Application and Analysis of Multicast Blocking Modelling in Fat-Tree Data Center Networks

Guozhi Li, Songtao Guo, Guiyan Liu, and Yuanyuan Yang

College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China

Correspondence should be addressed to Songtao Guo; guosongtao@cqu.edu.cn

Received 23 May 2017; Accepted 8 November 2017; Published 11 January 2018

1. Introduction

Recently, data center networks (DCNs) have been widely studied in both academia and industry due to the fact that their infrastructure can support various cloud computing services. The fat-tree DCN, as a special instance and variation of the Clos networks, has been widely adopted as the topology for DCNs since it can build large-scale traffic networks by only using fewer switches [1].

Multicast transmission is needed for efficient and simultaneous transmission of the same information copy to a large number of nodes, which is driven by many applications that benefit from execution parallelism and cooperation, such as the MapReduce type of application for processing data [2]. In fact, multicast is the parallel transmission of the data packets in complex network. For example, Google File System (GFS) is a distributed file system for massive data-intensive application in a multicast transmission manner [3].

There have been some studies on multicast transmission in fat-tree DCNs. The stochastic load-balanced multipath routing (SLMR) algorithm selects optimal path by obtaining and comparing the oversubscription probabilities of the candidate links, and it can balance traffic among multiple links by minimizing the probability of each link to face network blocking [4]. But the SLMR algorithm only studies unicast traffic. The bounded congestion multicast scheduling (BCMS) algorithm, an online multicast scheduling algorithm, is able to achieve bounded congestion as well as efficient bandwidth utilization even under worst-case traffic conditions in a fat-tree DCN [5]. Moreover, the scheduling algorithm fault rate (SAFR) reflects the efficiency level of scheduling algorithm. The larger the SAFR is, the lower efficiency the scheduling algorithm has. The SAFR in fat-tree DCNs increases faster with network blocking rate (NBR) compared with that in other DCNs as shown in Figure 1. In fact, the NBR reflects the degree of network blocking [6].

The scheduling processes in the existing scheduling algorithms [4–6] are based on the network state at current timeslot. They do not consider that network state may change when data flows begin to transfer after the current scheduling...
requests, which can reduce the algorithm execution time and to and greater than the three times of multicast bandwidth MSaMC failure due to delay error. In particular, we find the lowest blocking probability at next time-slot, which avoids the blocking probability and choose the downlinks with the link weight. In the downlink selection, we compare the state at next time-slot and the optimal downlink are selected.

Therefore, available links are selected based on the network weight in the multicast blocking model in fat-tree DCNs. The blocking probability at next time-slot and take them as link approximation.

The strategy is a local search algorithm that modifies Markov approximation to data center traffic engineering. The second strategies that construct Markov chains with different connection relationships. The first strategy just applies Markov approximation to data center traffic engineering. The second strategy is a local search algorithm that modifies Markov approximation.

In this paper, we adopt Markov chains to deduce the link blocking probability at next time-slot and take them as link weight in the multicast blocking model in fat-tree DCNs. Therefore, available links are selected based on the network state at next time-slot and the optimal downlink are selected by the link weight. In the downlink selection, we compare the blocking probability and choose the downlinks with lowest blocking probability at next time-slot, which avoids MSaMC failure due to delay error. In particular, we find that the remaining bandwidth of the selected uplinks is close to and greater than the three times of multicast bandwidth requests, which can reduce the algorithm execution time and save bandwidth consumption. Theoretical analysis shows the correctness of the strategy while simulation results show that MSaMC can achieve higher network throughput and lower average delay.

The contributions of the paper can be summarized as follows:

(i) We analyze why multicast blocking occurs in practical application. Afterwards, we present a novel way of multicast transmission forecasting and the multicast blocking model in fat-tree DCNs.

(ii) We put forward a multicast scheduling algorithm (MSaMC) to select the optimal uplinks and down-links. MSaMC not only ensures lower network blocking but also maximizes the utility of network bandwidth resources.

(iii) Theoretical analysis shows that the link blocking probability is less than 1/3 by our proposed MSaMC algorithm, and the multicast network can be non-blocking if the link blocking probability is less than 0.1.

The rest of the paper is organized as follows: Section 2 describes the detrimental effects of multicast blocking in fat-tree DCNs. Section 3 establishes the multicast blocking probability model in fat-tree DCNs and deduces the link blocking probability at next time-slot based Markov chains. In Section 4, we propose multicast scheduling algorithm with Markov chains (MSaMC) and analyze the complexity of MSaMC algorithm in Section 5. In Section 6, we evaluate the performance of MSaMC by simulation results. Finally, Section 7 concludes this paper.

2. Cause of Multicast Blocking

A fat-tree DCN as shown in Figure 2 is represented as a triple \( f(m, n, r) \), where \( m \) and \( r \) denote the number of core switches and edge switches, respectively, and \( n \) indicates the number of servers connecting to an edge switch. In fat-tree DCNs, all links are bidirectional and have the same capacity. We define the uplink as the link from edge switch to core switch and the downlink as the link from core switch to edge switch. A multicast flow request \( \omega \) can be abstracted as a triple \((i, D, \omega)\), where \( i \in \{1, 2, \ldots, r\} \) is the source edge switch and \( D \) denotes the set of destination edge switches by the multicast flow request \( \omega \). The number of destination edge switches with multicast flow request \( \omega \) is represented as \( |D| \). \(|D|\le r-1\), which is denoted as fanout \( f \). Note that the servers connecting to the same edge switch can freely communicate with each other, and the intraedge switch traffic can be ignored. Hence, both aggregation and edge layer can be seen as edge layer.

To illustrate the disadvantages of multicast blocking in fat-tree DCNs, a simple traffic pattern in a small fat-tree DCN is depicted in Figure 3. Suppose that there are two multicast flow requests, \( \omega_1 \) and \( \omega_2 \), and every flow request looks for available links by identical scheduling algorithm. Both flow \( \omega_1 \) and flow \( \omega_2 \) have a source server and two destination servers located at different edge switches, and the sum of both
Complexity

Figure 2: The topology of fat-tree DCNs.

Figure 3: The cause of multicast blocking.

is greater than the available link bandwidth. In particular, flow $\omega_1$ and flow $\omega_2$ forward through core switch 1 at the same time and are routed from core switch 1 to edge switch 2 through the same link by the scheduling algorithm, which will cause heavy blocking at the links connected to core switch 1. Therefore, the available bandwidth to each flow will suffer further reduction if the scheduler cannot identify heavy multicast blocking in the fat-tree DCNs.

Figure 3 also explains the main reason of multicast blocking. We can see that multicast blocking has occurred at the link between core switch 1 and edge switch 2. Clearly, before the blocking at the link is alleviated, other links cannot release the occupied bandwidth. This means that the links from edge switch 1 to core switch 1, from edge switch 1 to core switch 2, from core switch 2 to edge switch 3, and from edge switch 3 to core switch 1 are released until the multicast blocking is alleviated. However, the fat-tree DCNs cannot accept the long time to address the blocking due to the requirement for low latency.

In the fat-tree DCNs, different source servers may execute scheduling algorithm in the same time so that they may occupy the same link and the multicast blocking will inevitably occur. Hence, the multicast blocking is a common phenomenon in the applications of DCN so that network performance will be reduced. In addition, there are also many servers as hotspots of user access, which may cause data flow transfer by many to one. In fact, the key reason of multicast blocking is that the network link state at next time-slot is not considered. Several works have been proposed to solve the network blocking in the transmission of multicast packets in DCNs [12, 13]. As data centers usually adopt commercial switches that cannot guarantee network nonblocking, an efficient packet repairing scheme was proposed [12], which relies on unicast to retransmit dropped multicast packets caused by switch buffer overload or switching failure. Furthermore, the bloom filter [13] was proposed to compress the multicast forwarding table in switches, which avoids the multicast blocking in the data center network.

To the best of our knowledge, the exiting multicast scheduling algorithms only considered the network state at the current time-slot in DCNs; thus the delay error between the algorithm execution time and the beginning transferring time of data flow will make the scheduling algorithm invalid. Based on the consideration, we focus on the study of the multicast scheduling in the network state at next time-slot based on Markov chains.

3. Model and Probability of Multicast Blocking

In the section, we first establish the multicast blocking model based on the topology of fat-tree DCNs by using a similar approach. Then we deduce the blocking probability of available downlinks at next time-slot.

3.1. Multicast Subnetwork. A multicast bandwidth request corresponds to a multicast subnetwork in fat-tree DCNs, which consists of available core switches and edge switches for the multicast bandwidth request. The multicast subnetwork in Figure 4 has $f$ destination edge switches, $x$ available core switches, and $n \times f$ servers, where $1 \leq x \leq m$. In the process of multicast connection, the link weight of multicast subnetwork is denoted as the blocking probability
at next time-slot. Thus our goal is to obtain the link blocking probability for any type of multicast bandwidth request at next time-slot.

It is known that the fat-tree DCN is a typical large-scale network, where there are many available links that can meet the multicast connection request. When a link is available for a multicast bandwidth request \( \omega \), the blocking probability of the link at the current time-slot is given by \( p = \frac{\omega}{\mu} \), where \( \mu \) is the remaining bandwidth.

A multicast connection can be represented by the destination edge switches. Given a multicast bandwidth request \( \omega \) with fanout \( f \) \((1 \leq f < r)\), \( P(f) \) indicates the blocking probability for this multicast connection. We denote the blocking of available uplinks \( i \) as the events \( u_1, u_2, \ldots, u_x \), and the blocking of available downlinks between available core switches and the \( k \)th \((1 \leq k \leq f)\) destination edge switches as the events \( d_{i_1}, d_{k_2}, \ldots, d_{kx} \). All available links form a multicast tree rooted at the core switches that can satisfy the multicast connection in the multicast network. Other notations used in the paper are summarized in Notations.

3.2. Multicast Blocking Model. In the multicast subnetwork, we employ \( \epsilon \) to express the event that the request of multicast connection with fanout \( f \) cannot be satisfied in the network shown in Figure 4. We do not consider the links whose remaining bandwidth is less than multicast bandwidth request \( \omega \), since the link is not available when the multicast data flow \( \omega \) goes through the link. We let \( P(\epsilon \mid \phi) \) be the conditional blocking probability of state \( \phi \) and \( P(\phi) \) be the probability of state \( \phi \). Then the blocking probability of subnetwork for a multicast connection is given by

\[
P(f) = P(\epsilon) = \sum_{\phi} P(\phi) P(\epsilon \mid \phi). \tag{1}
\]

For the event \( \phi \), the data traffic of the uplinks does not interfere with each other; that is, the uplinks are independent. Therefore, we have \( P(\phi) = \phi^m \).

From the multicast blocking subnetwork in Figure 4, we can obtain the blocking property of the fat-tree DCNs; that is, the multicast bandwidth request \( \omega \) from a source edge switch to distinct destination edge switches cannot be achieved if and only if there is no any available downlink connecting all destination edge switches.

In that way, we take \( \epsilon' \) to denote the event that the multicast bandwidth request \( \omega \) with fanout \( f \) cannot be achieved in the available uplinks. Thus we can get

\[
P(\epsilon') = P(\epsilon \mid u_1, u_2, \ldots, u_x). \tag{2}
\]

An available downlink \( d_{ij} \), where \( 1 \leq i < f \) and \( 1 \leq j \leq x \), represents a link from a core switch to the \( i \)th destination edge switch. The event \( \epsilon' \) can be expressed by events \( d_{ij}'s \) as follows:

\[
\epsilon' = (d_{i_1} \cap d_{i_2} \cap \cdots \cap d_{ix}) \cup \cdots \\
\cup (d_{f_1} \cap d_{f_2} \cap \cdots \cap d_{fx}). \tag{3}
\]

Afterwards, we define that the blocking of downlinks connecting to each destination edge switch is event \( A = \{A_1, A_2, \ldots, A_f\} \); moreover, we have \( A_1 = (d_{i_1} \cap d_{i_2} \cap \cdots \cap d_{ix}) \). Thus we get

\[
\epsilon' = \bigcup_{i=1}^{f} A_i. \tag{4}
\]

Based on the theory of combinatorics, the inclusion-exclusion principle (also known as the sieve principle) is an equation related to the size of two sets and their intersection. For the general case of principle, in [14], let \( \{A_1, A_2, \ldots, A_f\} \) be finite set. Then we have

\[
\left| \bigcup_{i=1}^{f} A_i \right| = \sum_{i=1}^{f} |A_i| - \sum_{1 \leq i < j \leq f} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq f} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{f-1} |A_1 \cap A_2 \cap \cdots \cap A_f|. \tag{5}
\]
For the events \( A_1, A_2, \ldots, A_f \) in a probability space \((\Omega, F, P)\), we can obtain the probability of the event \( e' \)

\[
P(e') = \sum_{i=1}^{f} P(A_i) - \sum_{1 \leq i < j \leq f} P(A_i \cap A_j) + \sum_{1 \leq i < j < h \leq f} P(A_i \cap A_j \cap A_h) + \cdots
\]

\[
+ (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f),
\]

where \( P(A_i) \) denotes the probability of the event \( A_i \).

Combining (1) and (2) with (6), the multicast blocking model for a multicast connection with fanout \( f \) is given by

\[
P(f) = \sum_{k=1}^{m} \binom{m}{k} p^k q^{m-k} \left( \sum_{i=1}^{f} P(A_i) + \sum_{1 \leq i < j \leq f} P(A_i \cap A_j) + \sum_{1 \leq i < j < h \leq f} P(A_i \cap A_j \cap A_h) + \cdots
\]

\[
+ (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f) \right).
\]

From (6), \( \sum_{i<j \leq f} P(A_i \cap A_j) \geq \sum_{i<j<k \leq f} P(A_i \cap A_j \cap A_k) \), the following inequality can be derived:

\[
\sum_{i<j \leq f} P(A_i \cap A_j) - \sum_{1 \leq i < j < h \leq f} P(A_i \cap A_j \cap A_h) + \cdots
\]

\[
+ (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f) \geq 0.
\]

Therefore, the minimum blocking probability of the event \( e' \) is

\[
P_{\text{min}}(e') = \frac{f}{\sum_{i=1}^{f} P(A_i)},
\]

where \( P(A_f) = \prod_{i=1}^{x} P_{1,i} \).

Afterwards, we define \( P_{\text{min}}(f) \) as the minimum blocking probability of multicast subnetwork, and the number of available core switches is \( x \). Thus we get

\[
P_{\text{min}}(f) = \sum_{k=1}^{m} \binom{m}{k} p^k q^{m-k} \left( \frac{f}{\sum_{i=1}^{f} P(A_i)} \right)
\]

\[
= \sum_{k=1}^{m} \binom{m}{k} p^k q^{m-k} \left( \prod_{i=1}^{x} P_{1,i} \right),
\]

where \( x \leq m \).

It is not difficult to find from (10) that the minimum blocking probability \( P_{\text{min}}(f) \) is an increasing sequence with fanout \( f \). In other words, it is more difficult to realize a multicast bandwidth request with larger fanout since the number of core switches is less. Therefore, the minimum blocking probability with fanout \( f \) reflects the state of available link at next time-slot.

### 3.3 Link Blocking Probability at Next Time-Slot

In this subsection, we calculate the blocking probability of available link at next time-slot based on Markov chains theory. We randomly select a link denoted by the \( i \)th link to analyze.

In the multicast blocking model, we denote the current time-slot as \( t \), and the next time-slot as \( t+1 \). \( b_i(t) \) is the \( i \)th link occupied bandwidth at time-slot \( t \); that is, \( y_i(t) = b_i(t) \) is the sum of occupied bandwidth of all available downlinks at time-slot \( t \); namely, \( a(t) = \sum_{j=1}^{x} y_j(t) \), and \( y_i(t+1) \) refers to predicted occupied bandwidth of the \( i \)th link at time-slot \( t+1 \). In [15], the preference or uniform selection mechanism based on Markov chains is adopted for calculating the link blocking probability at next time-slot. Based on the mechanism, the probability \( P_i \) of the link incoming new flow at time-slot \( t+1 \) can be given by

\[
P_i = P_{\text{min}}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\text{min}}(f)) \cdot \frac{1}{f},
\]

where \( 1 \leq f \leq r \).

In addition, we do not consider the case that the bandwidth of available link is decreasing; namely, the bandwidth of available link is enough for multicast bandwidth request. If a multicast bandwidth request selects the \( i \)th link at time-slot \( t+1 \), it means \( y_i(t+1) \) will add \( \pi_i \), where \( 1 \leq i \leq M_i \), \( M_i \) is defined as increasing the maximum number of data flows. Then we let \( b_0 \) denote the probability of the \( i \)th link flow remaining unchanged or increasing at time-slot \( t+1 \); thus we can get

\[
P_i = \sum_{\pi_0}^M \left[ P_{\text{min}}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\text{min}}(f)) \cdot \frac{1}{f} \right] \cdot \pi_i
\]

\[
= \sum_{i=1}^{M} \frac{P_i^{\pi_i}}{1 - P_i},
\]

where \( i = 1, 2, \ldots, x \) and \( \pi_i = 0, 1, 1, M_i \).

According to (12), we will calculate one-step transition probability of a multicast flow denoted as \( P(y_i(t+1) = b_i + \pi_i \mid y_i(t) = b_i) \) which is a Markov process.

\[
P(y_i(t+1) = b_i + \pi_i \mid y_i(t) = b_i) = \frac{1}{P_i} \cdot \frac{P_{\text{min}}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\text{min}}(f)) \cdot \frac{1}{f}}{1 - P_i^{\pi_i}}
\]

where \( i = 1, 2, \ldots, x \).

In fact, \( P(y_i(t+1) = b_i + \pi_i \mid y_i(t) = b_i) \) indicates the link blocking probability at time-slot \( t+1 \), which is determined by \( P_i \) and \( \pi_i \). The link blocking probability will be small when \( \pi_i \) is small at time-slot \( t+1 \); otherwise, the link may be blocked at time-slot \( t+1 \). Therefore, the range of \( \pi_i \) is very important to our proposed multicast scheduling algorithm. In this paper, we assume that the multicast bandwidth request \( \omega \) is one data flow unit, and \( \pi_i \) is an integral multiple of multicast bandwidth request \( \omega \).
4. Multicast Scheduling Algorithm with Markov Chains

In the section, we will propose a multicast scheduling algorithm with Markov chains (MSaMC) in fat-tree DCNs, which aims to minimize the blocking probability of available links and improve the traffic efficiency of data flows in the multicast network. Then we give a simple example to explain the implementation process of MSaMC.

4.1. Description of the MSaMC

The core of MSaMC is to select the downlinks with minimum blocking probability at time-slot \( t + 1 \). Accordingly, the first step of the algorithm is to find the available core switches, denoted as the set \( T, |T| \leq f \).

Based on our theoretical analysis in Section 5, the multicast subnetwork may be blocked if it is less than \( 3\omega \); that is, \( u_i \geq 3\omega \).

The second step is to choose the appropriate core switch which is connected to the downlink with minimum blocking probability at time-slot \( t + 1 \) in each iteration. At the end of the iteration, we can transfer the core switches from the set \( T \) to the set \( T' \). The iteration will terminate when the set of destination edge switches \( D \) is empty. Obviously, the core switches in the set \( T' \) are connected to the downlinks with minimum blocking probability. And the set \( T' \) can satisfy arbitrary multicast flow request in fat-tree DCNs [5].

Based on the above steps, we will obtain a set of appropriate core switches \( T' \). Moreover, each destination edge switch in \( D \) can find one downlink from the set \( T' \) to be connected with the minimal blocking probability at time-slot \( t + 1 \).

4.2. An Example of the MSaMC

For the purpose of illustration, in the following, we give a scheduling example in a simple fat-tree DCN as shown in Figure 5. Assume that we have obtained the network state at time-slot \( t \) and made a multicast flow request \((1, (2, 3, 4), 50M)\). The link remaining bandwidth \( \mu \) and link blocking probability \( P \) at next time-slot are shown in Tables 1 and 2, respectively. The symbol \( \vee \) denotes available uplink and \( \times \) indicates unavailable link.

For clarity, we select only two layers of the network and give relevant links in each step.

As described in Section 4.1, the MSaMC is implemented by three steps. First, we take the remaining bandwidth of the uplink as \( u_i (u_i \geq 3 \times 50M) \) and find the set of available core switches; that is, \( T = \{2, 3, 4\} \). Secondly, we evaluate the blocking probability of relevant downlinks at time-slot \( t+1 \). In
5. Theoretical Analysis

In the section, we analyze the performance of MSaMC. By (9), we derived the blocking probability bound of multicast subnetwork, as shown in Lemma 1.

Lemma 1. In a multicast subnetwork, the maximum subnetwork blocking probability is less than 1/3.

\[
\text{max } P_{\text{min}}(f) = p \cdot \left(1 - \left(1 - p_{\text{min}}^x \right)^f \right) \leq \frac{1}{3} \left(1 - \frac{1}{3} \right)^f.
\]

Then we have max \( P_{\text{min}}(f) = 1/3 \) as \( f \to \infty \). This completes the proof. \( \square \)

The result of Lemma 1 is not related to the number of ports of switches. This is because the deduction of Lemma 1 is based on the link blocking probability \( p \), \( p = \omega / \mu \). However, the multicast bandwidth \( \omega \) and the link remaining bandwidth \( \mu \) will not be affected by the number of ports of switches. Therefore, Lemma 1 still holds when the edge switches have more ports. Moreover, the size of switch radix has no effect on the performance of MSaMC.

At time-slot \( t + 1 \), the data flow of available link will increase under the preference or uniform selection mechanism. In addition, the blocking probability of available link should have upper bound (maximum value) for guaranteeing the efficient transmission of multicast flow. Based on (7) and Lemma 1, we can get max \( P_i = 1/3 \) when the number of uplinks and downlinks are equal to 2, respectively. Clearly, this condition is a simplest multicast transmission model. In real multicast network, satisfying this condition. Therefore, Lemma 1 still holds when the edge switches have more ports. Moreover, the size of switch radix has no effect on the performance of MSaMC.
we compute the first-order derivative of (13) about $p_i$, where $i = 1, 2, \ldots, x$.

$$\frac{\partial}{\partial p_i} P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i) = -\frac{p_i^{\pi_i}}{1 - p_i^{M+1}} + \pi_i \cdot (1 - P_i) \cdot \frac{p_i^{\pi_i}}{p_i \cdot (1 - p_i^{M+1})} + \frac{(M_i + 1) \cdot (1 - P_i) \cdot p_i^{\pi_i} \cdot p_i^{M+1}}{p_i \cdot (1 - p_i^{M+1})^2}. \tag{16}$$

In (16), the third term is more than zero, and the second term is greater than the absolute value of the first term when $\pi_i \geq 3$; hence, we can obtain $P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i) > 0$. Therefore, $P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i)$ is a monotonously increasing function for $p_i$ when $\pi_i \geq 3$. The multicast flow request $\omega$ is defined as one data unit; evidently, $\pi_i \geq 3\omega$. In other words, the remaining bandwidth of available link can satisfy the multicast bandwidth request $\omega$ at time-slot $t + 1$ if $\mu \geq 3\omega$. This completes the proof. $\square$

On the basis of Theorem 2, the first step of Algorithm 1 is reasonable and efficient. The condition with $\mu \geq 3\omega$ not only ensures the sufficient remaining bandwidth for satisfying the multicast flow request but also avoids the complex calculation of uplink blocking probability. However, the downlink has data flow coming from other uplinks at any time-slot, which results in the uncertainty of downlink state at time-slot $t + 1$. Therefore, we take the minimum blocking probability at time-slot $t + 1$ as the selection target of optimal downlinks.

Due to the randomness and uncertainty of the downlink state, it is difficult to estimate the network blocking state at time-slot $t + 1$. Afterwards, we deduce the expectation that the $i$th downlink connects to the $j$th destination edge switch at time-slot $t + 1$, denoted by $e_i(t, b_j), \quad j = 1, 2, \ldots, f$. Given that the data flow in the $i$th downlink is $b_j$, we can obtain

$$e_i(t, b_j) = b_j + \frac{1}{P_{b, \pi_i = 1}} \sum_{\pi_i > 0}^{M_i} \pi_i \cdot p_i^{\pi_i},$$

where $P_{b, \pi_i = 1} = (1 - p_i^{M+1})/(1 - P_i), \quad i = 1, 2, \ldots, x$.

By (17), we conclude the following theorem which explains the average increase rate of data flow at each downlink.

**Theorem 3.** In a fat-tree DCN, the increased bandwidth of downlink is no more than two units on the average at time-slot $t + 1$.

**Proof.** We consider $\sum_{\pi_i = 0}^{M_i} P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i) = 1$, which means the flow increment of each link must be one element in set $\{0, 1, \ldots, M_i\}$.

Setting $A = \sum_{\pi_i = 0}^{M_i} \pi_i \cdot p_i^{\pi_i} = p_i^1 + \sum_{\pi_i = 2}^{M_i} \pi_i \cdot p_i^{\pi_i}$, we can get

$$A = \sum_{\pi_i = 0}^{M_i} \pi_i \cdot p_i^{\pi_i} = \sum_{\pi_i = 2}^{M_i} (\pi_i - 1) \cdot p_i^{\pi_i} + M_i \cdot p_i^{M+1}.$$

Through the subtraction of the above two equations, we can obtain $(1 - P_i) \cdot A = P_i^1 + \sum_{\pi_i = 2}^{M_i} p_i^{\pi_i} - M_i \cdot p_i^{M+1}. \quad \text{Then we have } A = (P_i^1 - M_i \cdot p_i^{M+1})/(1 - P_i) + (P_i^2 - M_i \cdot p_i^{M+1})/(1 - P_i^3).$$

Substituting it into (17), we can obtain

$$e_i(t, b_j) = b_j + \frac{1}{P_{b, \pi_i = 1}} \sum_{\pi_i = 0}^{M_i} \pi_i \cdot p_i^{\pi_i} = b_j + \frac{A}{P_{b, \pi_i = 1}} = b_j + \frac{P_i^1 - M_i \cdot p_i^{M+1} + P_i^2 - p_i^{M+1}}{1 - p_i^{M+1}} \quad \text{\quad} \tag{18}$$

$$+ \frac{p_i^2 - p_i^{M+1}}{(1 - P_i) \cdot (1 - P_i^3)} \quad \text{where } P_i < 1/3. \text{ By relaxing the latter two terms of (18), } e_i(t, b_j) \text{ can be rewritten as}$$

$$e_i(t, b_j) = b_j + \frac{P_i^1 - M_i \cdot p_i^{M+1}}{1 - p_i^{M+1}} \quad \text{\quad} \tag{19}$$

$$+ \frac{p_i^2 - p_i^{M+1}}{(1 - P_i) \cdot (1 - P_i^3)} < b_j + 2,$$

where $i = 1, 2, \ldots, x$. By merging (17) and (19), we have $b_j < e_i(t, b_j) < b_j + 2$, then $1 < e_i(t, b_j) - b_j < 1 < 3$. Hence, the downlink bandwidth will increase at least one unit data flow when the downlink is blocked.

When $M_i < e_i(t, b_j) - b_j + 1$, the number of increased data flows is larger than $M_i$; however, it is not allowed by the definition of $M_i$; thus we can obtain

$$P(y_i(t + 1) > e_i(t, b_j) \mid y_i(t) = b_j) = 0. \tag{20}$$

When $M_i \geq e_i(t, b_j) - b_j + 1$, we can get

$$P(y_i(t + 1) > e_i(t, b_j) \mid y_i(t) = b_j) = \frac{M_i}{\sum_{\pi_i = e_i(t, b_j) - b_j + 1}^{M_i} \pi_i \cdot p_i^{\pi_i}} \quad \text{\quad} \tag{21}$$

Equation (21) represents the downlink traffic capability at time-slot $t + 1$. When the value of (21) is very large, the blocking probability of downlink is higher, vice versa. To clarify the fact that the downlink has lower blocking probability at next time-slot, we have the following theorem.

**Theorem 4.** In the multicast blocking model of fat-tree DCNs, the downlink blocking probability at time-slot $t + 1$ is less than 0.125.
In order to show that the MSaMC manifests the lower blocking probability of downlink at time-slot $t+1$ under the different values of $M_i$, we provide the following comparison as shown in Figure 6.

In Figure 6, $P(y_i (t+1) > e_i (t, b_i) \mid y_i (t) = b_i)$ indicates the downlink blocking probability, and their values are not more than 0.125 for different $M_i$ and $P_i$. At the zero point, the blocking probability is close to zero unless $P_i > 0.1$. In real network, the condition of $P_i > 0.1$ is rarely. Therefore, the MSaMC has very lower blocking probability.

In the following, we analyze the time complexity of MSaMC. The first step of MSaMC takes the time complexity of $O(m)$ to identify available core switches. In the second step, the MSaMC needs to find the appropriate core switches. We need $O(f \cdot f)$ time to calculate the blocking probability of available downlinks at time-slot $t+1$ and select the appropriate core switches to the set $T'$, where $f \leq r-1$. In the end, we take $O(f + f)$ time to construct the optimal paths from source edge switch to destination edge switches. Thus the computational complexity of MSaMC is given by

$$O(m + f \cdot f + f + f) \leq O(m + (r-1)^2 + 2(r-1))$$

$$= O(r^2 + m - 1).$$

Note that the complexity of the algorithm is polynomial with the number of core switches $m$ and the number of edge switches $r$, which means that the computational complexity is rather lower if the fanout $f$ is very small. Therefore, the algorithm is time-efficient in multicasting scheduling.

### 6. Simulation Results

In this section, we utilize network simulator NS2 to evaluate the effectiveness of MSaMC in fat-tree DCNs in terms of the average delay variance (ADV) of links with different time-slots. Afterwards, we compare the performance between MSaMC and SLMR algorithm with the unicast traffic [4] and present the comparison between MSaMC and BCMS algorithm with the multicast traffic [5].

#### 6.1. Simulation Settings

The simulation network topology adopts 1024 servers, 128 edge switches, 128 aggregation switches, and 64 core switches. The related network parameters are set in Table 3. Each flow has a bandwidth demand with the bandwidth of 10 Mbps [4]. For the fat-tree topology, we consider mixed traffic distribution of both unicast and multicast traffic. For unicast traffic, the flow destinations of a source server are uniformly distributed in all other servers. The packet length is uniformly distributed between 800 and 1,400 bytes and the size of each multicast flow is equal [17, 18].

#### 6.2. Comparison of Average Delay Variance

In this subsection, we first define the average delay variance (ADV) and then compare the ADV of the uplink and downlink by the different number of packets.

**Definition 5 (average delay variance).** Average delay variance (ADV) $V$ is defined as the average of the sum of the transmission delay differences of the two adjacent packets in a multicast subnetwork; that is,

$$V = \frac{\sum_{i \in e} \sum_{j\in l} (T(t)_{ij} - T(t-1)_{ij})}{x},$$

where $x$ is the number of available links, $l$ is the number of packets in an available link, and $T(t)$ indicates the transmission delay of packet at time-slot $t$.

WE take ADV as a metric for the network state of multicast subnetwork. The smaller the ADV is, the more stable the network state is, vice versa.

Figure 7 shows the average delay variance (ADV) of links as the number of packets grows. As the link remaining bandwidth $\mu$ is taken as $\omega$ or $2\omega$, the average delay variance
6.3. Total Network Throughput. In the subsection, we set the length of time-slot $t$ as $\omega/S$ and $2(\omega/S)$. We can observe from the Figure 9(a) that MSaMC achieves better performance than the SLMR algorithm when the length of time-slot $t$ is $2(\omega/S)$. This is because MSaMC can quickly recover the network blocking, and thus it can achieve higher network throughput. In addition, the MSaMC cannot calculate the optimal path in real time when the length of time-slot $t$ is $\omega/S$; therefore, the SLMR algorithm provides the higher throughput.

Figure 9(b) shows throughput comparison of MSaMC and BCMS algorithm under mixed scheduling pattern. The throughput of BCMS algorithm is lower as the simulation time increases gradually. The multicast transmission of BCMS algorithm needs longer time to address the problem of network blocking; therefore, the throughput will decrease sharply if the network blocking cannot be predicted. In contrast, the MSaMC can predict the probability of network blocking at next time-slot and address the delay problem of dynamic bandwidth allocation. Therefore, the MSaMC can obtain higher total network throughput.

6.4. Average Delay. In this subsection, we compare the average end-to-end delay of our MSaMC, SLMR algorithm with the unicast traffic, and BCMS algorithm with mixed traffic over different traffic loads. Figure 10 shows the average end-to-end delay for the unicast and mixed traffic patterns, respectively.

We can observe from Figure 10 that, as the simulation time increases gradually, the MSaMC with $t = 2(\omega/S)$ has the lowest average delay than SLMR and BCMS algorithms for the two kinds of traffic. This is because SLMR and BCMS algorithms utilize more backtracksto eliminate the multicast blocking; therefore, they take more time to forward data flows to destination edge switches. In addition, we can also find that when the length of the time-slot is $2(\omega/S)$, our MSaMC has the minimum average delay. This is because the time-slot with length $2(\omega/S)$ can just ensure that data can be transmitted accurately to destination switches. The shorter time-slot with less than $2(\omega/S)$ will lead to the incomplete data transmission while the longer time-slot with more than $2(\omega/S)$ will cause the incorrect prediction for traffic blocking status.

7. Conclusions

In this paper, we propose a novel multicast scheduling algorithm with Markov chains called MSaMC in fat-tree data center networks (DCNs), which can accurately predict the link traffic state at next time-slot and achieve effective flow scheduling to improve efficiently network performance. We show that MSaMC can guarantee the lower link blocking at next time-slot in a fat-tree DCN for satisfying an arbitrary sequence of multicast flow requests under our traffic model. In addition, the time complexity analysis also shows that the performance of MSaMC is determined by the number of core switches $m$ and the destination edge switches $f$. Finally, we compare the performance of MSaMC with an existing unicast scheduling algorithm called SLMR algorithm and a well-known adaptive multicast scheduling algorithm called
BCMS algorithm. Experimental results show that MSaMC can achieve higher network throughput and lower average delay.

**Notations**

- $\omega$: Multicast bandwidth request about data flow
- $b_i$: The occupied bandwidth of $i$th link
- $\mu$: The remaining bandwidth of link
- $a$: The sum of occupied bandwidth
- $y$: The value of link weight
- $S$: Link bandwidth
- $M$: Increasing the maximum number of data flows
- $\pi$: Increasing the number of data flows
- $T$: The set of available core switches

**Conflicts of Interest**

The authors declare that they have no conflicts of interest.
Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities (XDJ2016A011, XDJ2015C010, XDJ2015D023, and XDJ2016D047), the National Natural Science Foundation of China (nos. 61402381, 61503309, 61772432, and 61772433), Natural Science Key Foundation of Chongqing (cstc2015jcyjBX0094), and Natural Science Foundation of Chongqing (CSTC2016JCYJA0449), China Postdoctoral Science Foundation (2016M600219), and Chongqing Postdoctoral Science Foundation (XM2016002).

References


