2012 IEEE 26th International Parallel and Distributed Processing Symposium

On Nonblocking Multirate Multicast Fat-tree Data Center Networks with Server Redundancy

Zhiyang Guo and Yuanyuan Yang
Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA

***Abstract*—Fat-tree networks have been widely adopted as network topologies in data center networks (DCNs). However, it is costly for fat-tree DCNs to support nonblocking multicast communication, due to the large number of core switches required. Since multicast is an essential communication pattern in many cloud services and nonblocking multicast communication can ensure the high performance of such services, reducing the cost of nonblocking multicast fat-tree DCNs is very important. On the other hand, server redundancy is ubiquitous in today's data centers to provide high availability of services. In this paper, we explore server redundancy in data centers to reduce the cost of nonblocking multicast fat-tree data center networks (DCNs). First, we present a multirate network model that accurately describes the communication environment of the fat-tree DCNs. Then, we show that the sufficient number of core switches for nonblocking multicast communication under the multirate model can be significantly reduced in arbitrary 2-redundant fat-tree DCNs, i.e., each server has exactly one redundant backup in the data center. We generalize the result to practical fat-tree DCNs where servers may have different number of redundant backups depending on the availability requirements of services they provide, and show that a higher redundancy level further reduces the cost of nonblocking multicast fat-tree DCNs. Finally, we propose a multicast routing algorithm with linear time complexity to configure multicast connections in fat-tree DCNs.**

***Index Terms*—Data center networks, network cost, fat-trees, folded-Clos, nonblocking, multicast, multirate, server redundancy.**

I. Introduction

Data centers form the backbone of a wide range of services offered via the Internet including Web-hosting, e-commerce, social networking, and a variety of more general services such as software as a service (SAAS), platform as a service (PAAS), and grid/cloud computing. Modern data centers, e.g., Microsoft's Azure platform, Google App engine and Amazon's EC2 platform, usually consist of tens of thousands of servers. These servers are interconnected using a specific data center networking (DCN) structure, which represents a significant portion of the total data center cost. According to [3], the fraction of networking cost of a data center can be as high as 15%, and is concentrated primarily on switches, routers and links. As data centers generally represent a significant investment in capital outlay and ongoing expense, reducing the cost of DCNs is a first order design concern for maximizing data center profits. The search for cost-efficient and scalable data center network fabrics that satisfy stringent communication requirements (e.g., nonblocking communication, full bisection bandwidth and robustness to failures) has motivated much research effort in recent years [1], [2].

The fat-tree, a special instance of the Clos network [5], has been widely adopted as the topology for DCNs [1], for the reason that it enables to build a large-scale communication network from many small commodity switches rather than fewer larger and more expensive switches. Due to the huge price difference between commodity and noncommodity switches, such property of the fat-tree provides a strong economical incentive. Fig. 1 illustrates the topology of a level-2 fat-tree, where each edge in the fat-tree network consists of two directed links, the lower level consists of r $(n+m)\times(n+m)$ edge switches and the upper level consists of m $(r \times r)$ core switches. We will use the notion $ftree(m,n,r)$ to denote such a fat-tree.

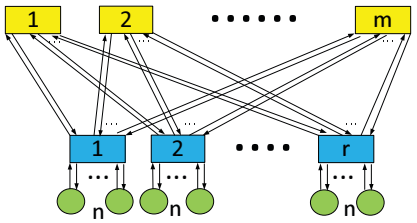


Fig. 1. A folded Clos/fat-tree network $ftree(m,n,r)$.

One of the primary concerns in a fat-tree DCN is that whether it can provide *full bisection bandwidth*, i.e., the capability to connect an arbitrary server with any other server(s) in the network at the full bandwidth of their local network interface. To achieve full bisection bandwidth, a fat-tree DCN has to be nonblocking, that is, it must be able to establish an arbitrary connection without causing any contention in the network at any time. As can seen in Fig. 1, given the number of edge switches r and the number of nodes associated with each edge switch n, the connection capacity of a fat-tree network $ftree(m,n,r)$ depends on the number of core switches m. Deploying more core switches will certainly improve the connection capacity of the fat-tree network, but will also increase the cost. Extensive research has been conducted on finding the most cost-effective values for m to achieve nonblocking communications in such Clos-type networks. While these studies indicate that nonblocking permutation communication can be achieved with a reasonably small number of core switches in a fat-tree network, nonblocking multicast fat-tree still has higher cost than its permutation counterpart.

To achieve fully nonblocking multicast communication, a fat-tree DCN has to be sufficiently provisioned to handle all types of multicast traffic. As all the simultaneous connections from an edge switch will compete for the same group of core switches when establishing connection paths, the demand for core switches is the most stringent when some edge switch is *congested*, i.e., all the servers associated

1530-2075/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPS.2012.96

with the edge switch are active simultaneously and requesting broadcast connections at the full bandwidth capacity of their local network interfaces. Such congestion at edge switches and the non-uniformity of multicast traffic pattern account for the high cost of nonblocking multicast fat-tree DCNs. In the meanwhile, many critical data center services require high bandwidth transmission between a host to a group of servers, such as redirecting search queries to a set of index servers and replicating file chunks in distributed systems. Since nonblocking multicast communication can provide guaranteed high performance for such services, it is very important to reduce the cost of nonblocking multicast fat-tree DCNs.

On the other hand, data centers today have extremely stringent availability requirements, as business operation downtime adversely affects not only the costs of recovery time and opportunity losses, but also the costs of company reputation and customer confidence. For some types of data centers (e.g., data centers for data storage or email), 6-nine (99.9999%) or almost error free is a common availability target [8]. To achieve near-perfect service availability, many data center implementations attempt to build server redundancy into the data center to eliminate single points of failure [9], [11]. Active servers (i.e., servers that are currently running applications) in such data centers have fully redundant instances provided as standby backups, which can be brought online immediately when their associated active servers fail. For example, when an active server with a particular application loses network access or crashes, the application will be unavailable until the problem is fixed. Data centers with server redundancy implemented can remedy this situation by immediately restarting the application on one of its back up servers without requiring administrative intervention, thus shield the external observation of the failure. Server redundancy, though incurs extra hardware cost, is necessary in guaranteeing overall system availability in data centers.

In this paper, we explore such server redundancy in data centers to reduce the cost of nonblocking multicast fat-tree DCNs. First, we present a multirate network model that accurately describes the communication environment in fat-tree DCNs. Then, we show that server redundancy in data centers can be used to effectively mitigate the congestion by evenly distributing active servers and standby backup servers among all edge switches. Therefore, the sufficient condition on the number of core switches for nonblocking multicast communication can be significantly reduced when the fat-tree DCN is 2-redundant, i.e., each active server has one redundant backup server. We also extend the result to practical fat-tree DCNs where servers may have different number of redundant backups depending on the availability requirements of the services they provide, and show that a higher redundancy level further reduces the cost of nonblocking multicast fat-tree DCNs. Next, we compare the sufficient multicast nonblocking condition for fat-tree DCNs with different sizes and server redundancy levels. The comparison results demonstrate the substantial cost saving of exploring server redundancy. Finally, we propose an efficient multicast routing algorithm with linear time complexity for satisfying multicast connection requests in fat-tree DCNs.

The rest of the paper is organized as follows. Section II discusses the related work. Section III briefly introduces the background of fat-tree DCNs and the multirate network model adopted in this paper. Section IV presents the sufficient multicast nonblocking condition on the number of core switches in redundant fat-tree DCNs under the multirate model. The proposed multicast routing algorithm and its complexity analysis are presented in Section V. Finally, Section VI concludes the paper.

II. Related Work

Nonblocking Clos-type networks have been extensively studied under different types of nonblocking capabilities. In general, there are three types of nonblocking capabilities. A network is *strictly nonblocking* if it is always possible to set up a new connection independent of existing connections and path search algorithms. A network is *rearrangeable nonblocking* if a new connection can always be established by rearranging the paths of existing connections. In addition, a network is *wide-sense nonblocking* when it is always possible to set up a new connection by suitably choosing the routes for the new connection, without disrupting the existing connections. As strictly nonblocking Clos-type networks generally have prohibitively high cost, and rearrangeably nonblocking Clos-type networks may disturb existing services and cause packet out-of-order problem, we will focus wide-sense nonblocking in this paper.

Multicast capability of three-stage Clos networks has been investigated in the context of telephone switching network. The most significant results on multicast nonblocking conditions are summarized as follows. Masson [12] and Hwang [13] gave the sufficient conditions on strictly nonblocking and rearrangeable nonblocking multicast Clos networks. Yang and Masson [6] gave a sufficient condition for wide-sense nonblocking multicast Clos networks denoted as $v(m, n, r)$, with $m > 3(n-1)\frac{\log r}{\log \log r}$, which yields the currently best available design for this type of multicast network. In [7], this condition was also shown to be necessary under several commonly used routing strategies.

The analysis of Clos networks for circuit switching telephone traffic adopts a single-rate network model, in which each link is dedicated to a single connection at one time. However, most modern networks operate in a packet switching manner, where packets from different connections are allowed to share a link through multiplexing technique. To accurately describe such networks, Melen and Turner [15] presented a multirate network model, which can support connections with different data rates and a connection can consume an fraction of the bandwidth of link carrying it, instead of exclusively occupying the link. They also gave the strictly nonblocking conditions for multirate permutation Clos networks. Liew and Chan improved the nonblocking condition for permutation Clos networks under both discrete bandwidth model and continuous bandwidth model [16]. Later, Yang [17] generalized the results on single-rate multicast networks to the multirate model, and gave the wide-sense nonblocking condition for multirate Clos networks under multicast traffic.

For a comprehensive summary on multirate Clos networks, readers may refer to [14].

With the emergence of data centers, the fat-tree has become a popular network topology for DCNs. The characteristics and requirements of data center communication pose new challenges and opportunities on the research of fat-tree networks. Much effort has been focused on understanding fat-tree networks by analyzing performance metrics such as blocking probability, or developing techniques to improve network performance [20], [21]. Various routing techniques including oblivious routing [18], multi-path routing [1] and adaptive routing [19], [23] have been proposed. In addition, some effort has been devoted to reconsidering the nonblocking condition for permutation fat-tree networks in cluster communication environments under the constraints of distributed control [22].

This paper differs from all previous work in the following three aspects. First, we approach this problem from a completely different angle and consider how to reduce the cost of nonblocking multicast fat-tree DCNs through exploring server redundancy in data centers. Second, previous work on nonblocking Clos networks are based on three-stage directional Clos networks. However, in fat-tree networks, a connection needs to be forwarded by core switches only when the corresponding source server and destination servers are associated with different edge switches. Therefore, a fat-tree cannot simply be considered as a symmetric Clos network with its input stage and output stage folded together. We take such difference into consideration in our work. Third, the proposed multirate network model addresses unique characteristics of fat-tree DCNs such as higher inter-level link bandwidth, which have not been considered in previous multirate multicast networks.

III. Preliminaries

In this section, we first introduce the background of fat-tree DCNs and server redundancy in data centers. Then, we present the multirate multicast network model. We also give some definitions, notations and observations that will be useful in our analysis of the nonblocking condition for multirate multicast fat-tree DCNs.

A. Fat-tree DCNs

Driven by the huge price gap between large non-commodity switches and small commodity switches, data center severs are usually connected by a tree hierarchy of small switches. Many DCNs adopt a special instance of a Clos topology called *fat-tree*. To connect a large number of servers, level-k ($k > 2$) fat-trees are adopted, which can be recursively built from basic level-2 fat-trees [1], [4].

Fig. 2 shows a level-2 fat-tree DCN, which consists of r $(n+m)\times(n+m)$ edge switches in the bottom level and m $(r\times r)$ core switches in the upper level. An edge switch connects to each core switch through two directed links. We denote the links from edge switches to core switches as *uplinks* and the links from core switches to edge switches as *downlinks*. Such a fat-tee network can interconnect $N = nr$ servers, with each server connected to an edge switch also through two directed links, denoted as *transmitting link* and *receiving link*, respectively.

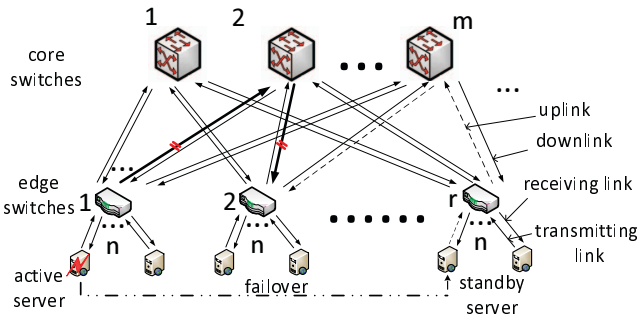


Fig. 2. A fat-tree data center network. When an active server crashes, its applications (whose connections are represented in bold lines) failover to its backup server (dashed lines) immediately.

Generally speaking, both edge switches and core switches in a fat-tree DCN allow all directly connected ports to communicate with each other at full port bandwidth. Therefore, a connection needs to be routed through core switches only when the corresponding source server and destination servers are associated with different edge switches. To reduce congestion between edge switches, fat-tree DCNs usually adopt inter-level links with higher bandwidth than that of transmitting/receiving links directly connected to each server. For example, each edge switch in [1] has some GigE ports (48-288) to connect to the servers, as well as some 10GigE ports to connect to the core switches that aggregate and transfer packets from all the associated servers. The core switches are equipped with 10GigE ports in order to provide large connection capacity between edge switches.

B. Server Redundancy in Data Centers

To provide high service availability in data centers, various server redundancy models have been proposed [11]. In this paper, we adopt a simple active/standby model, which has been implemented in practical data centers [9]. The model can be briefly described as follows. Each server has fully redundant instances provided in case of failure, among which we denote the server that is currently running applications as *active server*, and the redundant instances as *standby servers*. When an active server fails, one of its associated standby servers is brought online, and takes over all the applications from the active server, such a process is called *failover*. Fig. 2 also shows an example of failover.

Each active server and its corresponding standby backups are *identical* instances of each other, meaning that, they share homogeneous software/hardware configurations and data storage to ensure successful failover. In addition, they are required to be *independent* to avoid being taken off-line simultaneously, i.e., they should not share the same access switch, power supply, etc. In our analysis, we say a data center network *2-redundant*, if every server has another redundant instance provided in the data center network, under the constraints that they must be associated with different edge switches. By the same token, we say a data center network *k-redundant*, if every server has other $k-1$ independent redundant instances. In many practical data centers, severs may have different numbers of redundant instances depending on the availability

requirement of services provided [10]. We say such data center networks *general redundant*.

C. Multirate Network Model

In this subsection, we introduce the multirate multicast network model.

First, it is reasonable to assume all switches in a fat-tree DCN have multicast capability, that is, the ability to establish connection from an arbitrary input port to a group of output ports simultaneously. As a fat-tree network can be recursively constructed, each switch requiring multicast capability can be replaced by a multicast fat-tree network of the same size. Therefore, eventually, we may only need to build multicast capability into small commodity switch modules, which involves only minor increases in hardware complexity.

In the multirate network model, a connection can take only a fraction of the total bandwidth of a link, and the available bandwidth left on the link can be used for other connections. For example, assume two connections a and b share a link of bandwidth 1 by time division multiplexing (TDM), meaning that they take turns to send packets. If the packets from a occupy the link $\frac{3}{4}$ of time, and b takes the other $\frac{1}{4}$. Mathematically it is equivalent to saying "the bandwidth of flow a is 0.75 and the bandwidth of b is 0.25."

As mentioned earlier, in order to reduce congestion, many fat-tree DCNs adopt inter-level links with higher bandwidth than links directly connected to servers. To simplify notations, we set the bandwidth of transmitting/receiving links of each server to 1, and denote the normalized bandwidth of inter-level links as S. We define the *weight* of a link as the sum of transmission bandwidth of all connections passing through the link. Also, a link is called ω-*idle*, $0 \leq \omega \leq 1$, if there is *at least* ω available bandwidth left on the link. Note that the number of servers associated with an edge switch, n, generally ranges from hundreds to thousands due to the large scale of today's data centers, whereas it is impractical to use core switches with very high port bandwidth (e.g., an 128-port 10GigE switch would cost orders of magnitude higher than an 128-port GigE switch [1]), hence, we assume that S is much smaller than n in fat-tree DCNs.

A *multicast connection request with weight* ω is a connection request from a source server to a set of destination servers that requires ω transmission bandwidth. If a connection is destined for more than one destination servers on some edge switch, then it is only necessary for the source server to have one connection path to that switch, within which the path can be multicast to as many servers as necessary. Therefore, a multicast connection request can be described in terms of connections between a source server and a set of edge switches along with the transmission bandwidth requirement. Formally, a multirate multicast connection request from a server $i \in I = \{1, 2, \ldots, nr\}$ is denoted as (I_i, ω), where $I_i \subset \{1, 2, \ldots, r\}$ denotes the subset of edge switches to which server i is to be connected in the multicast connection. The weight of the multicast connection, denoted by ω, $0 \leq \omega \leq 1$, is the transmission bandwidth required by the connection.

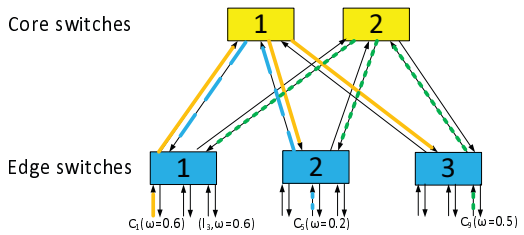


Fig. 3. A $ftree(2, 3, 3)$ multirate fat-tree network. There are three existing connections in the network $C_1(\omega = 0.6)$, $C_5(\omega = 0.2)$ and $C_9(\omega = 0.5)$ marked by lines of different colors and styles.

We consider a discrete bandwidth model as in [16], in which the weight ω of all connections belongs to a given finite set $B = \{b, 2b, \ldots, Db\}$. To simplify the notations, we shall always assume that $1/b$ is an integer and $Db = 1$ in our analysis, as the case that $1/b$ is not an integer can be derived similarly. Apparently, a multirate network reduces to a circuit switching network when $b = 1$. We call the number of destination edge switches that connection (I_i, ω) is connected to the *fanout* of the connection and denote it as $|I_i|$. Note that since only connection paths routed through core switches would affect the nonblocking condition, we can omit the connection paths within the same edge switch in our analysis. Hence, the maximum possible fanout of any multicast connection in a fat-tree DCN is $r - 1$. Meanwhile, we refer to a set of multicast connections as a multicast *assignment*, if the total weight of each transmitting/receiving link in the network does not surpass its bandwidth capacity. A nonblocking multicast network is a network that can realized all possible multicast assignments without congestion.

To fulfill a connection request (I_i, ω), the connection paths will be routed through a set of core switches via ω-idle uplinks, then reach the corresponding destination edge switches I_i via a set of ω-idle downlinks. Therefore, we need to look at the network state from the perspective of both uplinks and downlinks. For a given connection request (I_i, ω), we refer to the set of core switches connected by ω-idle uplinks from the edge switch associated with server i as the *available* core switches. Also, to characterize the connection state of a core switch j, $1 \leq j \leq m$, we define the term *destination set* $M_{j,\omega} \subseteq \{1, 2, \ldots, r\}$ as the set of edge switches that core switch j connects to through downlinks with weight *greater* than $1 - \omega$. In other words, the connection paths to the edge switches in $M_{j,\omega}$ *cannot* be established through core switch j with respect to connection request (I_i, ω).

To better illustrate these notations, we show a small multirate fat-tree network $ftree(2, 3, 3)$ in Fig. 3. We set the bandwidth capacity of all the links in the network to 1. There are three existing connections distributed among edge switches $C_1(\omega = 0.6)$, $C_5(\omega = 0.2)$ and $C_9(\omega = 0.5)$, marked by lines of different colors. Now, suppose there is a connection request $(I_3, \omega = 0.6)$ from server 3. We can see that only core switch 2 is *available* for the request, as only the uplink to core switch 2 is 0.6-idle. Also, according to the available bandwidth of each downlink, we can see that the destination sets are $M_{1,0.6} = \{2, 3\}, M_{2,0.6} = \{1, 2\}$, respectively.

In the next section, we will show that the cost of non-

blocking multicast fat-tree DCNs can be significantly reduced through exploring server redundancy in data centers.

IV. Nonblocking Condition for Multicast Fat-tree DCNs in Redundant Data Centers

In this section, we derive the sufficient multicast nonblocking condition on the number of core switches in redundant fat-tree DCNs under the multirate model. First, we study an ideal case that the fat-tree DCN is 2-redundant, that is, every server has one independent redundant instance provided in the data center network. Next, we generalize the results to practical fat-tree DCNs, where servers may have different numbers of redundant instances depending on the availability requirement of services provided. Finally, we compare the sufficient number of core switches for nonblocking multicast fat-tree DCNs with respect to various network sizes and redundancy levels. As will be seen, the comparison demonstrates substantial cost saving of exploring server redundancy.

A. Sufficient Condition on m for 2-Redundant Fat-tree DCNs

In this subsection, we show that the sufficient multicast nonblocking condition on the number of core switches can be significantly reduced when the fat-tree DCN is 2-redundant.

In a 2-redundant fat-tree DCN, each server has another independent redundant instance in the network. Given that they are identical instances, i.e., they share the same hardware/software configurations, we can assign either one of the two instances as active server and the other as standby backup. The following lemma shows that in arbitrary 2-redundant fat-tree DCNs, we can always find a way to assign active servers, such that the standby servers and active servers are evenly distributed among edge switches.

Lemma 1: In any 2-redundant fat-tree DCN, it is always possible to assign active servers to edge switches, such that each edge switch has $\frac{n}{2}$ active servers when n is even, or at most $\lceil \frac{n}{2} \rceil$ active servers when n is odd, where n is the number of servers associated with each edge switch.

Proof: We can model the edge switches and their associated severs in a 2-redundant fat-tree DCN $ftree(m, n, r)$ by a *Server Distribution Graph (SDG)*, $G(V, E)$, according to the following procedures. First, denote edge switch i as node $v_i \in V$ in the SDG, for $1 \leq i \leq r$. For each server s_k associated with edge switch i, if its redundant instance is associated with edge switch j, we add an edge $e_k \in E$ between node v_i and node v_j. Note that any two servers associated with the same edge switch cannot be the redundant instance of each other, as each active server and its standby backup server must be independent. Therefore, the resulting SDG for any 2-redundant data center is a multigraph (i.e., multiple edges between two nodes may exist) with each node having a degree of exactly n (i.e., n-regular). SDG is not necessarily connected, and each component of the graph is also an n-regular multigraph.

We then give an algorithm called *Eulerian Traversal* on the SDG to assign active servers, and show that the lemma holds for arbitrary 2-redundant fat-tree DCNs. First, we consider the case where n is even. In this case, every node in the SDG has an even degree. According to the property of Eulerian graphs, each component of the SDG contains an Eulerian cycle, which is a cycle that traverses every edge in a connected graph exactly once. It is easy to find an Eulerian cycle in $O(|V|+|E|)$ time for any Eulerian graph $G(V, E)$.

After obtaining an Eulerian cycle in each component of the SDG, we proceed as follows.

- For each Eulerian cycle, pick an arbitrary node, say, v_{i_1}, on the circle as the starting point.
- Traverse the circle from v_{i_1} in either clockwise or counterclockwise direction.
- Assign the edge starting from a node in the traversal to that node. For example, for a traversal $(v_{i_1} \overset{e_{i_1}}{\rightarrow} v_{i_2} \overset{e_{i_2}}{\rightarrow} \cdots \overset{e_{i_{k-1}}}{\rightarrow} v_{i_k} \overset{e_{i_k}}{\rightarrow} \cdots v_{i_1}$, where v_{i_k} and e_{i_k} are the k^{th} encountered node and edge, respectively, edge e_{i_k} is assigned to node v_{i_k}.

In total, each node v_i will be encountered $n/2$ times during the traversal according to the algorithm, as every node in the graph has a degree of n and each encounter accounts for a degree of 2. In addition, each encounter increases the number of edges assigned to node v_i by 1. Thus, node v_i has a total of $n/2$ edges assigned to it. We then choose the servers corresponding to the assigned edges in the SDG as active servers on edge switch i and the rest of servers as standby servers. Hence, the lemma holds when n is even.

We now prove that the lemma also holds when n is odd. In a 2-redundant fat-tree DCN $ftree(m, n, r)$, each server has exactly one independent redundant instance, thus there must be an even number of edge switches (i.e., r is even) given n is odd. Therefore, the SDG in this case is an n-regular multigraph with an even number of nodes. Grouping two nodes into a pair arbitrarily, we can find $r/2$ pairs of nodes in the SDG. Add a *virtual edge* between the two nodes in each pair, then the graph obtained is an $(n+1)$-regular multigraph. Since $n+1$ is even, the Eulerian traversal algorithm can be applied to the graph. We then ignore all the virtual edges and assign active servers for each edge switch in the same way as that when n is even. Clearly, there are at most $\lceil \frac{n}{2} \rceil$ active servers in each edge switch. ■

Lemma 1 shows that we can always evenly distribute active servers and standby servers among edge switches when the fat-tree DCN is 2-redundant. In a fat-tree DCN, all the simultaneous connections from an edge switch will compete for the same set of core switches when establishing connection paths. Hence, the demand for core switches is the most stringent when some edge switch is *congested*, i.e., all associated servers connected to the switch are active, and requesting broadcast connections with full bandwidth capacity of their transmitting links simultaneously. Such congestion at edge switches accounts for the large number of core switches for nonblocking multicast communication in fat-tree DCNs. Since at most half of the servers associated with each edge switches can be active simultaneously according to Lemma 1, the congestion at edge switches can be effectively mitigated. Notice that whether n is even or odd has negligible difference for a reasonably large fat-tree network, thus we will assume n is even in the following sections for clarity.

Suppose there is a new multicast request (I_i, ω) in the

$ftree(m,n,r)$ DCN, which is currently providing a set of multicast connections. Next, we focus on the network condition under which the given request can be satisfied. Suppose the new multicast connection request (I_i,ω) is requesting to connect to r' destination edge switches, where $r'=|I_i|, 1\leq r'<r$ and $\omega\in B$. Connection request (I_i,ω) will be routed to some core switches via a set of ω-idle uplinks, then routed towards the destination edge switches through a set of ω-idle downlinks. We need to look at the network state from the perspective of both uplinks and downlinks. To make the problem tractable, we set a routing constraint that every connection in the network can be routed through at most x $(1\leq x<r)$ core switches. As will be shown later, such constraint is indeed feasible given a sufficient number of available core switches.

First, we consider the number of uplinks that cannot be used by the connection request, or equivalent, the number of core switches that are *not* available to the given connection request. Let $J_2(\omega,x)$ denote the maximum number of inaccessible core switches regarding the given request (I_i,ω) in a 2-redundant fat-tree DCN, under the condition that every connection in the network is routed through at most x $(1\leq x<r)$ core switches. We have the following lemma regarding $J_2(\omega,x)$.

Lemma 2: Given a new multicast connection request (I_i,ω) in an arbitrary 2-redundant $ftree(m,n,r)$ DCN, we have

$$J_2(\omega,x)=\left\lfloor\frac{(n/2-\omega)x}{S-\omega+b}\right\rfloor.$$

Proof: Without loss of generality, assume the new connection request is from the first active server on the corresponding edge switch. By Lemma 1, there are $n/2$ active servers on the edge switch. Since the new connection request (I_i,ω) must be valid, that is, the transmitting link of the corresponding source server and the receiving links of the corresponding destination servers must be ω-idle, the most congested case would be that the transmitting link of the first active server currently carries connections of a total weight $1-\omega$, and each of the remaining $n/2-1$ active servers having multicast connections with full bandwidth of its transmitting link. Then, the sum of the weights of all the existing connections from the edge switch is $n/2-\omega$. Since we limit each connection to be routed through at most x core switches, the total weight on all the uplinks associated with the edge switch is at most $(n/2-\omega)x$.

If the current weight of an uplink out of the edge switch is at least $S-\omega+b$, it does not have sufficient bandwidth for the new connection request, hence, the corresponding core switch is inaccessible for the new connection request. Consequently, the number of uplinks out of the edge switch that carry a weight of at least $S-\omega+b$ cannot be more than $\left\lfloor\frac{(n/2-\omega)x}{S-\omega+b}\right\rfloor$. That proves the lemma. ■

We have consider the network state from the perspective of uplinks. Similarly, we can look at the network state from the perspective of downlinks, as shown in the following corollary.

Corollary 1: Given a new multicast connection request (I_i,ω) in an arbitrary 2-redundant $ftree(m,n,r)$ DCN, for each destination edge switch k in set I_i, there are at most $J_2(\omega,1)$ downlinks to edge switch k that cannot be used by the connection request.

Proof: For the same reason as in Lemma 2, the total weight of all the connections carried on the receiving links of all active servers in any destination edge switch in I_i is at most $n/2-\omega$. However, the difference here is that a connection can only be passed from *one* downlink to the receiving link of a server, that is, $x\equiv 1$. Therefore, we can use $J_2(\omega,1)$ to denote the number of downlinks with weights greater than $S-\omega$ to each destination edge switch. ■

Yang [6] gave the the necessary and sufficient condition that the connection request (I_i,ω) can be satisfied through a set of x available core switches without congestion, as shown in the following lemma.

Lemma 3: We can satisfy a multicast connection request (I_i,ω), using some x $(x\geq 1)$ available core switches, say, $j_1,j_2,\ldots,j_x$, from among the available core switches, if and only if

$$I_i\cap(\bigcap_{k=1}^{x}M_{j_k,\omega})=\emptyset.$$

Lemma 3 indicates that to satisfy a connection request (I_i,ω), a connection path from the source server to each of the destination edge switches must be available through at least one of the x available core switches. Yang and Masson [6] gave a method to find no more than x available core switches that satisfy the condition in Lemma 3. Using a similar technique, we can derive the following lemma.

Lemma 4: Given a new connection request (I_i,ω) with fan-out r', $1\leq r'<r$, in a 2-redundant $ftree(m,n,r)$ DCN, if there exist more than $m'=J_2(\omega,1)r'^{1/x}, 1\leq x\leq r'$ available core switches, then there will always exist x core switches through which this new connection request can be satisfied.

Proof: Suppose we have m' available core switches for a connection request (I_i,ω). Since we are only concerned with the destination edge switches in the connection request, we intersect the destination sets of these m' core switches with I_i, such that each destination set only includes the edge switches in I_i. The destination sets after intersection are still denoted as $M_{j,\omega}, j=1,2,\ldots,m'$ for convenience.

We apply a *minimum cardinality* rule when choosing the set of core switches to satisfy the connection request. In the first iteration, we find the core switch with the destination set of minimum cardinality, denoted as $M_{j_1,\omega}$. From Corollary 1, we know that there are at most $J_2(\omega,1)$ downlinks from core switches to each destination edge switch that cannot be used by the connection request. Therefore, there are at most $J_2(\omega,1)r'$ elements in all $M_{j,\omega}$s. The cardinality of the chosen destination set cannot be more than the average cardinality of all the destination sets, thus $|M_{j_1,\omega}|\leq\frac{J_2(\omega,1)r'}{m'}$. In the next iteration, we concentrate on the destination switches that cannot be reached through $M_{j_1,\omega}$. To do this, we intersect $M_{j_1,\omega}$ with each of the destination sets and obtain another m' sets, denoted as $M^{(1)}_{j,\omega}, j=1,2,\ldots,m'$. It is clear that there are at most $J_2(\omega,1)|M_{j_1,\omega}|$ elements in all these sets. We again choose the set with minimum cardinality $M_{j_2,\omega}$. By the same token, we have

$$|M_{j_2,\omega}|\leq\frac{J_2(\omega,1)|M_{j_1,\omega}|}{m'}\leq\left(\frac{J_2(\omega,1)}{m'}\right)^2 r'.$$

In general, in the k^{th} iteration ($1 \leq k \leq x$),

$$|M_{j_k,\omega}| \leq \left(\frac{J_2(\omega,1)}{m'}\right)^k r'.$$

In order to satisfy the connection request using no more than x core switches, we must have

$$|M_{j_x,\omega}| \leq \left(\frac{J_2(\omega,1)}{m'}\right)^x r' < 1.$$

By solving the inequality, we obtain that

$$m' > J_2(\omega,1) r'^{1/x}.$$

The lemma is thus proved. ■

We are now ready to give the sufficient nonblocking condition on the number of core switches m for a 2-redundant multicast fat-tree DCN $ftree(m,n,r)$.

Theorem 1: A 2-redundant $ftree(m,n,r)$ DCN is nonblocking for any multicast assignments under the multirate model, if

$$m > \min_{1\leq x<r} \max_{\omega\in B} \{J_2(\omega,x) + J_2(\omega,1)(r-1)^{1/x}\} \quad (1)$$

Proof: From Lemma 2, we know that $J_2(\omega,x)$ is the maximum possible number of core switches that are not available for a new connection request (I_i,ω), given that every connection is routed through at most x core switches. Also, Lemma 4 shows that a connection request (I_i,ω) with fanout r' will be satisfied through x available core switches, given that there are more than $J_2(\omega,1)r'^{1/x}$ available core switches. Since the fanout of any connection in the fat-tree DCN is at most $r-1$, we have that for a given x, if

$$m > J_2(\omega,x) + J_2(\omega,1)(r-1)^{1/x}$$

we can satisfy the new connection request. Considering all possible values of x and ω, we obtain the nonblocking condition for 2-redundant multicast fat-tree DCNs. ■

In addition, we can represent the nonblocking condition in Theorem 1 in terms of basic network parameters, as shown in the following corollary.

Corollary 2: A 2-redundant $ftree(m,n,r)$ DCN is nonblocking for any multicast assignments under the multirate model, if

$$m > \min_{1\leq x<r} \left\{ \left\lfloor \frac{(n/2-1)x}{S+b-1} \right\rfloor + \left\lfloor \frac{(n/2-1)}{S+b-1} \right\rfloor (r-1)^{1/x} \right\} \quad (2)$$

Proof: Based on our assumption, n is much larger than S in fat-tree DCNs. Thus, $J_2(\omega,x)$ is maximized when $\omega = 1$, which proves the corollary. ■

For given n and r, we can use Theorem 1 to find an optimum x such that a minimum m can be determined for nonblocking multicast communication in arbitrary 2-redundant fat-tree DCNs. Next, we extend the result to general redundant fat-tree DCNs.

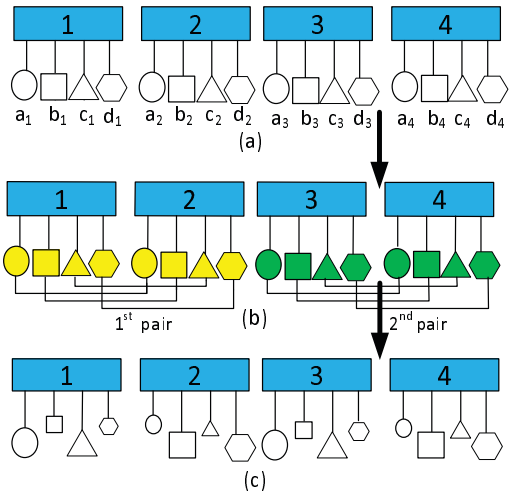


Fig. 4. Server assignment in a simple 4-redundant fat-tree DCN. (a) Server distribution, in which each server has 4 independent identical instances (including itself) denoted by the same shape; (b) Arbitrarily pair 4 identical instances of each server. Assume servers from different pairs are not identical instances, the resulting problem becomes *Assignment(1)*; (c) Solve the problem, then only consider the active servers (denoted by larger shapes in the figure) chosen for the next iteration.

B. Sufficient Condition on m for General Redundant Fat-tree DCNs

In this subsection, we extend the result to general redundant fat-tree DCNs, in which servers may have different numbers of redundant instances.

From Lemma 1, we know that it is always possible to distribute active servers and standby servers evenly in arbitrary 2-redundant fat-tree DCNs. Next, we show that the lemma can be extended to arbitrary k-redundant ($k > 2$) fat-tree DCNs. We begin with the assumption that $k = 2^i$.

Lemma 5: In an arbitrary k-redundant ($k = 2^i$) fat-tree DCN, it is always possible to assign active servers, such that each edge switch has at most $\frac{n}{k}$ active servers.

Proof: Notice that Lemma 1 is a special case of the above lemma when $i = 1$. We denote the server assignment for arbitrary 2-redundant fat-tree DCNs in Lemma 1 as *Assignment(1)* problem, and the server assignment for arbitrary k-redundant ($k = 2^i$) fat-tree DCNs as *Assignment(i)* problem. We prove the lemma by recursively using the Eulerian traversal algorithm. For better illustration, we also show an example of server assignment problem in a simple 4-redundant fat-tree DCN in Fig. 4.

For a server a in a k-redundant fat-tree DCN, there are k identical instances (including itself) independent of each other. We denoted the k instances as $a_1, a_2, \ldots, a_k$ (see Fig. 4(a) for an example). Since $k = 2^i$, we can group two identical instances together in a pair arbitrarily, and obtain 2^{i-1} pairs. Without loss of generality, we denote the obtained server pairs as $\{a_1, a_2\}, \ldots, \{a_{k-1}, a_k\}$.

For assignment purpose, we first assume that the two servers in each of the 2^{i-1} pairs are identical instances to each other, but servers from different pairs are not identical. For example, as shown in Fig. 4(b), a_1 and a_2 are treated as identical instances to each other (yellow nodes), but they are

not considered identical to a_3 and a_4 (green nodes). Under this assumption, we obtain an *Assignment(1)* problem, where each server has another identical instance. This problem can be solved by using the Eulerian traversal algorithm. After that we have each edge switch with $n/2$ associated active servers by Lemma 1. In the next iteration, we try to assign active servers only from the active servers selected in the previous iteration. In this way, each server now has $\frac{k}{2}$ identical instances (including itself), and the problem becomes an *Assignment(i-1)* problem. For example, in Fig. 4(c), if we only consider the active servers chosen, the problem would become an *Assignment(1)* problem.

Clearly, *Assignment(i)* problem can be solved in i iterations. Note that each iteration reduces the number of active servers on each edge switch by half. Therefore, there are $\frac{n}{2^i} = \frac{n}{k}$ active servers in each edge switch after i iterations. The lemma is thus proved. [1] ■

Lemma 5 shows that a higher redundancy level further reduces the congestion at edge switches in a fat-tree DCN. In fact, the condition that $k = 2^i$ in Lemma 5 can be removed. Let $J_k(\omega, x)$ denote the maximum number of inaccessible core switches for the new request (I_i, ω) in an arbitrary k-redundant fat-tree DCN, under the condition that every connection is routed through at most x core switches. We can have the following lemma regarding $J_k(\omega, x)$.

Lemma 6: Given a k-redundant $ftree(m, n, r)$ DCN, and a new multicast request (I_i, ω), we have

$$J_k(\omega, x) = \left\lfloor \frac{(\frac{n}{2^i} - \omega)x}{S - \omega + b} \right\rfloor,$$

where $i = \arg\max_j \{j | 2^j \leq k\}$.

Proof: It is straightforward to see that the server distribution graph (SDG) of a k-redundant fat-tree DCN always contains a subgraph that is the SDG of some 2^i-redundant fat-tree DCN, $i = \arg\max_j \{j | 2^j \leq k\}$. In other words, we can always arbitrarily choose 2^i out of k identical instances for each server in a k-redundant data center, then only assign active servers from these servers. The resulting problem would be an $Assignment(i)$ problem. By Lemma 5, the lemma holds. ■

In practice, depending on the availability requirement of services provided, different servers in data centers usually have a different number of redundant instances [10]. Next, we will generalize the result to general redundant fat-tree DCNs where servers have different redundancy levels.

Theorem 2: A general redundant $ftree(m, n, r)$ DCN is nonblocking for any multicast assignments under the multirate model, where every server is *at least* k-redundant, if

$$m > \min_{1 \leq x < r} \left\{ \left\lfloor \frac{(\frac{n}{2^i} - 1)x}{S + b - 1} \right\rfloor + \left\lfloor \frac{(n/2 - 1)}{S + b - 1} \right\rfloor (r - 1)^{1/x} \right\} \quad (3)$$

where $i = \arg\max_j \{j | 2^j \leq k\}$.

Given Lemma 6, the above theorem can be similarly proved as Corollary 2.

[1] We assume n is divisible by k implicitly. The proof for the case that n is not divisible by k is a trivial extension from the original proof, thus is omitted.

Theorem 2 shows that the sufficient nonblocking condition for general redundant multicast fat-tree DCNs depends on the least redundant servers. It is worth pointing out that Theorem 2 can be used flexibly in building practical fat-tree DCNs. For example, if it is not cost-effective to provide redundant backups for every server, data center owners have the option to select a set of highly redundant servers that host critical services, and make sure the fat-tree DCN is sufficiently equipped to guarantee nonblocking multicast communication for these servers based on Theorem 2. As long as high priority services are allowed to overtake low priority services when realizing connections, the network is guaranteed to be nonblocking for the selected set of servers (or equivalently, the selected set of services).

Given the values of n and r in a general redundant $ftree(m, n, r)$ DCN, we could use Theorem 2 to find the minimum m to build a nonblocking multicast fat-tree DCN. However, it is also of interest to determine a bound on m as an explicit function of n and r. The following Corollary addresses this issue.

Corollary 3: A general redundant $ftree(m, n, r)$ DCN is nonblocking for any multicast assignments under the multirate model, where every server is *at least* k-redundant, if

$$m > 3 \left\lceil \frac{(\frac{n}{2^i} - 1)}{S + b - 1} \right\rceil \frac{\log r}{\log \log r} \quad (4)$$

where $i = \arg\max_j \{j | 2^j \leq k\}$

Proof: The condition shown in Inequality (3) can be relaxed to

$$m > \min_{1 \leq x < r} \left\{ \left\lceil \frac{(\frac{n}{2^i} - 1)}{S + b - 1} \right\rceil (x + r^{1/x}) \right\}. \quad (5)$$

For any constant $u > 0$, we let $x = u\frac{\log r}{\log \log r}$ in (5). Then,

$$r^{1/x} = r^{\frac{u \log \log r}{\log r}} = (\log r)^{\frac{1}{u}}.$$

Letting $u = 2$, (5) can be written as

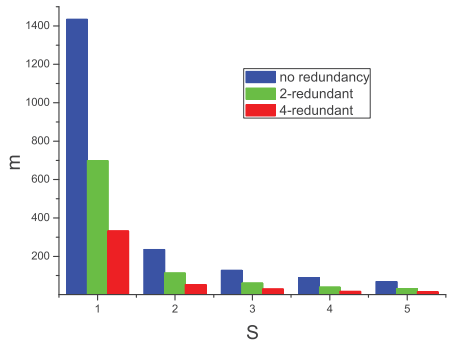
$$m > \left\lceil \frac{(\frac{n}{2^i} - 1)}{S + b - 1} \right\rceil \left[2\frac{\log r}{\log \log r} + (\log r)^{\frac{1}{2}} \right].$$

Since $\frac{\log r}{\log \log r}$ is of higher order than $(\log r)^{\frac{1}{2}}$, we have that

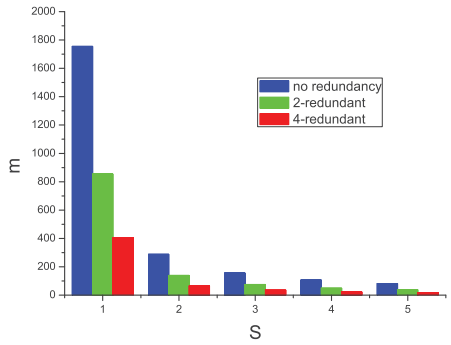
$$m > 3 \left\lceil \frac{(\frac{n}{2^i} - 1)}{S + b - 1} \right\rceil \frac{\log r}{\log \log r}$$

is sufficient for nonblocking multicast communication in fat-tree DCNs. ■

From the above proof, we can see that the bound given in Corollary 3 is in the same order as the original nonblocking condition in Theorem 2. Since a single-rate, circuit switching fat-tree network can be viewed as a special instance of the multirate network model, our nonblocking condition can also be applied to circuit switching fat-tree networks. If we set the the value of b and S to 1, as well as disregard server redundancy ($i = 0$) in Inequity (4), the multirate fat-tree network would reduce to a simple circuit switching network, and the condition in Corollary 3 would become $m > 3(n-1)\frac{\log r}{\log \log r}$, which is consistent with the bound for nonblocking multicast Clos networks obtained in [6].



(a) $n = 40, r = 128$



(b) $n = 40, r = 1024$

Fig. 5. Sufficient multirate multicast nonblocking condition on the number of core switches m in $ftree(m, n, r)$ DCNs with different sizes, redundancy levels and normalized inter-level link bandwidths S. (a) $n = 40, r = 128$; (b) $n = 40, r = 1024$.

C. Some Discussions

In this subsection, we compare the sufficient multicast nonblocking condition on the number of core switches m in fat-tree DCNs $ftree(m, n, r)$ with different sizes and redundancy levels. We also evaluate the impact of the inter-level link bandwidth S on the multicast nonblocking condition.

The main advantage of fat-tree networks is to build large interconnects from smaller switches. Hence, we adopt commodity edge switches with 40 duplex ports for connecting servers, i.e., $n = 40$. Considering the number of severs inside a data center can vary from several hundreds to tens of thousands, we investigate a medium size fat-tree DCN with $r = 128$ and a large size fat-tree DCN with $r = 1024$. We set the bandwidth of transmitting/receiving link of each server to 1, and denote the normalized inter-level link bandwidth as S. The smallest bandwidth fraction that a connection can carry, b, is set to 0.2.

Fig. 5 shows the sufficient multirate multicast nonblocking condition on the number of core switches m in $ftree(m, n, r)$ DCNs with different sizes, redundancy levels and normalized inter-level link bandwidths S. It can be observed from Fig. 5 that, for fat-tree DCNs of both sizes, the sufficient number of core switches m to support nonblocking multicast communication can be significantly reduced through exploring server redundancy. When the $ftree(m, n, r)$ DCN is at least 2-redundant, m can be reduced by about 50%, compared to the case without considering server redundancy. When the fat-tree DCN is at least 4-redundant, m can be further reduced by 75%. Considering the fact that core switches are usually the most expensive components in data center networks due to their large port count and high port speed, the cost saving towards building nonblocking multicast fat-tree DCNs through exploring server redundancy is very substantial.

To support a large number of ports in core switches, the method to build a level-2 nonblocking multicast fat-tree network can be recursively applied to build more levels of nonblocking fat-tree networks. In this case, server redundancy can reduce the cost of nonblocking multi-level fat-tree networks even further. For example, to obtain a level-3 nonblocking fat-tree network, a level-2 nonblocking fat-tree network block can be used to replace each core switch in the original level-2 fat-tree network. Since each core network block supports all types of multicast traffic without contention, it can be shown by induction that the recursively built larger network will also be nonblocking for multicast. The total cost saving by exploring server redundancy in this case would account for both requiring a fewer number of core network blocks and building each nonblocking core network block at lower cost, which is even more substantial than building a basic level-2 nonblocking fat-tree network.

Fig. 5 also shows that the normalized inter-level link bandwidth S has a significant impact on the multicast nonblocking condition. The sufficient number of core switches m for nonblocking multicast fat-tree DCNs drops drastically when S increases. The reason is that adopting higher inter-level link bandwidth allows more connections to be carried on each inter-level link simultaneously, thus reduces the number of core switches m required for nonblocking multicast communication. However, having high inter-level link bandwidth may require non-commodity, high speed core switches, which are generally orders of magnitude more expensive than commodity switches. With the results obtained in this paper, data center designers can find the most economical way to build nonblocking multicast fat-tree DCNs based on the current technology trend.

V. Multicast Routing Algorithm

In the last section, we have given the sufficient multicast nonblocking condition for redundant fat-tree DCNs under the multirate model. While our theorems can provide theoretical guideline for building cost-effective multicast fat-tree DCNs, it remains unclear that what routing algorithm should be adopted to ensure nonblocking multicast communication in such networks. Inspired by the theorems, in this section we provide an efficient multicast routing algorithm of linear time complexity, called *Minimum Cardinality* algorithm, for configuring multicast connections in fat-tree DCNs.

A. Minimum Cardinality Algorithm

The proposed algorithm takes a centralized fashion by leveraging the managed environment of data centers. A central controller collects the traffic demand of incoming connection requests, maintains the data center network states (e.g., the weight of links), computes the routing paths for each connection request, and configures the switches. The idea

TABLE 1
PSEUDO-CODE OF MINIMUM CARDINALITY ALGORITHM

Minimum Cardinality Algorithm:
Input: Connection request (I_i, ω), and network state;
Output: A set of core switches that satisfy connection request (I_i, ω).
Step 1:
If the transmitting link of the source server and the receiving links of the destination servers are not ω-idle
  then exit ((I_i, ω) is not a valid connection request);
else go to Step 2;
Step 2:
Choose m' available core switches and let $T_1, T_2, \ldots, T_{m'}$ be temporary destination sets for these available core switches;
for $j = 1$ **to** m' **do**
  $T_j = M_{j,\omega} \cap I_i$;
Step 3:
Let *core_switch* be the set for holding the labels of chosen core switches for realizing (I_i, ω);
core_switch = $\emptyset$;
repeat
  find $T_k (1 \leq k \leq m')$ such that $|T_k| = \min\{|T_1|, |T_2|, \ldots, |T_{m'}|\}$;
    min_set = T_k;
    core_switch = *core_switch* $\cup \{k\}$;
  if *min_set* $\neq \emptyset$ **then**
    for $j = 1$ to m' **do**
      $T_j = T_j \cap$ *min_set*;
until *min_set* = $\emptyset$;
Step 4:
Connect (I_i, ω) through the core switches stored in *core_switch* and update the weight of corresponding links;
End

of centralized network management is widely adopted in modern data center designs, such as Portland and VL2. Hence, our scheduling controller can be integrated into this type of network fabric controllers in the implementation.

Recall that, in the proof of Lemma 4, we show that given a sufficient number of core switches m', any connection request can be satisfied through at most x core switches, under the condition that minimum cardinality rule is adopted when choosing these core switches. We apply such an idea in the proposed minimum cardinality algorithm, the detail of which is described in Table 1.

Given a general redundant fat-tree DCN $ftree(m, n, r)$, we can use Eulerian traversal algorithm to assign active servers, such that there is always a sufficient number of available core switches left for any new valid connection request, provided that the multicast fat-tree DCNs satisfies the nonblocking condition established in Theorem 2, and each connection is established by using no more than x core switches. We can have a corresponding $x, (1 \leq x < r)$, which achieves minimum m, according to Theorem 2. Assume that there is an incoming connection request $(I_i, \omega), |I_i| = r' < r$, in a fat-tree DCN that is at least k-redundant, we can take a set of any $m' = J_k(\omega, 1) r'^{1/x} + 1$ available core switches, where $J_k(\omega, 1)$ is a constant determined by Lemma 6. Without loss of generality, we denote the destination sets of these core switches as $M_{1,\omega}, M_{2,\omega}, \ldots, M_{m',\omega}$.

Minimum Cardinality algorithm generates a set of core switches, through which the connection request (I_i, ω) can be satisfied. The first step is to check whether the connection request is valid or not. In the second step, temporary destination sets $T_1, T_2, \ldots, T_m$ are generated by intersecting I_i with the corresponding destination sets $M_{1,\omega}, M_{2,\omega}, \ldots, M_{m',\omega}$. The reason is that we are only concerned with the destinations that are to be reached by the connection request. The third step iteratively finds a set of core switches through which a portion of I_i can be realized. In each iteration of the third step, a core switch k with the *minimum cardinality* among the temporary destination sets is chosen. At the end of each iteration, each of the temporary destination sets is updated by intersecting with the newly selected temporary destination set T_k. The iteration continues until the chosen temporary destination set T_k becomes empty. Then the *core_switch* holds the labels of the core switches which can cover all the destinations in I_i. The last step realizes connection request (I_i, ω) by using the selected core switches in *core_switch*, and updates the destination sets.

By Theorem 2, we can guarantee that, for each new valid multicast connection request, there always exist m' available core switches, among which no more than x switches can be selected for realizing the multicast connection request. Therefore, with Minimum Cardinality algorithm, the multicast fat-tree DCN $ftree(m, n, r)$ is nonblocking when it satisfies the nonblocking condition on m in Theorem 2.

B. Time Complexity Analysis

The most involved part of the minimum cardinality algorithm is Step 3, which determines the time complexity of the algorithm. Note that, all the sets used in the algorithm are subsets of $\{1, 2, \ldots, r\}$, then they can be represented by a "bit-vector" data structure. In each iteration, the algorithm tries to find the minimum cardinality set *min_set*, then intersects it with all other m' temporary sets. Assuming each intersection is performed sequentially, the time complexity of each iteration is proportional to $m'|min\_set|$.

As shown in the proof of Lemma 4, we can have the following inequity regarding the cardinality of *min_set* after the first iteration

$$|min\_set_1| \leq \frac{J_k(\omega, 1) r'}{m'} < \frac{r'}{r'^{1/x}}.$$

Letting $\alpha = r'^{-1/x}$, the cardinality of *min_set* after the jth iteration satisfies

$$|min\_set_j| < \alpha^j r'.$$

As shown in Lemma 4, there are at most x iterations in Step 3, thus the total time for Step 3 is

$$O(m'r'(1+\alpha+\cdots+\alpha^{x'-1})) \leq O\left(\frac{m'r'}{1-\alpha}\right) = O\left(\frac{m'r'}{1-r^{-1/x}}\right)$$

We then have that the time complexity is $O(m'r')$ for Step 3, as well as for the entire algorithm. The value of x also affects the time complexity of the algorithm. For example, if we let $x = \log r$, then

$$r^{-1/x} \leq r^{-1/\log r} = \frac{1}{2}.$$

Clearly, as $J_k(\omega, 1)$ is a constant in the order of $O(n)$, the time complexity of the algorithm in this case is $O(nr) = O(N)$, which is linear to the network size. Note that, the intersection of the bit vector can be efficiently computed in parallel within constant time using a simple combination circuit. In that case, the time complexity algorithm can be further reduced.

VI. Conclusions

In this paper, we explore server redundancy in data centers to reduce the cost of nonblocking multicast fat-tree DCNs. We present a multirate network model that accurately describes the communication environment of fat-tree DCNs. Then, we show that the sufficient condition on the number of core switches for nonblocking multicast communication can be significantly reduced when the fat-tree DCN is 2-redundant, i.e., each server has one independent redundant instance in the network. We also generalize the results to practical fat-tree DCNs where servers may have different numbers of redundant instances. Based on the theorems, we provide an efficient multicast routing algorithm with linear time complexity to configure multicast connections in fat-tree DCNs. We also compare the sufficient multicast nonblocking condition on the number of core switches m in fat-tree DCNs with different sizes and redundancy levels. The comparison results demonstrate the substantial cost saving of exploring server redundancy.

Acknowledgments

This research work was supported in part by the U.S. National Science Foundation under grant numbers CCF-0915823 and CCF-0915495.

References

[1] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proc. ACM SIGCOMM'08,* Aug. 2008.

[2] L. Popa, S. Ratnasamy, G. Iannaccone, et al., "A Cost Comparison of Datacenter Network Architectures," *Proc. ACM Co-NEXT 2010,* pp. 16-28, 2010.

[3] A. Greenberg, J. Hamilton, D. A. Maltz and P. Patel, "The Cost of a Cloud : Research Problems in Data Center Networks," *ACM SIGCOMM CCR: Editorial note,* Jan. 2009.

[4] R.N. Mysore, A. Pamboris, N. Farrington, et al., "PortLand : A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," *Proc. ACM SIGCOMM'09,* pp. 39-50, Aug. 2009.

[5] C. Clos, "A Study of Nonblocking Switching Networks," *Bell System Technical Journal,* 32:406-424, 1953.

[6] Y. Yang and G.M. Masson, "Nonblocking Broadcast Switching Networks," *IEEE Transactions on Computers,* vol. 40, no. 9, pp. 1005-1015, Sept. 1991.

[7] Y. Yang and G.M. Masson, "The Necessary Conditions for Clos-type Nonblocking Multicast Networks," *IEEE Transactions on Computers,* vol. 48, pp. 1214-1227, 1999.

[8] M. Armbrust, A. Fox, R. Griffith, et al. "Above The Clouds: A Berkeley View of Cloud Computing." *Technical Report UCB/EECS-2009-28,* EECS Department, U.C. Berkeley, Feb. 2009.

[9] "Data Center High Availability Clusters Design Guide." *http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/HA_Clusters*

[10] N. Bonvin, T.G. Papaioannou and K. Aberer, "Cost-efficient and Differentiated Data Availability Guarantees in Data Clouds," *IEEE ICDE' 10*, pp. 980-983, Mar. 2010.

[11] S. Distefano, F. Longo, and M. Scarpa, "Availability Assessment of HA Standby Redundant Clusters," *29th IEEE Symposium on Reliable Distributed Systems,* pp. 265-274, Oct. 2010.

[12] G.M. Masson and B.W. Jordan, "Generalized Multi-stage Connection Networks," *Networks,* vol. 2, pp. 191-209, 1972.

[13] F.K. Hwang, "Rearrangeability of Multiconnection Three-stage Networks," *Networks,* vol. 2, pp. 301-306, 1972.

[14] J.S. Turner and R.Melen, "Multirate Clos networks," *IEEE Communications Magazine,* vol. 41, no. 10, pp. 38-44, Oct. 2003.

[15] R. Melen and J.s. Turner, "Nonblocking Multirate Networks," *SIAM Journal of Computing,* vol. 18, no. 2, pp. 301-13, Apr. 1989.

[16] S.C. Liew, Ming-Hung Ng and C.W. Chan, "Blocking and Nonblocking Multirate Clos Switching Networks," *IEEE/ACM Transactions on Networking,* vol. 6, no. 3, pp. 307-318, Jun. 1998.

[17] Y. Yang, "An Analysis Model on Nonblocking Multirate Broadcast Networks," *ACM International Conference on Supercomputing,* pp. 256-263, 1994.

[18] X. Yuan, W. Nienaber, Z. Duan and R. Melhem, "Oblivious Routing in Fat-Tree Based System Area Networks With Uncertain Traffic Demands," *IEEE/ACM Transactions on Networking,* vol. 17, no. 5, pp. 1439-1452, Oct. 2009.

[19] J. Kim, W. J. Dally and D. Abts, "Adaptive Routing in High-Radix Clos Network," *ACM SC'2006,* Nov. 2006.

[20] Y. Yang and J. Wang, "A More Accurate Analytical Model on Blocking Probability of Multicast Networks," *IEEE Transactions on Communications,* vol. 48, no. 11, pp. 1930-1936, Nov. 2000.

[21] A. Singh, " Load-Balanced Routing in Interconnection Networks," PhD thesis, Stanford University, 2005.

[22] X. Yuan, "On Nonblocking Folded-Clos Networks in Computer Communication Environments," *IEEE IPDPS 2011,* May 2011.

[23] D. Li, J. Yu, J. Yu and J. Wu, "Exploring Efficient and Scalable Multicast Routing in Future Data Center Networks," *IEEE INFOCOM 2011,* Mar. 2011.

[24] Y. Chen, S. Jain, V.K. Adhikari, et al, "A First Look at Inter-Data Center Traffic Characteristics via Yahoo! Datasets," *IEEE INFOCOM 2011,* Mar. 2011.