Computer Networks 132 (2018) 26–39

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Online unicasting and multicasting in software-defined networks

Meitian Huang[a], Weifa Liang[a,*], Zichuan Xu[b], Wenzheng Xu[c], Song Guo[d], Yinlong Xu[e]

[a] Research School of Computer Science, The Australian National University, Canberra, ACT, 0200, Australia
[b] School of Software, Dalian University of Technology, Dalian, 116024, PR China
[c] College of Computer Science, Sichuan University, Chengdu, 610065, PR China
[d] Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[e] School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230026, PR China

ARTICLE INFO

Article history:
Received 10 July 2017
Revised 19 November 2017
Accepted 24 December 2017
Available online 28 December 2017

Keywords:
Dynamic unicast and multicast request admissions
Network resource allocation
Online algorithms
Competitive ratio analysis
Ternary content addressable memory (TCAM)
Software-defined networks
Combinatorial optimization

ABSTRACT

Software-Defined Networking (SDN) has emerged as the paradigm of the next-generation networking through separating the control plane from the data plane. In a software-defined network, the forwarding table at each switch node usually is implemented by expensive and power-hungry Ternary Content Addressable Memory (TCAM) that only has limited numbers of entries. In addition, the bandwidth capacity at each link is limited as well. Provisioning quality services to users by admitting their requests subject to such critical network resource constraints is a fundamental problem, and very little attention has been paid. In this paper, we study online unicasting and multicasting in SDNs with an objective of maximizing the network throughput under network resource constraints, for which we first propose a novel cost model to accurately capture the usages of network resources at switch nodes and links. We then devise two online algorithms with competitive ratios $O(\log n)$ and $O(K^{\epsilon} \log n)$ for online unicasting and multicasting, respectively, where n is the network size, K is the maximum number of destinations in any multicast request, and ϵ is a constant with $0 < \epsilon \leq 1$. We finally evaluate the proposed algorithms empirically through simulations. The simulation results demonstrate that the proposed algorithms are very promising.

1. Introduction

Software-Defined Networking (SDN) has emerged as the next-generation networking paradigm that creates an opportunity to tackle a longstanding problem in traditional networks, by moving the network control logic from the underlying routers and switches to a logically centralized controller and offering the programmability of the network [1,7,11,17]. SDN now is becoming a key technology for the next-generation network architecture, including Internet backbone networks and data center networks such as Google's B4 [15]. However, one fundamental problem in the adoption of the SDN technology by network and cloud service providers is how to enable efficient data traffic routing in the network such that the network throughput is maximized, considering that not only do SDNs usually have both node and link resource capacity constraints but also user requests arrive into SDNs dynamically without the knowledge of future request arrivals.

Low-latency and high-performance matching of forwarding rules in each TCAM plays a vital role in terms of routing efficiency. Therefore, the forwarding table at each switch node typically is implemented by a special yet expensive memory – the Ternary Content Addressable Memory (TCAM) that supports fast, parallel lookups [17,26]. However, the number of entries in each such TCAM forwarding table is usually limited to several thousand [17]. This highly restricted capacity of TCAM has been recognized as the main bottleneck to the scalability of SDN [7,16,17,23]. Efficient utilization of forwarding tables to serve a scaling number of forwarding rules while meeting network resource capacity constraints is an important and challenging research topic. In addition, with ever-growing bandwidth demands by users, it urgently needs efficient routing mechanisms that take into account both the TCAM capacity at each switch node and the bandwidth capacity at each link in the network. Furthermore, the dynamics of online user requests without the knowledge of future request arrivals makes the design of efficient routing protocols for SDNs very difficult and challenging.

* Corresponding author.
E-mail addresses: u4700480@anu.edu.au (M. Huang), wliang@cs.anu.edu.au (W. Liang), z.xu@dlut.edu.cn (Z. Xu), wenzheng.xu@scu.edu.cn (W. Xu), song.guo@polyu.edu.hk (S. Guo), ylxu@ustc.edu.cn (Y. Xu).

https://doi.org/10.1016/j.comnet.2017.12.011
1389-1286/

In this paper, we study online unicasting and multicasting in SDNs with the aim to maximize network throughput (or the acceptance rate of requests), where a sequence of unicast or multicast requests arrive one by one without the knowledge of future arrivals. Each incoming request is either accepted or rejected immediately, depending on whether there are sufficient resources to meet its resource demands. Although extensive effort on online unicasting and multicasting in traditional networks has been conducted in the past, most studies considered either the node resource capacity [18,20,21] or the link resource capacity [2,24], none of the studies jointly took into account these two types of resource capacities: the forwarding table capacities at switch nodes and the bandwidth capacities at links. We will jointly take into account the resource capacities at both nodes and links. We will also propose novel cost models unifies the usage costs of these two types of resources simultaneously. It is noticed that there are several recent studies on unicast and multicast routing for SDNs [1,7,11,17]. They however only considered the forwarding table capacity without taking into account the link bandwidth constraint. For example, the authors in [1,7] studied unicast routing in SDNs by first reducing the node TCAM capacity constraint into the node-degree constraint, followed by finding a node-degree-constrained maximum flow for a given set of unicast routing requests. Such the reduction approach however is not applicable in practice, since the node TCAM capacity usually is far greater than the maximum degree of nodes in the network. Huang et al. [11] dealt with the admission of a single multicast request in SDNs. They reduced the problem of finding a multicast tree for the request to the problem of finding a node-degree-constrained multicast tree that is NP-hard [8]. An approximate solution to the latter problem returns an approximate solution to the former problem. In contrast, in this paper we consider dynamic admissions of a sequence of online unicast or multicast requests without the knowledge of future request arrivals. The mentioned existing algorithms clearly are not applicable to the problem of concern. Instead, new online algorithms for online unicasting and multicasting as well as the analysis techniques to the competitive ratios of online algorithms need to be developed.

The main contributions of this paper are summarized as follows. We study the online unicasting and multicasting problems in SDNs with the aim to maximize the network throughput, by taking both node and link capacities and user bandwidth demands into consideration. We first propose a novel cost model to accurately capture the usage costs of node and link resources in the admission of a sequence of unicast or multicast requests without the knowledge of future request arrivals. We then devise efficient online algorithms with provable competitive ratios for them. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are very promising. To the best of our knowledge, we are the very first to study online unicasting and multicasting in SDNs by taking both node and link constraints into consideration, and devise the very first online algorithms with provable competitive ratios for the problems. In particular, the construction of the auxiliary graph and assignment of their edge weights may be of independent interest and can be applied to other optimization problems in networks.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the system model, notions and notations, and problem definitions. Section 4 introduces a cost model for resource usages. Sections 5 and 6 propose online algorithms and analyze their competitive ratios for online unicasting and multicasting in software-defined networks, respectively. Section 7 evaluates the performance of the proposed algorithms by experimental simulation, and Section 8 concludes the paper.

2. Related work

There are several studies that addressed the challenges in traffic engineering and steering for incremental deployment of SDN-enabled devices in existing networks [1,5,25]. Most of these studies dealt with data traffic routing in SDNs, by considering the limited TCAM size at each switch node [7,11,16]. Specifically, Kanizo et al. [16] studied a unicast routing problem for a group of unicast requests in SDNs. To overcome the limitation of the forwarding table size, they proposed two approaches to decompose large SDN forwarding tables into several small ones and distribute these small tables across the network, while maintaining the overall SDN policy semantics. Huang et al. [10] tackled the limitation of TCAM size by selectively caching forwarding rules in local switches and forwarding packets to the centralized controller if necessary, while CacheFlow [17] was proposed to equip each switch with hardware memory implemented by TCAM and secondary software memory in order to create an abstraction of infinite space for forwarding rules. Cohen et al. [7] studied the bounded path-degree max flow problem in SDNs subject to the TCAM capacity constraint at switches. They proposed an approximate solution with high probability. Meanwhile, Huang et al. [11] studied the multicast problem in SDNs, by devising an approximation algorithm for finding a degree-constrained Steiner tree for a single multicast request. However, these mentioned studies only considered the switch node capacity for a given set of unicast requests or a single unicast or multicast request. The most recent work in [12] dealt with admissions of a set of multicast requests with the aim to minimize the total bandwidth consumption on realizing the requests, subject to both node and link capacity constraints. Huang et al. [13,14] studied the problem of placing consolidated middleboxes in an SDN with the objective of maximizing network throughput, assuming that only subset of switches attached with servers. They proposed efficient heuristic algorithms to jointly place the consolidated middleboxes of requests and route data traffic of these requests from their sources to the placed middleboxes before being forwarded to their destinations. Also, they assumed that there is an end-to-end delay constraint associated with each request. The request admission problem considered in that paper thus is essentially different from the one in this paper, as the system model is different. We here assume that each switch has a limited TCAM and there is not the end-to-end delay constraint imposed on each request. The solution in that paper thus cannot be extended to solve the problem in this paper. Xu et al. [27] considered user query processing among cloudlets in a wireless metropolitan area network such that the network throughput is maximized while the average access latency among user requests is minimized. Since that paper focused on load-balancing among cloudlets with computing capacity constraints, by assigning different user requests to different cloudlets, neither TCAM nor SDN has ever been considered in that paper. The solution in that paper is not applicable to the problem here either.

The aforementioned studies are essentially different from the one in this paper, where we deal with dynamic admissions of unicast or multicast requests without the knowledge of their future arrivals, by considering both the TCAM capacity at each switch node and the bandwidth capacity at each link in an SDN. In particular, the joint consideration of these two different types of network resources when performing online request admission is much more challenging, in comparison with existing works that only one type of resource is considered and all requests are given in advance, as some admitted requests may still occupy the resources when a new request arrives, thereby leaving less available resources for later arrived requests. Existing studies failed to address the following three important aspects in dynamic admissions of requests. First, due to the dynamic changes of network resources,

there desperately needs a cost metric to measure the dynamic consumptions of various network resources and these resources utilization. Building an accurate cost metric to capture such dynamics is crucial, which then can be used to guide efficient resource allocations for incoming requests. The key to developing such a cost metric is to accurately model the availability and utilization of each resource. An exponential function of a specific resource and its utilization rate is an excellent candidate of this cost metric, which has been adopted for online request routing in many different types of networks including ATM and virtual circuit networks [2,24], and ad hoc and wireless sensor networks [18,20,21]. Second, the joint consideration of both the forwarding table capacity at each switch node and the bandwidth capacity at each link makes the cost modeling of resource usages in SDNs more difficult. Third, the joint consideration of resources at both nodes and links complicates the analysis of the proposed solution, since the performance analysis (i.e., the competitive ratios) of existing online algorithms for online unicasting and multicasting in ATM networks, virtual circuit networks [2,24], and wireless sensor networks [20] are only based on the cost modeling of a single type of resource at either nodes or links.

3. Preliminaries

In this section we first introduce the system model, we then introduce the notions and notations, and we finally define the problem precisely.

3.1. System model

We consider a software-defined network (SDN) $G = (V, E)$, where V is the set of SDN-enabled switch nodes, and E is the set of links that connect the switches. There is an SDN controller co-located with a switch node in G to handle the admission of unicast or multicast requests, by installing forwarding rules to the forwarding tables at switch nodes and allocating bandwidth on the links along the routing paths or trees in G for the requests. Each switch node $v \in V$ is equipped with a TCAM forwarding table for packet forwarding, and the table capacity is L_v rule entries. Each link $e = (u, v) \in E$ has a bandwidth capacity B_e (or $B_{(u, v)}$).

3.2. TCAM and routing rule matching in SDNs

The flow table at each SDN switch contains a list of *rules* that determine how packets have to be processed by the switch. Each rule consists of three main components: a *matching pattern, an actions field*, and a *priority* [3]. The purpose of matching pattern is to test if a packet belongs to a flow according to the attributes of the packet such as its source IP address, communication protocol, etc. All packets that are matched by the same matching pattern are considered to belong to the same flow. The actions specified in the actions field of a rule are applied to every packet of the corresponding flow. Some common actions including forwarding, dropping, or rewriting the packets. As a packet may match multiple matching patterns, each rule is also associated with a priority and only the actions of the matching rule with the highest priority are applied. It is worth noting that a rule may contains other additional components, such as a *counter*, which is used to keep track of the number of packets that has been processed according to this rule.

We adopt the TCAM usage model from [4,22]. That is, TCAM only stores the matching pattern and priority of every rule, whereas other components of the rule are stored in external memory (e.g., Static Random Access Memory (SRAM) [3]). For each incoming packet, the switch will first undergo a parallel lookup operation to find a matching rule with the highest priority. Having found (the index of) the matching rule for the packet, the action part of the rule is retrieved from SRAM and the specified actions are applied to the packet. Consequently, only one forwarding entry needs to be stored in TCAM for both unicast and multicast requests. Due to the significantly higher cost and smaller capacity of TCAM, in this paper, we focus on the capacity constraint of TCAM.

3.3. User routing requests

We consider two types of user routing requests: unicast and multicast requests that arrive into the system one by one. Let $r_k = (s_k, t_k; b_k)$ be the kth unicast request arrived into the system, where b_k is the bandwidth resource demand of r_k from s_k to t_k. Similarly, let $r_k = (s_k, D_k; b_k)$ be the kth multicast request, where s_k is the source switch node, D_k is the destination set of switch nodes with $D_k \subseteq V$, and b_k is the amount of demanded bandwidth by the request. Following existing studies [1,7,10], we assume that the bandwidth demand of a request can be derived from historical traces of the same or similar request demands. Even the demand of a request varies over time, it can be predicted accurately by making use of its historical traces. For example, we can adopt a popular prediction method such as the auto-regressive moving average method for this purpose. We further assume that the demanded bandwidth b_k by each request r_k is an integer and at least one unit bandwidth, i.e., $b_k \geq 1$ for any k.

Given that the forwarding table at each switch node and bandwidth resource at each link are limited, an incoming request will be admitted by the system only if there is a routing path (for the unicast request) or a multicast tree (for the multicast request) that has enough available resources to meet its demands. The system will allocate its demanded resources to each admitted request, and an admitted request will release the resources allocated to it back to the system once it finishes. Otherwise, if the network cannot meet the resource demands of a request, the request will be rejected. A rejected request can be re-submitted to the network, and it can be admitted if there are sufficient available resources in the network to meet its demand.

3.4. Competitive ratios of online algorithms

Let OPT and S be the values of an optimal offline solution and the solution delivered by an online algorithm $\mathcal{A}$ for a given sequence of requests without the knowledge of future request arrivals, respectively. The *competitive ratio* of algorithm $\mathcal{A}$ is ξ if $\frac{S}{OPT} \geq \frac{1}{\xi}$ for any instance of the maximization problem. Ideally, the competitive ratio is expected to be a small constant. However, the performance of the SDN can be as bad as $O(n)$ if there is no admission control on requests, since some requests may consume a lot of resources in the network, later arrived requests cannot be admitted due to lack of demanded resources, where n is the network size.

3.5. Problem definitions

Given a software-defined network $G = (V, E)$, where each $v \in V$ be a switch node equipped with a TCAM forwarding table of size L_v for packet routing, and let B_e be the bandwidth capacity of link $e \in E$, and a sequence of unicast requests $(s_k, t_k; b_k)$ arrives into the system one by one without the knowledge of future arrivals, *the network capacity maximization problem for online unicasting* in G is to maximize the network throughput (the accumulated bandwidth of admitted unicast requests), subject to resource capacity constraints on switch nodes and links in the network.

The network capacity maximization problem for online multicasting can be defined similarly. That is, given an SDN $G = (V, E)$, let each $v \in V$ be a switch node equipped with a TCAM of forwarding

table of size L_v for packet routing, and let B_e be the bandwidth capacity of each link $e \in E$. Given a sequence of multicast requests (s_k, D_k; b_k) that arrives into the system one by one without the knowledge of future arrivals, the problem is to maximize the network throughput (the accumulated bandwidth of admitted multicast requests), subject to resource constraints on switch nodes and links in the network.

4. The usage costs of resources of links and nodes

Given a software-defined network $G = (V, E)$, a metric is needed to model the usage costs of resources at its switch nodes and links. One important characteristic of such resource usages is that *the marginal costs* of resource usages inflate with the increase on the workloads of the resources. Compared with a lightly-loaded switch node, a heavily-loaded switch node will spend more time and consume more energy on matching a forwarding rule for an incoming packet, because more rules in such a heavily-loaded switch node need to be considered. For example, in the process of installing a forwarding rule into a switch node with the TCAM capacity of 2000 entries, existing forwarding rules must be matched to make sure the new forwarding rule does not exist in the TCAM prior to its installation. This process takes five seconds for the first 1000 entries, while it takes almost two minutes for the next 1000 entries [17], which directly translates into more energy costs. Thus, when admitting a request, we should make use of lower-cost and under-loaded nodes and links to admit the request, rather than over-loaded ones. Through this observation, we will make use of exponential functions to model the costs of resource usages, which are the functions of available amounts and the resource workloads. Since usages of over-loaded/saturated resources usually incur higher costs by the adopted exponential functions, it can efficiently avoid the usage of overloaded/saturated resources. The exponential functions for the usage costs of edge and node resources are given as follows.

Let $L_v(k)$ and $B_e(k)$ be the numbers of available entries in the forwarding table at switch node $v \in V$ and the residual bandwidth on link $e \in E$, respectively, when the kth request r_k arrives. We use an exponential function to model the cost $c_v(k)$ of using the resource at each switch node v by request r_k, which is defined as

$$c_v(k) = L_v(\alpha^{1-\frac{L_v(k)}{L_v}} - 1), \tag{1}$$

where $\alpha > 1$ is a constant to be determined later, and $1 - \frac{L_v(k)}{L_v}$ is *the utilization ratio* of the forwarding table of v when request r_k arrives.

The cost $c_e(k)$ of using the bandwidth of link $e \in E$ by request r_k can be similarly defined as

$$c_e(k) = B_e(\beta^{1-\frac{B_e(k)}{B_e}} - 1), \tag{2}$$

where $\beta > 1$ is a constant that is similar to α to be determined later, and $1 - \frac{B_e(k)}{B_e}$ is the utilization ratio of the bandwidth of link e when request r_k arrives.

5. Online algorithm for dynamic unicast routing

In this section, we first describe an online algorithm for the network capacity maximization problem for online unicasting. We then analyze the competitive ratio and time complexity of the proposed algorithm.

5.1. Online algorithm

The basic idea of the online algorithm is to transform the network capacity maximization problem for online unicasting in the original SDN G into the problem of a series of finding a shortest path in edge-weighted, directed auxiliary graphs. The key is how to jointly consider node and edge costs in a unified way in the construction of each auxiliary graph $G'(k)$ for request r_k.

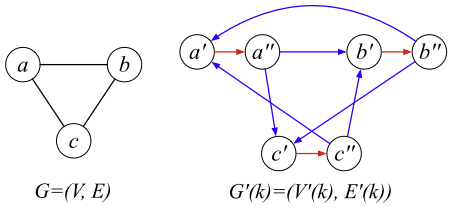


Fig. 1. The construction of $G'(k) = (V'(k), E'(k))$ for an SDN $G = (V, E)$ for online unicasting when request r_k arrives.

For each unicast request r_k with (s_k, t_k, b_k), to model the usage costs of the resources at nodes and links in the network G by admitting r_k, an edge-weighted, directed graph $G'(k) = (V'(k), E'(k); \omega)$ will be constructed from $G = (V, E)$, and a shortest path in $G'(k)$ from node s'_k to node t'_k will be found. Note that the weight of each edge in $G'(k)$ is *the normalized usage cost* of its corresponding switch node or link, which is an exponential function of the available amount and the workload of the resource at the node or the link. In the following, we first detail the construction of $G'(k)$. We then devise an efficient online algorithm for the network capacity maximization problem for online unicasting.

The edge-weighted, directed graph $G'(k) = (V'(k), E'(k); \omega)$ is constructed from $G = (V, E)$ as follows. For each switch node $v \in V$, two nodes v' and v'' are added to $V'(k)$, i.e., $V'(k) = \{v', v'' \mid v \in V\}$, and a directed edge $\langle v', v'' \rangle$ is added to $E'(k)$. For each link $(u, v) \in E$, two directed edges $\langle u'', v' \rangle$ and $\langle v'', u' \rangle$ are added to $E'(k)$, i.e., $E'(k) = \{\langle v', v'' \rangle \mid v \in V\} \cup \{\langle v'', u' \rangle, \langle u'', v' \rangle \mid (u, v) \in E\}$. For brevity, we refer to the edges in $G'(k)$ derived from switch nodes and links of G as the *node-derived edges* and *link-derived edges*, and denote by $E'_v(k)$ and $E'_e(k)$ the sets of node-derived edges and link-derived edges in $G'(k)$, respectively. Clearly, $E'(k) = E'_e(k) \cup E'_v(k)$ and $E'_e(k) \cap E'_v(k) = \emptyset$. Fig. 1 illustrates the construction of $G'(k)$.

Depending on its type (node-derived or link-derived edge) and the usage cost of the resource it represents, each edge $e \in E'(k)$ is assigned a weight as follows.

$$\omega_e(k) = \begin{cases} \frac{c_v(k)}{L_v} = \alpha^{1-\frac{L_v(k)}{L_v}} - 1 & \text{if } e = \langle v', v'' \rangle \in E'_v(k), \\ \frac{c_{u,v}(k)}{B_{(u,v)}} = \beta^{1-\frac{B_{(u,v)}(k)}{B_{(u,v)}}} - 1 & \text{if } e = \langle u'', v' \rangle \in E'_e(k), \end{cases}$$

where the weight of each node-derived edge reflects the forwarding table entry usage on its corresponding switch node, while the weight of each link-derived edge reflects the bandwidth usage on its corresponding link. Notice that the weight of each edge is the normalized usage cost of the resource it represents.

An edge $e' \in E'_e(k)$ derived from a link $e \in E$ is omitted if its residual bandwidth is strictly less than b_k, because e cannot meet the bandwidth demand of request r_k, and thus plays no role in the admission of the kth request. For simplicity, the resulting graph after pruning some edges from it is still denoted as $G'(k)$. Let $P(k)$ be a shortest path in $G'(k)$ from s'_k to t'_k. Following the construction of $G'(k)$, the edges in $P(k)$ are the node-derived and link-derived edges alternatively.

To maximize the network throughput (the accumulated bandwidth of all admitted requests), *an admission control policy* will be adopted. That is, when the length of a shortest path in $G'(k)$ from s'_k to t'_k for each incoming request r_k is greater than a given threshold, the request will be rejected no matter whether there are sufficient resources to admit the request. We here adopt the following admission control policy: a unicast request r_k will be rejected, if (i) the weighted sum of node-derived edges in $P(k)$ is greater than σ_v; or (ii) the weighted sum of the link-derived edges

in $P(k)$ is greater than σ_e, where σ_v $(= |V| - 1 = n - 1)$ and σ_e $(= |V| - 1 = n - 1)$ are pre-determined thresholds. In other words, an incoming request r_k is admitted if and only if there exists a shortest path $P(k)$ in $G'(k)$ from s'_k to t'_k such that $P(k)$ meets the following requirements: (i) $\sum_{e=\langle v', v''\rangle \in P(k) \cap E'_v(k)} \omega_e(k) \leq \sigma_v$; and (ii) $\sum_{e=\langle v'', u'\rangle \in P(k) \cap E'_e(k)} \omega_e(k) \leq \sigma_e$. The detailed algorithm for online unicasting is given in Algorithm 1.

Algorithm 1 Online routing algorithm for unicast requests.

Input: a software define network $G = (V, E)$ and the kth unicast request $r_k = (s_k, t_k; b_k)$;

Output: Maximize the network throughput by admitting or rejecting each arriving unicast request r_k. If admitted, a routing path for r_k will be found.

1: /* Ensure that the switch table of each node v can contain at least one entry */
2: **for** each node $v \in V$ **do**
3: **if** the table size at node v is full, i.e., $L_v(k) = 0$ **then**
4: Remove v and its incident links from G;
5: **end if**
6: **end for**
7: /* Ensure that the residual bandwidth capacity of each link e is at least b_k */
8: **for** each link $e \in E$ **do**
9: **if** the residual bandwidth at link e is less than b_k, i.e., $B_e(k) < b_k$ **then**
10: Remove e from G;
11: **end if**
12: **end for**
13: Construct an edge-weighted, directed graph $G'(k) = (V'(k), E'(k); \omega)$ from the resulting subgraph of G and calculate the edge weights according to the resource utilization when request r_k arrives;
14: Find a shortest path $P(k)$ in $G'(k)$ from s'_k to t'_k;
15: **if** $P(k)$ does not exist **then**
16: Reject unicast request r_k;
17: **else**
18: **if** $(\sum_{e \in P(k) \cap E'_v(k)} \omega_e(k) \leq \sigma_v)$ and $(\sum_{e \in P(k) \cap E'_e(k)} \omega_e(k) \leq \sigma_e)$ **then**
19: Admit unicast request r_k with $P(k)$ as its routing path;
20: Update the residual resource capacities of links in E and nodes in V;
21: **else**
22: Reject unicast request r_k;
23: **end if**
24: **end if**

5.2. Algorithm analysis

In the following we analyze the performance of the proposed algorithm. Let $L_{\min}$ be the minimum TCAM size, i.e., $L_{\min} = \min\{L_v \mid v \in V\}$, let $B_{\min}$ be the minimum bandwidth capacity among links, i.e., $B_{\min} = \min\{B_e \mid e \in E\}$, and $b_{\max}$ the maximum bandwidth demand by any unicast request, i.e., $b_{\max} = \{b_{k'} \mid 1 \leq k' \leq k\}$.

We first show the upper bound on the cost of admitted unicast requests as of the arrival of request r_k by Algorithm 1 by the following lemma.

Lemma 1. *Given an SDN $G = (V, E)$ with forwarding table capacity L_v at each switch node $v \in V$ and link bandwidth capacity B_e at each link $e \in E$, denote by $\mathcal{S}(k)$ the set of unicast requests admitted by the online algorithm, Algorithm 1, until the arrival of request r_k. Then, the cost sums of nodes and links in G are*

$$\sum_{v \in V} c_v(k) \leq |\mathcal{S}(k)|(\sigma_v + n - 1)\log\alpha, \tag{3}$$

and

$$\sum_{e \in E} c_e(k) \leq \mathbb{B}(k)(\sigma_e + n - 1)\log\beta, \tag{4}$$

respectively, where α and β are constants with $2|V| \leq \alpha \leq 2^{L_{\min}}$ and $2|V| \leq \beta \leq 2^{B_{\min}/b_{\max}}$, and $\mathbb{B}(k) = \sum_{k' \in \mathcal{S}(k)} b_{k'}$.

See the proof in Appendix A.

We then provide a lower bound on the length of the routing path to which an optimal offline algorithm routes a request that is rejected by the online algorithm in the following lemma.

Lemma 2. *Let $\mathcal{R}(k)$ be the set of unicast requests that are admitted by an optimal offline algorithm yet rejected by the online algorithm, Algorithm 1, prior to the arrival of unicast request r_k, and let $P_{opt}(k')$ be the routing path in $G'(k)$ found by the optimal offline algorithm for request $r_{k'} \in \mathcal{R}(k)$. Then, for any request $r_{k'} \in \mathcal{R}(k)$, we have*

$$\sum_{e \in P_{opt}(k')} \omega_e(k') \geq \min\{\sigma_v, \sigma_e\} = |V| - 1 = n - 1, \tag{5}$$

where α and β are constants with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$.

See the proof in Appendix A.

We finally have the following theorem.

Theorem 1. *Given an SDN $G = (V, E)$ with both node and link capacities $L_v(\cdot)$ and $B_e(\cdot)$ for all $v \in V$ and $e \in E$, assume that there is a sequence of unicast requests $r_1 = (s_1, t_1; b_1), \ldots, r_k = (s_k, t_k; b_k)$ arriving one by one without the knowledge of future arrivals. There is an online algorithm, Algorithm 1, with the competitive ratio of $2(\gamma \log\alpha + \log\beta) + 1$ for the network capacity maximization problem for online unicasting, which takes $O(|V|^2)$ time for each request, where α and β are constants with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$, and $\gamma = \frac{B_{\min}}{\log\beta}$ is a value with $1 < \gamma \leq \frac{B_{\min}}{\log(2n)}$.*

See the proof in Appendix A.

6. Online algorithm for multicast routing

In this section, we deal with the network capacity maximization problem for online multicasting. We first propose an efficient online algorithm for the problem. We then analyze the competitive ratio and time complexity of the proposed algorithm.

6.1. Online algorithm

The basic idea of the proposed algorithm is to respond to each incoming multicast request r_k $(= (s_k, D_k; b_k))$ by either admitting or rejecting it, according to an admission control policy. To model the node and link resource usages by admitting multicast request r_k, an auxiliary edge-weighted, directed graph $G'(k) = (V'(k), E'(k); \omega)$ will be constructed, where the weight of each node-derived or link-derived edge of $G'(k)$ reflects the availability and utilization of the resource in that node or link of G.

In order to reduce the resource consumption of admitting a multicast request r_k and admit more future requests, a multicast tree in $G'(k)$ rooted at the source s'_k and spanning all destination nodes in D_k will be found if it exists. If the weighted sums of all node-derived edges and link-derived edges in the multicast tree are less than their corresponding thresholds, then the request will be admitted; otherwise, it will be rejected. In the following, we detail the construction of $G'(k)$, and then devise an online algorithm

for the network capacity maximization problem for online multicasting.

Given an SDN $G(V, E)$ with node and link capacities $L_v(\cdot)$ and $B_e(\cdot)$, respectively, an auxiliary edge-weighted, directed graph $G'(k) = (V'(k), E'(k); w)$ for each multicast request r_k is constructed, and the construction of $G'(k)$ is identical to the construction of $G'(k)$ for the kth unicast request as shown in Fig. 1. That is, for each switch node $v \in V$, two nodes v' and v'' are added to $V'(k)$, and there is a directed edge $\langle v', v'' \rangle$ added to $E'(k)$. For each edge $(u, v) \in E$, two directed edges $\langle v'', u' \rangle$ and $\langle u'', v' \rangle$ are added to $E'(k)$, i.e., $V'(k) = \{v', v'' \mid v \in V\}$ and $E'(k) = \{\langle v', v'' \rangle \mid v \in V\} \cup \{\langle v'', u' \rangle, \langle u'', v' \rangle \mid (u, v) \in E\}$.

Now, given a multicast request r_k with $(s_k, D_k; b_k)$, the problem is transformed to finding a multicast tree $T(k)$ in $G'(k)$ rooted at s'_k and spanning all nodes in $D'_k = \{u' \mid u \in D_k\}$ such that the weighted sum of the edges in $T(k)$ is minimized, which is a classic NP-hard *directed Steiner tree problem*. As it is very unlikely to find an exact solution for it in polynomial time, an approximate solution suffices, by applying the approximation algorithm in [6]. The approximation ratio of the approximation algorithm is $|D_k|^{\epsilon}$, and its running time is a polynomial function of n and $\frac{1}{\epsilon}$, where ϵ is a constant with $0 < \epsilon \leq 1$. The choice of ϵ will determine the accuracy of the solution obtained and the running time of the algorithm. For the sake of simplicity, in the rest of this paper, we abuse the notation of $T(k)$ and denote it as the corresponding multicast tree in G rooted at s_k and spanning all terminal nodes in D_k too.

To prevent admitting some multicast requests that will degrade the performance of the proposed online algorithm significantly, an admission control policy will be adopted. That is, a multicast request r_k is admitted only if it meets (i) $\sum_{e \in T(k) \cap E'_v(k)} \omega_e(k) \leq \sigma_v$; and (ii) $\sum_{e \in T(k) \cap E'_e(k)} \omega_e(k) \leq \sigma_e$, where $\sigma_v = \sigma_e = |V| - 1 = n - 1$.

The detailed online algorithm for the network capacity maximization problem for online multicasting is given in Algorithm 2.

Algorithm 2 Online routing algorithm for multicast requests.

Input: An SDN $G = (V, E)$ and an incoming multicast request r_k with $(s_k, D_k; b_k)$ and $D_k \subseteq V$;

Output: Maximize the network throughput by admitting or rejecting each arriving multicast request r_k. If admitted, a routing multicast tree for r_k will be found.

1: /* Ensure that the switch table of each node v has at least $|D_k|$ available entries */;
2: **for** each node $v \in V$ **do**
3: **if** the available table size at node v is zero **then**
4: Remove v and its incident links from G;
5: **end if**
6: **end for**
7: /* Ensure that the residual bandwidth capacity of each link e is at least b_k */
8: **for** each link $e \in E$ **do**
9: **if** the residual bandwidth at link e is less than b_k, i.e., $B_e(k) < b_k$ **then**
10: Remove e from G;
11: **end if**
12: **end for**
13: Construct an edge-weighted, directed graph $G'(k) = (V'(k), E'(k); \omega)$ from the resulting subgraph of $G$and calculate the edge weights according to the resource utilization when request r_k arrives;
14: Find an approximate multicast tree $T(k)$ in $G'(k)$ rooted at s'_k and spanning all nodes in D'_k, by applying the algorithm due to Charikar et~al. in [6];
15: **if** $T(k)$ does not exist **then**
16: Reject multicast request r_k;
17: **else**
18: **if** $(\sum_{e \in T(k) \cap E'_v(k)} \omega_e(k) \leq \sigma_v)$ and $(\sum_{e \in T(k) \cap E'_e(k)} \omega_e(k) \leq \sigma_e)$ **then**
19: Admit multicast request r_k with $T(k)$;
20: Update the residual resource capacities of links in E and nodes in V;
21: **else**
22: Reject request r_k;
23: **end if**
24: **end if**

6.2. Algorithm analysis

In the following, we analyze the competitive ratio and time complexity of Algorithm 2. Recall that $L_{\min}$ is the minimum TCAM size among switch nodes, i.e., $L_{\min} = \min\{L_v \mid v \in V\}$, $B_{\min}$ is the minimum bandwidth capacity among links, i.e., $B_{\min} = \min\{B_e \mid e \in E\}$, $b_{\max}$ is the maximum bandwidth demand by any multicast request, K is the maximum number of terminal nodes in any multicast request, i.e., $K = \max\{|D_{k'}| \mid 1 \leq k' \leq k\}$, and ϵ is a fixed value with $0 < \epsilon \leq 1$. We start with the following lemma.

Lemma 3. *Given an SDN $G = (V, E)$ with node capacity L_v for each switch node $v \in V$ and link bandwidth capacity B_e for each link $e \in E$, denote by $\mathcal{S}(k)$ the set of multicast requests admitted by the online algorithm, Algorithm 2, until the arrival of multicast request r_k. Then, the cost sums of nodes and of links of G when multicast request r_k arrives are*

$$\sum_{v \in V} c_v(k) \leq |\mathcal{S}(k)|(\sigma_v + n - 1)\log\alpha, \tag{6}$$

and

$$\sum_{e \in E} c_e(k) \leq \mathbb{B}(k)(\sigma_e + n - 1)\log\beta, \tag{7}$$

respectively, where α and β are constants with $2|V| \leq \alpha \leq 2^{L_{\min}}$ and $2|V| \leq \beta \leq 2^{B_{\min}/b_{\max}}$, and $\mathbb{B}(k) = \sum_{k' \in \mathcal{S}(k)} b_{k'}$.

See the proof in Appendix B.

We then show the lower bound on the cost sum of node-derived and link-derived edges of the multicast tree $T(k')$ for a multicast request $r_{k'}$, which is admitted by an optimal offline algorithm but rejected by the online algorithm, as follows.

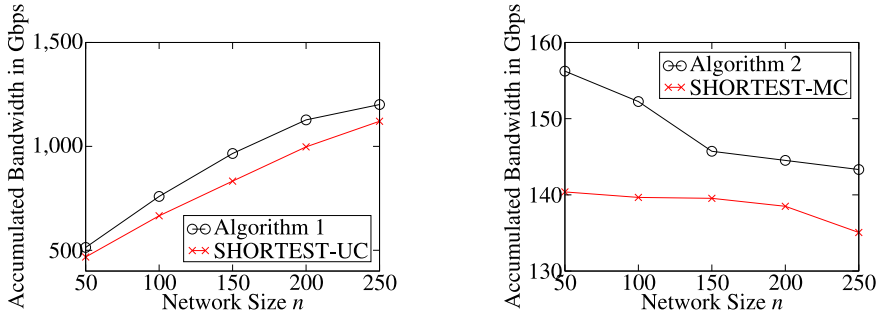
Lemma 4. *Let $\mathcal{R}(k)$ be the set of multicast requests admitted by an optimal offline algorithm yet rejected by the online algorithm, Algorithm 2, prior to the arrival of multicast request r_k. Let $T_{opt}(k')$ be the multicast tree in $G'(k)$ found by the optimal offline algorithm for request $r_{k'} \in \mathcal{R}(k)$. Then, for each multicast request $r_{k'} \in \mathcal{R}(k)$, we have $\sum_{e \in T_{opt}(k')} \omega_e(k') \geq \frac{\min\{\sigma_v, \sigma_e\}}{K^{\epsilon}}$, where α and β are constants with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$.*

See the proof in Appendix B.

We finally have the following theorem.

Theorem 2. *Given an SDN $G = (V, E)$ with both node and link capacities $L_v(\cdot)$ and $B_e(\cdot)$ for all $v \in V$ and $e \in E$, assume that there is a sequence of multicast requests $r_k = (s_k, D_k; b_k)$ arriving one by one without the knowledge of future arrivals with $D_k \subseteq V$. There is an online algorithm, Algorithm 2, with the competitive ratio of $2K^{\epsilon}(\gamma \log\alpha + \log\beta) + 1$ for the network capacity maximization problem for online multicasting, which takes $O((|V| + |E|)^{1/\epsilon} K^{2/\epsilon})$ time, where α and β are constants with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$, $\gamma = B_{\min}/\log\beta$, and ϵ is a constant with $0 < \epsilon \leq 1$.*

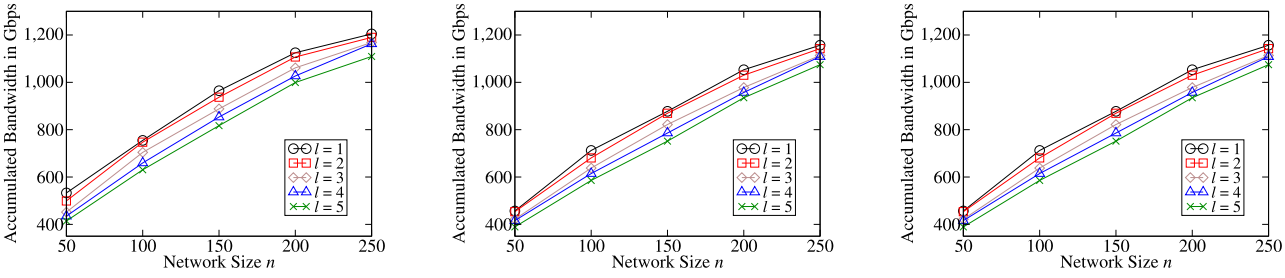
See the proof in Appendix B.



(a) Accumulated bandwidth for online unicasting

(b) Accumulated bandwidth for online multicasting

Fig. 2. The accumulated bandwidth delivered by different algorithms in networks.



(a) The performance of `Algorithm 1` by varying β $(= 2^l n)$ when $\alpha = 2^1 n$

(b) The performance of `Algorithm 1` by varying β $(= 2^l n)$ when $\alpha = 2^3 n$

(c) The performance of `Algorithm 1` by varying β $(= 2^l n)$ when $\alpha = 2^5 n$

(d) The performance of `Algorithm 1` by varying α $(= 2^l n)$ when $\beta = 2^1 n$

(e) The performance of `Algorithm 1` by varying α $(= 2^l n)$ when $\beta = 2^3 n$

(f) The performance of `Algorithm 1` by varying α $(= 2^l n)$ when $\beta = 2^5 n$

Fig. 3. The performance of Algorithm 1 for online unicasting by varying α and β, when $\sigma_v = \sigma_e = n - 1$.

7. Performance evaluation

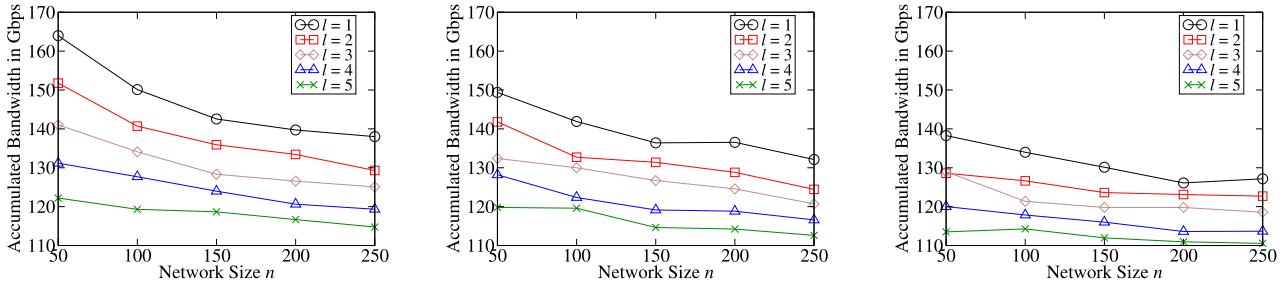
In this section, we evaluate the performance of the proposed algorithms through experimental simulations. We also investigate the impact of important parameters on the performance of the proposed algorithms.

7.1. Environment settings

We consider networks with 50, 100, 150, 200, and 250 nodes, respectively. For each network size, 30 network instances are generated, using the tool GT-ITM [9]. The size of TCAM L_v of each switch node $v \in V$ varies from 500 to 5000, and the bandwidth capacity B_e of each link $e \in E$ varies from 1,000 Mbps to 10,000 Mbps [16,19,23]. The bandwidth demand b_k of each unicast or multicast request r_k is randomly drawn between 1 Mbps and 50 Mbps. The number of destinations in a multicast request is randomly chosen between 1% and 15% of the network size. The value in each figure is the mean of the results out of 30 network instances with 30 different sequences of 50,000 unicast requests or 20,000 multicast requests.

To speed up the running time of the proposed algorithm, Algorithm 2, we employ a simpler approach based on the single-source shortest path algorithm to find an approximate, minimum Steiner tree, instead of the time-consuming approximation algorithm in [6] that delivers a better solution.

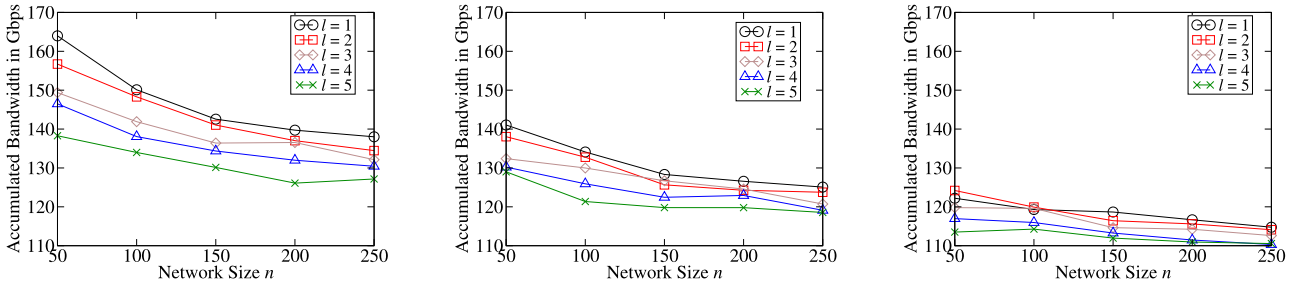
Since the proposed algorithms in this paper are the very first ones that jointly consider node and edge capacities in SDNs, comparing the performance of the proposed algorithms with those only considered either one type of resource capacity may be unfair. To evaluate the performance of the proposed algorithms against the benchmarks, we here propose two heuristics `SHORTEST-UC` and `SHORTEST-MC` for online unicasting and multicasting respectively. Specifically, for each request r_k, the heuristic algorithms first remove the links and nodes from the network G that do not have enough residual resources to support the admission of the request, and then assign each link the same weight. `SHORTEST-UC` finds a shortest path with the minimum number of links from the



(a) The performance of `Algorithm 2` by varying β $(= 2^l n)$ when $\alpha = 2^1 n$

(b) The performance of `Algorithm 2` by varying β $(= 2^l n)$ when $\alpha = 2^3 n$

(c) The performance of `Algorithm 2` by varying β $(= 2^l n)$ when $\alpha = 2^5 n$



(d) The performance of `Algorithm 2` by varying α $(= 2^l n)$ when $\beta = 2^1 n$

(e) The performance of `Algorithm 2` by varying α $(= 2^l n)$ when $\beta = 2^3 n$

(f) The performance of `Algorithm 2` by varying α $(= 2^l n)$ when $\beta = 2^5 n$

Fig. 4. The performance of Algorithm 2 for online multicasting by varying α and β when $\sigma_v = \sigma_e = n - 1$.

source to the destination of request r_k, while `SHORTEST-MC` finds a single-source shortest path tree spanning all destination nodes rooted at the source and spanning all destinations of a multicast request r_k.

7.2. Performance evaluation of different algorithms

We first evaluate the proposed two online algorithms Algorithms 1 and 2 against algorithms `SHORTEST-UC` and `SHORTEST-MC`, by varying network size n from 50 to 250 while keeping other parameters fixed, i.e., $\alpha = \beta = 2n$ and $\sigma_e = \sigma_v = n - 1$.

Fig. 2 plots the performance curves of different algorithms, from which it can be seen both Algorithms 1 and 2 outperform `SHORTEST-UC` and `SHORTEST-MC` in all cases. Specifically, Algorithm 1 outperforms algorithm `SHORTEST-UC`. As shown in Fig. 2(a), Algorithm 1 delivers 10% more accumulated bandwidth than that of `SHORTEST-UC`. Meanwhile, Algorithm 2 delivers more accumulated bandwidth and admits more requests than that of `SHORTEST-MC`, too. In particular, As shown in Fig. 2(b), the accumulated bandwidth delivered by Algorithm 2 is 10% more than that by `SHORTEST-MC` when $n = 50$. However, Algorithm 2 still delivers 9% more accumulated bandwidth compared with `SHORTEST-MC` when the network size is 250. The reason is that with the growth of the network size n, the number of destinations in each multicast request increases, requiring more node and link resources for the request realization.

7.3. Parameter impacts on algorithmic performance

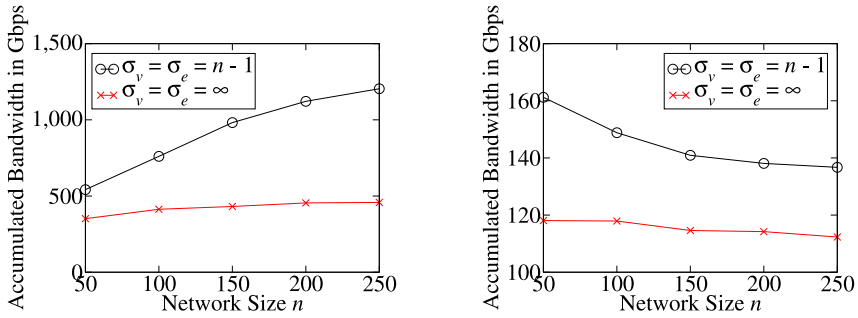
We first investigate the impact of parameters α and β on the performance of the proposed algorithms, by varying them from $2^1 n$ to $2^5 n$ while keeping $\sigma_v = \sigma_e = n - 1$. Figs. 3 and 4 plot the performance curves of Algorithms 1 and 2, by varying either α or β while fixing the other.

It can be seen from Fig. 3(a) to (c) that when α is fixed, the larger the value of β, the less the accumulated bandwidth delivered by Algorithm 1 and vice versa. For instance, when $\alpha = 2^1 n$ and $n = 50$, Algorithm 1 with $\beta = 2^1 n$ delivers 15% more the accumulated bandwidth than itself with $\beta = 2^5 n$. It also can be seen that the performance gap of Algorithm 1 under different α and β is flat with the increase of network size n. Similarly, Fig. 4 plots the performance curves of Algorithm 2, by varying the values of exactly one of α and β each time, from which it can be seen when α is fixed, the larger the value of β, the less the accumulated bandwidth delivered by Algorithm 2 and vice versa.

We then study the impact of the admission thresholds σ_v and σ_e on the performance of Algorithms 1 and 2. Fig. 5 plots the performance curves of Algorithms 1 and 2 with and without the admission control thresholds, from which it can be seen that both of them with admission control significantly outperform themselves without the admission control. Specifically, for online unicasting, the performance gap of Algorithm 1 with and without the thresholds becomes larger and larger with the increase in network size n, as shown in Fig. 5(a). For example, the ratio of the accumulated bandwidths delivered by Algorithm 1 with and without the admission control grows from 1.25 to 2.5 when $n = 25, 250$ respectively. For the online multicasting, the performance gap of Algorithm 2 with and without the admission control is relatively stable, as shown in Fig. 5(b). The difference in the accumulated bandwidth only drops from approximately 30 Gbps to approximately 24 Gbps for a monitoring period when the network size increases from 100 to 250.

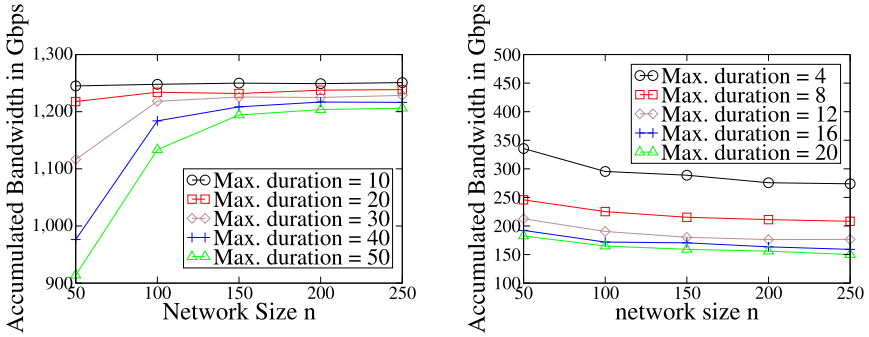
7.4. Impact of request implementation durations on the performance of algorithms

We now investigate the impact of the durations (numbers of time slots) of admitted requests on the performance of the proposed online algorithms Algorithms 1 and 2, by varying the maxi-



(a) Accumulated bandwidth by `Algorithm 1` (b) Accumulated bandwidth by `Algorithm 2`

Fig. 5. The accumulated bandwidth delivered by Algorithm 1 for online unicasting and Algorithm 2 for online multicasting with thresholds $\sigma_v = \sigma_e = n - 1$ and without the thresholds $\sigma_v = \sigma_e = \infty$ when $\alpha = \beta = 2n$.



(a) Accumulated bandwidth by `Algorithm 1` (b) Accumulated bandwidth by `Algorithm 2`

Fig. 6. The network throughput delivered by Algorithm 1 for online unicasting and Algorithm 2 for online multicasting by varying the maximum duration of admitted requests in terms of numbers of time slots.

mum duration of admitted requests, where each admitted request will release the resources allocated to it when it departs from the network. From Fig. 6 (a)-(b), we can see that the longer the maximum duration is, the less the number of requests the proposed online algorithms admit. The reason behind is that longer resource holding durations by the admitted requests result in fewer resources for later request admissions, which limits the ability of the network to admit more requests. Specifically, Fig. 6 (a) indicates that the impact of the maximum duration of admitted unicast requests is significant on the performance of the algorithm when the network size is small, because a small network has fewer resources, it cannot accommodate more requests if the currently admitted requests do not depart from it. On the other hand, from Fig. 6 (b), we can see that the gap between the performance of the online multicast algorithm is roughly the same for all network sizes when the maximum duration of admitted requests changes, due to the fact that multicast requests require much more resources and readily consume all the resources in the network even if some admitted requests depart from the network shortly.

8. Conclusions

In this paper, we studied online unicasting and multicasting in SDNs with resource capacity constraints on switch nodes and links, and user request bandwidth demands. We first proposed a novel cost model to model the usage costs of different resources. We then devised novel online algorithms with provable competitive ratios for online unicasting and multicasting. We finally evaluated the performance of the proposed algorithms through experimental simulation. Simulation results indicate that the proposed algorithms are very promising.

Acknowledgments

We really appreciate the three anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us improve the quality and presentation of the paper greatly. It is acknowledged that the work of Wenzheng Xu was supported by the National Natural Science Foundation of China under Grant No. 61602330.

Appendix A

Proof for Lemma 1

Proof. Consider a unicast request $r_{k'} \in \mathcal{S}(k)$ admitted by the online algorithm. Then, for any switch node $v \in V$, we have

$$\begin{aligned} & c_v(k'+1) - c_v(k') \\ &= L_v\left(\alpha^{1-\frac{L_v(k'+1)}{L_v}} - 1\right) - L_v\left(\alpha^{1-\frac{L_v(k')}{L_v}} - 1\right) \\ &= L_v\left(\alpha^{1-\frac{L_v(k'+1)}{L_v}} - \alpha^{1-\frac{L_v(k')}{L_v}}\right) \\ &= L_v\alpha^{1-\frac{L_v(k')}{L_v}}\left(\alpha^{\frac{L_v(k')-L_v(k'+1)}{L_v}} - 1\right) \\ &\le L_v\alpha^{1-\frac{L_v(k')}{L_v}}\left(\alpha^{\frac{1}{L_v}} - 1\right) \end{aligned} \tag{8}$$

$$= L_v \alpha^{1-\frac{L_v(k')}{L_v}} \left(2^{\frac{1}{L_v}\log\alpha} - 1\right)$$
$$\leq L_v \alpha^{1-\frac{L_v(k')}{L_v}} (\log\alpha / L_v) \quad (9)$$

$$= \alpha^{1-\frac{L_v(k')}{L_v}} \log\alpha. \quad (10)$$

where Inequality (8) holds because at most one routing entry is added to the forwarding table of node v, and Inequality (9) holds because $2^x - 1 \leq x$ with $0 \leq x \leq 1$, and $(1/L_v)\log\alpha \leq (1/L_v)L_{\min} \leq (1/L_v)L_v = 1$.

Similarly, for any edge $e \in E$, we have

$$\begin{aligned} & c_e(k'+1) - c_e(k') \\ &= B_e\left(\beta^{1-\frac{B_e(k'+1)}{B_e}} - 1\right) - B_e\left(\beta^{1-\frac{B_e(k')}{B_e}} - 1\right) \\ &= B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(\beta^{\frac{B_e(k')-B_e(k'+1)}{B_e}} - 1\right) \\ &\leq B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(\beta^{\frac{b_{k'}}{B_e}} - 1\right), \end{aligned} \quad (11)$$

$$= B_e\beta^{1-\frac{B_e(k')}{B_e}}(2^{\frac{b_{k'}}{B_e}\log\beta} - 1)$$
$$\leq \beta^{1-\frac{B_e(k')}{B_e}} \cdot b_{k'} \cdot \log\beta, \quad (12)$$

where Inequality (11) follows since at most $b_{k'}$ bandwidth units of link e for request $r_{k'}$ are reserved, and Inequality (12) follows because $\frac{b_{k'}}{B_e}\log\beta \leq \frac{b_{k'}}{B_e} \cdot \frac{B_{\min}}{b_{\max}} \leq \frac{b_{k'}}{B_e} \cdot \frac{B_e}{b_{k'}} = 1$, and $2^x - 1 \leq x$ with $0 \leq x \leq 1$.

We then calculate the cost sum of all nodes or links of G when admitting request $r_{k'}$. Notice that if an edge in $G'(k')$ is not in $P(k')$, its cost does not change after the admission of request $r_{k'}$. The difference in the cost sum of nodes before and after admitting request $r_{k'}$ thus is

$$\begin{aligned} & \sum_{v\in V}\left(c_v(k'+1) - c_v(k')\right) \\ &= \sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} (c_v(k'+1) - c_v(k')) \\ &\leq \sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} \left(\alpha^{1-\frac{L_v(k')}{L_v}} \cdot \log\alpha\right), \quad \text{by Inequality (10)} \\ &= \log\alpha \sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} \left(\left(\alpha^{1-\frac{L_v(k')}{L_v}} - 1\right) + 1\right) \\ &= \log\alpha \sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} \left(w_{\langle v',v''\rangle}(k') + 1\right) \\ &= \log\alpha \left(\sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} \omega_{\langle v',v''\rangle}(k') + \sum_{\langle v',v''\rangle \in P(k')\cap E'_v(k')} 1\right) \\ &\leq (\sigma_v + (n-1))\log\alpha, \end{aligned} \quad (13)$$

where Inequality (13) holds because request $r_{k'}$ is admitted only if it meets the admission control policy (i), and any routing path in $G'(k')$ contains no more than $n-1$ node-derived edges. Similarly, the cost sum of edges by request $r_{k'}$ is

$$\begin{aligned} & \sum_{e\in E}\left(c_e(k'+1) - c_e(k')\right) \\ &= \sum_{e' \in P(k')\cap E'_e(k')} (c_e(k'+1) - c_e(k')) \\ &\leq \sum_{e' \in P(k')\cap E'_e(k')} (\beta^{1-\frac{B_e(k')}{B_e}} \cdot b_{k'} \cdot \log\beta), \quad \text{by Inequality (12)} \\ &= b_{k'}\log\beta \sum_{e' \in P(k')\cap E'_e(k')} ((\beta^{1-\frac{B_e(k')}{B_e}} - 1) + 1) \\ &= b_{k'}\log\beta \left(\sum_{e' \in P(k')\cap E'_e(k')} (\beta^{1-\frac{B_e(k')}{B_e}} - 1) + \sum_{e' \in P(k')} 1\right) \\ &= b_{k'}\log\beta \left(\sum_{e' \in P(k'\cap E'_e(k'))} \omega_{e'}(k') + \sum_{e' \in P(k')\cap E'_e(k')} 1\right) \\ &\leq b_{k'} \cdot (\sigma_e + n - 1)\log\beta. \end{aligned} \quad (14)$$

where Inequality (14) holds because request $r_{k'}$ is admitted only if it meets the admission control policy (ii), and follows the fact that any routing path in $G'(k')$ contains no more than $n-1$ link-derived edges. Notice that $c_v(1) = c_e(1) = 0$ for all $v \in V$ and $e \in E$. Thus, the cost sum of all nodes when request r_k arrives is

$$\begin{aligned} \sum_{v\in V} c_v(k) &= \sum_{k'=1}^{k-1}\sum_{v\in V}(c_v(k'+1) - c_v(k')) \\ &= \sum_{k'\in\mathcal{S}(k)}\sum_{v\in V}(c_v(k'+1) - c_v(k')) \\ &\leq \sum_{k'\in\mathcal{S}(k)} ((\sigma_v + (n-1))\log\alpha) \\ &= ((\sigma_v + (n-1))\log\alpha) \sum_{k'\in\mathcal{S}(k)} 1 \\ &= |\mathcal{S}(k)|(\sigma_v + (n-1))\log\alpha, \end{aligned} \quad (15)$$

where Inequality (15) follows from Inequality (13). Likewise, the cost sum of all edges for routing $|\mathcal{S}(k)|$ unicast requests by the online algorithm is

$$\begin{aligned} \sum_{e\in E} c_e(k) &= \sum_{k'=1}^{k-1}\sum_{e\in E}(c_e(k'+1) - c_e(k')) \\ &= \sum_{k'\in\mathcal{S}(k)}\sum_{e\in E}(c_e(k'+1) - c_e(k')) \\ &\leq \sum_{k'\in\mathcal{S}(k)} (b_{k'}(\sigma_e + (n-1))\log\beta) \\ &= ((\sigma_e + (n-1))\log\beta) \sum_{k'\in\mathcal{S}(k)} b_{k'} \\ &\leq \mathbb{B}(k)(\sigma_e + (n-1))\log\beta, \end{aligned} \quad (16)$$

where $\mathbb{B}(k) = \sum_{k'\in\mathcal{S}(k)} b_{k'}$, and Inequality (16) follows from Inequality (14). □

Proof for Lemma 2

Proof. Suppose that given an SDN G, there are an optimal offline algorithm and the proposed online algorithm for the problem of concern in G. Assuming that each request is revealed to both algorithms one by one. Each algorithm either admits a request by allocating the demanded resources to the request or rejects the request. The optimal offline algorithm may reject a request that is admitted by the proposed online algorithm or vice versa. Thus, for a given monitoring period, the proposed online algorithm and the optimal offline algorithm may accept different sets of requests, resulting in a difference between the resource availability in the resulting networks by these two algorithms when a request arrives. Denote by $G_{opt}(k')$ and $G(k')$ the networks to which the optimal offline algorithm and the proposed online algorithm are applied prior to the arrival of request $r_{k'}$, respectively.

Consider a unicast request $r_{k'}$ that is admitted by the optimal offline algorithm yet rejected by the proposed online algorithm. Since $r_{k'}$ is admitted by the optimal offline algorithm, it

means that the optimal offline algorithm is able to admit $r_{k'}$ into $G_{opt}(k')$ using a path $P_{opt}(k')$ in $G_{opt}(k')$. We can check whether $G'(k')$ contains $P_{opt}(k')$, and there are exactly two cases. Case 1: every node and edge in $P_{opt}(k')$ has its corresponding edges in $G'(k')$, or case 2: at least one node or one link in $P_{opt}(k')$ does not have a corresponding edge in $G'(k')$, because its available resource cannot meet the demand of $r_{k'}$. In the following, we will argue that in both cases, $\sum_{e\in P_{opt}(k')}\omega_e(k') \geq \min\{\sigma_v, \sigma_e\} = |V| - 1 = n-1$, where $\omega_e(k')$ is calculated based on the availability and workload of the resource in $G(k')$.

Case 1: If every node and edge in $P_{opt}(k')$ has a corresponding edge in $G'(k')$ as well, there is at least one path in $G'(k')$ from $s'_{k'}$ to $t'_{k'}$, namely the one corresponds to $P_{opt}(k')$. Consequently, the proposed online algorithm must be able to find a path $P(k')$ such that $P(k')$ can meet the resource demand of request $r_{k'}$ and $\sum_{e\in P(k')}\omega_e(k') \leq \sum_{e\in p}\omega_e(k')$ for any path p in $G'(k')$ from $s'_{k'}$ to $t'_{k'}$ including $P_{opt}(k')$. Since $r_{k'}$ is rejected by the online algorithm, the length of $P(k')$ is no less than the given threshold, i.e., $\sum_{e\in P(k')}\omega_e(k') \geq \min\{\sigma_v, \sigma_e\}$. Thus, $\min\{\sigma_v, \sigma_e\} \leq \sum_{e\in P(k')}\omega_e(k') \leq \sum_{e\in P_{opt}(k')}\omega_e(k')$.

Case 2: At least one node or one edge in $P_{opt}(k')$ does not have a corresponding edge in $G'(k')$. Recall that a node or a link is not included only if its residual capacity cannot satisfy the resource demands of $r_{k'}$. This case thus can be further divided into two subcases: (a) if there is a node with no available table entry to route the message for the unicast request, then there is a node-derived edge $e' = \langle v', v''\rangle \in P_{opt}(k')$ in $G'(k')$ such that $L_v(k') < 1$. Consequently, the length of $P_{opt}(k')$ is greater than σ_v:

$$\begin{aligned}\sum_{e\in P_{opt}(k')}\omega_e(k') &\geq \omega_{\langle v',v''\rangle}(k') = \alpha^{1-\frac{L_v(k')}{L_v}} - 1\\ &> \alpha^{1-\frac{1}{L_v}} - 1, \quad \text{since } L_v(k') < 1\\ &\geq \alpha^{1-\frac{1}{\log\alpha}} - 1, \quad \text{since } 2n \leq \alpha \leq 2^{L_{\min}} \leq 2^{L_v}\\ &= \frac{\alpha}{2} - 1 \geq \sigma_v, \quad \text{by the assumption of that } \alpha \geq 2n.\end{aligned}$$

(b) If there is an edge in any routing path found by the online algorithm without sufficient bandwidth to route request $r_{k'}$, then there exists an edge $e' = \langle v'', u'\rangle \in P_{opt}(k')$ in $G'(k')$ such that $B_{(v,u)}(k') < b_{k'}$. Therefore, the length of $P_{opt}(k')$ is greater than σ_e:

$$\begin{aligned}\sum_{e\in P_{opt}(k')}\omega_e(k') &\geq \omega_{\langle v'',u'\rangle}(k') = \beta^{1-\frac{B_{(v,u)}(k')}{B_{(v,u)}}} - 1\\ &> \beta^{1-\frac{b_{k'}}{B_{(u,v)}}} - 1, \quad \text{since } B_{(v,u)}(k') < b_{k'}\\ &\geq \beta^{1-\frac{1}{\log\beta}} - 1, \quad \text{since } 2n \leq \beta \leq 2^{B_{\min}/b_{\max}} \leq 2^{B_{(u,v)}/b_{k'}}\\ &= \frac{\beta}{2} - 1 \geq \sigma_e, \quad \text{by the assumption of that } \beta \geq 2n.\end{aligned}$$

□

Proof for Theorem 1

Proof. Let $\mathbb{B}_{opt}(k)$ be the total bandwidth of requests admitted by an optimal offline algorithm when request r_k arrives, we then have

$$\begin{aligned}(n-1)(\mathbb{B}_{opt}(k) - \mathbb{B}(k)) &\leq (n-1)\sum_{k'\in\mathcal{R}(k)} b_{k'}\\ &= \sum_{k'\in\mathcal{R}(k)} b_{k'}(n-1)\\ &\leq \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{e\in P_{opt}(k')}\omega_e(k')\right)\\ &\leq \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{e\in P_{opt}(k)}\omega_e(k)\right) \qquad (17)\end{aligned}$$

$$\begin{aligned}&= \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')}\omega_{\langle v',v''\rangle}(k) + \sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')}\omega_{\langle u'',v'\rangle}(k)\right)\\ &= \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')}\frac{c_v(k)}{L_v} + \sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')}\frac{c_{(u,v)}(k)}{B_{(u,v)}}\right)\\ &= \sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')} b_{k'}\frac{c_v(k)}{L_v}\\ &\quad + \sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')} b_{k'}\frac{c_{(u,v)}(k)}{B_{(u,v)}}\\ &\leq \sum_{v\in V} c_v(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')}\frac{b_{k'}}{L_v}\\ &\quad + \sum_{(u,v)\in E} c_{(u,v)}(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')}\frac{b_{k'}}{B_{(u,v)}} \qquad (18)\end{aligned}$$

$$\begin{aligned}&= \sum_{v\in V} c_v(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')} b_{k'}}{L_v}\\ &\quad + \sum_{(u,v)\in E} c_{(u,v)}(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')} b_{k'}}{B_{(u,v)}}\\ &\leq \sum_{v\in V} c_v(k)\gamma + \sum_{e\in E} c_{(u,v)}(k) \qquad (19)\end{aligned}$$

$$= \gamma\sum_{v\in V} c_v(k) + \sum_{e\in E} c_{(u,v)}(k) \qquad (20)$$

$$\begin{aligned}&\leq \gamma|S(k)|(\sigma_v + (n-1))\log\alpha\\ &\quad + \mathbb{B}(k)(\sigma_e + (n-1))\log\beta, \quad \text{by Lemma 1}\\ &= 2(n-1)(\gamma|S(k)|\log\alpha + \mathbb{B}(k)\log\beta). \qquad (21)\end{aligned}$$

Notice that Inequality (17) holds because the utilization of each resource does not decrease and consequently the weight of any edge in $G'(k)$ does not decrease with more request admissions, i.e., $\omega_e(k') \leq \omega_e(k)$ for any edge $e \in E'(k)$ and any k' with $1 \leq k' \leq k$. Inequality (18) holds since $\sum_{i=1}^{p}\sum_{j=1}^{q} A_iB_j \leq \sum_{i=1}^{p} A_i \sum_{j=1}^{q} B_j$ with $A_i \geq 0$ and $B_j \geq 0$.

The proof of Inequality (20) proceeds as follows. For any switch node $v \in V$, each forwarding table entry can be used to admit a request with bandwidth at most $b_{\max} (\leq B_{\min}/\log\beta = \gamma)$, and the forwarding table at each node v has L_v entries. Thus, the accumulated bandwidth of all unicast requests using switch node v as their relay node is no more than $L_v \cdot b_{\max}$. Hence, the accumulated bandwidth of all admitted requests through node v by an optimal offline algorithm is no more than $L_v \cdot b_{\max}$, i.e., $\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')} b_{k'} \leq L_v \cdot b_{\max} \leq \gamma \cdot L_v$, and $(\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v(k')} b_{k'})/L_v \leq \gamma$. Meanwhile, all algorithms, including optimal offline algorithms for the problem of concern, the total amount of bandwidth used in any link is no more than its capacity, thus, for every link $e \in E$, the accumulated bandwidth of all admitted requests on it by an optimal offline algorithm is no more than its capacity, i.e., $\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')} b_{k'} \leq B_e$. Therefore, $(\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e(k')} b_{k'})/B_e \leq 1$. By Inequality (21), we have

$$\begin{aligned}\frac{\mathbb{B}_{opt}(k) - \mathbb{B}(k)}{\mathbb{B}(k)} &\leq 2(\gamma\frac{|S(k)|}{\mathbb{B}(k)}\log\alpha + \log\beta)\\ &\leq 2(\gamma\log\alpha + \log\beta), \qquad (22)\end{aligned}$$

where the last step follows because $\mathbb{B}(k) = \sum_{k'\in S(k)} b_{k'}$ and $b_{k'} \geq 1$. From Inequality (22), we have

$$\frac{\mathbb{B}_{opt}(k)}{\mathbb{B}(k)} \le 2(\gamma \log\alpha + \log\beta) + 1,$$
$$= O(\log n), \text{ when } \alpha = \beta = 2|V| = 2n. \quad (23)$$

The auxiliary graph $G'(k) = (V'(k), E'(k))$ contains $|V'(k)|$ $(= 2|V|)$ nodes and $|E'(k)|$ $(= |V| + 2|E|)$ edges, its construction thus takes $O(|V'(k)| + |E'(k)|) = O(|V| + |E|)$ time. In addition, finding a shortest path in $G'(k)$ takes $O(|V'(k)|^2) = O(|V|^2)$ time. Algorithm 1 therefore takes $O(|V|^2 + |V| + |E|) = O(|V|^2)$ time. □

Appendix B

Proof for Lemma 3

Proof. Consider an admitted multicast request $r_{k'} \in \mathcal{S}(k)$ by the online algorithm. If the edge derived from a switch node $v \in V$ is not in $T(k')$, then $c_v(k'+1) - c_v(k') = 0$. The cost sum of all nodes in G for admitting multicast request $r_{k'}$ is

$$\begin{aligned}
&\sum_{v\in V}(c_v(k'+1) - c_v(k')) \\
&= \sum_{v\in V\cap T(k')}(c_v(k'+1) - c_v(k')) \\
&= \sum_{v\in T(k')}\left(L_v\left(\alpha^{1-\frac{L_v(k'+1)}{L_v}} - 1\right) - L_v\left(\alpha^{1-\frac{L_v(k')}{L_v}} - 1\right)\right) \\
&= \sum_{v\in T(k')}\left(L_v\alpha^{1-\frac{L_v(k')}{L_v}}\left(\alpha^{\frac{L_v(k')-L_v(k'+1)}{L_v}} - 1\right)\right) \\
&\le \sum_{v\in T(k')}\left(L_v\alpha^{1-\frac{L_v(k')}{L_v}}\left(\alpha^{\frac{1}{L_v}} - 1\right)\right) \quad (24)
\end{aligned}$$

$$= \sum_{v\in T(k')}\left(L_v\alpha^{1-\frac{L_v(k')}{L_v}}\left(2^{\log\alpha\cdot\frac{1}{L_v}} - 1\right)\right)$$
$$\le \sum_{v\in T(k')} L_v\alpha^{1-\frac{L_v(k')}{L_v}}\cdot\log\alpha\cdot\frac{1}{L_v} \quad (25)$$

$$\le (\sigma_v + (n-1))\log\alpha, \quad (26)$$

where Inequality (24) holds, because only one forwarding table entry in the switch table at node v is required to admit multicast request $r_{k'}$ [4,22]. Inequality (25) follows, as $2^x - 1 \le x$ with $0 \le x \le 1$, and $\alpha \le 2^{L_{\min}} \le 2^{L_v/d_v}$, and Inequality (26) holds from the fact that request $r_{k'}$ is admitted only if the admission control policy (i) is met, and a multicast tree cannot have more than $n-1$ node-derived edges.

Notice that $c_v(1) = 0$ for all $v \in V$. The cost sum of nodes by routing all requests in $\mathcal{S}(k)$ is

$$\begin{aligned}
\sum_{v\in V} c_v(k) &= \sum_{k'=1}^{k-1}\sum_{v\in V}(c_v(k'+1) - c_v(k')) \\
&= \sum_{k'\in\mathcal{S}(k)}\sum_{v\in V}(c_v(k'+1) - c_v(k')) \\
&\le \sum_{k'\in\mathcal{S}(k)}\big(\sigma_v + (n-1)\big)\log\alpha, \quad \text{by Inequality (26)} \\
&= |\mathcal{S}(k)|(\sigma_v + (n-1))\log\alpha.
\end{aligned}$$

Thus, Inequality (6) holds.

Inequality (7) can be similarly proven by adopting the technique for Inequality (14), omitted. □

Proof for Lemma 4

Proof. Suppose that given an SDN G and there are an optimal offline algorithm and the proposed online algorithm for the network capacity maximization problem for online multicasting. Assuming that each request is revealed to both algorithms one by one. Each algorithm can either admit the request by allocating the demanded resources to it or reject it. However, the optimal offline algorithm may reject a request that is admitted by the proposed online algorithm, or vice versa. Thus, for a given monitoring period, the proposed online algorithm and the optimal offline algorithm may accept different sets of requests, resulting in a difference between the resource availability in the resulting networks by these two algorithms when a request arrives. Denote by $G_{opt}(k')$ and $G(k')$ the networks to which the optimal offline algorithm and the proposed online algorithm are applied prior to the arrival of request $r_{k'}$, respectively.

Consider a multicast request $r_{k'}$ that is admitted by the optimal offline algorithm yet rejected by the proposed online algorithm. Since $r_{k'}$ is admitted by the optimal offline algorithm, it means that the optimal offline algorithm is able to admit $r_{k'}$ into $G_{opt}(k')$ using a tree $T_{opt}(k')$ in $G_{opt}(k')$. We check whether $G'(k')$ derived from $G(k')$ contains $T_{opt}(k')$, and there are exactly two cases. Case 1: every node and edge in $T_{opt}(k')$ has a corresponding edge in $G'(k')$, or case 2: at least one node or one link in $T_{opt}(k')$ does not have a corresponding edge in $G'(k')$, because its available resource capacity cannot meet the demand of $r_{k'}$. In the following, we will show that for both cases, $\sum_{e\in T_{opt}(k')}\omega_e(k') \ge \frac{\min\{\sigma_v,\sigma_e\}}{K^\epsilon}$, where $\omega_e(k')$ is calculated based on the availability and workload of the resource in $G(k')$.

Case 1. If every node and edge in $T_{opt}(k')$ has a corresponding edge in $G'(k')$, there is at least one tree in $G'(k')$ that roots at $s'_{k'}$ and spans the nodes in $D_{k'}$, namely the one corresponds to $T_{opt}(k')$. Consequently, the proposed online algorithm must be able to find a tree $T(k')$ in $G'(k')$ such that $T(k')$ can meet the resource demand of request $r_{k'}$ and $\sum_{e\in T}\omega_e(k') \ge \frac{\sum_{e\in T(k')}\omega_e(k')}{|D_{k'}|^\epsilon} \ge \frac{\sum_{e\in T(k')}\omega_e(k')}{K^\epsilon}$ for any tree T in $G'(k')$ that roots at $s'_{k'}$ and spans the nodes in $D_{k'}$, including $T_{opt}(k')$. Since $r_{k'}$ is rejected by the online algorithm, the weighted sum of edges in $T(k')$ is no less than the given threshold, i.e., $\frac{\sum_{e\in T(k')}\omega_e(k')}{K^\epsilon} \ge \frac{\min\{\sigma_v,\sigma_e\}}{K^\epsilon}$. Thus, $\sum_{e\in T_{opt}(k')}\omega_e(k') \ge \frac{\sum_{e\in T(k')}\omega_e(k')}{K^\epsilon} \ge \frac{\min\{\sigma_v,\sigma_e\}}{K^\epsilon}$.

Case 2. At least one node or one edge in $T_{opt}(k')$ does not have a corresponding edge in $G'(k')$. Recall that a node or a link is not included only if its residual capacity cannot satisfy the resource demands of $r_{k'}$. This case thus can be further divided into two subcases: (a) if there is not any multicast tree with sufficient available table entries to route the message of the request, then there must have a node-derived edge $e' = \langle v'', w_{vu}\rangle \in T_{opt}(k') \cap E'_v(k')$ derived from node $v \in V$ such that $L_v(k') < d_v \le K$. Consequently, the weighted sum of edges in $T_{opt}(k')$ is greater than σ_v/K^ϵ as follows.

$$\begin{aligned}
\sum_{e\in T_{opt}(k')}\omega_e(k') &\ge \omega_{\langle v',v''\rangle}(k') = \alpha^{1-\frac{L_v(k')}{L_v}} - 1 \\
&> \alpha^{1-\frac{d_v}{L_v}} - 1, \quad \text{since } L_v(k') < d_v \\
&\ge \alpha^{1-\frac{1}{\log\alpha}} - 1, \quad \text{since } 2n \le \alpha \le 2^{L_{\min}} \le 2^{L_v/d_v} \\
&= \frac{\alpha}{2} - 1 \ge \sigma_v \ge \frac{\sigma_v}{K^\epsilon}.
\end{aligned}$$

(b) If there is not any multicast tree with sufficient bandwidth on one of its links to admit multicast request k', then the weighted sum of edges in $T_{opt}(k')$ is greater than σ_e/K^ϵ, which can be proved using the similar method as the one for online unicasting in the proof body of Lemma 2, and thus omitted. □

Proof for Theorem 2

Proof. Let $\mathbb{B}_{opt}(k)$ be the total bandwidth of all admitted multicast requests by an optimal offline algorithm. Combining Lemmas 3 and

4, we have

$$\begin{aligned}
\frac{(n-1)}{K^{\epsilon}}(\mathbb{B}_{opt}(k)-\mathbb{B}(k)) &\le \frac{(n-1)}{K^{\epsilon}}\sum_{k'\in\mathcal{R}(k)} b_{k'} \\
&= \sum_{k'\in\mathcal{R}(k)} b_{k'}\frac{(n-1)}{K^{\epsilon}} \\
&\le \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{e\in T_{opt}(k')}\omega_e(k')\right), \quad \text{by Lemma 4} \\
&= \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')}\omega_{\langle v'',w_{vu}\rangle}(k')\right. \\
&\quad \left.+\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')}\omega_{\langle w_{vu},u'\rangle}(k')\right) \\
&\le \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')}\omega_{\langle v'',w_{vu}\rangle}(k)\right. \\
&\quad \left.+\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')}\omega_{\langle w_{vu},u'\rangle}(k)\right) \qquad (27)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{k'\in\mathcal{R}(k)} b_{k'}\left(\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')}\frac{c_v(k)}{L_v}+\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')}\frac{c_{(v,u)}(k)}{B_{(v,u)}}\right) \\
&= \sum_{k'\in\mathcal{R}(k)}\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')} b_{k'}\frac{c_v(k)}{L_v} \\
&\quad+\sum_{k'\in\mathcal{R}(k)}\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')} b_{k'}\frac{c_{(v,u)}(k)}{B_{(v,u)}} \\
&\le \sum_{v\in V} c_v(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')}\frac{b_{k'}}{L_v} \\
&\quad+\sum_{(v,u)\in E} c_{(v,u)}(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')}\frac{b_{k'}}{B_{(v,u)}} \\
&= \sum_{v\in V} c_v(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v'',w_{vu}\rangle\in T_{opt}(k')\cap E'_v(k')} b_{k'}}{L_v} \\
&\quad+\sum_{(v,u)\in E} c_{(v,u)}(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle w_{vu},u'\rangle\in T_{opt}(k')\cap E'_e(k')} b_{k'}}{B_{(v,u)}} \\
&\le \sum_{v\in V} c_v(k)\gamma+\sum_{e\in E} c_e(k) \le \gamma\sum_{v\in V} c_v(k)+\sum_{e\in E} c_e(k) \\
&\le \gamma|S(k)|(\sigma_v+(n-1))\log\alpha+\mathbb{B}(k)(\sigma_e+(n-1))\log\beta \\
&= 2(n-1)(\gamma|S(k)|\log\alpha+\mathbb{B}(k)\log\beta) \qquad (28)
\end{aligned}$$

where Inequality (27) follows because the resource utilization is always nondecreasing.

From Inequality (28), we have

$$\frac{\mathbb{B}_{opt}(k)-\mathbb{B}(k)}{\mathbb{B}(k)} \le 2K^{\epsilon}\left(\frac{|\mathcal{S}(k)|}{\mathbb{B}(k)}\gamma\log\alpha+\log\beta\right). \qquad (29)$$

Hence,

$$\begin{aligned}
\frac{\mathbb{B}_{opt}(k)}{\mathbb{B}(k)} &\le 2K^{\epsilon}\left(\frac{|\mathcal{S}(k)|}{\mathbb{B}(k)}\gamma\log\alpha+\log\beta\right)+1, \\
&\le 2K^{\epsilon}(\gamma\log\alpha+\log\beta)+1, \text{since } \frac{|\mathcal{S}(k)|}{\mathbb{B}(k)}\le 1, \\
&\le 2K^{\epsilon}(\gamma\log(2n))+1, \text{if } \alpha=\beta=2|V|=2n, \\
&= O(K^{\epsilon}\log n).
\end{aligned}$$

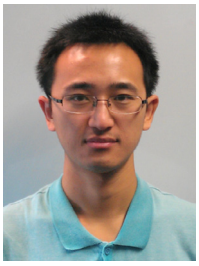
The rest is to analyze the time complexity of Algorithm 2. Recall that given an SDN $G=(V,E)$, we first construct an auxiliary graph $G'(k)=(V'(k),E'(k))$, which contains $|V'(k)|$ $(=|V|+|E|)$ nodes and $|E'(k)|$ $(=|V|+4|E|)$ edges. Thus, the construction of $G'(k)$ takes $O(|V|+|E|)$ time. It then takes $O(|V'(k)|^{1/\epsilon}|D_k|^{2/\epsilon})$ time to find an approximate Steiner tree in $G'(k)$ for a multicast request r_k, by employing the algorithm in [6] As a result, Algorithm 2 takes $O((|V|+|E|)^{1/\epsilon}K^{2/\epsilon})$ time. □

References

[1] S. Agarwal, M. Kodialam, T.V. Lakshman, Traffic engineering in software defined networks, in: Proc. of INFOCOM, IEEE, 2013.
[2] J. Aspnes, Y.A. Yossi, A. Fiat, S. Plotkin, O. Waarts, On-line routing of virtual circuits with applications to load balancing and machine scheduling, J. ACM 44 (3) (1997) 486–504.
[3] S. Banerjee, K. Kannan, Tag-in-tag: efficient flow table management in SDN switches, in: Proc. of International Conference on Network and Service Management, IEEE, 2014.
[4] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, M. Horowitz, Forwarding metamorphosis: fast programmable match-action processing in hardware for sdn, in: Proc. of SIGCOMM, ACM, 2013.
[5] Z. Cao, M. Kodialam, T.V. Lakshman, Traffic steering in software defined networks: planning and online routing, in: Proc. of SIGCOMM Workshop Distrib. Cloud Comput. (DCC'14), ACM, 2014.
[6] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, M. Li, Approximation algorithms for directed Steiner problems, J. Algorithms 33 (1) (1998) 73–91. Elsevier.
[7] R. Cohen, L. Eytan, J. Naor, D. Raz, On the effect of forwarding table size on SDN network utilization, in: Proc. of INFOCOM, IEEE, 2014.
[8] M. Furer, B. Raghavachari, Approximating the minimum-degree Steiner tree to within one of optimal, J. Algorithms 17 (3) (1994) 409–423.
[9] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.
[10] H. Huang, S. Guo, P. Li, W. Liang, Cost minimization for rule caching in software defined networks, IEEE Trans. Parallel Distrib. Syst. 27 (4) (2016) 1007–1016.
[11] L. Huang, H. Hung, C. Lin, D. Yang, Scalable Steiner tree for multicast communications in software-defined networking, Comput. Res. Repos. (CoRR) (2014). Abs/1404.3454.
[12] L. Huang, H. Hsu, S. Shen, D. Yang, W. Chen, Multicast traffic engineering for software-defined networks, in: Proc. of INFOCOM, IEEE, 2016.
[13] M. Huang, W. Liang, Z. Xu, M. Jia, S. Guo, Throughput maximization in software-defined networks with consolidated middleboxes, in: Proc of 41st Annual IEEE Conference on Local Computer Networks (LCN), IEEE, 2016.
[14] M. Huang, W. Liang, Z. Xu, S. Guo, Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes, IEEE Trans. Netw. Serv. Manage. 14 (3) (2017) 631–645.
[15] S. Jain, et al., B4: experience with a globally-deployed software defined WAN, in: Proc. of SIGCOMM, ACM, 2013.
[16] Y. Kanizo, D. Hay, I. Keslassy, Palette: distributing tables in software-defined networks, in: Proc. of INFOCOM, IEEE, 2013.
[17] N. Katta, O. Alipourfard, J. Rexford, D. Walker, Infinite cacheflow in software-defined networks, in: Proc. of HotSDN, ACM, 2014.
[18] K. Kar, M. Kodialam, T.V. Lakshman, L. Tassiulas, Routing for network capacity maximization in energy-constrained ad hoc networks, in: Proc. of INFOCOM, IEEE, 2003.
[19] D. Kreutz, F.M.V. Ramos, P. Verissimo, et al., Software-defined networking: a comprehensive survey, Proc. of IEEE 103 (1) (2015) 14–76. IEEE
[20] W. Liang, X. Guo, On-line multicasting for network capacity maximization in energy-constrained ad hoc networks, IEEE Trans. Mobile Comput. 5 (9) (2006) 1215–1227.
[21] W. Liang, Y. Liu, On-line data gathering for maximizing network lifetime in sensor networks, IEEE Trans. Mobile Comput. 6 (1) (2007) 2–11.
[22] T. Mishra, S. Sahni, PETCAM - a power efficient tcam architecture for forwarding tables, IEEE Trans. Comput. 61 (1) (2012) 3–17.
[23] B.A.A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: past, present, and future of programmable networks, Commun. Surv. Tutor. 16 (3) (2014) 1617–1634. IEEE.
[24] S. Plotkin, Competitive routing of virtual circuits in ATM networks, IEEE J. Sel. Areas Commun. 13 (6) (1995) 1128–1136.
[25] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, SIMPLE-fying middlebox policy enforcement using SDN, in: Proc. of SIGCOMM, ACM, 2013.
[26] E. Spitznagel, D. Taylor, J. Turner, Packet classification using extended TCAMs, in: Proc. of ICNP, IEEE, 2003.
[27] Z. Xu, W. Liang, W. Xu, M. Jia, S. Guo, Efficient algorithms for capacitated cloudlet placements, IEEE Trans. Parallel Distrib. Syst. 27 (10) (2016) 2866–2880.

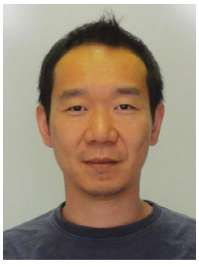
Meitian Huang currently is a Ph.D. candidate in Computer Science at the Australian National University. His research interests include software-defined networking and cloud computing. He received the B.Sc. degree with the first class Honours in Computer Science at the Australian National University in 2015.

Weifa Liang (M'99–SM'01) received the Ph.D. degree from the Australian National University in 1998, the M.E. degree from the University of Science and Technology of China in 1989, and the B.Sc. degree from Wuhan University, China in 1984, all in computer science. He is currently a full professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, cloud computing, Software-Defined Networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

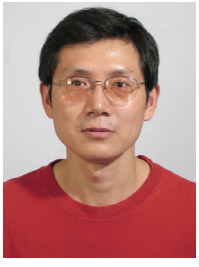


Zichuan Xu received his Ph.D. degree from the Australian National University in 2016, M.E. and B.Sc. degrees from Dalian University of Technology in China in 2011 and 2008, all in Computer Science. He was a research associate at University College London. He currently is an Associate Professor in School of Software at Dalian University of Technology. His research interests include cloud computing, software-defined networking, wireless sensor networks, algorithmic game theory, and optimization problems.

Wenzheng Xu (M'15) received the B.Sc., M.E., and Ph.D. degrees in computer science from Sun Yat-Sen University, Guangzhou, P.R. China, in 2008, 2010, and 2015, respectively. He currently is a Special Associate Professor at the Sichuan University and was a visitor at the Australian National University. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory. He is a member of the IEEE.



Song Guo (M'02-SM'11) received his Ph.D. in computer science from University of Ottawa. He is currently a full professor at Department of Computing, The Hong Kong Polytechnic University. Prior to joining PolyU, he was a full professor with the University of Aizu, Japan. His research interests are mainly in the areas of cloud and green computing, big data, wireless networks, and cyber-physical systems. He has published over 300 conference and journal papers in these areas and received multiple best paper awards from IEEE/ACM conferences. His research has been sponsored by JSPS, JST, MIC, NSF, NSFC, and industrial companies. Dr. Guo has served as an editor of several journals, including IEEE TPDS, IEEE TETC, IEEE TGCN, IEEE Communications Magazine, and Wireless Networks. He has been actively participating in international conferences serving as general chair and TPC chair. He is a senior member of IEEE, a senior member of ACM, and an IEEE Communications Society Distinguished Lecturer.



Yinlong Xu received his B.S. in Mathematics from Peking University in 1983, and M.S. and Ph.D. in Computer Science from University of Science and Technology of China (USTC) in 1989 and 2004, respectively. He is currently a full professor with the School of Computer Science and Technology at USTC. Prior to that, he served the Department of Computer Science and Technology at USTC as an assistant professor, a lecturer, and an associate professor. His research interests include network coding, wireless networks, combinatorial optimization, design and analysis of parallel algorithms, and parallel programming tools. He received the Excellent Ph.D. Advisor Award of Chinese Academy of Sciences in 2006.