Efficient Multicast Routing on BCube-Based Data Centers

Junjie Xie[1], Deke Guo[1], Jia Xu[2], Lailong Luo[1] and Xiaoqiang Teng[1]
[1]Science and Technology on Information Systems Engineering laboratory,
National University of Defense Technology
Changsha, Hunan 410073 - China
[e-mail: guodeke@gmail.com]
[2]School of Computer, Electronics and Information, Guangxi University
Nanning, Guangxi 530003 – China
[e-mail: xujia.neu@gmail.com]
*Corresponding author: Deke Guo

Received July 24, 2014; revised October 9, 2014; accepted October 22, 2014; published December 31, 2014

Abstract

Multicast group communication has many advantages in data centers and thus is widely used by many applications. It can efficiently reduce the network traffic and improve the application throughput. For the multicast application in data centers, an essential problem is how to find a minimal multicast tree, which has been proved to be NP-hard. In this paper, we propose an approximation tree-building method for the minimal multicast problem, named HD（Hamming Distance）-based multicast tree. Consider that many new network structures have been proposed for data centers. We choose three representative ones, including BCube, FBFLY, and HyperX, whose topological structures can be regarded as the generalized hypercube. Given a multicast group in BCube, the HD-based method can jointly schedule the path from each of receiver to the only sender among multiple disjoint paths; hence, it can quickly construct an efficient multicast tree with the low cost. The experimental results demonstrate that our method consumes less time to construct an efficient multicast tree, while considerably reduces the cost of the multicast tree compared to the representative methods. Our approach for BCube can also be adapted to other generalized hypercube network structures for data centers after minimal modifications.

Keywords: Data Center, Multicast, Generalized Hypercube, Approximation Algrithm

This work is partially supported by the National Basic Research Program (973 program) under Grant No. 2014CB347800, the NSFC under Grant Nos. 61422214, 61402494, the Program for New Century Excellent Talents in University, and the Preliminary Research Funding of NUDT under Grant No. JC10-05-02.

http://dx.doi.org/10.3837/tiis.2014.12.006

1. Introduction

Data centers provide core infrastructure supporting many types of cloud applications. Some of these applications, such as Web mail, Web search and interactive games, are run based on the traffic pattern of group communication at the application layer, which is referred to as the application-layer multicast. There are also plenty of applications, including structured storage system, distributed file system and distributed execution engine, are implemented based on the group communication at the network layer, which is thus called the network-layer multicast. This shows that Multicast is popular at both two layers. Considering the application-layer multicast can be translated to the network-level multicast in the data center network [1][1], this paper focuses on improving the execution efficiency of the multicast conducted at the network layer.

Although the deployment of Multicast benefits group communications by significantly reducing network traffic and improving application throughput, designing and implementing network-layer multicast still faces many obstacles. On the other hand, the development of data centers provides an opportunity for deploying multicast in a managed network environment. Nowadays, many efforts have been done on designing novel network structures for data centers, such as BCube [2], Fat-Tree [3], VL2 [4], DCell [5], CamCube [6], BCN [7], FiConn [8], FBFLY [9] and HyperX [10]. Compared with traditional network sturcutres, the newly propounded network structures have higher link density to ensure that there are more equivalent node-disjoint paths between every pair of servers [11]. In such cases, as pointed out in [12], existing Internet-level multicast technologies cannot well accommodate to the future data center networks. In this paper, we aim to design efficient multicast routing schemes in data center networks which are based on the concept of generalized hypercube, such as BCube, FBFLY, and HyperX.

The traditional receiver-driven scheme is an intrinsic way to establish a multicast tree in a data center. In particular, given a multicast group, each receiver maintains a shortest path to the sender such that a multicast tree can be achieved by simply combining such shortest paths. As we will show, such a straightforward method causes a certain degree of waste of network and link resources. The minimal Steiner-tree (MST) problem [13] is another possible way to construct an efficient multicast tree for every multicast group in a data center. However, the existing approximation algorithms for constructing an MST, cannot fully utilize the topology information of data centers with densely connected networks to refine the muticast protocols. Moreover, the time complexity of MST construction algorithms is $O(m \times N^2)$, where m and N denote the size of the multicast group and the size of the data center, respectively. This is too high for online generating an multicast tree for each multicast group inside large-scale data centers. The most relevant work to us is [11], which revolves around ESM, a top-down construction method for efficient multicast tree in BCube data centers. Our research reveals that the ESM is a practical but not very outstanding method to derive an efficient multicast tree in terms of the number of links in the tree and the time complexity.

Bearing this in mind, we propose an efficient and effective method, named *HD*(Hamming Distance)-*based method*, for constructing multicast trees for a large amount of multicast groups in a large-scale data center. Extensive simulation results show that our proposal apparently outperforms both of the ESM method [12] and the Steiner-tree algorithm [13] in terms of the number of links in each multicast tree and the time complexity of constructing multicast trees.

The rest of the paper is organized as follows. In Section 2, we summarize the multicast problem and analyze the existing methods. In Section 3, we propose a novel and efficient method, named HD-based method, to construct the multicast tree. In Section 4, we evaluate our proposed solution with related works using large-scale simulations. Finally, the paper is concluded in Section 5.

2. Background and Related Work

2.1 Network Structure of Data Centers

As pointed in [7], many new network structures have been proposed for data centers which are roughly divided into two categories, namely the switch-centric network structures and the server-centric network structures. The first category organizes switches into a given structure. The second category is to put the interconnection intelligence on servers. The *k*-ary *n*-flat flattened butterfly (FBFLY) [9] is a scalable yet low-diameter switch-centric network structure. The key insight behind FBFLY is to interconnect all homogeneous high port switches with the generalized multi-dimensional hypercube [14].

On the other hand, BCube [2] is a representative server-centric network structure that is constructed in an iterative way. Specifically, BCube(*n*, 0) is constructed by connecting *n* servers using an *n*-port switch. More generally, BCube(*n*,*k*), $(k \geq 1)$, is built based on *n* BCube(*n*,*k*-1)*s* and n^k *n*-port switches. **Fig. 1** shows an illustrative example of BCube (4,1), which is composed of 16 servers and 2 layer switches. Essentially, the underlying network structure of BCube is also a generalized hypercube. Actually, there exist other data centers whose network structures can also be viewed as a generalized hypercube, such as HyperX [10].

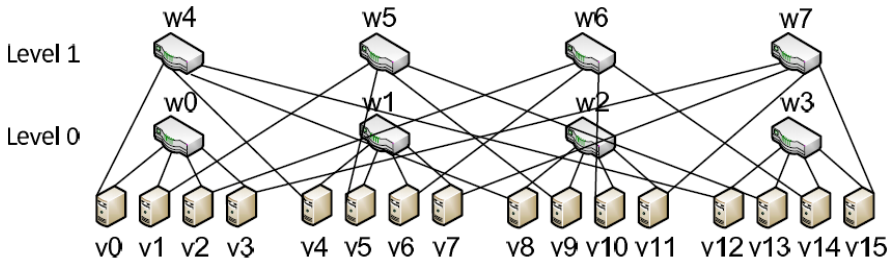


Fig. 1. An illustrative example of BCube(4,1).

In this paper we focus on solving the multicast problem in BCube [2]. That is because BCube supports various bandwidth-intensive applications by speeding up one-to-one, one-to-several and one-to-all traffic patterns, and by providing high network capacity for all-to-all traffic. When failure rate increases due to the malfunction of those servers, switches or links, BCube displays a graceful performance decrease. Besides, the TCP/IP protocol stack can be seamlessly integrated with BCube, and the BCube packet forwarding can be efficiently implemented in both hardware and software. Moreover, BCube is fault tolerant and easily to achieve load balancing.

The topology of BCube (*n*, *k*) can be abstracted as a *k*+1 dimensional *n*-ary generalized hypercube with the same set of servers. For example, two servers with labels $\{x_k \ldots x_i \ldots x_1 x_0\}$

and $\{y_k...y_i...y_1y_0\}$ are deemed as the mutual one-hop neighbors in the *i*-th dimension, if their labels are different only in the *i* dimension, where x_i , $y_i \in \{0,1,...,n-1\}$ with $0 \le i \le k$. Therefore, a server has *n*-1 one-hop neighbors in each dimension. The only difference is that such two servers are directly connected with each other in a generalized hypercube while being connected to the common *i* level switch in BCube(*n*, *k*). In such case, each server is indirectly connected with its neighbors in any given dimension via a common switch. Additionally, we define two servers being the mutual *j*-hop neighbors if their labels differ in *j* dimensions for $0 \le j \le k+1$.

2.2 Problem statement

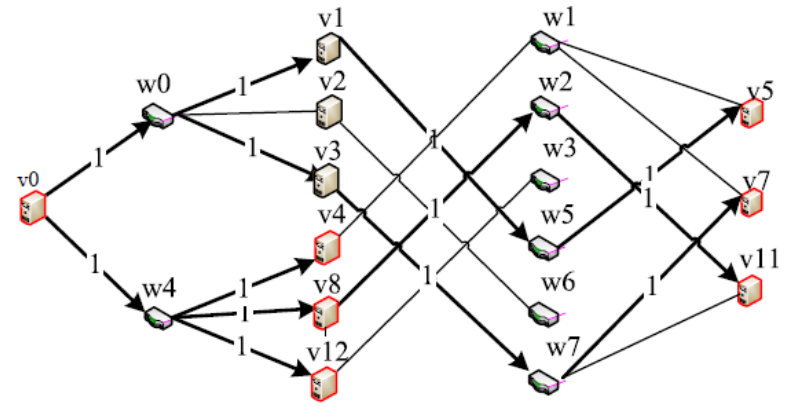
It is well known that multicast group members consists of *m* members, i.e., a source node s and *m*-1 receivers, denoted by $r_1, r_2, ..., r_{m-1}$. The source node delivers the same content to all receivers based on a multicast tree, whose cost is the weight sum of each link.

Multicast has been widely used by many applications in data centers. For web search services [15], the incoming user query is directed to a set of indexing servers to look up its matching documents [16]. Multicast can help accelerate the directing process and reduce the response time. There are also some bandwidth-intensive applications, such as Distributed file systems (like GFS [17] and HDFS), being extensively deployed in data centers. In these systems, files are divided into many fix-sized data chunks, with typical size of 64M or 100 M. Each data chunk is then delivered to servers from at least two different racks to improve the system reliability. Under this circumstance, Multicast can significantly save the inter-rack bandwidth for such bandwidth-hungry applications. To sum up, multicast is able to significantly save the network traffic, improve the application throughput and reduce the probability of the package congestion in the backbone net.

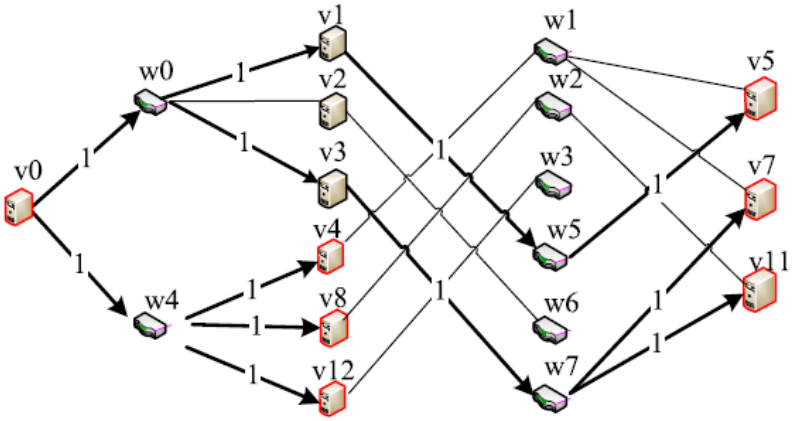
In this paper, we try to build an efficient multicast tree with the lowest cost covering all members of a given multicast group. This is well-known as the Minimal Steiner Tree (MST) problem, which is NP-hard in general graphs as well as the generalized hypercube. Therefore, the construction of a multicast tree with the lowest cost is NP-hard in data centers with generalized hypercube network structure, such as BCube, FBFLY and HyperX. In this paper, we focus on proposing approaches to approximate the optimal solution.

2.3 Existing approximation methods

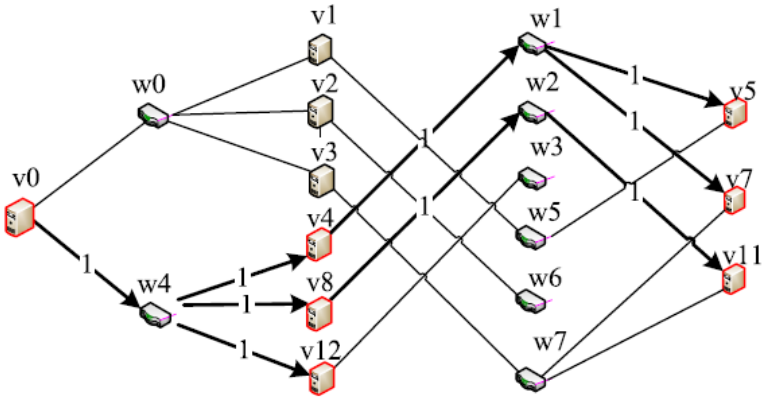
Little efforts have been paid on constructing efficient multicast trees in data centers. We start with representative approximation methods for solving the MST problem. Although some approximation methods for the MST problem obtain better approximation ratio, they exhibit higher computation complexity compared to the one proposed in [13]. In [13], a fast construction algorithm of approximation MST is presented. Firstly, a complete graph containing all multicast group members is constructed, based on which a minimum spanning tree is derived. Then, each virtual link in the minimal spanning tree is replaced by its corresponding shortest path in practice. Finally, the minimum spanning tree is derived based on the resultant graph again. Specifically, unnecessary edges are deleted until all the leaves in the graph are multicast group members. We have implemented such algorithm in **Section 4**.



(a) A Steiner-tree algorithm tree with 13 links.



(b) An ESM tree with 12 links.



(c) A HD-based tree with 9 links.

Fig. 2. Three multicast trees derived by different methods, given a multicast group with the sender v_0 and the receiver set $\{v_4, v_5, v_7, v_8, v_{11}, v_{12}\}$ in Bcube(4,1).

The way to build an efficient multicast tree should ensure that those paths between the sender and receivers share links as more as possible. For the Steiner-tree algorithm in [13], the step of replacing the virtual link by the real shortest path has strong randomicity. Under the

setting of BCube, there are many equal paths between every pair of nodes. The Steiner-tree algorithm, however, just randomly selects one path from multiple equal shortest paths between a pair of nodes. Since there is no cooperation between those two nodes in such path selection behavior, the resultant multicast tree cannot share more links. **Fig. 2**(a) plots a multicast tree resulting from the Steiner-tree algorithm. It is clear that its cost is larger than that of multicast trees as shown in **Fig. 2**(b) and **2**(c). Moreover, the time complexity of such algorithm is $O(m \times N^2)$, where m is the number of receivers. In large-scale data centers, i.e. with a large value of N, this time complexity is apparently too high for online generating an efficient multicast tree.

By exploiting the topological feature of modern data center networks, Li et. al. developed an approximation algorithm [12] , named ESM, to build efficient multicast minimal trees in a source-to-receiver expansion way. Particularly, for BCube (n,k) with the sender s, a set of servers is selected from stage 2 in the group spanning graph such that the selected servers are covered by both s and a single switch at stage 1. Assume that the server set at stage 2 is E, and the selected switch at stage 1 is W. The tree node set for BCube (n,k) is the union of the tree node sets for those $|E|$+1 BCube (n,k-1)s. Then the tree node set in each BCube (n,k-1) is derived by dividing it into several BCube (n,k -2)s. This process iterates until all the BCube (n,0)s are obtained.

Given a sender v_0 and a receiver set $\{v_4, v_5, v_7, v_8, v_{11}, v_{12}\}$, **Fig. 2**(b) shows the multicast tree built by the ESM method [12]. This multicast tree occupies less links than the multicast tree shown in **Fig. 2**(a). Although ESM is a dedicated method for building multicast tree in data centers with Bube topology, it possesses some weaknesses. For example, it is still unknown that how to select servers in E. Consider that BCube (4,1) consists of 4 BCube(4,0)s. When we select the sources of BCube(4,0)s, we can select $\{v_0, v_1, v_2, v_3\}$ or $\{v_0, v_4, v_8, v_{12}\}$. We will get different multicast trees by choosing different sources. Therefore, the ESM method loses some opportunities to share more links.

In this paper, we construct a more efficient multicast tree in a bottom to top manner, i.e. we derive a multicast tree starting from the receivers towards the only source. As shown in **Fig. 2**(c), it is clear that the multicast tree generated by our method, incurs the lowest cost compared to the other two methods, owing to the lowest number of active links.

3. Efficient Multicast Tree Construction

In this paper, we propose a HD-based method to efficiently approximate the minimal multicast tree in data centers based on the generalized hypercube, such as BCube. The number of bits corresponding to two different values of the identifier is called the Hamming distance of the two identifiers. Taking identifiers 10101 and 00110 as an example, the value in the identifier 10101 differs from that in the identifier 00110 by the first, fourth and fifth bit. So, the Hamming distance between these two identifiers is 3. Considering the N bits of the identifier can be represented by a vertex of an n-dimensional hypercube, the Hamming distance of two identifiers is the minimum number of edges between the two vertices in the hypercube, or the shortest distance between the two vertices. Intuitively, two nodes can share some links and may be connected with the same relay node, when the values of their identifiers are different in the same bit. Based on such intuition, we propose the HD-based method.

Given a multicast group in BCube, the sender, receivers and other inner switches or servers of the multicast tree form a multi-stage graph. The multi-stage graph has k+1 stages for the

BCube(n,k). In this multi-stage graph, there exists only one sender at stage 0. The multicast receivers may appear at any stage and all servers at stage k+1 are just some receivers. Generally, only multicast members are insufficient to establish a multicast tree. The basic idea of our method is to discover additional nodes as less as possible for each stage from stage k to stage 1.

To ease the description of our HD-based method, we first give some definitions.

Definition 1: For a server at stage i, we replace the value of its identifier in one dimension with that of the sender's identifier. This generates the identifier of a server at stage i-1, which is defined as the predecessor of the original server.

Definition 2: For any server at stage i, where $i \in \{1,2,...,k+1\}$, there exist multiple predecessors at stage i-1. The selected predecessor at stage i-1 in the tree is called a real predecessor and is a new predecessor if it is not a receiver in the multicast group.

Definition 3: Given a multicast group, the servers in the tree at stage i can be divided into two parts, i.e. those receivers at stage i and the new predecessors of servers at stage i+1.

We first calculate the Hamming distance between the identifiers of the sender and each receiver. Accordingly, we can infer the location of each receiver in the multi-stage graph. That is to say, the receiver is suppose to be at stage i if its Hamming distance to the sender is i, i.e. the server's identifier differs in i dimensions with the sender's identifier. As aforementioned, the servers at the last stage (i.e., in the leaf level of the tree) are receivers. We then start from the last stage and identify the predecessors of servers at each stage in the tree. Note that a server at stage i has multiple predecessors at stage i-1, as discussed in **Definition 2**. For example, the server v_7 in **Fig. 2**(c) has two predecessors, v_3 and v_4. To get the minimal multicast tree, the challenging issue is how to select the real predecessor for each server at stage i, where $0 \le i \le k+1$. We prefer to make the branches of the resulting tree as less as possible, so as to form an efficient multicast tree.

According to the **Definition 1**, we need to choose one dimension $d \in \{0,1,...,k\}$, for each receiver at stage $i = k+1$, so as to find its real predecessor at stage i-1. To achieve the goal, for any given server at stage i, the dimension d should meet the following three conditions:

- First, the identifier of a server should be different from the identifier of the sender in the dimension d.
- Second, the identifier of the predecessor of a server must be the same with the identifier of the sender in the dimension d.
- Third, the selection of the dimension d should minimize the number of servers at stage i-1 in the tree. Note that those servers at stage i-1 are given in **Definition 3**.

As shown in **Fig. 2**(c), the real predecessor of v_7 at stage 2 should be v_3 if d=0, and it should be v_4 if d=1. For a multicast group with the sender v_0 and the receiver set $\{v_4, v_5, v_7, v_8, v_{11}, v_{12}\}$, the servers at stage 2 in the multicast tree are $\{v_5, v_7, v_{11}\}$. When d=1, the real predecessors of such servers are $\{v_4, v_8\}$, and all servers appeared at stage 1 are $\{v_4, v_8, v_{12}\}$. When d=0, the predecessors of such servers are $\{v_1, v_3\}$, and all servers appearing at stage 1 are $\{v_1, v_3, v_4, v_8, v_{12}\}$. It is clear that d=1 incurs less number of servers at stage 1 than d=0. Therefore, we select d=1 at stage 2. Consequently, the sender v_0 will deliver data to v_7 via v_4 since v_4 is the real predecessor of v_7.

Bearing these points in mind, we propose a HD-based method to construct an efficient multicast tree. The detail of our HD-based method is listed in **Algorithm 1**. For a multicast group in BCube, we first calculate the Hamming distance from each receiver to the sender *s*. Accordingly, we can derive the stage where each receiver of the multicast graph locates at the multi-stage graph. We then iteratively infer those real predecessors at the next stage for all servers from stage *k*+1 to stage 1 in the multicast tree. At stage *k*+1, we determine the value of *d* from a set $\{0,1,...,k\}$, which minimizes the number of servers at stage *k* compared to other possible values of *d*. As a result, we pick out those servers which should appear at stage *k*, as shown in **Definition 3**. In the same way, we then choose all servers at stage *k*-1. This process iterates until we get all servers at stage 1. The server at stage 0 is just the root of the resulting efficient multicast tree.

In BCube(*n*, *k*), the time complexity of our HD-based algorithm is $O(k^2 \times h)$, where $h = \max\{m_j, j \in \{1,2,...,k,k+1\}\}$, m_j denotes the number of servers at stage *j* in the tree.

Algorithm 1: Multicast tree construction

Require: a *k*+1 dimensional *n*-ary generalized hypercube

```
randomly select a set of receivers;
for i=0 to receivers.length do
    compute the Hamming distance w between the sender and each receiver;
    append receiver i to dishop[w];
for i=k+1 to 1 do
    for m=0 to k do
        for j=0 to doshop[i].length do
            count the number of receivers in the tree at stage i;
    select d=m that makes the number of servers in the tree at stage i least;
    append new predecessors to dishop[i-1];
    count the number of active links between stage i and stage i-1;
count the number of links in the tree.
```

The proposed methodologies for BCube can be adapted to other data centers with generalized hypercube structure (such as FBFLY [9] and HyperX [10]) after minor modifications. The core of the HD-based method is the identifier of the server. For a flattened butterfly (FBFLY) topoloty with *k*-ary *n*-flat topology, the server in this topology can be encoded as $\{x_{n-1}...x_i...x_1x_0\}, (0 \le x_i < k)$. For a regular (*L*, *S*, *K*, *T*) HyperX, a server can be identified by $\{s_{L-1}...s_i...s_1s_0t\}, (0 \le t < T, 0 \le s_i < S)$ in the data center. After encoding servers, we can calculate the Hamming distance between each pair of servers, and then apply the HD-based method to construct a multicast tree.

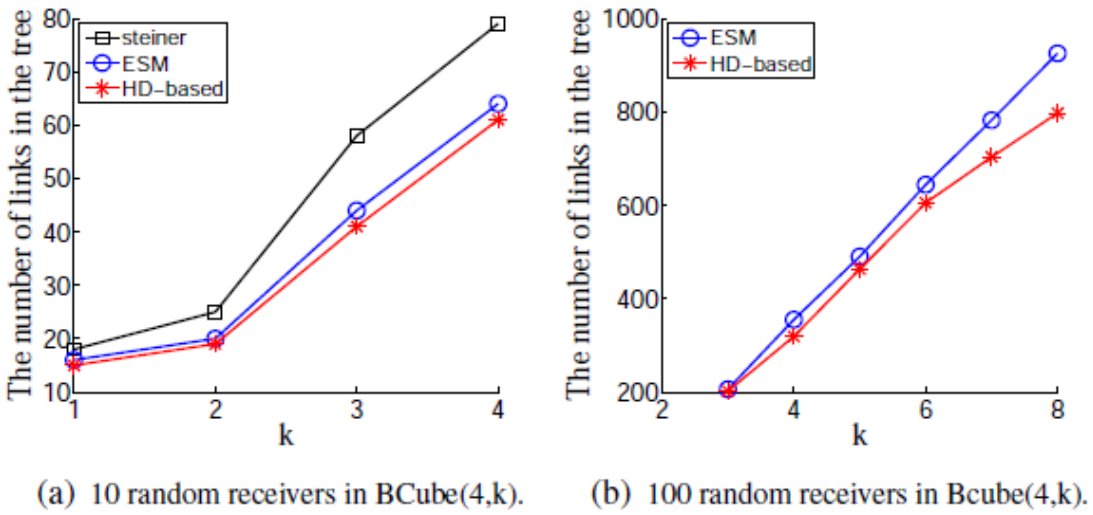
4. Performance Evaluation

In this section, we evaluate the performance of our HD-based method by comparing with two related multicast tree construction algorithms, namely the ESM algorithm [12] and the Steiner-tree algorithm [13], under different settings of data centers and multicast groups.

4.1 Impact of the Size of Data Centers

To evaluate the impact of the data center size, we start with a multicast consisting of the sender v_0 and 10 random receivers in BCube(4, k), where k changes from 1 to 4. Given such a multicast group, we generate multicast trees by using the three different methods. We then calculate the number of links in different trees. In the same way, we conduct such an evaluation 200 rounds and then achieve the number of links in the tree on average.

It is well known that the less the number of links is, the more efficient the multicast tree is. **Fig. 3**(a) shows that the multicast tree constructed by the ESM occupies less links than the multicast tree due to the Steiner-tree algorithm. Additionally, the number of links in the tree built by our HD-based method is less than that tree in the case of ESM. Therefore, the HD-based method can generate a more efficient multicast tree than the ESM [12] and the Steiner-tree [13] algorithms. To evaluate the performance of our HD-based method in larger data centers, we further do more experiments as follows.



(a) 10 random receivers in BCube(4,k). (b) 100 random receivers in Bcube(4,k).

Fig. 3. The impact of data center size on the cost of resulting multicast trees.

We set the sender as v_0 and select 100 random receivers in BCube(4, k), where k ranges from 3 to 8. Similarly, we generate the multicast trees by using the ESM and our HD-based methods under each configuration. For a multicast group in each data center, we conduct 200 rounds of experiments for each method. We then get the average number of links in the resulting multicast tree for each method. **Fig. 3**(b) clearly demonstrates the HD-based method incurs less links than the ESM [12] method. When there are 100 random receivers, the HD-based method can save 128 links than the ESM in BCube(4,8).

To summarize, our HD-based approach incurs less links in the resulting multicast tree compared to the ESM [12] and the Steiner-tree [13] algorithms, irrespective the size of data center. Therefore, our HD-based method can construct a more efficient multicast tree than the ESM and the Steiner-tree algorithms.

4.2 Impact of the Size of Multicast Groups

We further evaluate the performance of our HD-based method when varying the size of multicast group. We set the sender as v_0 and vary the number of randomly selected receivers from 10 to 100 in BCube(4,3). We then generate multicast trees through the three building methods in each setting. After conducting 200 rounds of experiments in each setting, we calculate the average number of links in the resulting multicast trees built by each of three methods. **Fig. 4**(a) plots the cost of different multicast trees. We can see that the multicast tree of the HD-based method utilizes less links than that of the Steiner-tree algorithm and the ESM method. Note that the advantage of the HD-based method is not obvious than the ESM. The root cause is that the data center is too small, just 256 servers. Therefore, we evaluate the HD-based method and the ESM in larger data centers.

We consider a data center of BCube(8,5) that hosts 262,144 servers. In this setting, we still use v_0 as the sender of multicast group and change the number of receivers from 100 to 3,000. Additionally, we conduct 200 rounds of experiments for each method under each configuration.

Accordingly, we achieve the average number of links in the multicast trees resulting from two different methods, the ESM and the HD-based methods. As shown in **Fig. 4**(b), the number of links in the tree from our HD-based method is considerably less than that tree from the ESM. It is worthy noticing that when the number of multicast receivers is 3,000, our HD-based method saves 1,200 links on average than the ESM. Such evidences demonstrate that our HD-based method outperforms the ESM.

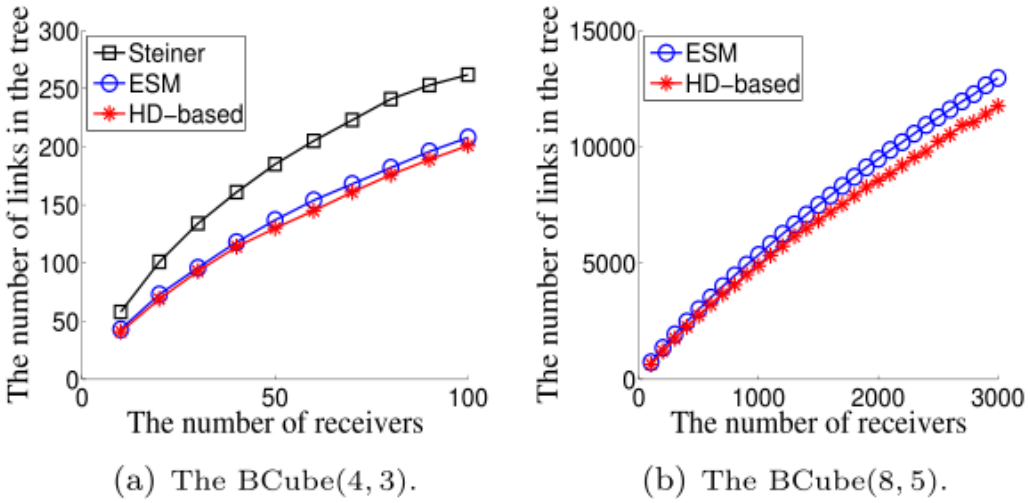


Fig. 4. The impact of multicast group size on the cost of multicast trees.

To summarize, **Fig. 4**(a) shows that, our HD-based method is always better than the Steiner-tree [13] and the ESM [12] algorithms,irrespective the number of receivers. Note that those methods result in the same multicast tree if a multicast group uses all servers in the data center. **Fig. 4**(b) shows that our HD-based method always occupies less links than the ESM method for large-scale data centers. It thus generates a more efficient multicast tree than the other two method, regardless of the size of multicast group.

4.3 Computational Time Complexity

In data centers, a fast generation algorithm is essential for finding an efficient multicast tree for each large number of multicast groups simultaneously, especially in large-scale data centers. It is well known that the time complexity of the Steiner-tree algorithm [13] is $O(h \times N^2)$, where h is the number of the receivers and N is the number of servers in a data center. Due to the increasing scale of data centers, the Steiner-tree algorithm [13] cannot generate efficient multicast trees online for large number of multicast groups simultaneously. To find a faster algorithm, Li et al. has proposed an approximate method, the ESM [12], for the minimal multicast tree. The time complexity of ESM in BCube(n,k) is $O(N)$, where $N = n^{k+1}$, and N is the number of servers in data centers. The time complexity of the ESM is lower than the Steiner-tree algorithm. However, many data centers, such as data centers of Google, host very large number of servers. Consequently, the ESM still costs much time to find an efficient multicast tree. Particularly, when the number of receivers is small, it is not worth to cost so much time.

The time complexity of our HD-based method is $O(k^2 \times h)$, where $h = \max\{m_j, j \in \{1,2,...,k,k+1\}\}$, m_j denotes the number of servers at stage j in the tree. Consider that the value of k is relatively small and h is just part of the total number of receivers in a multicast group. Therefore, our HD-based method consumes less time to construct an efficient multicast tree than the ESM [12] and the Steiner-tree [13] algorithms. In the multi-stage graph, the number of servers is $C_n^i \times (n-1)^i$, in the stage i. Hence, the inequation $h < (n-1)^{k+1}$ holds. In BCube(8,3), $k^2 \times h < k^2 \times (n-1)^{k+1} = 3^2 \times 7^4 < 8^4 = n^{k+1} = N$. Therefore, if the multicast group is not large compared to the size of a data center, our algorithm consumes less time than the ESM and the Steiner-tree algorithms. Actually, the number of receivers of a multicast group in data centers is not too large, as discussed in [12]. Therefore, our HD-based method can significantly reduce the construction time of the multicast tree than the other two methods.

5. Conclusion

Multicast has many advantages in data centers and thus is widely used by many applications. To generate an efficient multicast tree in data centers, we propose an approximation tree-building method, named HD-based multicast trees, for the minimal multicast tree problem. The topological structures of data centers, such as BCube, FBFLY, and HyperX, are the generalized Hypercube. Given a multicast group, the HD-based method can jointly schedule the path from each of receiver to the unique sender among multiple disjoint paths; hence, it can quickly construct an efficient multicast tree with low costs. The experimental results demonstrate that our method consumes less time to construct an efficient multicast tree, while considerably reduces the cost of the multicast tree compared to other representative methods.

References

[1] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, and etc., "Dr. Multicast: Rx for Data Center Communication Scalability," in *Proc. of ACM Eurosys'10*, April, 2010. Article (CrossRef Link).

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

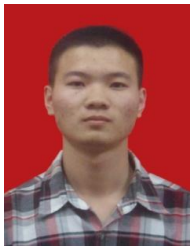
63–74, August 2009. Article (CrossRef Link).
[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. of ACM SIGCOMM*, pp. 63–74, August, 2008. Article (CrossRef Link).
[4] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proc. of ACM SIGCOMM*, pp. 51–62, August, 2009. Article (CrossRef Link).
[5] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: Ascalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, 2008, pp. 75–86. Article (CrossRef Link).
[6] H. Abu-Libdeh, P. Costa, A. Rowstron, G. OShea, and A. Donnelly, "Symbiotic routing in future data centers," *ACM SIGCOMM Computer Communication Review*, 2010, pp. 51–62. Article (CrossRef Link).
[7] D. Guo, T. Chen, D. Li, Y. Liu, X. Liu, and G. Chen, "Bcn: Expansible network structures for data centers using hierarchical compound graphs," in *Proc. of IEEE INFOCOM*, pp. 61–65, April, 2011. Article (CrossRef Link).
[8] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, and J. Wu, "Scalable and cost-effective interconnection of data-center servers using dual server ports," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 102–114, 2011. Article (CrossRef Link).
[9] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. of ACM ISCA*, pp. 338–347, 2010. Article (CrossRef Link).
[10] J. H. Ahn, N. L. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proc. of ACM/IEEE Conference on High Performance Computing(SC)*, pp. 1–11, 2009. Article (CrossRef Link).
[11] D. Guo, T. Chen, D. Li, M. Li, Y. Liu, and G. Chen, "Expansible and Cost-Effective Network Structures for Data Centers Using Dual-port Servers," IEEE Transactions on Computers (**TC**), vol. 62, no. 7, pp. 1303-1317, 2013. Article (CrossRef Link).
[12] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "Esm: efficient and scalable data center multicast routing," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 3, pp. 944–955, 2012. Article (CrossRef Link).
[13] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981. Article (CrossRef Link).
[14] 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. Article (CrossRef Link).
[15] J. Chen, H. Wang, D. Towey, et al. "Worst-input mutation approach toweb services vulnerability testing based on SOAP messages," Tsinghua Science and Technology, vol. 19, no.5, pp. 429-441, 2014. Article (CrossRef Link).
[16] C. Chen, S. Koliai and G. Gao. "Exploitation of locality for energy efficiency for breadth first search in fine-grain execution models," Tsinghua Science and Technology, vol. 18, no. 6, pp. 636-646, 2013. Article (CrossRef Link).
[17] S. Ghemawat, H. Gobio, and S. Leungm, "The google file system," in *Proc. of ACM SOSP*, pp. 29–43, 2003. Article (CrossRef Link).

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. Email: xiejunjie06@gmail.com

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, wireless and mobile systems, and interconnection networks. He is a member of the ACM and the IEEE. Email: guodeke@gmail.com

Jia Xu is currently an assistant professor with School of Computer, Electronics and Information at Guangxi University, China. Since she received the Ph.D. degree in Computer Science and Technology from Northeastern University in October 2013, she has been working as a postdoctoral research fellow at Advanced Digital Science Center (ADSC), Illinois at Singapore. Dr. Xu is a member of IEEE, ACM and China Computer Federation (CCF). Her research interests include topology optimization for data center, big data query processing and data privacy protection. Email: xujia.neu@gmail.com

Lailong Luo received the B.S. degree in National University of Defense 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. His research interests include distributed systems, data centers and interconnection networks. Email: luolailong09@163.com



Xiaoqiang Teng received B.S. degree from Shenyang University of Technology, in 2013. He is currently pursuing the M.S. degree in the School of Information System and Management, National University of Defense Technology. His research interests include wireless sensor network, data centers and mobile computing. He is a student member of the ACM. Email: tengxiaoqiang13@163.com