Received: 18 February 2019 | Revised: 22 August 2019 | Accepted: 9 November 2019
DOI: 10.1002/cpe.5633

WILEY

RESEARCH ARTICLE

Priority-based online flow scheduling for network throughput maximization in software defined networking

Liang Liu[1,2] | Songtao Guo[1,3,4] | Guiyan Liu[1] | Yue Zeng[1]

[1]College of Electronic and Information Engineering, Southwest University, Chongqing, China
[2]School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China
[3]Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China
[4]College of Computer Science, Chongqing University, Chongqing, China

Correspondence
Songtao Guo, College of Computer Science, Chongiqng University, Chongqing 400044, China.
Email: songtao_guo@163.com

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61772432 and 61772433; Fundamental Research Funds for the Central Universities, Grant/Award Number: 2019CDYGZD004; Natural Science Foundation of Chongqing, Grant/Award Number: cstc2018jcyjA0644 and cstc2018jcyjA0743; Science and Technology Project Fund of Chongqing Education Commission, Grant/Award Number: KJQN201800640 and kj1502003

Summary

Data transmission in current networks is usually associated with strict priority enforcement for the purpose of quality of service (QoS). Under the case that priority flow requests are injected into the network sequentially without the information of future flow request arrivals, it is a challenging to achieve network throughput maximization for on-line flow requests under the joint constraints of the flow's priority, bandwidth demand, and resource capacity. Software Defined Networking (SDN) can effectively solve the flow scheduling equilibrium problem between the priority of dynamic flow requests and the maximization of network throughput. Therefore, in this paper, we study on-line flow request admission in SDN, the goal of which is to maximize the network throughput under the constraints of critical network bandwidth resources, flow priority, and bandwidth demands. First, we present the concept of flow routing cost and profit and a model to characterize the cost of using link resources and routing paths. Then, we propose an efficient on-line priority flow scheduling algorithm (OPFSA) to solve priority flow request scheduling problem and analyze the competitive ratio of OPFSA. Our on-line algorithm can reach throughput within $O(\frac{1}{\log n})$ of the highest possible throughput that can be achieved by an off-line algorithm, where n is the number of node in the network. Finally, experimental results demonstrate that compared with SHORTEST-SC, our proposed algorithm can enhance the cumulative bandwidth about 9% and 40% when general network size is 30 and 170 nodes, respectively, and improve the throughput about 25% in Fat-tree network when pod size is 4.

KEYWORDS

flow scheduling, on-line routing algorithm, priority flow request, software defined networking, throughput maximization

1 | INTRODUCTION

Software Defined Networking (SDN) has become an emerging network paradigm that separates networking control logic from underlying routers and switches into logically centralized controllers and provides network programmability.[1] This separation between the control plane and data plane simplifies policy enforcement, enhances network flow dynamic control and scheduling, and improves network bandwidth utilization in the network.[2-4] In an SDN, the controller has a centralized grasp of the network status information and the topology view of the entire network, which can dynamically change the transmission path of the service data flow according to the network status for maximizing network throughput. Thus, SDN has been regarded as a key technology of next generation network architectures. Thus far, it has been used in many large-scale data center networks (DCNs) and Internet-backbone networks, for example, Google's B4.[5]

Nowadays, a variety of applications running on the network such as on-line shopping, voice call streaming, and on-line gaming are greatly enriching people's daily lives, but they also bring vast challenges to network management. From the user's point of view, some of them hope they can get higher priority and high-quality and consistent flow services. From the perspective of internet service provider (ISP), according to the current traffic charging strategy, the greater the network throughput (the traffic exchanges over a period of time), the greater the ISP's revenue. ISPs always expect to maximize network throughput in order to obtain higher profits. Therefore, when a series of flow requests arrive, they expect their network infrastructure to first accept large flows to enter the network so as to achieve higher network throughput.

However, due to that the user's demands are various, it is possible that the user's flow bandwidth request is small but the priority is high such as driverless control flow and voice call flow. It is also possible that the user request has high flow bandwidth demand, but the priority of the flow is low such as file transfer flow. When these two kinds of flow requests arrive at the same time, it is nontrivial to schedule which flow request first; therefore, it is a promising topic to obtain a balance between flow priority scheduling and throughput maximization. Traditional networks are hard to satisfy the above expectations due to their poor flexibility and lack of fine-grained optimization of network bandwidth resources and traffic manipulation.[6]

With advantages of SDN's centralized global network control and fine-grained flow scheduling, some flow scheduling schemes have been presented. Most existing works on on-line routing problems considered either the link bandwidth constraint[7-9] or the priority flow queue scheduling.[10,11] Guo et al[7] investigated the traffic engineering in an SDN/OSPF hybrid network, and Lee et al[8] considered the influences of different application on the algorithm performance of bandwidth guarantee, but they did not take into account application priority. Yang[11] paid more attention to difference service of applications. He utilized the theory of flow queue scheduling to improve the weight-loop algorithm WRR and realized the goal of higher priority flow obtaining higher bandwidth guarantee. However, this method requires additional queue scheduling, which increases the workload of bandwidth guarantee system.

In addition, the joint optimization of flow-route choice in the control plane and flow-update scheduling in the data plane has also been gotten much attention. Xu et al[9] studied the real-time route update in SDN based on joint flow-route choice and flow-update scheduling. However, owing to the very high complexity of the method, it is difficult to implement in practice. Due to that the network resources in SDN are dynamically assigned, the availability of bandwidth-resources has great flexibility. If the network has a series of priority flow requests which arrive one after another without the information of future flow request arrivals, it is a challenge to determine which requests should be accepted and which requests should be rejected because the accepted requests will acquire bandwidth-resources and seriously hinder admission of later flow requests.

In this paper, by jointly considering the priority of each flow request and the bandwidth capacity of each link without the information of future flow request arrivals, we study on-line flow scheduling problem in SDN to maximize network throughput. First, we assign a cost function that is exponential in its current load to each edge. Second, we construct a subgraph from original network when one flow enters.[12] Then, we suppose that each flow request has a correlative profit and make the profit proportional to the product of bandwidth and priority. In this case, the higher the priority, the more important the flow. The throughput maximization means the ISP's profit maximization.

Furthermore, we propose an on-line priority flow scheduling algorithm (OPFSA) to solve the problem that determines which requests will be accepted and which ones will be rejected in on-line way. We suppose that flow requests arrive one after another without the information of future flow request arrivals. Each flow request indicates an explicit source and destination nodes, as well as the bandwidth requirements and priority. The choice that OPFSA either rejects or accepts the request depends on the allocation of the required bandwidth along flow request paths. Finally, we evaluate the performance of our proposed algorithm in terms of the competitive ratio, which is the ratio of the profit achieved by our on-line algorithm over other off-line algorithms.[13,14] To the best of our knowledge, it is the first work on on-line priority flow scheduling jointly considering the priority and bandwidth requirement of flows in a cost-profit way in SDN.

The remainder of this paper is organized as follows. Section 2 reviews the related, work and Section 3 introduces the system model and the problem definition. In Section 4, we propose the on-line algorithms and analyze the competitive ratio of flows routing by considering both the priority and bandwidth requirement of flows and disallowing no-rerouting in SDN. In Section 5, we evaluate the performance of the proposed algorithm, and Section 6 concludes this paper.

2 | RELATED WORK

Based on the advantages of SDN Technology, traffic engineering in SDN has attracted much attention from the academic community.[2,15]

Some works focused on flow splitting routing in the hybrid SDN.[7,16-22] Specifically, Xu et al[19] studied a throughput-maximization routing for incremental deployment of hybrid network. In order to reduce the complexity of routing management for arbitrary flow splitting, they solved the multicommodity h-splittable flow routing problem by a depth-first-search method and a randomized rounding mechanism, where $h \geq 1$. Nakasan et al[20] studied the multipath transport of splitting flows in SDN. However, in fact, some flows such as voice flows are not suitable for multipath transmission. Thus, in this paper, we do not consider the strategy of flow splitting. Tajiki et al[22] proposed the scheme which reassigns flows to the Label-Switched Paths (LSPs) to highly utilize the network resources in SDN-based MPLS hybrid networks. Then, they mathematically formulated two optimization problems and proposed a heuristic algorithm to improve the performance of the scheme. Because rerouting can degrade network performance, we do not consider rerouting in this paper. Gushchin et al[17] supposed that the routing of a flow request can be split into multiple paths and proposed a two-stage local optimization method to solve flow routing problem. Guo et al[7] explored the traffic engineering in an SDN/OSPF hybrid network. They changed the OSPF weights and flow splitting ratio of the SDN nodes so that the controller can arbitrarily split the flows which is coming into the SDN nodes; furthermore, they proposed an innovative algorithm called SOTE to achieve lower maximum link utilization.

Part of the existing works paid attention to flow scheduling in SDN, which involves network function virtualization(NFV).[15,23-27] Particularly, Huang et al[26] studied how to find a cost-optimal routing path which passes through the middleboxes in their orders in the service chain of the request, with the aim of maximizing the network throughput as well as subject to various bandwidth constraints such as Ternary Content

Addressable Memory (TCAM) capacity constraint in SDNs. Kanizo et al[23] investigated unicast routing problem on a set of requests in SDN. Li et al[24] focused on a congestion-free routing strategy in software defined data center networks by resorting to the global view of the data center network. They proposed a time-slot allocation scheme and computed the corresponding routing paths to conduct the coming packets. Huang et al[25] considered a joint optimization problem of middlebox selection and routing with the objective of maximizing the throughput in SDN and presented a polynomial algorithm based on the Markov approximation technique. Xu et al[28] proposed an algorithm of anycast routing (ARFU) based on the scheme of fully polynomial time approximation and a mechanism of single-source nonsplittable flow routing to obtain high throughput anycast routing. Cao et al[15] investigated the routing problem of flow policy-aware in SDN by supposing that the flow needs to pass a given network functions sequence. Li et al[18] solved flow requests resource provision problem which contains Virtualization Network Functions Service Chain by a linear programming technique with randomized rounding, the goal of which is to maximize the number of accepted flow requests in the whole cloud data center.

Different from the above works which either considered the link capacity, TCAM capacity, or other resource constraints for a single group of flow scheduling requests, in this paper, we study on-line routing problem of priority flow requests under the capacity at each link and the priority of each flow request in the case of no knowledge of future request arrivals in advance. Since the reception and rejection of flow is judged by online, in order to guide the network resource allocation of incoming flow requests, a good cost metric which accurately indicates the consumption and utilization of network resources is very significant. The essential requirement of this cost metric is that it can accurately describe the utilization and availability of resources. Due to that the marginal cost of using resources increases with the increase of incoming flows, the exponential function of particular resource's utilization is a better candidate for the cost metric. The similar cost metric function has been used in the networks with different types of resources such as on-line request routing in virtual circuit and ATM networks,[29,30] node energy of on-line data collection in wireless sensor and ad hoc networks,[31] unicast on-line flow request,[32] and multicast on-line flow request.[33] In addition, performing on-line routing in SDN simultaneously considers the priority of each flow request and the bandwidth of each link while it does not utilize the queue scheduling, which makes it more difficult to model the cost of routing path in SDN. Furthermore, we analyze the competitive ratio of our on-line algorithm by jointly taking into account the priority and the request bandwidth of each flow.

3 | NETWORK MODEL AND PROBLEM FORMULATION

In this section, we will introduce the network model and then give the problem formulation.

3.1 | Network model

SDN includes two main hardware facilities: SDN Controller (SDN-C) and SDN-enabled switch (SDN-S). The SDN-C is a logically centralized device which is in charge of control functions.[34] One or a few controllers can manage an SDN and route flow requests by forwarding rules which have been installed into the routing tables of SDN-S by SDN-C. In addition, the SDN-S is also responsible for allocating bandwidth on links along the routing paths in a network. The SDN-S constitutes the data plane of an SDN, and responsible for data forwarding. The logic for forwarding the packets is determined by the SDN-C and is implemented through the flow table at an SDN-S. We represent the data plane as an undirected graph $G = (V, E)$, where V denotes a set of n SDN-enabled switch nodes making up the network and E denotes the set of m links connecting the switches. The graph G is supposed to be mesh-connected, with multiple paths connecting each pair of nodes. Each link $e \in E$ is associated with a fixed bandwidth capacity u_e. The notations used in this paper are summarized in Table 1.

3.2 | User routing requests

We suppose that the time is separated into equal time slots and SDN controller schedules flow requests at the beginning of each time slot. Let $S(t)$ be the set of arrived flow requests in time slot t. Each flow request specifies a definite amount of bandwidth demand and priority requirement to route its traffic in G from a source switch to a destination switch. Let $f_i \in S(t)$ be the ith flow request, represented by a quintuple $f_i = (s_i, t_i, r_i, \beta_i, \rho(i))$, where s_i is its source node, t_i is its destination node, r_i is request bandwidth, β_i is the priority of flow request, which is defined as an integer, and $\rho(i)$ is the profit that the algorithm gains if it accepts to route this flow and is denoted by the product of r_i and β_i, ie, $\rho(i) = r_i\beta_i$. In practice, in order to facilitate the allocation of reasonable numerical priority for the request, we add a fixed integer $N \geq n$ for adjusting the magnitude of the flow priority β, which is described as follows:

$$\rho(i) = r_i N \beta_i. \tag{1}$$

Let $u_{\min}$ be the minimum link-bandwidth capacity among all links, ie, $u_{\min} = \min\{u_e | e \in E\}$, and let $\beta = max_i\{\beta(i)\}$ be the maximum priority of all flow requests. We suppose that the bandwidth demand r_i of each flow request is not less than one unit, ie, $r_i \geq 1$. In addition, similar to the work of Aspnes et al,[29] to design an on-line algorithm with logarithmic competition in networks, we suppose that

$$r_i \leq \frac{u_{\min}}{\log\alpha}, \tag{2}$$

TABLE 1 Notations table

Notation	Definition
G	Directed graph with node set V and link set E
n	Size of node set V
m	Size of directed link set E
u_e	Capacity of link e
f_i	The ith flow request
s_i	Ingress node for flow i
t_i	Egress node for flow i
r_i	The requested bandwidth for flow i
β_i	The priority for flow i
$\rho(i)$	Profit that the algorithm gains if accepting to route flow i
N	Parameter for adjusting the magnitude of flow priority
u_{min}	Minimum bandwidth capacity among links
β	Maximum priority of all flow requests
α	Parameter which is equal $2N\beta + 2$
p_i	Routing path for flow i
P_i	Routing path set for flow i
ω	Weight of link e
$\lambda_e(k)$	Relative load of the link e before the kth flow request is coming
$u_e(k)$	Residual bandwidth on link e before the kth flow request arriving
$c_e(k)$	Cost of using the resource at link e by flow request k

where $\alpha = 2N\beta + 2$. Inequality (2) implies that the requested bandwidth is obviously smaller than the capacity of minimum link in the network. In practice, the minimum link capacity in the network is usually much larger than the largest flow request bandwidth. For example, according to the work of Huang et al,[12] the bandwidth capacity of each link is at least 1000M. We assume $N = 1, \beta = 3, \alpha = 2N\beta = 6$; thus, $\frac{u_{\min}}{\log\alpha} \approx 387M$. In most cases, the request bandwidth is in the range of 0 to 100M. Even if it cannot be satisfied, in SDN, we can easily split a flow to multiple flows to satisfy this constraint. As long as there is a routing path p_i to meet the resource demands of a flow request and the path cost is less than the profit, this flow request will be accepted; otherwise, it will be rejected.

3.3 | Usage cost of link resources

Given $G = (V, E)$, let $\lambda_e(k)$ be the relative load of link $e \in E$ before the kth flow request is coming, which is defined as follows:

$$\lambda_e(k) = \sum_{e\in P_i, i<k} \frac{r_i}{\mu_e} = 1 - \frac{u_e(k)}{u_e}, \tag{3}$$

where P_i denotes the routing path of the ith request and $u_e(k)$ denotes the residual bandwidth on link $e \in E$, which is readily available because the controller can obtain global information for all links in SDN. In addition, the capacity constraints of links will be guaranteed, ie, $\lambda_e(k) \le 1$. Another constraint is that the algorithm must be on-line. In the case that the on-line algorithm does not know the information of the future flow request arrivals, the decision on accepting or rejecting a request f_i must be made at its start point. Once a flow is accepted and routed, this flow cannot be rerouted.

Furthermore, a metric is needed to model the usage costs of links. With the network accept more and more flow requests, the marginal cost of links resource usage significantly inflates with the increase of the load of links, that is to say, the margin cost of links resource usage is not linearly increased with the workload of links. As aforementioned in Section 1, it is a better candidate for the cost metric that an exponential function of a particular resource's utilization. Here, we use a similarly exponential function to model the cost $c_e(k)$ of using the bandwidth resource at each link e by flow request k, which is defined as follows:

$$c_e(k) = u_e(\alpha^{\lambda_e(k)} - 1) = u_e\left(\alpha^{1-\frac{u_e(k)}{u_e}} - 1\right). \tag{4}$$

We can observe from (4) that the larger the proportion of the occupied link resources over the all link resource in the network, the higher the risk that the resource capacity constraint will be violated, and the packet loss rate, delay, and jitter of the network will increase significantly as well. Therefore, when a flow request is accepted, we should use the links with lower usage cost of link resources (or the cheaper links) to receive the flow request.

3.4 | Problem formulation

We consider the case that a sequence of priority flow requests arrive at the network one after another without knowing the information of future flow request arrivals. Given undirected graph $G = (V, E)$, the network throughput maximization problem for on-line flow request by jointly considering the flow's priority and request bandwidth in G is to maximize the cumulative profit of flow requests which have been successfully accepted, subject to bandwidth constraints in G, ie, to accept as many flow requests which are weighted by their profits as possible. We use a binary variable x_{ip} to depict whether path p is chosen for flow i. When a request enters the system, it is either accepted ($x_{ip} = 1$) and carried on a single path or rejected ($x_{ip} = 0$). The maximization problem is formulated as the following mixed integer programming problem:

$$\begin{aligned} &\textit{Maximize} \quad \rho \overset{\Delta}{=} \sum_i \sum_{p \in P_i} r_i N \beta_i x_{ip} \\ &S.t. \begin{cases} C1: r_i \le \frac{u_{min}}{log\alpha}, & \forall i \\ C2: \sum\limits_{p \in P_i} x_{ip} \le 1, & \forall i \\ C3: \sum\limits_{p \in P_i, e \in p} r_i x_{ip} \le u_e, & \forall i, \forall e \\ C4: \sum\limits_{e \in p_i} \frac{r_i}{u_e} c_e(i) \le \rho(i), & \forall i \\ C5: x_{ip} \in \{0,1\}, & \forall i, \forall p \end{cases} . \end{aligned} \tag{5}$$

In (5), the objective is to maximize the amount of total accepted flow traffics. Constraint C1 specifies the requested link bandwidth constraint. Constraint C2 means the flow indivisibility constraint, ie, the flow is not splittable. Link capacity constraint C3 ensures that the traffic load on each link $e \in E$ should not exceed its capacity u_e, ie, congestion-free. Constraint C4 indicates the achievable profit constraint, ie, the achievable profit of each flow i should be no less than its routing path cost. Due to the on-line dynamic change of the network and the unpredictability of the flow, we will propose the on-line flow scheduling algorithm to solve the maximization problem.

4 | ON-LINE ALGORITHM AND COMPETITIVE RATIO ANALYSIS

In this section, we will present an on-line priority flow scheduling algorithm by solving the network throughput maximization problem and analyze the competitive ratio of the on-line algorithm.

4.1 | On-line priority flow scheduling algorithm

In this section, we consider a series of priority flow requests that arrive one after another and propose an on-line priority flow scheduling algorithm (OPFSA) based on the proposed usage cost model, which is described in Algorithm 1. In the case without the information of future flow request arrivals, we need to determine which requests are to be accepted and then find a routing path for each accepted flow.

Algorithm 1 On-line Priority Flow Scheduling Algorithm (OPFSA)

```
Input: software define networking $G = (V, E)$, the capacity $u_e$ of each edge in $G$, the $i$-th priority flow request $f_i = (s_i, t_i, r_i, \beta_i, \rho(i))$, and the fixed parameter $N$.
Output: Accept or reject the priority-flow request, if accepted, the algorithm will deliver a routing path for this priority-flow request.
 1: Remove links $e \in E$ whose residual bandwidth is less than $r_i$ from $G$ and obtain a subgraph $G' = (V', E')$;
 2: Assign each edge $e' \in E'$ with weight $\omega$ and get an edge-weighted subgraph $G' = (V', E'; \omega)$;
 3: Calculate the cost of each link by Equation (4);
 4: Look for a shortest path in $G'$ from $s_i$ to $t_i$ ;
 5: if $r_i \le \frac{u_{min}}{log\alpha}$ and $\sum_{e \in p_i} \frac{r_i}{u_e} c_e(i) \le \rho(i)$ then
 6:     accept request $f_i$ using $p_i$ as the routing path;
 7:     set: $\lambda_e(i+1) = \lambda_e(i) + \frac{r_i}{u_e} = 1 - \frac{u_e(i)}{u_e} + \frac{r_i}{u_e}$;
 8: else
 9:     Reject the request;
10: end if
```

More specifically, in Algorithm 1, we first remove links $e \in E$ whose residual bandwidth is less than r_i (line 1) and construct a subgraph $G' = (V', E')$ of G. This is because these links cannot meet the bandwidth requirement of f_i and they cannot play any role when the ith flow request is accepted. Then, we assign each edge $e' \in E'$ with weight $\omega = (\alpha^{\lambda_e(j)} - 1) = (\alpha^{1-\frac{u_e(j)}{u_e}} - 1)$ and get an edge-weighted subgraph $G' = (V', E'; \omega)$ (line 2). Thirdly, we calculate the cost for this edge, defined by $c_e(j) = u_e\omega = u_e(\alpha^{\lambda_e(j)} - 1) = u_e(\alpha^{1-\frac{u_e(j)}{u_e}} - 1)$ (line 3).

The corresponding shortest path of each priority flow request $f_i = (s_i, t_i, r_i, \beta_i, \rho(i))$ from node s_i to node t_i in the graph $G' = (V', E'; \omega)$ is always available. In fact, we may find an arbitrary path to route each accepted flow request. However, the length of a path to route the flow, ie, the sum of weighted links in the path, should be less than the profit when this flow is accepted. Therefore, the shortest path is the best choice for convenience (line 4).

After that, in order to accept high priority as many high bandwidth requests as possible, the following admission control policy will be adopted. In lines 5-10, if the length of weighted edges of a flow request f_i is greater than $\rho(i)$, then this flow request will be rejected. Otherwise, the algorithm will route f_i on the path with smallest weighted average of these costs. More precisely, if $e \in p_j$, then the contribution of link e to the cost of the path is computed by $\frac{r_i}{u_e} c_e(j)$. If there exists a path p_j whose cost is bounded by the profit $\rho(i)$, then this path is used to route the request. Otherwise, the request is rejected. That is to say, when a priority flow f_i arrives, if the Algorithm 1 finds a path p_j between s_i and t_i in G' and the cost of path p_j meets the following requirement:

$$\sum_{e \in p_j} \frac{r_i}{u_e} c_e(j) \le \rho(i), \tag{6}$$

then this flow request will be accepted.

In the following, we give the analysis of the complexity of Algorithm 1.

Theorem 1 (Time complexity of OPFSA). *The time complexity of OPFSA is* $O(|E| + |V| log(|V|))$.

Proof. Given $G = (V, E)$, in OPFSA, when a flow arrives, we first need to construct an edge-weighted graph and the complexity is $O(|V| + |E|)$. Then, we need to calculate the relative costs on links, and the complexity is also $O(|V| + |E|)$. Finally, we need to execute the Shortest Path Algorithm, and the time-complexity is $O(|E| + |V| log(|V|))$. Therefore, the time-complexity of OPFSA is $O(|E| + |V| log(|V|))$. □

4.2 | Competitive Ratio Analysis

In this section, we give the analysis of the OPFSA. Firstly, we prove that the Algorithm 1 does not break the capacity constraint of link. Secondly, we present the competitive ratio between on-line algorithm and optimal off-line algorithm.

In general, the capacity constraint is always satisfied. This is because when an edge is closely saturated, its cost will become very high so that it will never be used for routing. Let A denote the set of indices of priority flow requests that are satisfied by OPFSA, ie, $A = \{i : p_i \ne \phi\}$. Then, the following lemma holds.

Lemma 1. *Given an SDN* $G = (V, E)$ *with link bandwidth capacity* u_e *for each link* $e \in E$, *all edges will not violate the link capacity constraint, that is to say,* $A = \{i : p_i \ne \phi\}$, *for each link* $e \in E$, $\sum_{i \in A, e \in p_i} r_i \le u_e$.

Proof. We employ the proof method by contradiction. Let f_j be the first flow on edge e that makes the relative load on edge e exceed 1. In other words, f_j is the first flow on edge e that makes the available capacity of edge e less than r_j. According to the definition of relative load, we have $\lambda_e(j) > 1 - \frac{r_j}{\mu_e}$. As aforementioned, we assume that $r_j \le \frac{u_{\min}}{\log\alpha}$, and we can conclude that $\frac{r_j}{\mu_e} \le \frac{u_{\min}}{\mu_e \log\alpha}$. Therefore, $\lambda_e(j) > 1 - \frac{r_j}{\mu_e} \ge 1 - \frac{u_{\min}}{\mu_e \log\alpha} \ge 1 - \frac{1}{\log\alpha}$. Since $c_e(j) = u_e(\alpha^{\lambda_e(j)} - 1)$, we can get

$$\begin{aligned} c_e(j)/u_e &= \alpha^{\lambda_e(j)} - 1 \\ &\ge \alpha^{1 - \frac{1}{\log\alpha}} - 1 \\ &= \alpha/2 - 1 = \beta N. \end{aligned} \tag{7}$$

According to inequality (7) and equation (1), we have

$$\frac{r_j}{u_e} c_e(j) \ge \beta N r_j \ge \rho(j). \tag{8}$$

Hence, due to that the routing cost of the request f_j is greater than the profit, we can obtain that the request could not use link e. □

In the following, we will use the sum of link costs to lower the bound of total profit by our algorithm.

Lemma 2. *Given an SDN* $G = (V, E)$ *with link bandwidth capacity* u_e *for each link* $e \in E$, *let* k *be the index of last priority flow request; then, we have*

$$2 \log\alpha \sum_{j \in A} \rho(j) \ge \sum_{e \in p_k} c_e(k + 1). \tag{9}$$

Proof. We will employ the induction method to complete the proof. When k is equal to 0, because the values of inequality (9) are 0, the inequality (9) is always true. In addition, the refused priority request flows do not change the value of inequality (9). Therefore, for any accepted request f_j, the proof of inequality (9) can be transformed into the proof of the following inequality:

$$2\rho(j)\log\alpha \geq \sum_{e\in p_j}(c_e(j+1)-c_e(j)). \tag{10}$$

According to the definition of link usage cost, for each link $e \in p_j$, we have

$$\begin{aligned} c_e(j+1)-c_e(j) &= u_e\left(\alpha^{\lambda_e(j)+\frac{r_j}{u_e}}-\alpha^{\lambda_e(j)}\right) \\ &= u_e\alpha^{\lambda_e(j)}\left(\alpha^{\frac{r_j}{u_e}}-1\right) \\ &= u_e\alpha^{\lambda_e(j)}\left(2^{\log\alpha^{\frac{r_j}{u_e}}}-1\right)\Big). \end{aligned} \tag{11}$$

By assumption (2), we have $r_j \leq \frac{u(e)}{\log\alpha}$. Furthermore, we can get $0 \leq \frac{r_j}{u_e}\log\alpha = \log\alpha^{\frac{r_j}{u_e}} \leq 1$. Since $2^x - 1 \leq x$ for $0 \leq x \leq 1$, we have $2^{\log\alpha^{\frac{r_j}{u_e}}} - 1 \leq \log\alpha^{\frac{r_j}{u_e}}$; then, we can conclude

$$\begin{aligned} c_e(j+1)-c_e(j) &= u_e\alpha^{\lambda_e(j)}\left(2^{\log\alpha^{\frac{r_j}{u_e}}}-1\right)\Big) \\ &\leq u_e\alpha^{\lambda_e(j)}\log\alpha^{\frac{r_j}{u_e}} \\ &= \alpha^{\lambda_e(j)}r_j\log\alpha \\ &= (c_e(j)/u_e+1)r_j\log\alpha \\ &= \left(c_e(j)\frac{r_j}{u_e}+r_j\right)\log\alpha. \end{aligned} \tag{12}$$

The last equality holds because $\alpha^{\lambda_e(j)} = c_e(j)/u_e + 1$ by equation (4). The change of the link cost c_e is due to the fact that the request f_j is accepted. Thus, when request f_j arrives, the total cost of all links can be given by

$$\begin{aligned} \sum_{e\in p_j}(c_e(j+1)-c_e(j)) &\leq \sum_{e\in p_j}\left(c_e(j)\frac{r_j}{u_e}+r_j\right)\log\alpha \\ &\leq \log\alpha\left(\sum_{e\in p_j}c_e(j)\frac{r_j}{u_e}+\sum_{e\in p_j}r_j\right) \\ &\leq \log\alpha\left(\rho(j)+\sum_{e\in p_j}r_j\right) \\ &\leq \log\alpha\left(\rho(j)+|p_j|r_j\right) \\ &\leq 2\rho(j)\log\alpha. \end{aligned} \tag{13}$$

The last two inequalities hold because the maximum value of $|p_j|$ is $n-1$ and $|p_j|r_j \leq (n-1)r_j \leq r_jN\beta_j = \rho(j)$. □

In the following, we will present a cost lower bound on the route-path of the last priority flow request which is rejected by the on-line algorithm but is routed by the optimal off-line algorithm.

Lemma 3. *We use Q to represent the set of the priority flow requests which are accepted by an optimal off-line algorithm, however, rejected by the Algorithm 1, and let $l = max\{Q\}$ be the last flow request in Q. We have*

$$\sum_{j\in Q}\rho(j) \leq \sum_e c_e(l). \tag{14}$$

Proof. We use p'_j to represent the path which is used by the optimal off-line algorithm to route f_j for each $f_j \in Q$, the cost $c_e(j)$ of routing the flow f_j that is rejected by the on-line algorithm is monotonically increasing with the number of flows, which implies

$$\begin{aligned} \rho_j &\le \sum_{e\in p'_j} c_e(j)\frac{r_j}{u_e} \\ &\le \sum_{e\in p'_j} c_e(l)\frac{r_j}{u_e}. \end{aligned} \tag{15}$$

Summing the costs of all flow requests f_j for all $j \in Q$, we can get

$$\begin{aligned} \sum_{j\in Q}\rho(j) &\le \sum_{j\in Q}\sum_{e\in p'_j} c_e(l)\frac{r_j}{u_e} \\ &\le \sum_e c_e(l)\sum_{j\in Q, e\in p'_j}\frac{r_j}{u_e} \\ &\le \sum_e c_e(l). \end{aligned} \tag{16}$$

The last inequality means that the off-line algorithm cannot violate the relative load of link e at any time, ie, $\sum_{j\in Q, e\in p'_j}\frac{r_j}{u_e} \le 1$. □

According to Lemma 1 and Lemma 2, we can get the following theorem.

Theorem 2. *Given an SDN $G = (V, E)$, we suppose that the capacity of each link $e \in E$ is u_e and there is a sequence of priority flow requests $f_i = (s_i, t_i, r_i, \beta_i, \rho(i))$, where $i = 1, 2, \dots, n$, arriving one after another without the information of future flow request arrivals, the on-line OPFSA never violates the capacity constraint, and will obtain at least $1/2log(2a)$-fraction of the cumulative profit by the optimal off-line algorithm for the network throughput maximization problem by jointly considering bandwidth and priority demand.*

Proof. The cumulative profit of optimal off-line algorithm is bounded by

$$\sum_{i\in A}\rho(i) + \sum_{j\in Q}\rho(j). \tag{17}$$

According to Lemma 3, the upper bound of this profit can be given by

$$\sum_{i\in A}\rho(i) + \sum_e c_e(l). \tag{18}$$

Because $c_e(k+1)$ represents the last cost of edge e and the cost of link is increasing monotonically, we have $c_e(k+1) \ge c_e(l), \forall e \in E$. Together with Lemma 2, the profit by the optimal off-line algorithm will be bounded by

$$\begin{aligned} \sum_{i\in A}\rho(i) + \sum_e c_e(l) &\le \sum_{i\in A}\rho(i) + 2\log\alpha\sum_{i\in A}\rho(i) \\ &\le (2\log\alpha + 1)\sum_{i\in A}\rho(i) \\ &\le 2\log(2\alpha)\sum_{i\in A}\rho(i). \end{aligned} \tag{19}$$

According to inequality (14) and (19), we can get

$$\sum_{i\in A}\rho(i) + \sum_{j\in Q}\rho(j) \le 2\log(2\alpha)\sum_{i\in A}\rho(i), \tag{20}$$

that is,

$$\sum_{i\in A}\rho(i) \ge \frac{1}{2\log(2\alpha)}\left(\sum_{i\in A}\rho(i) + \sum_{j\in Q}\rho(j)\right). \tag{21}$$

Therefore,

$$\frac{|A|}{|A| \cup |Q|} \geq \frac{1}{2\log(2\alpha)}. \tag{22}$$

Note that the competitive ratio of OPFSA is determined by parameter α. When $\alpha = n$, the competitive ratio of OPFSA is $O(logn)$. □

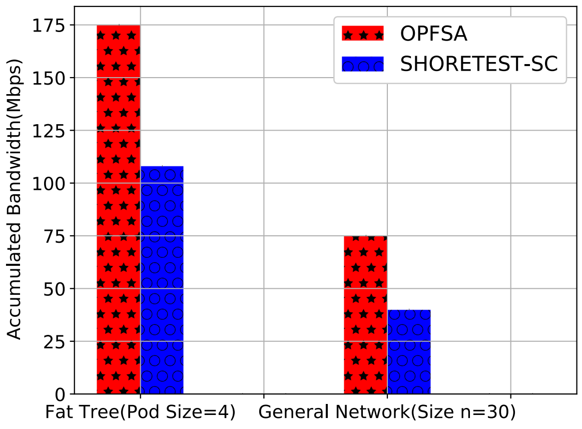

FIGURE 1 The cumulative bandwidth delivered by different algorithms for fat tree(4 pods) and general network(30 nodes) respectively in mininet, with parameters $\alpha = 2n + 2$ and $\beta_i = 1, \rho = n$, assuming that there are 20 flow requests

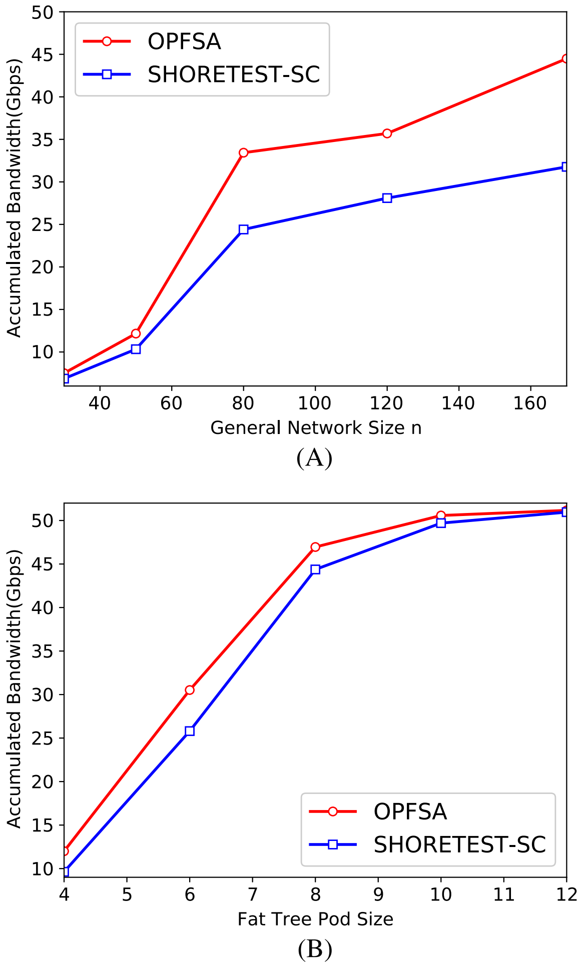

FIGURE 2 The cumulative bandwidth delivered by different algorithms for different network structure and network size n, while parameters $\alpha = 2n + 2$ and $\beta_i = 1, \rho = n$, assuming that there are 2000 flow requests. A, Cumulative bandwidth by OPFSA and SHORTEST-SC in general networks; B, Cumulative bandwidth by OPFSA and SHORTEST-SC in Fat-tree networks

5 | PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed on-line OPFSA through extensive experiments. We first describe the experimental environment and the compared algorithms and then explore the impact of important parameters on the performance of the proposed algorithm.

5.1 | Environment settings

Due to that the OPFSA is served for the online flow scheduling, the key idea of the OPFSA is to use the value of an exponential function to represent the usage cost (weight) of link in the network, ie, the value of the exponential function increases with the increase of the links load. Thus, we compare the proposed algorithm with an on-line heuristic algorithm, SHORTEST-SC. Especially, the SHORTEST-SC firstly removes the links from the network which do not have sufficient capacity to route the incoming request f_i and then assigns each link the same weight such as 1. SHORTEST-SC will always search for the shortest route path of request f_i in the network. Then, we evaluate the impact of some parameters on the weights of links, which results in the influence on the proposed algorithm OPFSA.

In order to evaluate the performance of the proposed algorithm in the case of large-scale flow requests, we will use the average of the results out of 20 network instances with 20 different sequences of 30 000 flow requests as the value in each figure. It is difficult to run such a large-scale simulation in mininet. However, in order to prove the effectiveness of the proposed algorithm in mininet, we firstly implement a small-scale simulation of the algorithms. We construct two small-scale network topologies which has 4 pods in fat tree and 30 nodes in general network, respectively. The bandwidth capacity u_e of each link $e \in E$ is randomly distributed between 100 Mbps to 1000 Mbps. We utilize "iperf" to generate 20 flows randomly, and the size of each flow is set as 1 to 50 M randomly. The data are the average of 10 network instances for each topology. The experimental results are shown in Figure1. We can conclude that the OPFSA algorithm is obviously superior to the SHORTEST-SC. Then, we implement the algorithms in Python-simulation under large-scale network and traffic conditions, and these algorithms are performed on a computer with a core i5-2350 M CPU, 8GB of RAM. We consider the general networks with 30, 50, 80, 120, and 170 nodes, respectively, and the Fat-tree data center networks (DCNs) with 4, 6, 8, 10, and 12 pods, respectively. For each network size, we randomly

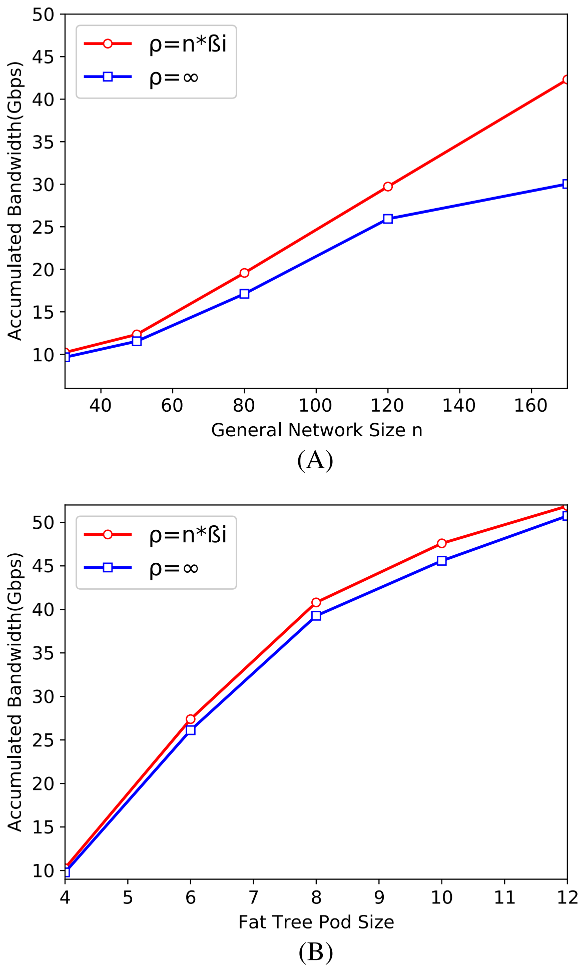

FIGURE 3 The cumulative bandwidth delivered by OPFSA of the general network and Fat-tree network with profit constraints $\rho = n\beta_i$ and without profit constraints $\rho = \infty$ for a monitoring period consisting of 2000 flow requests when $\alpha = 2n\beta + 2$ and every flow priority β_i is randomly 1, 3 or 5. A, Impact of parameters ρ and α on OPFSA

generate some links, of which the total number is the square of the number of nodes divided by 4. Without loss of generality, we set $N = n$. In addition, 20 network instances are randomly generated using the Python Network Library. The bandwidth capacity u_e of each link $e \in E$ is randomly distributed between 1000 Mbps to 10000 Mbps.[12,35] We randomly assign bandwidth demand r_i from 1 Mbps to 50 Mbps for each flow request f_i.

The priority β_i of each request is randomly assigned an integer of 1, 3, or 5. In the experiments, we use the average of 20 network instances with 20 different series of 30 000 flow requests. It is noted that Assumption 2 is always satisfied for any flow request, ie, $r_i \leq \frac{u_{\min}}{\log\alpha}$, where $u_{\min} = 1000$.

5.2 | Comparison of different algorithms

In this section, we will evaluate the performance on cumulative bandwidth of the proposed on-line algorithm OPFSA against the SHORTEST-SC algorithm with the fixed flow priority, that is, $\beta_i = 1$, and other two parameters $\alpha = 2n + 2$ and $\rho = n$ change with network size. Figure 2 plots the performance of the two algorithms, and it can be seen that the proposed OPFSA always outperforms SHORTEST-SC. Specifically, Figure 2A indicates that OPFSA delivers 9% and 40% more cumulative bandwidth than the SHORTEST-SC algorithm when general network has 30 and 170 nodes, respectively, and Figure2B indicates that OPFSA delivers 25% more cumulative bandwidth than the SHORTEST-SC algorithm when Fat-tree pod size is 4.

In addition, Figure 2A shows that with the expansion of network size, the gap of the cumulative bandwidth between the OPFSA and the SHORTEST-SC becomes larger and larger. This is because with the growth of the network size n, the number of flows that are accepted to enter the network is also increasing, but the whole capacity of the network is gradually saturated. On the contrary, Figure2B shows that with the growth of the number of pods in Fat-tree network, the gap of the cumulative bandwidth between the OPFSA and SHORTEST-SC becomes smaller and smaller. This is because with the growth of the pod number, the number of links in the Fat-tree increases significantly, which makes the capacity of the whole network far from saturation. Therefore, all request flows are accepted to enter the network in both OPFSA and SHORTEST-SC.

5.3 | Impact of parameters ρ and α on OPFSA

In this subsection, we mainly consider the impact of two parameters ρ and α on our proposed Algorithm OPFSA.

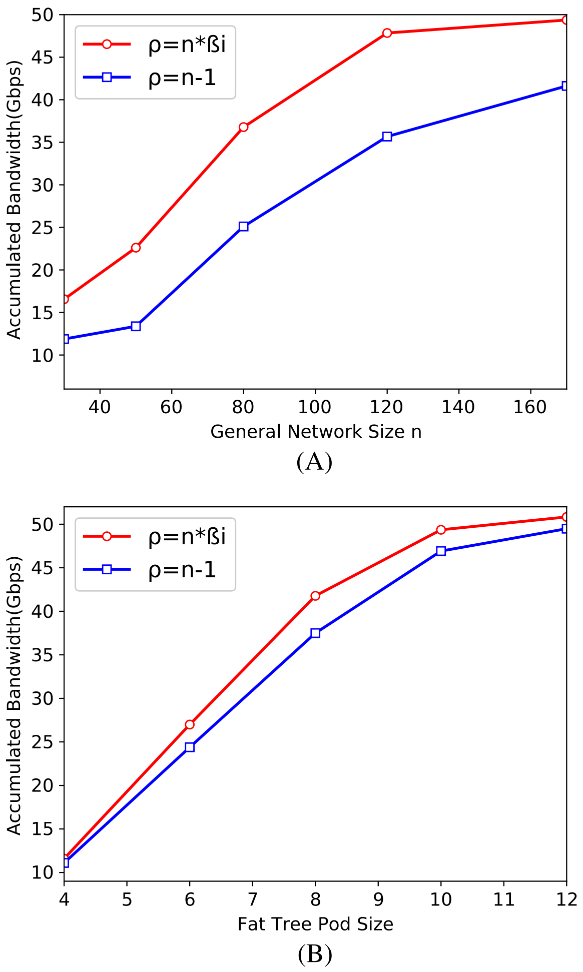

FIGURE 4 The cumulative bandwidth delivered by OPFSA of the general network and Fat-tree network with profit constraints $\rho = n\beta_i$ and $\rho = n - 1$ respectively for a monitoring period consisting of 2000 flow requests when $\alpha = 2n\beta + 2$ and every flow priority β_i is randomly 1, 3, or 5. A, Accumulated bandwidth by OPFSA with $\rho = n\beta_i$ and $\rho = n - 1$, respectively, in general network; B, Accumulated bandwidth by OPFSA with $\rho = n\beta_i$ and $\rho = n - 1$, respectively, in Fat-tree network

Firstly, we evaluate the impact of the admission control profit parameter ρ on the performance of OPFSA in terms of the cumulative bandwidth. The goal of admission control is to prevent the acceptance of flow request with large costs relative to profit. When $\rho < \infty$ is bounded, even if there are sufficient available link resources for a flow request, this flow request is very likely to be rejected.

Figure 3 shows the performance of OPFSA with and without profit constraint, and it can be seen that the OPFSA with admission control significantly outperforms that without admission control in two network structures. To be specific, for the OPFSA, the performance gap of OPFSA with profit constraint and without constraint profit in general network becomes larger and larger as shown in Figure 3A with the growth of network size *n*, eg, the cumulative bandwidth ratio which delivered by OPFSA with and without the profit constraint increases from 1.05 when $n = 30$ to 1.4 when $n = 170$. For the Fat-tree network, the performance gap of OPFSA with and without the admission control profit constraint is steady shown in Figure 3B. The difference on the cumulative bandwidth just increases from 500 Mbps to 1100 Mbps when the pod size grows from 4 to 12.

Secondly, we evaluate the influence of the admission control profit parameter with $\rho = n\beta_i$ and $\rho = n - 1$, respectively, on the performance of OPFSA in terms of the cumulative bandwidth. Figure 4 shows the performance of OPFSA with $\rho = n\beta_i$ and $\rho = n - 1$ constraint, respectively, and it can be seen that OPFSA with $\rho = n\beta_i$ admission control significantly outperforms $\rho = n - 1$ admission control. Specifically, OPFSA with $\rho = n\beta_i$ profit constraint delivers 28% more cumulative bandwidth than $\rho = n - 1$ profit constraint. Similarly, in Fat-tree network, OPFSA with $\rho = n\beta_i$ profit constraint delivers 4% more the cumulative bandwidth than with $\rho = n - 1$ profit constraint.

Then, we analyze the impact of parameters α on the cumulative bandwidth of OPFSA, by varying α from $2^1(n\beta_i + 1)$ to $2^5(n\beta_i + 1)$ while setting $\rho = n\beta_i$. Figure 5 plots the cumulative bandwidth of OPFSA by varying the value of α. It can be seen from Figure 5A that when the value of α is larger, the cumulative bandwidth delivered by OPFSA for different general network sizes *n* becomes less . For example, OPFSA with $2^1(n\beta_i + 1)$ delivers 26% more the cumulative bandwidth than with $2^5(n\beta_i + 1)$. Similarly, in Fat-tree network, OPFSA with $2^1(n\beta_i + 1)$ delivers 30% more the cumulative bandwidth than with $2^5(n\beta_i + 1)$. It also can be noted that the cumulative bandwidth gap of OPFSA under different values of α is steady with the growth of network size *n*.

Finally, we explore the impact of parameters α on the flow rejection ratio of the proposed algorithm by varying α from $2^1(n\beta_i + 1)$ to $2^5(n\beta_i + 1)$ and let $\rho = n\beta_i$. Figure 6 plots the flow rejection ratio of OPFSA by varying the value of α. It can be noted from Figure 6A that when the value of α is more larger, the more the flow rejection ratio delivered by OPFSA for different general network size *n*. For instance, OPFSA with $2^5(n\beta_i + 1)$

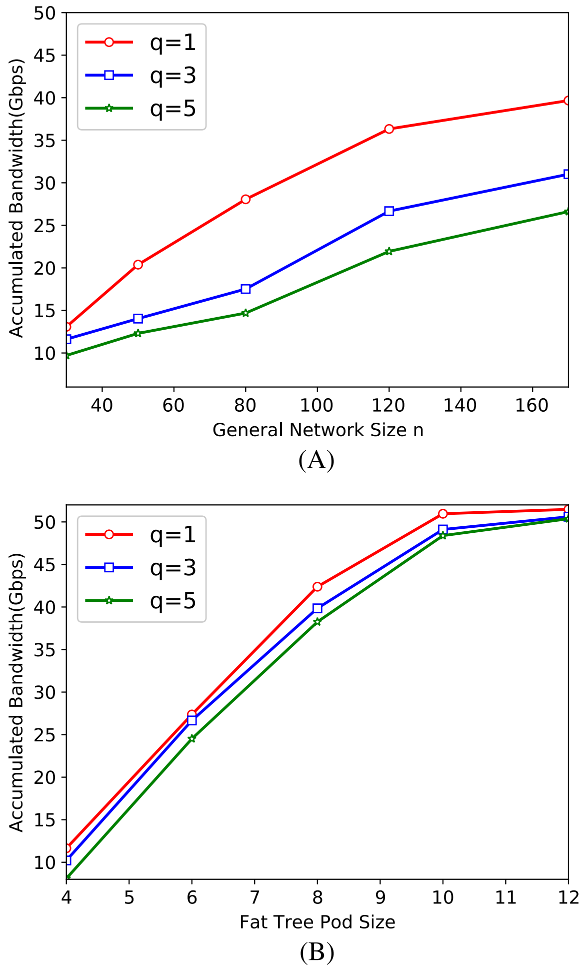

FIGURE 5 The cumulative bandwidth of OPFSA of the general network and Fat-tree network with different network size *n* by varying α, when $\rho = n\beta_i$ and β_i=1, 3, or 5 randomly. A, The cumulative bandwidth of OPFSA in general network by varying $\alpha = 2^q(n\beta + 1)$ with *q*=1, 3, and 5, respectively; B, The cumulative bandwidth of OPFSA in Fat-tree network by varying $\alpha = 2^q(n\beta + 1)$ with *q*=1, 3, and 5, respectively

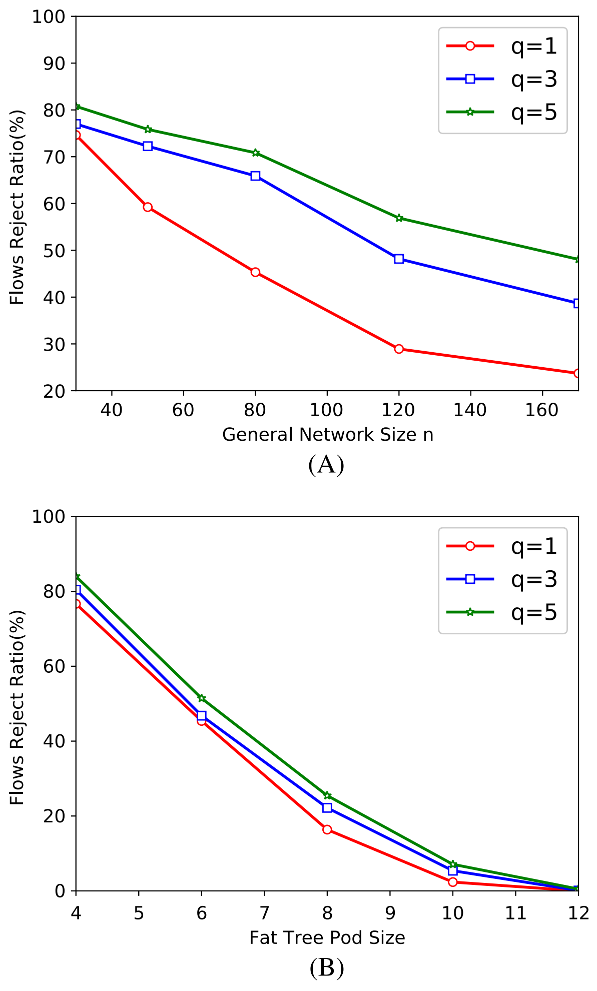

FIGURE 6 The flow rejection ratio of OPFSA of the general network and Fat-tree network with different network size *n* by varying α, when $\rho = n\beta_i$ and β_i=1, 3, or 5 randomly. A, The flow rejection ratio of OPFSA in general network by varying $\alpha = 2^q(n\beta + 1)$ with *q*=1, 3, and 5, respectively; B, The flow rejection ratio of OPFSA in Fat-tree network by varying $\alpha = 2^q(n\beta + 1)$ with *q*=1, 3, and 5, respectively

delivers 8% more flow rejection ratio than with $2^1(n\beta_i + 1)$. It is the same in Fat-tree network that OPFSA with $2^5(n\beta_i + 1)$ delivers 4% more flow rejected ratio than with $2^1(n\beta_i + 1)$. It can also be noticed that the flow rejection ratio gap of OPFSA based on different values of α is steady with the growth of network size *n*.

6 | CONCLUSION

Priority flow scheduling in current networks is very important for improving quality of service (QoS). When a series of priority flow requests are injected into the network one by one without the arrival information of future traffic requests, it is challenging to achieve the flow scheduling equilibrium between the priority of dynamic flow request and the maximization of network throughput. In this paper, we study dynamic priority flow route scheduling in SDN under the constraints of link capacity, flow request priority, and flow bandwidth demand. We present a cost model to reflect the usage costs of link resources and routing path and propose an efficient on-line priority flow scheduling algorithm (OPFSA) by solving flow throughput maximization problem. Then, we analyze the competitive ratio of OPFSA theoretically. Finally, we evaluate the performance of the proposed algorithm through experiments. The results show that the proposed algorithm OPFSA can improve the cumulative bandwidth about 9% and 40% compared with the SHORTEST-SC when general network size is 30 and 170 nodes, respectively, and in Fat-tree network, as well as network throughput about 25% compared with SHORTEST-SC when the Fat-tree pod size is 4.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under grants 61772432 and 61772433, the Fundamental Research Funds for the Central Universities under grant 2019CDYGZD004, the Natural Science Foundation of Chongqing under grants cstc2018jcyjA0644 and cstc2018jcyjA0743, and the Science and Technology Project Fund of Chongqing Education Commission under grants KJQN201800640 and kj1502003.

ORCID

Songtao Guo https://orcid.org/0000-0001-6741-4871

REFERENCES

1. Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):69-74.
2. Agarwal S, Kodialam M, Lakshman TV. Traffic engineering in software defined networks. Paper presented at: IEEE Conference on Computer Communications; 2013; Turin, Italy.
3. Katta N, Alipourfard O, Rexford J, Walker D. Infinite cacheflow in software-defined networks. Paper presented at: 3rd Workshop on Hot Topics in Software Defined Networking; 2014; Chicago, IL.
4. Kreutz D, Ramos FMV, Verssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. *Proc IEEE*. 2015;103(1):14-76.
5. Jain S, Kumar A, Mandal S, et al. B4: experience with a globally-deployed software defined WAN. *SIGCOMM Comput Commun Rev* 2013; 43(4):3-14.
6. Nunes BAA, Mendonca M, Nguyen XN, Obraczka K, Turletti T. A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Commun Surv Tutor*. 2014;16(3):1617-1634.
7. Guo Y, Wang Z, Yin X, Shi X, Wu J. Traffic engineering in SDN/OSPF hybrid network. Paper presented at: IEEE 22nd International Conference on Network Protocols; 2014; Raleigh, NC.
8. Lee J, Turner Y, Lee M, et al. Application-driven bandwidth guarantees in datacenters. *SIGCOMM Comput Commun Rev* 2014;44(4):467-478.
9. Xu H, Yu Z, Li X-Y, Qian C, Huang L, Jung T. Real-time update with joint optimization of route selection and update scheduling for SDNs. Paper presented at: IEEE 24th International Conference on Network Protocols; 2016; Singapore.
10. Yang X, Vaidya NH. Priority scheduling in wireless ad hoc networks. Paper presented at: 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing; 2002; Lausanne, Switzerland.
11. Yang S. Research on dynamic bandwidth allocation algorithm for multi-service networks. *Commun Tech*. 2011;44(4):82-84.
12. Huang M, Liang W, Xu Z, Xu W, Guo S, Xu Y. Dynamic routing for network throughput maximization in software-defined networks. Paper presented at: 35th Annual IEEE International Conference on Computer Communications; 2016; San Francisco, CA.
13. Buchbinder N, Naor JS. *The Design of Competitive Online Algorithms aia a Primal-Dual Approach*. Hanover, MA: Publishers Inc; 2009. Foundations and Trends in Theoretical Computer Science.
14. Sleator DD, Tarjan RE. Amortized efficiency of list update and paging rules. *Commun ACM*. 1985;28(2):202-208.
15. Cao Z, Kodialam M, Lakshman TV. Traffic steering in software defined networks. *ACM SIGCOMM Comput Commun Rev*. 2014;44(4):312-418.
16. Zhang J, Xi K, Luo M, Chao HJ. Load balancing for multiple traffic matrices using sdn hybrid routing. Paper presented at: IEEE 15th International Conference on High Performance Switching and Routing; 2014; Vancouver, Canada.
17. Gushchin A, Walid A, Tang A. Scalable routing in SDN-enabled networks with consolidated middleboxes. Paper presented at: ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization; 2015; London, United Kingdom.
18. Li Y, Phan LTX, Loo BT. Network functions virtualization with soft real-time guarantees. Paper presented at: 35th Annual IEEE International Conference on Computer Communications; 2016; San Francisco, CA.
19. Xu H, Li X-Y, Huang L, Deng H, Huang H, Wang H. Incremental deployment and throughput maximization routing for a hybrid SDN. *IEEE/ACM Trans Netw* 2017;25(3):1861-1875.
20. Nakasan C, Ichikawa K, Iida H, Uthayopas P. A simple multipath OpenFlow controller using topology-based algorithm for multipath TCP. *Concurrency Computat Pract Exper*. 2017;29(13).
21. Tanha M, Sajjadi D, Ruby R, Pan J. Traffic engineering enhancement by progressive migration to SDN. *IEEE Commun Lett*. 2018;22(3):438-441.
22. Tajiki MM, Akbari B, Mokari N, Chiaraviglio L. SDN-based resource allocation in MPLS networks: A hybrid approach. *Concurrency Computat Pract Exper*. 2019;31(8).
23. Kanizo Y, Hay D, Keslassy I. Palette: Distributing tables in software-defined networks. Paper presented at: IEEE Conference on Computer Communications; 2013; Turin, Italy.
24. Li Y, Li W, Chen H, Guo D, Zhang T, Qu T. Congestion-free routing strategy in software defined data center networks. *Concurrency Computat Pract Exper*. 2015;27(18):5735-5748.
25. Huang H, Guo S, Wu J, Li J. Joint middlebox selection and routing for software-defined networking. Paper presented at: IEEE International Conference on Communications; 2016; Kuala Lumpur, Malaysia.
26. Huang M, Liang W, Xu Z, Guo S. Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes. *IEEE Trans Network Serv Manag*. 2017;14(3):631-645.
27. Jia M, Liang W, Huang M, Xu Z, Ma Y. Routing cost minimization and throughput maximization of NFV-enabled unicasting in software-defined networks. *IEEE Trans Netw Serv Manag*. 2018;15(2):732-745.
28. Xu H, Li X, Huang L, Wang J, Leng B. High-throughput anycast routing and congestion-free reconfiguration for SDNs. Paper presented at: IEEE/ACM 24th International Symposium on Quality of Service; 2016; Beijing, China.
29. Aspnes J, Azar Y, Fiat A, Plotkin S, Waarts O. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J ACM*. 1997;44(3):486-504.
30. Plotkin S. Competitive routing of virtual circuits in ATM networks. *IEEE J Sel Areas Communi*. 1995;13(6):1128-1136.
31. Liang W, Liu Y. Online data gathering for maximizing network lifetime in sensor networks. *IEEE Trans Mob Comput*. 2007;6(1):2-11.
32. Kar K, Kodialam M, Lakshman TV, Tassiulas L. Routing for network capacity maximization in energy-constrained ad-hoc networks. Paper presented at: 22nd Annual Joint Conference of the IEEE Computer and Communications Societies; 2003; San Francisco, CA.

33. Liang W, Guo X. Online multicasting for network capacity maximization in energy-constrained ad hoc networks. *IEEE Trans Mob Comput*. 2006;5(9):1215-1227.
34. Gude N, Koponen T, Pettit J, et al. NOX: towards an operating system for networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(3):105-110.
35. Kreutz D, Ramos FMV, Verssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. *Proc IEEE*. 2015;103(1):14-76.

How to cite this article: Liu L, Guo S, Liu G, Zeng Y. Priority-based online flow scheduling for network throughput maximization in software defined networking. *Concurrency Computat Pract Exper*. 2020;32:e5633. https://doi.org/10.1002/cpe.5633