hybrid topologies.

When nodes are added or eliminated from R3, the routing tables need to be updated. Note that, unlike other topologies where addition or deletion of a node may affect the global routing, R3 suffers the least since it limits the influence of topology alteration into the specific random cluster. Here is an example, for Fat-Tree, if one of the core switches breakdowns, then the routing tables of all nodes that belong to the subtree rooted from the failed switch will be updated to suit the new topology. On the contrary, when 00001 fails in Fig.2, only those nodes of 00 cluster may need to update routing table, thus will never impact nodes in other random clusters.

5 Topology optimization

Given the number of servers in a data center, a typical question is how to allocate the ports of each switch when establishing the data center networking structure. That is, how many ports each switch should be allocated to connect with servers? Note that the remained ports of each switch are utilized to form a networking structure among all switches. There lies a trade-off between the amount of switches and the network diameter, since the increasing number of switches leads to decreased network diameter at the cost of incurring more investment [18]. Meanwhile, the ratio of node degree to the network diameter is a classical problem for topology design. How much randomness is optimal for both routing and networking? In the design of our hybrid topologies, more structured links can ease the routing. On the contrary, networking can benefit from more randomness, since random topologies can naturally support the incremental expansion with low diameter. Therefore, the proposed hybrid topologies, i.e., R3, need further optimization.

5.1 Related factors

Given the number of servers and that of ports at each switch, how many ports each switch should be allocated to connect with servers and other switches, respectively, so as to realize the minimum APL with an acceptable amount of switches? In R3, to minimize average path length, we must concern at least three impact factors. As a hybrid topology combine r-RRGs and GHC via the compound graph theory, the diameter of R3 is decided by three factors. That is the dimension of GHC, $m_1, m_2, \cdots, m_s$, the node degree of RRGs, r, and the number of nodes in each RRG, $n_1, n_2, \cdots, n_t$, where $N{=}m_s{*}m_{s-1}{*}\cdots{*}m_1$. The dimension influences the redundancy of routing path and the amount of structured links in routing paths. Moreover, it dominates the hamming distance between any pair of nodes, x and y, in terms of number of hops between them [8]. The other two factors determine the number of relay nodes inside each cluster along the path. APL is a network level measurement, so we calculate it and integrate such factors to reveal their influence on the network diameter.

5.2 Optimization strategy of topologies

In our hybrid topology, each routing path consists of two parts, the structured links and some random link in random clusters on the path.

Theorem 3. *In R3, let APL_{ghc}, APL_{rrg}, and APL_{r3} denote the average length of routing in GHC, RRGs and R3, respectively. Then we have $APL_{r3}{=}APL_{ghc}{+}(APL_{ghc}{+}1){*}APL_{rrg}$.*

Proof. In such a routing path, there exist APL_{ghc} structured links, and $APL_{ghc}{+}1$ random clusters among which $APL_{ghc}{-}1$ random clusters are relay. In each of such clusters, the path length is APL_{rrg} on average. Thus, Theorem 3 is proved. □

Theorem 4. *For a $G(m_s, m_{s-1}, \cdots, m_1)$, $N{=}m_s{*}m_{s-1}{*}\cdots{*}m_1$ denotes the number of nodes, x_l denotes the number of node pairs, each of which exhibits the distance of l. We then have*

$$x_l{=}(N/2)\sum_{i_1=1}^{s-l+1}\cdots\sum_{i_j=j}^{s-l+j}\cdots\sum_{i_l=l}^{s}[(m_{i_1}{-}1)\cdots(m_{i_l}{-}1)]$$

$$APL_{ghc}{=}(2\sum_{l=1}^{s} l{*}x_l)/(N(N{-}1)$$

Proof. Consider a node A, denoted as $y_s y_{s-1}\cdots y_1$, in this GHC, If another node B is l hops away from node A, the coordinates just differ in l dimensions Thus, APL_{ghc} can be calculated naturally based on x_l. Thus Theorem 4 is proved. □

Note that different structures exhibit varied APL_{rrg} when the node degree and number of nodes do not change [9]. The default value of APL_{rrg} is set to $\log_{r-1} n$, where n and r denote the amount of nodes and the degree of each node, respectively. Theorems 3 and 4 derive a model of the network diameter for R3. We consider a special case that $n_1{=}n_2{=}\cdots{=}n_t{=}n$. We build this optimal model with the total server number T, the port count of each switch p as input. α and β denote the number of ports each switch allocates to connect with other switches and servers, respectively. Let $t \in [1, T]$ be the total number of used switches. The topology optimization can be modeled as follows:

$$\min APL_{r3} \tag{2}$$

s.t.

$$\begin{cases} T \le s * \beta \\ t \le n * \prod m_i \\ \sum_{i=1}^{s}(m_i - 1){+}1 \le n \\ 2 \le \alpha{+}\beta \le p \\ 1 \le r \le \alpha{-}1 \end{cases} \tag{3}$$

In this model, the minimum APL is searched in the domain of feasible solution with five constraints satisfied. The first inequality promises that the number of servers that our topology can accommodate is not less than the input T. The second inequality constraints the number of switches the topology can accommodate is more than the number of switches that actually used. The third inequality guarantees that our routing algorithm can be built successfully. While, the last constraint reveals that each switch should leave at least one port for structured links. In fact, this model is a nonlinear integer programming problem, which is NP-hard. According to the gradient optimal algorithm, we utilize an associated tool (ModelCenter [19]) to search the minimum result. Fig. 3 (a) shows that the optimized APL of R3 increases when T varies from 1000 to 6000. This result is reasonable because more servers need to be accommodated. It, however, fluctuates at 4500 and 5500 because there are more switches and less ports link with servers in these two cases. For this reason, the topology exhibits lower APL at the cost of increasing the investment due to more switches. The gradient optimal algorithm searches towards the fastest decline direction and find the minimal value. Fig.3(b) indicates that, given T=2000 and p= 48, the search process terminates when β=15, and the minimum APL is 3.71. Furthermore, we evaluate the amount of switches after APL is optimized. Fig.3(b) and Fig.3(c) demonstrate the marginal effect around the extreme point. When s=62, APL=3.78 is a little bit higher than 3.71. However, 138 switches are required so as to reach the extreme point. This will double the investment and obviously not cost-effective. So, whether the minimum APL is the best choice depends on designers.

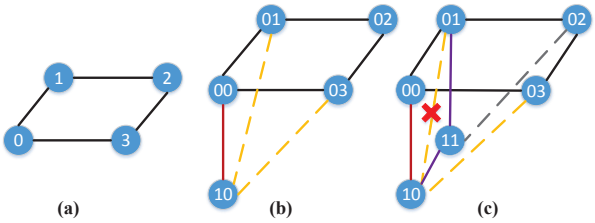


Fig. 4 A example of expansion by adding extra clusters.

6 Incremental expansion of topology

Incremental expansion is essential to data centers since they are usually required to accommodate arbitrary number of servers on demand. For data centers based on our hybrid topologies, two methods can be employed to realize the incremental expansion, i.e. expansion by adding nodes in an existing random cluster and expansion by inserting a new random cluster.

6.1 Expansion within an existing random cluster

Recall that random topologies like RRG and scale-free network support the incremental expansion naturally, new nodes can be added one by one. For RRG, when a new node is added in, several existing nodes break up their links and connect to the new one [5]. For a scale-free network, according to its generation algorithm, a new node will be linked with m existing nodes, which are selected based on the preferential attachment principle [6]. To keep from the unbalance and bottleneck, those random clusters with the minimum number of nodes are chosen to host new nodes. With more and more nodes are added into those selected clusters, the length

of inner-identifier needs to be increased to maintain consistency in the whole network. In other words, the length of in inner-identifier is decided by the maximum number of node in random clusters. In Fig.2, the number of nodes in each cluster is no more than 8, a 3-digit inner-identifier is feasible. Once a new node is added into cluster 00, the maximum number of node is 9. Thus, the existing 3-digit inner-identifier is replaced by a 4-digit inner-identifier. That is, existing inner-identifiers are updated by adding a new digit in the front of them. Theoretically, the number of node that each random cluster can accommodate is unlimited. However, if the random clusters have too many nodes, the structured links may be bandwidth bottleneck since these random clusters are interconnected with structured links only.

6.2 Expansion by extra random cluster

When all random clusters in the existing topology are saturated, a new cluster is required to accommodate upcoming new servers. Existing structured topologies are usually extended level by level, we propose a novel expansion strategy to implement the incremental expansion. More precisely, we expand an existing hybrid topology just like the way hypercube does, but in an incremental way.

Definition 5. *In a n-dimensional hypercube, a $n-1$-dimensional coordinate system can be established to identify each node according to its building rule. We call a pair of nodes are corresponding nodes if their coordinates are different only at the first digit.*

As shown in Fig.4(b), a new cluster 10 and an existing cluster 00 are called a pair of corresponding clusters. Algorithm 3 depicts a framework of cluster level expansion. When we need to add a new cluster, a function *COOREXTEND* judges whether the coordinate system requires more digits, then a function ADDREGULARLINK(G,New[i]) adds structured links to the new cluster, according to $G's$ construction rules. Then those unnecessary links are deleted by a DELETE function. In ADDRESTLINK(G, $New[i]$), those links, whose destination endpoints do not exist yet are linked to *New[i]'s* corresponding clusters to ensure the partial regularity of G. In this especial way, structured topologies can be incrementally expanded while remaining their fundamental characteristics via using corresponding clusters. We depict an example in Fig.4(a), when a new cluster wants to add into a quadrangle, the coordinate of existing clusters are increased one digit. The new cluster is identified as 10, whose corresponding cluster is 00. Thus, cluster 10 is linked with cluster 00. According to the quadrangle rule, cluster 10 should connect with clusters 11 and 13. Such two clusters, however, do not exist yet; hence, cluster 10 connects with the corresponding clusters of 11 and 13, i.e., 01 and 03 instead. When other three clusters arrive, we just repeat these operations and finally extend the quadrangle to a cube.

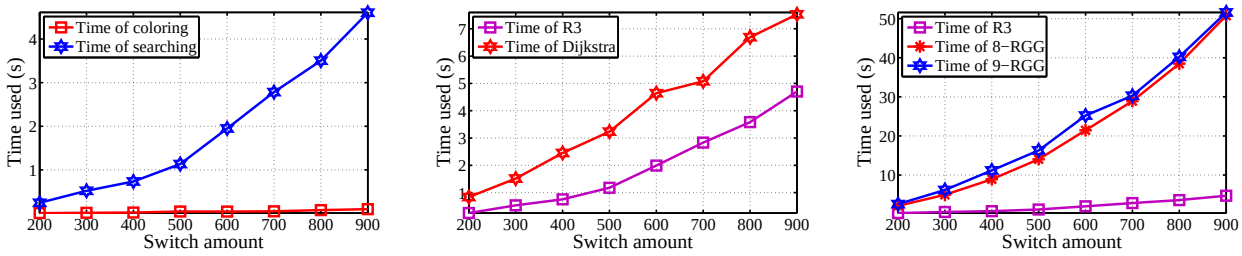
Algorithm 3 Expansion with Extra Random Clusters, G

Require: The regular topology, G; The amount of new random clusters, n.

```
1:  Let New[n] be those n new clusters that will be added;
2:  for i=0 to n do
3:      if COOREXTEND(G) then
4:          Each inter-identifier is added a digit in the front;
5:      Assign prefix to New[i];
6:      ADDREGULARLINK(G, New[i]);
7:      DELETE(G, New[i]);
8:      ADDRESTLINK(G, New[i]);
9:  return The new structured topology G.
10: function COOREXTEND(G)
11:     if r + 1>the amount the existing prefix can identify then
12:         Return True;
13:     else
14:         Return False;
15: function DELETE(G, New)
16:     Let neighbor be the neighbors of New;
17:     Let corespd be the corresponding node of New;
18:     for j=0 to New.neighboramount do
19:         if corespd linked with neighbor[j] then
20:             Delete this link;
        return G;
21: function ADDREGULARLINK(G, New)
22:     Let corespd be the corresponding node of New;
23:     Add link from New to corespd;
24:     Calculate New's neighbors denoted as reg;
25:     for s=0 to reg.size do
26:         Add link from New to reg[s];
        return G;
27: function ADDRESTLINK(G, New)
28:     Calculate New's rest neighbors denoted as rest;
29:     Let corespd[e] be the corresponding node of rest[e];
30:     for a=0 to rest.size do
31:         Add link from New to corespd[a];
        return G;
```

7 Performance evaluation

In this section, we simulate our hybrid topologies to evaluate the routing flexibility, cabling cost and network performance. Typically, we compare R3 with a fully structured GHC and a fully random Jellyfish topology, respectively. Note that we report all the average result over 100 rounds of simulations for each performance metric.



(a) Time consumption of routing and coloring. (b) Our routing method and Dijkstra algorithm. (c) Time consumption of R3 and Jellyfish.

Fig. 5 Time consumption of routing process.

7.1 Routing flexibility

Building routing table in a large-scale data center is a tough work for those fully or partially random DCNs, due to huge number of links and potential paths between any pair of switches. To measure the time consumption due to find the shortest path between each switch pair, we conduct a series of experiments with different settings of switches.

We constructs R3 with different amount of switches, ranging from 200 to 900. In each of these R3s, 20 switches form a 8-RRG random cluster, and such random clusters are connected via structured links. Since our routing method is coloring-based, the time consumption of routing consists of two parts, i.e., the coloring time and the routing time. During the coloring period, each structured link is colored with one color to record the nodes used to link each pair of random clusters. Additionally, the process of finding the shortest paths between every switch pairs also brings additional time consumption. As demonstrated in Fig.5(a), compared with the coloring time, the routing time contributes most of the total time. Fig.5(b) reports that our routing method outperforms the traditional Dijkstra algorithm in R3. More precisely, the coloring-based routing algorithm reduces half of time consumption compared to the Dijkstra algorithm on average. The reason is that, compared with traditional Dijkstra algorithm which is time-consuming, the coloring-based routing algorithm limits the Dijkstra algorithm inside each random cluster only. Furthermore, as reported in Fig.5(c), R3 consumes much less time compared to the Jellyfish topology. Note that the building clusters of our hybrid R3 topology are 8-RRGs. After introducing structured links into the topology, the total number of links in R3 is between that of 8-RRG Jellyfish and that of 9-RRG Jellyfish. The evaluation result, however, demonstrates that the routing time of R3 is less than that of 8-RRG and 9-RRG Jellyfish in many times.

Additionally, the routing flexibility is important in DCNs where failures of commodity devices are very common. In R3, once a switch breaks down, the coloring-based routing algorithm will immediately derive another available path and update involved routing table, dynamically.

7.2 Cabling cost

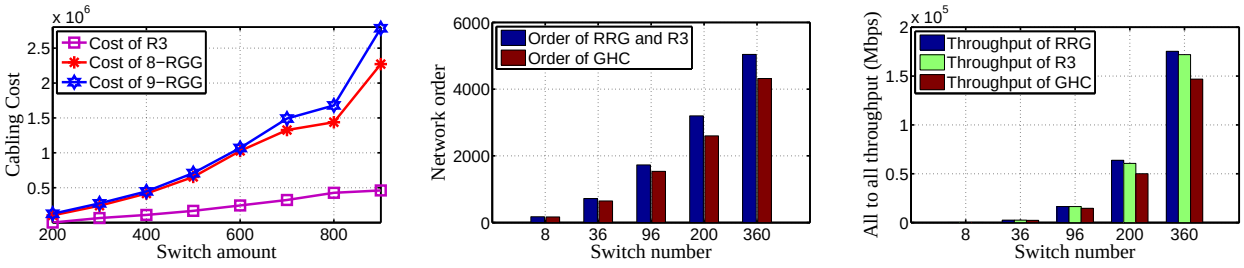
In DCNs, huge number of links are utilized to interconnect large-scale switches and servers to form a designed topology. For each of our hybrid topologies, those long-distance random links will significantly increase the cabling cost and complexity. In this section, we calculate the total length of all cables in R3 and Jellyfish to evaluate the cabling cost.

Racks in a data center are placed as a matrix for cooling and maintaining purpose. To minimize the total length of all cables, we suppose that racks are placed as a quadrate or quadrate-like structure. Given the amount of switches, we calculate the length and width according to the quadrate-like location strategy first. Then, we calculate the total length of all links via the Pythagorean Theorem. Meanwhile, we assume that links between racks are underground distribution such that the geographical distance can be calculated as the link distance.

Reasonably, we use the distance between any pair of racks as the metric of cabling cost. Fig.6(a) depicts the cabling cost of our hybrid topology compared to that of Jellyfish. For fairness, we compare R3 with 8-RRG Jellyfish and 9-RRG based Jellyfish, respectively. R3 causes much less cabling cost than 8-RRG as well as 9-RRG based Jellyfish. This result is reasonable since with the increasing number of switches, more remote links are introduced into Jellyfish. The distance between any pair of switches in R3, however, is predictable and less than that in Jellyfish.

7.3 Network performance

We compare R3 with two extreme topologies, the generalized hypercube and Jellyfish, which are representative ones of fully structured and random topologies, respectively. To



(a) Cabling cost comparison with Jellyfish. (b) Network order. (c) All to all throughput.

Fig. 6 Performance comparison between R3, RRG(Jellyfish) and GHC.

evaluate their performance, we vary the amount of 24-ports switches from n=8 to 360. We calculate the maximal amount of servers they can accommodate, i.e., network order, and evaluate the throughput under all-to-all traffic. At the best case, each switch inside a random cluster has a structured edge to connect with another switch in other clusters. So, R3 has the same network order with RRG. But for GHCs, the network order depends on its dimensions. Consider n=200 as an example and there is 200=2*2*2*5*5. Each switch should reserve at least 11 ports for linking with other switches, the remainder 13 ports link with servers. Consequently, the network order is at most 200∗13=2600. To construct the densest R3, we use 5*5 GHC and 7-RRG, which lead to a hybrid topology, which can accommodate 16∗200=3200 servers and the maximal node degree is 8. For fairness, a 8-degree Jellyfish is built as a reference. Fig.6(b) depicts the resulting network order. It is clear that both R3 and Jellyfish can accommodate large number of servers. This is reasonable because both R3 and Jellyfish have larger design space than GHC.

Indeed, the throughput of DCNs is affected by not only the topology, but also the bandwidth allocation strategy [20] [21].For a given DCN, different bandwidth allocation strategies result in different network performance. In this paper, to comprehensively reveal the impact of topologies, our experiments focus on the topologies under the same bandwidth allocation strategy. We first verify the network performance of R3 under different amount of switches, and then compare the network throughput with Jellyfish and GHC under all-to-all traffics. Typically, the bandwidth of each link is set to be 1000Mbps, and the data rate of each server is 100kbps. We monitor each flow via the flow monitor function in NS3, and obtain the total throughput by summing up the data rate of each flow. Fig.6(c) plots that the network throughput of R3 is always a little bit less than that of Jellyfish but much more than that of GHC. Thus R3 integrates the advantages of both GHC and Jellyfish while abandons their weakness.

8 Discussion

In this paper, we investigate a family of hybrid topologies, which combine structured topologies with random topologies seamlessly via the compound graph. To fully understand the hybrid designing methodology, we discuss the following issues further.

Rethinking the routing algorithm. An edge-coloring based routing algorithm in Section 4.2 is designed for fast and accurate routing tables. In effect, such a routing algorithm is propagable since Theorem 1 guarantees that at most Δ+1 kind of colors are sufficient to color all edges. Namely, the novel routing algorithm works well even though R3 selects different structured topology. Let m denotes the average number of nodes in random clusters, $|E|$ and $|V|$ denote the number of edges and vertex in the chosen structured topology, respectively. Then the time complexity of our routing algorithm is $O(|E|*|V|))+O(|V|*m^2)$, where $O(|E|*|V|))$ is the time complexity of coloring [22] and $O(|V|*m^2)$ is the time complexity of routing inner random clusters.

Dedicated integer programming model. Note that, the integer programming model in Section 5.2 is used to find the best parameter setting of a given structured topology, the number of servers and amount of ports per switch. So, different structured topologies result in different integer programming models to describe the hybrid topologies precisely.

Incremental expansion of our hybrid topologies. As for the expansion issues for hybrid topologies, we propose two appropriate methods, i.e., the expansion within an existing random cluster and the expansion by adding an extra random cluster in Section 6. The new added servers ought to be allocated into the existing random clusters evenly other than embed them into one or several clusters in bathes. Besides, which expansion strategy should be employed remains an open problem for the designers so that the flexibility and

design space will be guaranteed.

The experiment methodology. Our experiments concentrate on evaluating the performance of R3, RRG and GHC since R3 is constructed via combining RRG and GHC seamlessly. Different selections of structured and random topologies will definitely result in different performance. The performance of each of other hybrid topologies is similar with R3, i.e., its performance falls between that of used structured topology and that of the utilized random topology. Thus, the major motivation of our experiment is to prove that the proposed hybrid designing methodology combines the benefits of both structured and random topologies together while avoid their weakness successfully. Additionally, we leave the comparison with other topologies as one of our future work.

Constraints of hybrid DCNs. Besides the aforementioned benefits, the proposed hybrid DCNs also face a few potential limitations. First of all, three constraints must be satisfied to achieve a hybrid topology as elaborated in Section 3.2, i.e., the structured and random topologies need careful selection. Second, if there exist too many nodes in a random cluster, the boundary nodes of that cluster may be overloaded and thus may result in congestion. In other words, the designer must concern the capacity of each boundary node when building or upgrading the hybrid data center network. At last, even though there are parallel paths between any pair of nodes, large number of long-distance flows may lead to bottleneck at those structured links. Thus, more bandwidth should be allocated to these structured links.

9 Related work

Existing DCNs networking topologies can be roughly classified into five categories, i.e. switch-centric data centers, server-centric data centers, modular data centers, random data centers and wireless data centers.

In switch-centric data centers, routing and interconnection are realized by switches, which form dedicated structured topology, such as generalized hypercube, Torus, compound graph, tree and so on. Fat-Tree [1], F10 [23], VL2 [2] belong to this category. With the development of optical communication, optical packaging technology is introduced into switch-centric DCNs [24]. These optical links improve bandwidth greatly, but the associated control strategy becomes complex.

Note that switches and routers are expensive, while commodity server and mini-switch are cheap; hence, it is cost-saving to build a DCN just with servers and mini-switches. In server-centric data centers, routing and interconnection are realized by servers since servers are competent to cache and forward flows. Usually, server-centric DCNs are recursively defined and extended level by level. BCube [25], DCell [11] are all server-centric DCNs, but their network capacity are limited by the count of NIC ports at each server.

To ease the development of data centers, module has replaced racks as the basic building block of large-scale data centers. These modules integrate the power system, cool system and thousand servers inside a container. By further interconnecting a given number of such modules via a dedicated topology, an efficient, controllable, and elastic data center can be built. MDCube [26] and uFix [27] are two representative proposals. They utilize the remaining NICs at servers to interconnect those modules systematically.

For random DCNs, random links interconnect remote nodes together, hence, they shorten the network diameter. Typically, Jellyfish [5] and Scafida [6] are proposed based on the random regular graph and scale-free network, respectively. The advantage of random DCN is the characteristic of incremental expansion, which means that we can add servers one by one other than level by level. Routing in such random topologies, however, is difficult and time-consuming.

Recently, wireless communication technologies are introduced into DCNs. Therefore, the cabling cost will be considerably eliminated and the network bandwidth will be increased. In literature [28], a remote wireless channel between any pair of racks can be established by reflecting wireless signals via a mirror from source to destination. FireFly [29] goes further, it forecasts the traffic demand and adjust the topology dynamically in a short time period. Wireless DCNs supports unicast transmission well, but fails to accomplish other transmission models such as broadcast, multicast and shuffle.

10 Conclusion

In this paper, we investigate a family of compound graph based hybrid topologies for data centers, which combine the advantages of both structured topologies and random topologies. We propose a coloring-based routing algorithm to enable the shortest path and shorten the routing time-consumption, effectively. Meanwhile, we propose an integer programming model to derive the optimal R3 topology, with given number of servers and amount of ports per switch. To enable the incremental expansion, besides adding servers one by one inside a random cluster, we propose to extend the network scale by adding new random clusters one by one. The evaluation results indicate that our hybrid topologies consume

much less routing time and incur less cabling-cost than Jellyfish, and achieve better throughput than GHCs. In summary, our proposal achieves the easy-routing, incremental expansible and high throughput, simultaneously. Thus, our hybrid topologies are competent to large-scale data centers.

Acknowledgements The authors would like to thank anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation for Outstanding Youth under grant No.61422214, the National 973 Basic Research Program under grant No.2014CB347800, the Program for New Century Excellent Talents in University, and the Distinguished Young Scholars of National University of Defense Technology under grant No.JQ14-05-02.

References

1. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.
2. A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.
3. D. Guo, T. Chen, D. Li, M. Li, Y. Liu, and G. Chen, "Expandable and cost-effective network structures for data centers using dual-port servers," *Computers, IEEE Transactions on*, vol. 62, no. 7, pp. 1303–1317, 2013.
4. J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proc. SOCC*, ACM, October 2011.
5. A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly." in *Proc. USENIX NSDI*, San Jose, USA, April 2012.
6. L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 5, pp. 4–12, 2010.
7. A. Singla, P. B. Godfrey, and A. Kolla, "High throughput data center topology design," in *Proc. USENIX NSDI*, Seattle, USA, April 2014.
8. L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Transactions on Computers*, vol. 100, no. 4, pp. 323–333, 1984.
9. B. Bollobás, *Random graphs*. Springer, 1998.
10. F. Reitz, M. Pohl, and S. Diehl, "Focused animation of dynamic compound graphs," in *Proc. IEEE IV*, Barcelona,Spain, July 2009.
11. C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.
12. D. Guo, H. Chen, Y. He, H. Jin, C. Chen, H. Chen, Z. Shu, and G. Huang, "Kcube: A novel architecture for interconnection networks," *Information Processing Letters*, vol. 110, no. 18, pp. 821–825, 2010.
13. D. Guo, C. Li, J. Wu, and X. Zhou, "Dcube: A family of high performance modular data centers using dual-port servers," *Elsevier Journal of computer communication*, vol. 53, pp. 13–25, 2014.
14. J. Xie, D. Guo, J. Xu, L. Luo, and X. Teng, "Efficient multicast routing on bcube-based data centers," *KSII transactions on internet and information systems*, vol. 8, no. 12, pp. 4343–4355, 2014.
15. J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. Macmillan London, 1976, vol. 6.
16. D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.
17. H. Aljazzar and S. Leue, "k^*: A heuristic search algorithm for finding the k shortest paths," *Artificial Intelligence*, vol. 175, no. 18, pp. 2129–2154, 2011.
18. E. Giannini, F. Botta, P. Borro, D. Risso, P. Romagnoli, A. Fasoli, M. Mele, E. Testa, C. Mansi, V. Savarino *et al.*, "Platelet count/spleen diameter ratio: proposal and validation of a non-invasive parameter to predict the presence of oesophageal varices in patients with liver cirrhosis," *Gut*, vol. 52, no. 8, pp. 1200–1205, 2003.
19. "PHX ModelCenter," http://www.phoenix-int.com/software/phx-modelcenter.php/, 2014.
20. J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proc. IEEE INFOCOM*, Turin, Italy, April 2013.
21. J. Guo, F. Liu, X. Huang, J. Lui, and Hu, "On efficient bandwidth allocation for traffic variability in datacenters," in *Proc. IEEE INFOCOM*, Toronto, Canada, April 2014.
22. J. Misra and D. Gries, "A constructive proof of vizing's theorem," *Information Processing Letters*, vol. 41, no. 3, pp. 131–133, 1992.
23. V. Liu, D. Halperin, A. Krishnamurthy, and T. E. Anderson, "F10: A fault-tolerant engineered network," in *Proc. USENIX NSDI*, Lombard, USA, April 2013.
24. K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 498–511, 2012.
25. C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.
26. H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: a high performance network structure for modular data center interconnection," in *Proc. CoNEXT*, New York, USA, December 2009.
27. D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," in *Proc. ICNP*, Vancouver, Canada, October 2011.
28. X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: flexible wireless links for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 443–454, 2012.
29. N. H. Azimi, Z. A. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: a reconfigurable wireless data center fabric using free-space optics," in *Proc. SIGCOMM*, Chicago, USA, August 2014.

Lailong Luo received the B.S. degree in school of information system and management from National University of Defence Technology, Changsha, China, in 2013. He is currently working toward the M.S. degree in College of Information System and Management, National University of Defense Technology, Changsha, China. His research interests include data centers and software defined networks.

Deke Guo received the B.S. degree in industry engineering from Beijing University of Aeronautic and Astronautic, Beijing, China, in 2001, and the Ph.D. 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, software-defined networking, data center networking. He is a member of the ACM and the IEEE.

Wenxin Li received the B.E. degree from the School of Computer Science and Technology, Dalian University of Technology, Dalian, China, in 2012. Currently, he is a Ph.D. candidate in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His research interests include data center networks and cloud computing. His research interests include data center networks and cloud computing.

Tian Zhang, born in 1994, Changsha, China, a sophomore student from School of Information Management of Wuhan University, Wuhan, China. His major is Information Management and Information System, and his research interest is mainly on Wireless Sensor Networks and Data Center Networking. Besides, he also works on Indoor Localization with mentor and partners in National University of Defense Technology, Changsha, China.

Junjie Xie received the B.S. degree in computer science and technology from Beijing Institute of Technology, Beijing, China, in 2013. He is currently working toward the M.S. degree in College of Information System and Management, National University of Defense Technology, Changsha, China. His research interests include distributed systems, data centers, software defined networks and interconnection networks.

Xiaolei Zhou received the B.A. degree from the Information Management Department, Nanjing University, P.R. China, in 2009, and the MS degree in military science from the National University of Defense Technology, P.R. China, in 2011. He is currently working toward the PhD degree in the School of Information System and Management, National University of Defense Technology, P.R. China. His current research interests include indoor localization, wireless sensor networks and data center networking.