Hindawi
Complexity
Volume 2018, Article ID 7563170, 12 pages
https://doi.org/10.1155/2018/7563170

Research Article

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

Academic Editor: Dimitri Volchenkov

Multicast can improve network performance by eliminating unnecessary duplicated flows in the data center networks (DCNs). Thus it can significantly save network bandwidth. However, the network multicast blocking may cause the retransmission of a large number of data packets and seriously influence the traffic efficiency in data center networks, especially in the fat-tree DCNs with multirooted tree structure. In this paper, we build a multicast blocking model and apply it to solve the problem of network blocking in the fat-tree DCNs. Furthermore, we propose a novel multicast scheduling strategy. In the scheduling strategy, we select the uplink connecting to available core switch whose remaining bandwidth is close to and greater than the three times of bandwidth multicast requests so as to reduce the operation time of the proposed algorithm. Then the blocking probability of downlink in the next time-slot is calculated in multicast subnetwork by using Markov chains theory. With the obtained probability, we select the optimal downlink based on the available core switch. In addition, theoretical analysis shows that the multicast scheduling algorithm has close to zero network blocking probability as well as lower time complexity. Simulation results verify the effectiveness of our proposed multicast scheduling algorithm.

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 time-slot. They do not consider that network state may change when data flows begin to transfer after the current scheduling

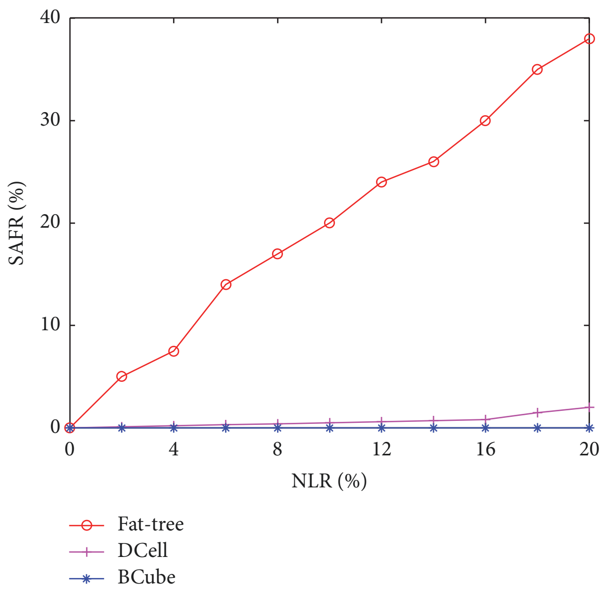


Figure 1: The relationship between network blocking rate (NBR) and scheduling algorithm fault rate (SAFR) in different DCNs.

process is finished. This may lead to the network load imbalance because the bandwidth of multicast connection has not been allocated dynamically [7]. Therefore, we develop an efficient multicast scheduling algorithm to achieve the scheduling of network flows at the network state of next time-slot in fat-tree DCNs.

However, since the network state at next time-slot is probabilistic and not deterministic, it is difficult to predict the network state of next time-slot from the present state with certainty and find a deterministic strategy. The Markov chains can be employed to predict network state, even though state transition is probabilistic [8]. Thus the next network states can be assessed by the set of probabilities in a Markov process [9]. The evolution of the set of probability essentially describes the underlying dynamical nature of a network [10]. In [11], the authors proposed a scheme by using Markov approximation, which aims at minimizing the maximum link utilization (i.e., the link utilization of the most blocked link) in data center networks. Moreover, the scheme provides two 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 downlinks. 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 nonblocking 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 ω can be abstracted as a triple (i, D, ω), 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 ω. The number of destination edge switches with multicast flow request ω is represented as $|D|$, $|D| \leq r - 1$, 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, ω_1 and ω_2, and every flow request looks for available links by identical scheduling algorithm. Both flow ω_1 and flow ω_2 have a source server and two destination servers located at different edge switches, and the sum of both

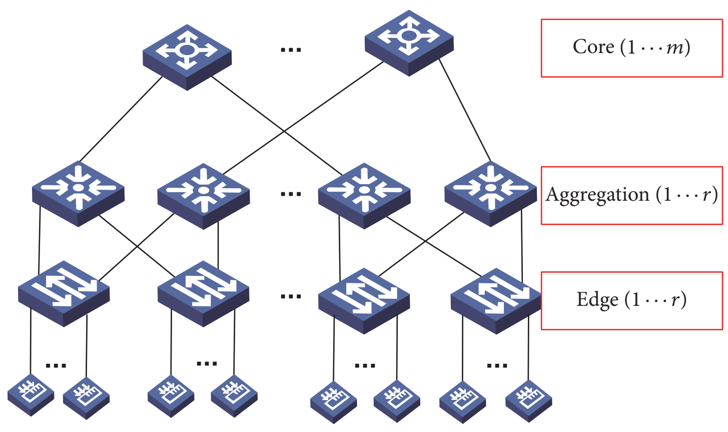


FIGURE 2: The topology of fat-tree DCNs.

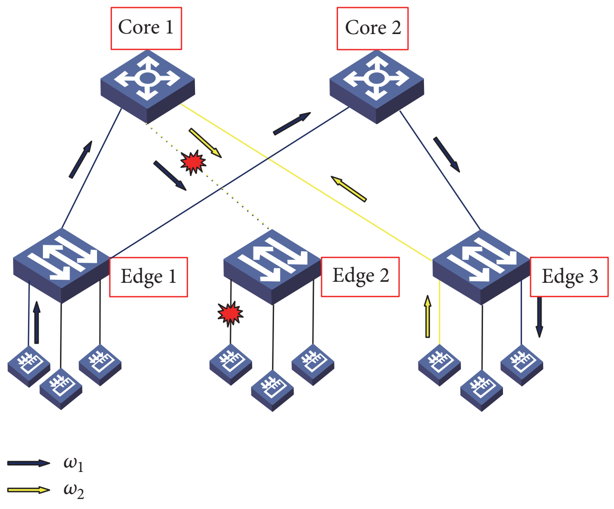


FIGURE 3: The cause of multicast blocking.

is greater than the available link bandwidth. In particular, flow ω_1 and flow ω_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

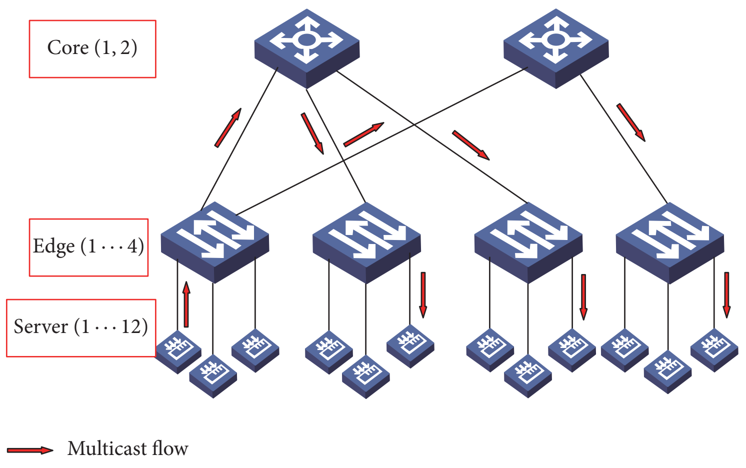


Figure 4: The multicast subnetwork.

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 ω, the blocking probability of the link at the current time-slot is given by $p = \omega/\mu$, where μ is the remaining bandwidth.

A multicast connection can be represented by the destination edge switches. Given a multicast bandwidth request ω with fanout f $(1 \le f < r)$, $P(f)$ indicates the blocking probability for this multicast connection. We denote the blocking of available uplink i as the events $u_1, u_2, \dots, u_x$, and the blocking of available downlinks between available core switches and the kth $(1 \le k \le f)$ destination edge switches as the events $d_{k1}, d_{k2}, \dots, d_{kx}$. 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 ϵ 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 ω, since the link is not available when the multicast data flow ω goes through the link. We let $P(\epsilon \mid \phi)$ be the conditional blocking probability of state ϕ and $P(\phi)$ be the probability of state ϕ. 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 ϕ, the data traffic of the uplinks does not interfere with each other; that is, the uplinks are independent. Therefore, we have $P(\phi) = q^k p^{m-k}$.

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 ω 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 ϵ' to denote the event that the multicast bandwidth request ω with fanout f cannot be achieved in the available uplinks. Thus we can get

$$P(\epsilon') = P(\epsilon \mid u_1, u_2, \dots, u_x). \tag{2}$$

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

$$\begin{aligned} \epsilon' &= (d_{11} \cap d_{12} \cap \cdots \cap d_{1x}) \cup \cdots \\ &\quad \cup (d_{f1} \cap d_{f2} \cap \cdots \cap d_{fx}). \end{aligned} \tag{3}$$

Afterwards, we define that the blocking of downlinks connecting to each destination edge switch is event $A = \{A_1, A_1, \dots, A_f\}$; moreover, we have $A_1 = (d_{11} \cap d_{12} \cap \cdots \cap d_{1x})$. 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 $\{A_1, A_2, \dots, A_f\}$ be finite set. Then we have

$$\begin{aligned} \left|\bigcup_{i=1}^{f} A_i\right| &= \sum_{i=1}^{f} |A_i| - \sum_{1 \le i < j \le f} |A_i \cap A_j| \\ &\quad + \sum_{1 \le i < j < h \le f} |A_i \cap A_j \cap A_h| - \cdots \\ &\quad + (-1)^{f-1} |A_1 \cap A_2 \cap \cdots \cap A_f|. \end{aligned} \tag{5}$$

For the events $A_1, A_1, \ldots, A_f$ in a probability space (Ω, F, P), we can obtain the probability of the event ϵ'

$$\begin{aligned} P(\epsilon') = &\sum_{i=1}^{f} P(A_i) - \sum_{1\le i<j\le f} P(A_i \cap A_j) \\ &+ \sum_{1\le i<j<h\le f} P(A_i \cap A_j \cap A_h) - \cdots \\ &+ (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f), \end{aligned} \tag{6}$$

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

$$\begin{aligned} P(f) = &\sum_{k=1}^{m} \binom{m}{k} p^k q^{m-k} \left(\sum_{i=1}^{f} P(A_i) \right. \\ &- \sum_{1\le i<j\le f} P(A_i \cap A_j) + \sum_{1\le i<j<h\le f} P(A_i \cap A_j \cap A_h) \\ &\left. - \cdots + (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f) \right). \end{aligned} \tag{7}$$

From (6), $\sum_{1\le i<j\le f} P(A_i \cap A_j) \ge \sum_{1\le i<j<h\le f} P(A_i \cap A_j \cap A_h)$, the following inequality can be derived:

$$\begin{aligned} &\sum_{1\le i<j\le f} P(A_i \cap A_j) - \sum_{1\le i<j<h\le f} P(A_i \cap A_j \cap A_h) + \cdots \\ &+ (-1)^{f-1} P(A_1 \cap A_2 \cap \cdots \cap A_f) \ge 0. \end{aligned} \tag{8}$$

Therefore, the minimum blocking probability of the event ϵ' is

$$P_{\min}(\epsilon') = \sum_{i=1}^{f} P(A_i), \tag{9}$$

where $P(A_1) = \prod_{k=1}^{x} p_{1k}$.

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

$$\begin{aligned} P_{\min}(f) &= \sum_{k=1}^{x} \binom{x}{k} p^k q^{x-k} \left(\sum_{i=1}^{f} P(A_i) \right) \\ &= \sum_{k=1}^{x} \binom{x}{k} p^k q^{x-k} \left(\sum_{i=1}^{f} \prod_{j=1}^{x} p_{ij} \right), \end{aligned} \tag{10}$$

where $x \le m$.

It is not difficult to find from (10) that the minimum blocking probability $P_{\min}(f)$ 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 ith 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 is the ith link occupied bandwidth at time-slot t; that is, $y_i(t) = b_i$. $a(t)$ 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 ith 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

$$P_i = P_{\min}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\min}(f)) \cdot \frac{1}{f}, \tag{11}$$

where $1 \le f \le r$.

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 ith link at time-slot $t + 1$, it means $y_i(t + 1)$ will add π_i, where $1 \le \pi_i \le M_i$, M_i is defined as increasing the maximum number of data flows. Then we let P_{b_i} denote the probability of the ith link flow remaining unchanged or increasing at time-slot $t + 1$; thus we can get

$$\begin{aligned} P_{b_i} &= \sum_{\pi_i=0}^{M_i} \left[P_{\min}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\min}(f)) \cdot \frac{1}{f} \right]^{\pi_i} \\ &= \sum_{\pi_i=0}^{M_i} P_i^{\pi_i} = \frac{1 - P_i^{M_i+1}}{1 - P_i}, \end{aligned} \tag{12}$$

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

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.

$$\begin{aligned} &P(y_i(t+1) = b_i + \pi_i \mid y_i(t) = b_i) \\ &\quad = \frac{1}{P_{b_i}} \cdot \left(P_{\min}(f) \cdot \frac{b_i}{a(t)} + (1 - P_{\min}(f)) \cdot \frac{1}{f} \right)^{\pi_i} \\ &\quad = \frac{(1 - P_i) \cdot P_i^{\pi_i}}{1 - P_i^{M_i+1}}, \end{aligned} \tag{13}$$

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 π_i. The link blocking probability will be small when π_i is small at time-slot $t + 1$; otherwise, the link may be blocked at time-slot $t + 1$. Therefore, the range of π_i is very important to our proposed multicast scheduling algorithm. In this paper, we assume that the multicast bandwidth request ω is one data flow unit, and π_i is an integral multiple of multicast bandwidth request ω.

```
Input: Incoming flow $(i, D, \omega)$, link remaining bandwidth $\mu$, the number of destination edge switches $|D|$, $\pi_i = 3\omega$.
Output: Multicast links with the minimum blocking probability.
(1) // Step 1: identify available core switches
(2) for $i = 1$ to $m$ do
(3)     Select an uplink $u_i$;
(4)     if $u_{\mu_i} \geq 3\omega$ and $|T| \leq |D|$ then
(5)         Select the core switch $i$ and add it into the set $T$;
(6)     end if
(7) end for
(8) // Step 2: select appropriate core switches
(9) Calculate the blocking probability of available downlinks at time-slot $t + 1$, $P_i(t + 1)$, by equation (13);
(10) for $j = 1$ to $|D|$ do
(11)     Find the core switch(es) in $T$ that are connected to a destination edge switch in $D$;
(12)     if There are multiple core switches to be found then
(13)         Select the core switch with the minimum blocking probability and deliver it to the appropriate set of core switches $T'$;
(14)     else
(15)         Deliver the core switch to the set $T'$;
(16)     end if
(17)     Remove destination edge switches that the selected core switch from $D$ can reach;
(18)     Update the set of remaining core switches in $T$;
(19) end for
(20) // Step 3: establish the optimal pathes
(21) Connect the links between source edge switch and destination edge switches through appropriate core switches in the set $T'$;
(22) Send configuration signals to corresponding devices in multicast subnetwork;
```

Algorithm 1: Multicast scheduling algorithm with Markov chains (MSaMC).

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, $|T| \leq f$. We take the remaining bandwidth of the ith uplink as u_{μ_i}. Based on our theoretical analysis in Section 5, the multicast subnetwork may be blocked if it is less than 3ω; that is, $u_{\mu_i} \geq 3\omega$.

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 $t + 1$. The third step is to establish the optimal path from source edge switch to destination edge switches through the appropriate core switches. The state of multicast subnetwork will be updated after the source server sends the configuration signals to corresponding forwarding devices. The main process of the MSaMC is described in Algorithm 1.

Table 1: Link remaining bandwidth (M).

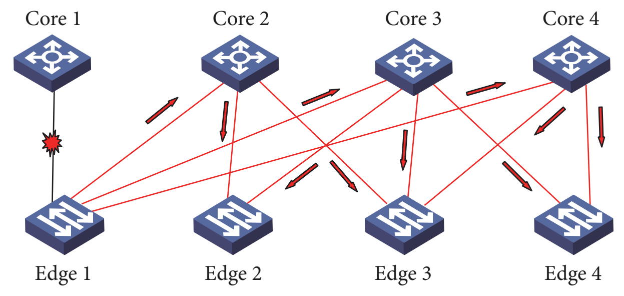
	C1	C2	C3	C4
E1	90	300	600	800
E2	600	700	800	200
E3	750	400	350	700
E4	500	200	150	500

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 μ and link blocking probability P at next time-slot are shown in Tables 1 and 2, respectively. The symbol $\surd$ 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.

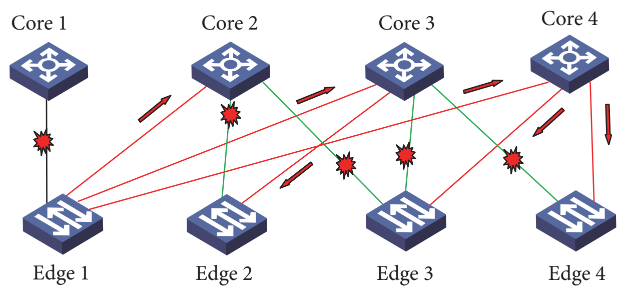
As described in Section 4.1, the MSaMC is implemented by three steps. Firstly, we take the remaining bandwidth of the uplink as u_μ ($u_{\mu_i} \geq 3 \times 50M$) and find the set of available core switches; that is, $T = \{2, 3, 4\}$. Secondly, we evaluate the blocking probability of relevant downlinks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

Table 2: The link blocking probability at next time-slot (%).

	C1	C2	C3	C4
E1	×	9	5	4
E2	×	4	3	7
E3	×	6	7	4
E4	×	9	10	5



(a) The links with satisfying the multicast flow request $(1, (2, 3, 4), \omega)$



(b) The selected optimal paths by the MSaMC

Figure 5: An example of the MSaMC.

effect, the blocking probability of downlink at time-slot $t + 1$ from core switch 2 to destination switch 2 is higher than that from core switch 3 to destination switch 2; therefore, we select the latter downlink as the optimal path. Subsequently, the core switch 3 is put into the set T'. Similarly, we get the core switch 4 for the set T'. Finally, the optimal path is constructed and the routing information is sent to the source edge switch 1 and core switches (3, 4).

In Figure 5(a), the link remaining bandwidth from edge switch 1 to core switch 1 is no less than $150M$. By the above way, we find that the optimal path for a pair of source edge switch and destination edge switch is source edge switch 1 → core switch 3 → destination edge switch 2, source edge switch 1 → core switch 4 → destination edge switch 3, and source edge switch 1 → core switch 4 → destination edge switch 4, as shown in Figure 5(b).

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.*

Proof. We take the remaining bandwidth of uplink to be no less than 3ω by the first step of Algorithm 1, and thus the maximum value of link blocking probability p is 1/3; in other words, the available link remaining bandwidth just satisfies the above condition; that is, $u_\mu = 3\omega$.

From (9) and De Morgan's laws [16], we can obtain the probability of event ϵ'

$$\begin{aligned} P_{\min}\left(\epsilon'\right) &= 1 - \prod_{i=1}^{f} P\left(\overline{d_{i1} \cap d_{i2} \cap \cdots \cap d_{ix}}\right) \\ &= 1 - \prod_{i=1}^{f} \left(1 - P\left(d_{i1} \cap d_{i2} \cap \cdots \cap d_{ix}\right)\right) \\ &= 1 - \prod_{i=1}^{f} \left(1 - \prod_{k=1}^{x} p_{d_{ik}}\right) = 1 - \left(1 - p^x\right)^f . \end{aligned} \tag{14}$$

Therefore, based on (10), the subnetwork blocking probability is maximum when the number of uplinks is 1. Thus we can obtain

$$\begin{aligned} \max P_{\min}(f) &= p \cdot \left(1 - \left(1 - p_{\min}^x\right)^f\right) \\ &= \frac{1}{3}\left(1 - \left(1 - \frac{1}{3}\right)^f\right). \end{aligned} \tag{15}$$

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

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 ω and the link remaining bandwidth μ 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 $P_i \ll 1/3$ is a general condition.

In addition, P_i is proportional to $P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i)$; namely, the link blocking probability will increase as the multicast flow gets larger. Therefore, $P(y_i(t + 1) = b_i + \pi_i \mid y_i(t) = b_i)$ is monotonously increasing for p_i.

Theorem 2. *As the remaining bandwidth of available link μ is no less than 3ω, the multicast flow can be transferred to f destination edge switches.*

Proof. For each incoming flow, by adopting the preferred selection mechanism in selecting the ith link, when $\pi_i \geq 1$, we compute the first-order derivative of (13) about p_i, where $i = 1, 2, \ldots, x$.

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

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 monotonously increasing function for p_i when $\pi_i \geq 3$. The multicast flow request ω 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 ω at time-slot $t + 1$ if $\mu \geq 3\omega$. This completes the proof. □

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 ith downlink connects to the jth destination edge switch at time-slot $t + 1$, denoted by $e_i(t, b_i)$, $j = 1, 2, \ldots, f$. Given that the data flow in the ith downlink is b_i, we can obtain

$$e_i(t, b_i) = \sum_{\pi_i=0}^{M_i} \left((b_i + \pi_i) \cdot P\left(y_i(t+1) = b_i + \pi_i \mid y_i(t) = b_i\right)\right) = b_i + \frac{1}{P_{b_i}} \sum_{\pi_i=1}^{M_i} \pi_i \cdot P_i^{\pi_i}, \quad (17)$$

where $P_{b_i} = (1 - P_i^{M_i+1})/(1 - P_i)$, $i = 1, 2, \ldots, x$.

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 $A = \sum_{\pi_i=1}^{M_i} \pi_i \cdot P_i^{\pi_i} = P_i + \sum_{\pi_i=2}^{M_i} \pi_i \cdot P_i^{\pi_i}$, we can get $P_i \cdot A = \sum_{\pi_i=1}^{M_i} \pi_i \cdot P_i^{\pi_i+1} = \sum_{\pi_i=2}^{M_i} (\pi_i - 1) \cdot P_i^{\pi_i} + M_i \cdot P_i^{M_i+1}$.

Through the subtraction of the above two equations, we can obtain $(1 - P_i) \cdot A = P_i + \sum_{n_i=2}^{M_i} P_i^{\pi_i} - M_i \cdot P_i^{M_i+1}$. Then we have $A = (P_i - M_i \cdot P_i^{M_i+1})/(1 - P_i) + (P_i^2 - M_i \cdot P_i^{M_i+1})/(1 - P_i)^2$. Substituting it into (17), we can obtain

$$e_i(t, b_i) = b_i + \frac{1}{P_{b_i}} \sum_{\pi_i=1}^{M_i} \pi_i \cdot P_i^{\pi_i} = b_i + \frac{A}{P_{b_i}} = b_i + \frac{P_i - M_i \cdot P_i^{M_i+1}}{1 - P_i^{M_i+1}} + \frac{P_i^2 - P_i^{M_i+1}}{(1 - P_i)\left(1 - P_i^{M_i+1}\right)}, \quad (18)$$

where $P_i < 1/3$. By relaxing the latter two terms of (18), $e_i(t, b_i)$ can be rewritten as

$$e_i(t, b_i) = b_i + \frac{P_i - M_i \cdot P_i^{M_i+1}}{1 - P_i^{M_i+1}} + \frac{P_i^2 - P_i^{M_i+1}}{(1 - P_i)\left(1 - P_i^{M_i+1}\right)} < b_i + 2, \quad (19)$$

where $i = 1, 2, \ldots, x$.

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

When $M_i < e_i(t, b_i) - b_i + 1$, 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

$$P\left(y_i(t+1) > e_i(t, b_i) \mid y_i(t) = b_i\right) = 0. \quad (20)$$

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

$$P\left(y_i(t+1) > e_i(t, b_i) \mid y_i(t) = b_i\right) = \sum_{\pi_i = e_i(t,b_i) - b_i + 1}^{M_i} P\left(y_i(t+1) = e_i(t, b_i) \mid y_i(t) = b_i\right) = \sum_{\pi_i = e_i(t,b_i) - b_i + 1}^{M_i} \frac{1}{P_{b_i}} \cdot P_i^{\pi_i} = \frac{P_i^{e_i(t,b_i) - b_i + 1} - P_i^{M_i+1}}{1 - P_i^{M_i+1}}. \quad (21)$$

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.*

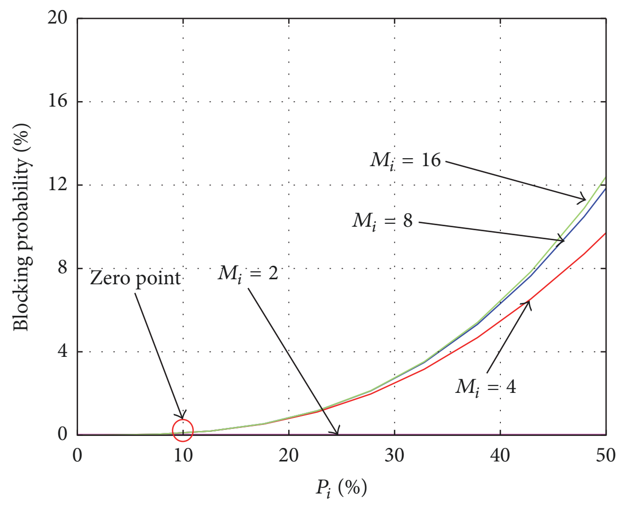


FIGURE 6: Downlink blocking probability comparison in different M_is.

Proof. Based on (21), we take the minimum value of M_i as 2. Thus we get

$$\begin{aligned} &P\left(y_i(t+1) > e_i\left(t, b_i\right) \mid y_i(t) = b_i\right) \\ &= \frac{P_i^{e_i(t,b_i)-b_i+1} - P_i^{M_i+1}}{1 - P_i^{M_i+1}} < \frac{(1/3)^3 - (1/3)^{(3+1)}}{1 - (1/3)^{(3+1)}} \\ &= 0.125. \end{aligned} \quad (22)$$

This completes the proof. □

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 $f \le r - 1$. 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

$$\begin{aligned} O\left(m + f \cdot f + f + f\right) &\le O\left(m + (r-1)^2 + 2(r-1)\right) \\ &= O\left(r^2 + m - 1\right). \end{aligned} \quad (23)$$

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 multicast scheduling.

TABLE 3: Parameter setting.

Parameter	Description
Platform	NS2
Link bandwidth	1 Gbps
RTT delay	0.1 ms
Switch buffer size	64 KB
TCP receiver buffer size	100 segments
Simulation time	10 s

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,

$$V = \frac{\sum_{i \in x} \sum_{j \in l} \left(T(t)_{ij} - T(t-1)_{ij}\right)}{x}, \quad (24)$$

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 μ is taken as ω or 2ω, the average delay variance

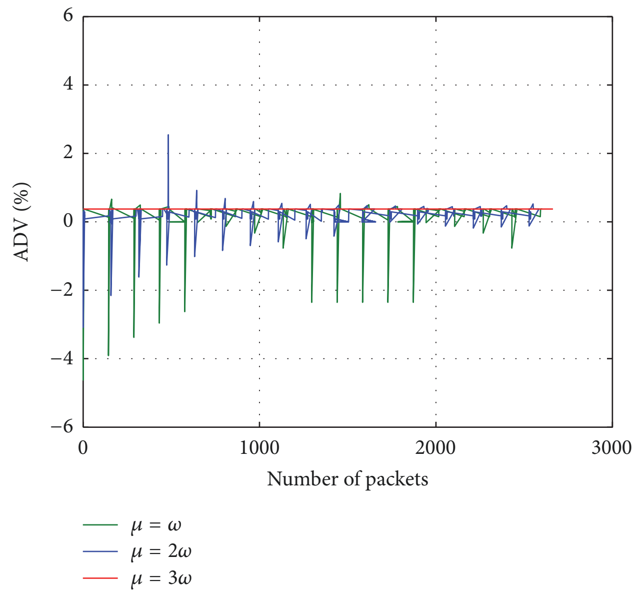


Figure 7: Average delay variance (ADV) comparison among the link of different remaining bandwidth.

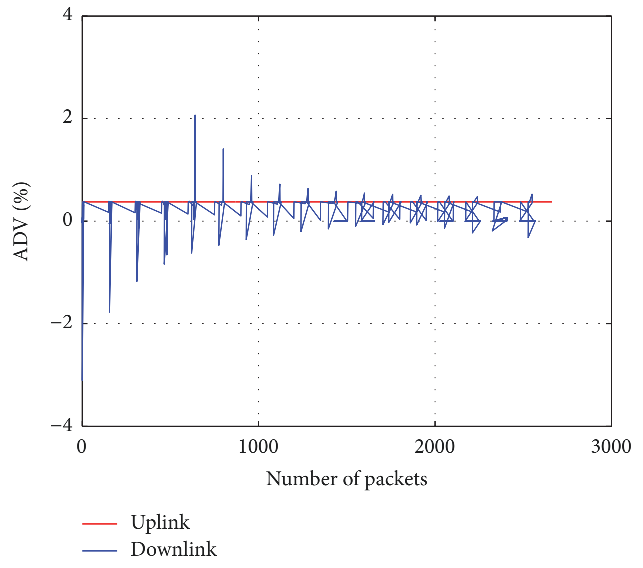


Figure 8: Average delay variance (ADV) comparison between uplink and downlink.

has bigger jitter. This is because the link remaining bandwidth cannot satisfy the multicast flow request ω at time-slot $t+1$. The average delay variance is close to a straight line when the link remaining bandwidth is 3ω, which implies that the network state is very stable. Therefore, the simulation result manifests that the optimal value of the link remaining bandwidth μ is 3ω.

From Figure 8, we observe that the jitter of uplink ADV is smaller than that of the downlink ADV. This is because the fat-tree DCN is a bipartition network; that is, the bandwidth of the uplink and downlink is equal. However, the downlink load is higher than the uplink load in the multicast traffic; therefore, the uplink state is more stable.

6.3. Total Network Throughput. In the subsection, we set the length of time-slot t as ω/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 ω/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 throughout 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 backtracks to 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

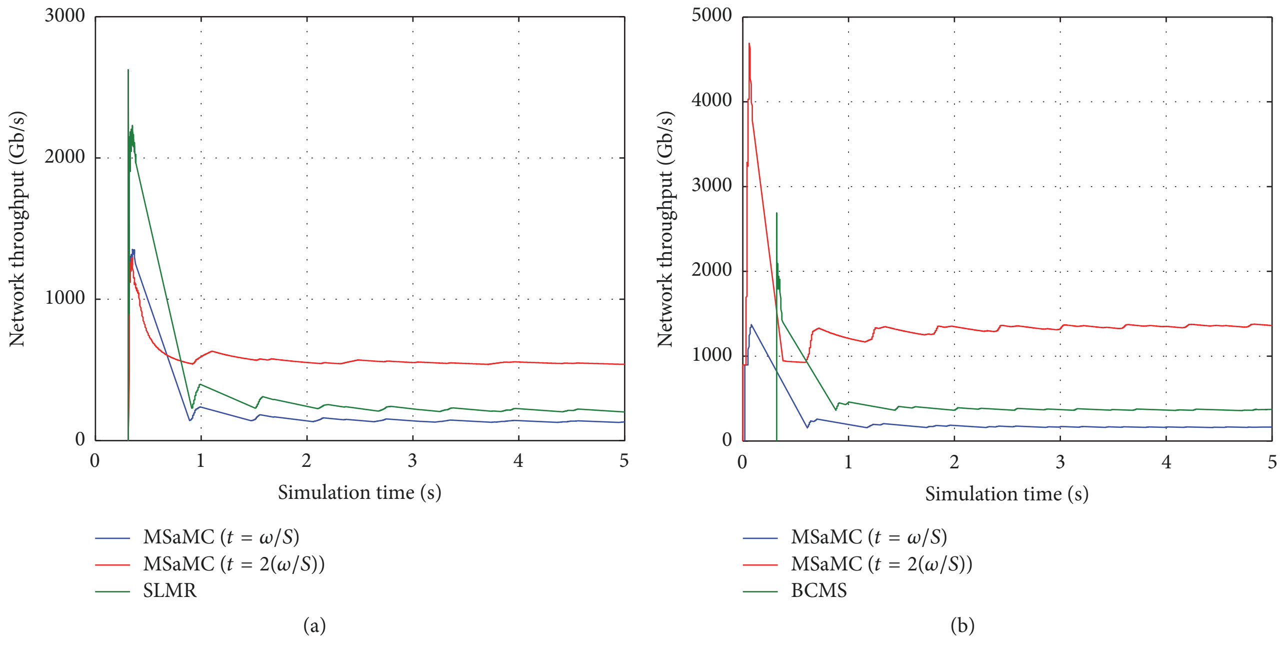


FIGURE 9: Network throughput comparison.

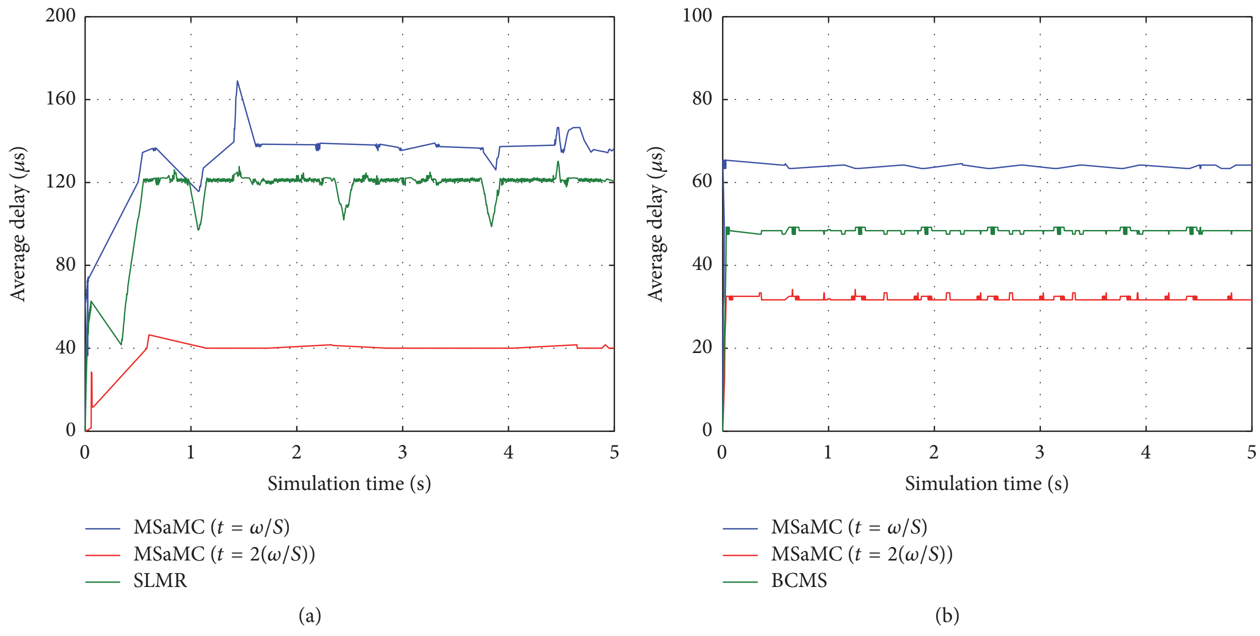


FIGURE 10: Average delay comparison.

BCMS algorithm. Experimental results show that MSaMC can achieve higher network throughput and lower average delay.

Notations

ω:	Multicast bandwidth request about data flow
b_i:	The occupied bandwidth of ith link
μ:	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
π:	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 (XDJK2016A011, XDJK2015C010, XDJK2015D023, and XDJK2016D047), 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 (2016M592619), and Chongqing Postdoctoral Science Foundation (XM2016002).

References

[1] J. Duan and Y. Yang, "Placement and Performance Analysis of Virtual Multicast Networks in Fat-Tree Data Center Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 3013–3028, 2016.

[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," *Acm Sigops Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[4] O. Fatmi and D. Pan, "Distributed multipath routing for data center networks based on stochastic traffic modeling," in *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control, ICNSC 2014*, pp. 536–541, USA, April 2014.

[5] Z. Guo, *On The Design of High Performance Data Center Networks*, Dissertations and Theses - Gradworks, 2014.

[6] H. Yu, S. Ruepp, and M. S. Berger, "Out-of-sequence prevention for multicast input-queuing space-memory-memory clos-network," *IEEE Communications Letters*, vol. 15, no. 7, pp. 761–765, 2011.

[7] G. Li, S. Guo, G. Liu, and Y. Yang, "Multicast Scheduling with Markov Chains in Fat-Tree Data Center Networks," in *Proceedings of the 2017 International Conference on Networking, Architecture, and Storage (NAS)*, pp. 1–7, Shenzhen, China, August 2017.

[8] X. Geng, A. Luo, Z. Sun, and Y. Cheng, "Markov chains based dynamic bandwidth allocation in diffserv network," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1711–1714, 2012.

[9] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, "The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem," *SIAM Review*, vol. 48, no. 4, pp. 681–699, 2006.

[10] T. G. Hallam, "David G. Luenberger: Introduction to Dynamic Systems, Theory, Models, and Applications. New York: John Wiley & Sons, 1979, 446 pp," *Behavioural Science*, vol. 26, no. 4, pp. 397-398, 1981.

[11] K. Hirata and M. Yamamoto, "Data center traffic engineering using Markov approximation," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 173–178, Da Nang, Vietnam, January 2017.

[12] D. Li, M. Xu, M.-C. Zhao, C. Guo, Y. Zhang, and M.-Y. Wu, "RDCM: Reliable data center multicast," in *Proceedings of the IEEE INFOCOM 2011*, pp. 56–60, China, April 2011.

[13] D. Li, H. Cui, Y. Hu, Y. Xia, and X. Wang, "Scalable data center multicast using multi-class bloom filter," in *Proceedings of the 2011 19th IEEE International Conference on Network Protocols, ICNP 2011*, pp. 266–275, Canada, October 2011.

[14] P. J. Cameron, "Notes on counting: An introduction to enumerative combinatorics," *Urology*, vol. 65, no. 5, pp. 898–904, 2012.

[15] R. Pastor-Satorras, M. Rubi, and A. Diaz-Guilera, "Statistical mechanics of complex networks," *Review of Modern Physics*, vol. 26, no. 1, 2002.

[16] A. P. Pynko, "Characterizing Belnap's logic via De Morgan's laws," *Mathematical Logic Quarterly*, vol. 41, no. 4, pp. 442–454, 1995.

[17] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proceedings of the 1st Workshop: Research on Enterprise Networking, WREN 2009, Co-located with the 2009 SIGCOMM Conference, SIGCOMM'09*, pp. 65–72, Spain, August 2009.

[18] C. Fraleigh, S. Moon, B. Lyles et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003.