# Dynamic Routing for Network Throughput Maximization in Software-Defined Networks

Meitian Huang[†], Weifa Liang[†], Zichuan Xu[†], Wenzheng Xu[‡†], Song Guo[¶], and Yinlong Xu[∗]

† Australian National University, Canberra, ACT 2601, Australia
‡ Sichuan University, Chengdu 610065, P. R. China
¶ Aizu University, Japan
∗ University of Science and Technology of China, Hefei 230026, P. R. China

Email: u4700480@anu.edu.au, wliang@cs.anu.edu.au, edward.xu@anu.edu.au, wenzheng.xu3@gmail.com, sguo@u-aizu.ac.jp, ylxu@ustc.edu.cn

***Abstract*—Software-Defined Networking (SDN) has emerged as the paradigm of the next-generation networking through separating the data control plane from the data plane. The forwarding routing table at each of its switch nodes is usually implemented by expensive and power-hungry Ternary Content Addressable Memory (TCAM) that only has limited number of entries, and the bandwidth at each of its links is bounded too. Under this new network architecture, providing a quality service to users by admitting user requests to meet their resource demands is challenging, and very little attention has ever been paid in this regard. In this paper, we will study online unicast and multicast request admissions in SDNs with the aim to maximize the network throughput under both critical network resources and user bandwidth demand constraints, for which we first propose a novel model to characterize the usage costs of node and link resources. We then devise efficient online algorithms for unicast and multicast requests. We also analyze the competitive ratios of the proposed online algorithms, which are $O(\log n)$ and $O(K^{\epsilon} \log n)$ for unicasting and multicasting, respectively, where $n$ is the network size, $K$ is the maximum number of members in a multicast request, and $\epsilon$ 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.**

## I. Introduction

Despite of their widespread adoption, traditional networks have been considered time-consuming and error-prone to configure according to increasingly sophisticated high-level policies and costly to reconfigure in response to faults, load, and changes [16]. The vertical integration between the control and data planes exacerbates the problems even further. Software-Defined Networking (SDN) is an emerging networking paradigm that creates an opportunity for solving this longstanding problems 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. In addition to simplifying policy enforcement and network (re)configuration and evolution, such a separation between the control plane and the data plane also paves the way for dynamic control and management of packet forwarding and processing in switches, which is expected to ease network management and improve network capacity utilization as well as delay-and-loss performance [1], [5], [8], [11]. SDN thus is becoming a key technology for the next-generation network architecture, and has been applied to many large scale networks, including Internet backbone networks and data-center networks, such as Google's B4 [9].

In a software-defined network, the centralized controller makes global routing decisions and translates high-level policies into forwarding rules to be installed in the forwarding tables at switches. Because of their greater flexibility, forwarding rules in SDN is more complex and requires more storage space compared with the forwarding rules in traditional networks. To match incoming packets and intricate forwarding rules as fast as possible, the forwarding table at each switch is thus normally implemented by the Ternary Content Addressable Memory (TCAM) that supports fast, parallel lookups. TCAM however is expensive [11] and power hungry [20], the capacity of forwarding tables is typically limited to several thousand entries [11]. Such highly restricted capacity of forwarding tables has been recognized as a bottleneck to the scalability of SDN [5], [10], [16], and efficient utilization of forwarding tables to serve a scaling number of forwarding rules while satisfying network policies and constraints is a challenging and important research topic. In addition, network infrastructure providers have expanded at a fast pace in recent years, constantly striving to meet increasingly higher and rapidly changing user demands [17]. To operate a public service at such a large scale, effective management of network bandwidth resources becomes vital, since any service disruption may lead to substantial monetary losses. Therefore, there is an urgent need for a routing scheme that pays as much attention to the forwarding table capacity constraint as it does to other capacity constraints such as the bandwidth capacity of each link.

In this paper, we study novel and challenging online unicast and multicast routing problems in SDNs, with the aim of maximizing network throughput, by jointly considering the forwarding table capacity at each switch and the bandwidth capacity at each Internet link. Unlike most existing studies on online unicast and multicast routing problems in traditional networks that considered either the node capacity [12], [14], [15] or the link bandwidth constraint [2], [18], we take into account both the forwarding table capacity constraint on switches and the bandwidth constraint on links in the network. This is much more challenging due to the need of innovative cost models that can accurately capture the usage costs of two different types of resources and new techniques to analyze the performance of proposed online algorithms. Meanwhile, there are only a few attentions paid to online unicast and multicast routing in SDNs. For example, several studies for unicast routing in SDN explored the capacitated forwarding table [1], [5], [8]. Most of these studies first reduce the node capacity constraint into the node-degree constraint of the network, and then find a node-degree-constrained maximum flow for unicast requests [1], [5]. Such reductions however are limited and may not be applicable in practice, since the node capacity constraint is usually far greater than the maximum degree of nodes. In addition, the only study on a single session online multicast in SDNs that we are aware of is in [8], where the authors considered how to construct a node-degree constrained multicast tree for one multicast request, not a sequence of online multicast requests as we deal with in this paper. In contrast to [1], [5], [8], we consider much more generic online unicast and multicast routing problems in SDNs, assuming that requests arrive one by one without the knowledge of future arrivals. To the best of our knowledge, this is the very first study of online unicast and multicast routing in SDNs, providing guaranteed competitive ratios. The algorithm design and analysis techniques are of independent interest, and they may be applied to other online optimization problems in networks.

The main contributions of this paper are summarized as follows. We are the first to study the admissions of dynamic unicast and multicast requests in SDNs with the aim to maximize the network throughput, by taking both switch node and link capacities and user bandwidth demands into consideration. Specifically, we first propose a novel cost model to capture the usage costs of node and link resources in admissions of a sequence of unicast or multicast requests without the knowledge of future request arrivals. We then devise efficient algorithms for unicast and multicast capacity maximization problems, and analyze their competitive ratios. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are very promising.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III introduces the system model, notions and notations, and problem definitions. Sections IV introduces a cost model for resource usages. Sections V and VI propose online algorithms and analyze their competitive ratios for unicast and multicast routing in software-defined networks, respectively. Section VII evaluates the performance of the proposed algorithms by experimental simulation, and Section VIII concludes the paper.

## II. Related Work

SDN has been envisioned as the next generation network architecture in easing the management of a network and improving network performance. While some studies focused on addressing the challenges in traffic engineering and steering for incremental deployment of SDN-enable devices in existing networks [1], [3], [19], tackling the limited flow table sizes has become the theme of many studies [5], [8], [10]. Specifically, Kanizo *et al.* [10] studied a unicast routing problem for a group of requests in SDN. To overcome the limitation of the forwarding table size, they proposed two approaches to decompose large SDN forwarding tables into several small ones and then distribute these small tables across the network, while maintaining the overall SDN policy semantics. On the other hand, Huang *et al.* [7] tackled the limitation of TACM by selectively caching forwarding rules in local switches and forwarding packets to the centralized controller if necessary. Cohen *et al.* [5] studied the bounded path-degree max flow problem in SDNs subject to the capacity constraint on switch devices. They proposed an approximate solution with high probability. Meanwhile, Huang *et al.* [8] studied the multicast problem in SDNs by devising an approximation algorithm for finding a degree-constrained Steiner tree for one multicast request. However, these mentioned works only considered the switch node capacity for a single group of unicast requests or a single multicast request, which is substantially different from our work in this paper, where we consider online routing of unicast or multicast requests without the knowledge of future request arrivals by taking into account both the capacity at each switch node and the bandwidth capacity at each link.

Since network resources in SDNs are allocated dynamically, the availability of resources exhibits significant elasticity. Given a sequence of unicast or multicast requests that arrive one by one without the knowledge of future request arrivals, determining which requests to be admitted while others should be rejected is nontrivial, as the admitted requests will acquire the resources and heavily impede the later request admissions. Thus, there is a desperate need of a cost metric to measure the network resource consumptions and their utilization, guiding the resource allocations for incoming requests. The key of such a cost metric is to accurately model the availability and utilization of the resources. An exponential function of a specific resource and its utilization ratio is an excellent candidate of the cost metric. This function has been used as a cost metric of online request routing for many different types of networks with different resources, including link bandwidth for online unicast and multicast routing in ATM and virtual circuit networks [2], [18], and node energy for online data gathering [15], unicasting [12], and multicasting [14] in ad hoc and wireless sensor networks. On the other hand, performing online routing in SDNs involves taking into account both the forwarding table at each switch node and the bandwidth at each link simultaneously, making the cost modeling of resource usages in SDNs more difficult. Furthermore, the joint consideration of resources at both nodes and links complicates the analysis of propose solution, since the analyses of the performance (i.e., the competitive ratios) of existing online algorithms for unicast and multicast requests in ATM networks, virtual circuit networks [2], [18], and wireless sensor networks [14] are only based on the cost modeling of a single type of resource at either nodes or links. Thus, in this paper, we develop new techniques to analyze the competitive ratios of online algorithms for unicast and multicast requests.

## III. Preliminaries

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

### A. System model

We consider a software-defined network $G = (V, E)$, where $V$ is the set of SDN-enabled switch nodes, and $E$ is the set of Internet links that connect the switches. Assume that there is an SDN controller for network $G$ to route unicast or multicast requests by installing forwarding rules into the routing tables in switches and allocating bandwidth on links along the routing paths or trees in $G$. Each switch $v \in V$ is equipped with a TCAM forwarding table for packet forwarding, and the table capacity is $L_v$ rule entries. Each link $e \in E$ has a bandwidth capacity $B_e$. Denote by $d_v$ the degree of a switch node $v$ in $G$, i.e., $d_v = |\{u \mid (u, v) \in E\}|$, and let $d_{\max}$ $(= \max\{d_v \mid v \in V\})$ be the maximum degree of switch nodes in $G$.

## B. *User routing requests*

We consider unicast and multicast requests that arrive into the system one by one. Denote by $(s_k, d_k; b_k)$ the $k$-th unicast request, where $s_k$ is its source switch, $d_k$ is its destination switch, and $b_k$ is the number of bandwidth units it needs. Similarly, denote by $(s_k, D_k; b_k)$ the $k$-th multicast request, where $s_k$ is its source switch, $D_k$ is the set of its destination switches, and $b_k$ is the number of bandwidth units it needs. We assume that the bandwidth demanded by each request $r_k$ is at least one unit, i.e., $b_k \geq 1$ for any $k$.

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) to meet its resource demands. Otherwise, the request will be *rejected*.

## C. *Problem definitions*

Given a software-defined network $G = (V, E)$, where $V$ is the set of SDN-enabled switches and $E$ is the set of Internet links between the switches. Under the assumption that a sequence of unicast requests $(s_k, d_k; b_k)$ 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 accumulated bandwidth of unicast requests that have successfully been admitted.

The network capacity maximization problem for online multicasting can be defined similarly. For a sequence of multicast requests $(s_k, D_k; b_k)$ that arrives into the system one by one without the knowledge of future multicast request arrivals, *the network capacity maximization problem for online multicasting* is to maximize the accumulated bandwidth of multicast requests that have successfully been admitted.

## IV. 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 its switch nodes and links. One important characteristic of the resource usage on both switch nodes and links in $G$ is that the marginal costs of resource usage inflate with the increase on the workloads of the resources. A heavily-loaded switch node will spend time and energy on matching a forwarding rule for an incoming network packet compared with a lightly-loaded one, because more rules need to be considered in such a heavily-loaded switch. Thus, when admitting a request, we should make use of cheaper nodes and links to admit the request in terms of usage costs of node and link resources. Let $L_v(k)$ and $B_e(k)$ be the number of available entries in the routing table at switch node $v \in V$ and the residual bandwidth on link $e \in E$, respectively, when the $k$-th requests arrives. We use an exponential function to model the cost $c_v(k)$ of using the resource at each switch node $v$ by request $k$, which is defined as follows.

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

where $\alpha > 1$ is a tuning parameter to be determined later, and $1 - \frac{L_v(k)}{L_v}$ is the utilization ratio of the forwarding table of $v$ when request $k$ arrives. Similarly, the cost $c_e(k)$ of using the bandwidth of link $e \in E$ by request $k$ is defined as

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

where $\beta > 1$ is another tuning parameter similar to $\alpha$, and $1 - \frac{B_e(k)}{B_e}$ is the utilization ratio of the bandwidth of link $e$ when request $k$ arrives.

## V. Online Algorithm for Unicast Routing

In this section, we first describe an online algorithm for the online unicast capacity maximization problem. We then analyze the competitive ratio of the proposed algorithm.

## A. *Online algorithm*

In the following we propose an online algorithm for the network capacity maximization problem for online unicasting, based on the proposed usage cost model. Given a sequence of unicast requests arriving one by one without the knowledge of future arrivals, we first decide which requests to be admitted, and then find a routing path for each admitted request.

The basic idea behind the proposed online algorithm is to find a shortest routing path for each admitted request, where *the length of a routing path* is the weighted sum of nodes and links in the path. To model the usage costs of the resources at nodes and links in the network, an edge-weighted, directed graph $G' = (V', E'; \omega)$ will be constructed from $G$. For each unicast request $k$ with $(s_k, d_k, b_k)$, a corresponding shortest path in $G'$ from node $s'_k$ to node $d'_k$ will be found. Note that the length of an edge 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.

To maximize the accumulated bandwidth capacity of all admitted requests, *an admission control policy* will be adopted. That is, when the length of a routing path is above a given threshold, the request will be rejected. In the following we detail the construction of $G'$ and the online algorithm for the network capacity maximization problem for online unicasting.

The edge-weighted, directed graph $G' = (V', E'; \omega)$ is constructed from $G$ as follows. For each switch node $v \in V$, two nodes $v'$ and $v''$ are added to $V'$, i.e., $V' = \{v', v'' \mid v \in V\}$, and a directed edge $\langle v', v'' \rangle$ is added to $E'$. For each link $(u, v) \in E$, two directed edges $\langle u'', v' \rangle$ and $\langle v'', u' \rangle$ are added to $E'$, i.e., $E' = \{\langle v', v'' \rangle \mid v \in V\} \cup \{\langle v'', u' \rangle, \langle u'', v' \rangle \mid (u, v) \in E\}$. For brevity, we refer to the edges in $G'$ that are derived from switch nodes of $G$ as the *node-derived edges* and the edges in $G'$ that are derived from the links of $G$ as the *link-derived edges*, and denote by $E'_v$ and $E'_e$ the sets of node-derived edges and link-derived edges, respectively. Clearly, $E' = E'_e \cup E'_v$ and $E'_e \cap E'_v = \emptyset$. Depending on its type (node-derived or link-derived edge) and the usage cost of the resource it represents, each edge $e \in E'$ is then assigned a weight as follows.

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

where the weight of each node-derived edge reflects the forwarding table size constraint on its corresponding switch node, while the weight of each link-derived reflects the bandwidth capacity constraint on its corresponding link.

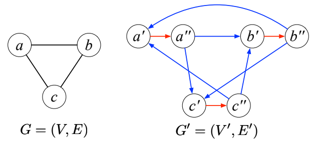


Fig. 1. The construction of $G'(V', E')$ for an SDN $G(V, E)$.

The edge $e' \in E'_e$ derived from an Internet link $e \in E$ is omitted if the residual bandwidth of $e$ is strictly less than $b_k$, because $e$ cannot meet the bandwidth demand of request $k$, and thus plays no role in admitted the $k$-th request. For simplicity, the resulting graph after pruning some edges from it is still denoted as $G'$. Let $P(k)$ be a shortest path in $G'$ from $s'_k$ to $d'_k$. Following the construction of $G'$, the edges in $P(k)$ are the node-derived and link-derived edges alternatively. To avoid admitting some requests that may undermine the performance of the SDN, the following admission control policy will be adopted: a unicast request $k$ will be rejected, if (i) the length of weighted node-derived edges in $P(k)$ is greater than $\sigma_v$, or (ii) the length of the weighted link-derived edges in $P(k)$ is greater than $\sigma_e$, where $\sigma_v$ $(= |V|-1 = n-1)$ and $\sigma_e$ $(= |V|-1 = n-1)$ are pre-determined thresholds. In other words, an incoming request $k$ is admitted if there exists a shortest path $P(k)$ between $s'_k$ and $d'_k$ in $G'$ such that $P(k)$ meets the following requirements:

(i) $\sum_{e=\langle v',v''\rangle \in P(k)\cap E'_v} \omega_e(k) \le \sigma_v$,
(ii) $\sum_{e=\langle v'',u'\rangle \in P(k)\cap E'_e} \omega_e(k) \le \sigma_e$.

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 $k$-th unicast request $(s_k, d_k; b_k)$;
Output: Admit or reject the request, if admitted, a routing path for the request will be delivered.
1: Construct an edge-weighted, directed graph $G' = (V', E'; \omega)$ from a subgraph of $G$ by removing links with residual bandwidth less than $b_k$;
2: Find a shortest path $P(k)$ in $G'$ from $s'_k$ to $d'_k$;
3: if $P(k)$ does not exist then
4:     Reject unicast request $k$;
5: else
6:     If $(\sum_{e \in P(k)\cap E'_v} \omega_e(k) \le \sigma_v)$ and $(\sum_{e\in P(k)\cap E'_e} \omega_e(k) \le \sigma_e)$, then admit request $k$ using $P(k)$ as the routing path; otherwise, reject the request.
7: end if.
```

## *B. Competitive ratio analysis of the online algorithm*

We now analyze the performance of the proposed algorithm. We make use of the following notations. $L_{\min}$ is the minimum table size in the network, i.e., $L_{\min} = \min\{L_v \mid v \in V\}$, $B_{\min}$ is the minimum bandwidth capacity among links, i.e., $B_{\min} = \min\{B_e \mid e \in E\}$, and $b_{\max}$ is the maximum bandwidth demand by any unicast request. We start by showing the upper bound on the cost of nodes and links of $G$ of all admitted unicast requests by Algorithm 1 as of the arrival of request $k$.

**Lemma 1.** *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 unicast requests admitted by the online algorithm, Algorithm 1, until the arrival of request $k$. Let $\alpha$ and $\beta$ be two given values with $2|V| \le \alpha \le 2^{L_{\min}}$ and $2|V| \le \beta \le 2^{B_{\min}/b_{\max}}$. Then, the cost sums of nodes and links are*

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

*and*

$$\sum_{e\in E} c_e(k) \le \mathbb{B}(k)(\sigma_e + n - 1)\log\beta, \tag{2}$$

*respectively, when request $k$ arrives.*

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

$$\begin{aligned}
& c_v(k'+1) - c_v(k') = L_v\big(\alpha^{1-\frac{L_v(k'+1)}{L_v}} - \alpha^{1-\frac{L_v(k')}{L_v}}\big) \\
& = L_v \alpha^{1-\frac{L_v(k')}{L_v}} \big(\alpha^{\frac{L_v(k')-L_v(k'+1)}{L_v}} - 1\big) \\
& \le L_v \alpha^{1-\frac{L_v(k')}{L_v}} \big(\alpha^{\frac{1}{L_v}} - 1\big) && (3)\\
& = L_v \alpha^{1-\frac{L_v(k')}{L_v}} \big(2^{\frac{1}{L_v}\log\alpha} - 1\big) \le L_v \alpha^{1-\frac{L_v(k')}{L_v}} (\log\alpha / L_v) && (4)\\
& = \alpha^{1-\frac{L_v(k')}{L_v}} \log\alpha. && (5)
\end{aligned}$$

where Inequality (3) holds because at most one routing entry is added to the routing table of node $v$, and Inequality (4) holds because $2^x - 1 \le x$ with $0 \le x \le 1$, and $(1/L_v)\log\alpha \le (1/L_v)L_{\min} \le (1/L_v)L_v = 1$.

For any edge $e \in E$, we have

$$\begin{aligned}
& c_e(k'+1) - c_e(k') = B_e \beta^{1-\frac{B_e(k')}{B_e}} \big(\beta^{\frac{B_e(k')-B_e(k'+1)}{B_e}} - 1\big) \\
& \le B_e \beta^{1-\frac{B_e(k')}{B_e}} \big(\beta^{\frac{b_{k'}}{B_e}} - 1\big), && (6)\\
& = B_e \beta^{1-\frac{B_e(k')}{B_e}} \big(2^{\frac{b_{k'}}{B_e}\log\beta} - 1\big) \le \beta^{1-\frac{B_e(k')}{B_e}} \cdot b_{k'} \cdot \log\beta, && (7)
\end{aligned}$$

where Inequality (6) follows since at most $b_{k'}$ bandwidth units of link $e$ for request $k'$ are reserved, and Inequality (7) follows because $\frac{b_{k'}}{B_e}\log\beta \le \frac{b_{k'}}{B_e} \cdot \frac{B_{\min}}{b_{\max}} \le \frac{b_{k'}}{B_e} \cdot \frac{B_e}{b_{k'}} = 1$.

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

$$\begin{aligned}
& \sum_{v\in V} (c_v(k'+1) - c_v(k')) \\
& = \sum_{\langle v',v''\rangle \in P(k')\cap E'_v} (c_v(k'+1) - c_v(k')) \\
& \le \sum_{\langle v',v''\rangle \in P(k')\cap E'_v} \big(\alpha^{1-\frac{L_v(k')}{L_v}} \cdot \log\alpha\big), \quad \text{by Inequality (5)} \\
& = \log\alpha \sum_{\langle v',v''\rangle \in P(k')\cap E'_v} \Big(w_{\langle v',v''\rangle}(k') + 1\Big) \\
& = \log\alpha \Big( \sum_{\langle v',v''\rangle \in P(k')\cap E'_v} \omega_{\langle v',v''\rangle}(k') + \sum_{\langle v',v''\rangle \in P(k')\cap E'_v} 1 \Big) \\
& \le (\sigma_v + (n-1))\log\alpha, && (8)
\end{aligned}$$

where Inequality (8) holds because request $k'$ is admitted only if it meets the admission control policy (i), and any routing path in $G'$ contains no more than $n-1$ node-derived edges.

Similarly, the cost sum of edges by routing request $k'$ is

$$\sum_{e\in E} (c_e(k'+1) - c_e(k')) \le \; b_{k'} \cdot (\sigma_e + n - 1)\log\beta. \tag{9}$$

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 $k$ arrives is

$$\begin{aligned}\sum_{v\in V} c_v(k) &= \sum_{k'=1}^{k-1}\sum_{v\in V}(c_v(k'+1) - c_v(k'))\\ &= \sum_{k'\in\mathcal{S}(k)}\sum_{v\in V}(c_v(k'+1) - c_v(k'))\\ &\le \sum_{k'\in\mathcal{S}(k)}((\sigma_v + (n-1))\log\alpha), \quad \text{by Inequality (8)}\\ &= |\mathcal{S}(k)|(\sigma_v + (n-1))\log\alpha.\end{aligned}$$

Likewise, the cost sum of all edges for routing $|\mathcal{S}(k)|$ unicast requests by the online algorithm is

$$\sum_{e\in E} c_e(k) \le \mathbb{B}(k)(\sigma_e + (n-1))\log\beta.$$

■

We then provide a lower bound on the length of the routing path to which an optimal offline algorithm routes any 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 $k$, and let $P_{opt}(k')$ be the routing path in $G'$ found by the optimal offline algorithm for request $k' \in \mathcal{R}(k)$. Assume that $\alpha$ with $2|V| = 2n \le \alpha \le 2^{L_{\min}}$ and $\beta$ with $2|V| = 2n \le \beta \le 2^{B_{\min}/b_{\max}}$ are given values, then, for any request $k' \in \mathcal{R}(k)$, we have*

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

*Proof:* Given a unicast request $k'$, if it is admitted by the optimal offline algorithm but rejected by the online algorithm, then either (i) the length of the routing path for request $k'$ delivered by the online algorithm is no less than the given threshold; or (ii) there is lack of node and/or link resources for its admission.

Case (i) although the online algorithm is able to find a routing path $P(k')$, request $k'$ is rejected due to the fact that the length of $P(k')$ is beyond the given threshold. Let $P_{opt}(k')$ be the routing path delivered by an optimal offline algorithm. The length of $P_{opt}(k')$ cannot be less than that of the shortest path $P(k')$. Therefore, the length of $P_{opt}(k')$ path is no less than the given threshold by the admission control policy, i.e., its length is no less than $\min\{\sigma_v, \sigma_e\} = n - 1$.

Case (ii) this case 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 $G'$ such that $L_v(k') < 1$. Consequently, the length of $P_{opt}(k')$ is greater than $\sigma_v$:

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

(b) If there is an edge in any routing path found by the online algorithm without sufficient bandwidth to route request $k'$, then there exists an edge $e' = \langle v'', u'\rangle \in P_{opt}(k')$ in $G'$ such that $B_{(v,u)}(k') < b_{k'}$. Based on the fact that $2n \le \beta \le 2^{B_{\min}/b_{\max}} \le 2^{B_{(u,v)}/b_{k'}}$, we can apply the same logic as above to show the length of $P_{opt}(k')$ is greater than $\sigma_e$. ■

Having Lemmas 1 and 2, we 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 $(s_1, d_1; b_1), \ldots, (s_k, d_k; b_k)$ arriving one by one without the knowledge of future arrivals. There is an online algorithm, Algorithm 1, for the network capacity maximization problem for online unicasting with the competitive ratio of $2(\gamma\log\alpha + \log\beta) + 1$, if both $\alpha$ and $\beta$ are given values with $2|V| = 2n \le \alpha \le 2^{L_{\min}}$ and $2|V| = 2n \le \beta \le 2^{B_{\min}/b_{\max}}$, where $\gamma = \frac{B_{\min}}{\log\beta}$ is a value with $1 < \gamma \le \frac{B_{\min}}{\log(2n)}$.*

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

$$\begin{aligned}&(n-1)(\mathbb{B}_{opt}(k) - \mathbb{B}(k)) \le (n-1)\sum_{k'\in\mathcal{R}(k)} b_{k'}\\ &= \sum_{k'\in\mathcal{R}(k)} b_{k'}(n-1) \le \sum_{k'\in\mathcal{R}(k)} b_{k'}\Big(\sum_{e\in P_{opt}(k')}\omega_e(k')\Big)\\ &\le \sum_{k'\in\mathcal{R}(k)} b_k\Big(\sum_{e\in P_{opt}(k)}\omega_e(k)\Big) \qquad (10)\\ &= \sum_{k'\in\mathcal{R}(k)} b_{k'}\Big(\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v}\omega_{\langle v',v''\rangle}(k)\\ &\qquad + \sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e}\omega_{\langle u'',v'\rangle}(k)\Big)\\ &= \sum_{k'\in\mathcal{R}(k)} b_{k'}\Big(\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v}\frac{c_v(k)}{L_v}\\ &\qquad + \sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e}\frac{c_{(u,v)}(k)}{B_{(u,v)}}\Big)\\ &= \sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v} b_{k'}\frac{c_v(k)}{L_v}\\ &\quad + \sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e} b_{k'}\frac{c_{(u,v)}(k)}{B_{(u,v)}}\\ &\le \sum_{v\in V} c_v(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v}\frac{b_{k'}}{L_v}\\ &\quad + \sum_{(u,v)\in E} c_{(u,v)}(k)\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e}\frac{b_{k'}}{B_{(u,v)}}\\ &= \sum_{v\in V} c_v(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v} b_{k'}}{L_v}\\ &\quad + \sum_{(u,v)\in E} c_{(u,v)}(k)\frac{\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e} b_{k'}}{B_{(u,v)}}\\ &\le \sum_{v\in V} c_v(k)\gamma + \sum_{e\in E} c_{(u,v)}(k) = \gamma\sum_{v\in V} c_v(k) + \sum_{e\in E} c_{(u,v)}(k)\\ &\qquad (11)\\ &\le \gamma|S(k)|(\sigma_v + (n-1))\log\alpha\\ &\quad + \mathbb{B}(k)(\sigma_e + (n-1))\log\beta, \quad \text{by Lemma 1}\end{aligned}$$

$$= 2(n-1)(\gamma|S(k)|\log\alpha + \mathbb{B}(k)\log\beta). \quad (12)$$

Notice that Inequality (10) holds because the utilization of each resource does not decrease and consequently the weight of any edge in $G'$ does not decrease with more request admissions, i.e., $\omega_e(k') \le \omega_e(k)$ for any edge $e \in E'$ and any $k'$ with $1 \le k' \le k$. The proof of Inequality (11) proceeds as follows. For any switch node $v \in V$, each routing table entry can be used to admit a request with bandwidth at most $b_{\max}(\le B_{\min}/\log\beta = \gamma)$, and the routing 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., $\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v} b_{k'} \le L_v \cdot b_{\max} \le \gamma \cdot L_v$. Hence, $(\sum_{k'\in\mathcal{R}(k)}\sum_{\langle v',v''\rangle\in P_{opt}(k')\cap E'_v} b_{k'})/L_v \le \gamma$. 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 bandwidth 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., $\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e} b_{k'} \le B_e$. Therefore, $(\sum_{k'\in\mathcal{R}(k)}\sum_{\langle u'',v'\rangle\in P_{opt}(k')\cap E'_e} b_{k'})/B_e \le 1$.

By Inequality (12), we have

$$\frac{\mathbb{B}_{opt}(k) - \mathbb{B}(k)}{\mathbb{B}(k)} \le 2(\gamma\frac{|S(k)|}{\mathbb{B}(k)}\log\alpha + \log\beta)$$
$$\le 2(\gamma\log\alpha + \log\beta), \quad (13)$$

where the last step follows because $\mathbb{B}(k) = \sum_{k'\in S(k)} b_{k'}$ and $b_{k'} \ge 1$. From Inequality (13), we have

$$\frac{\mathbb{B}_{opt}(k)}{\mathbb{B}(k)} \le 2(\gamma\log\alpha + \log\beta) + 1. \quad (14)$$

■

Notice that the competitive ratio of Algorithm 1 is determined by parameters $\alpha$ and $\beta$. When $\alpha = \beta = 2n$, the competitive ratio of Algorithm 1 is $2(B_{\min} + \log(2n)) + 1 = O(\log n)$.

## VI. 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 of the proposed algorithm.

### A. *Online algorithm*

The basic idea of the proposed algorithm is to respond to each incoming multicast request $k$ with $(s_k, D_k; b_k)$ by either admitting or rejecting it, according to an admission control policy. To model the node and link resource consumptions of the admission of multicast request $k$, an auxiliary edge-weighted, directed graph $G' = (V', E'; \omega)$ will be constructed, where the weight of each node-derived or link-derived edge in $G'$ will reflect the availability and the utilization of a resource. A multicast tree in $G'$ rooted at the source $s'_k$ and spanning all terminal nodes in $D_k$ will be found if it exists. If the weighted sums of all its node-derived edges and all its link-derived edges are also less than their corresponding thresholds, then the request will be admitted; otherwise, it will be rejected.

Given an SDN $G(V,E)$ with node and link capacities $L_v(\cdot)$ and $B_e(\cdot)$, respectively, the auxiliary edge-weighted, directed graph $G' = (V', E'; w)$ is constructed as follows. For each switch node $v \in V$, two nodes $v'$ and $v''$ are added to $V'$, and there is a directed edge $\langle v', v''\rangle$ added to $E'$. For each edge $(u,v) \in E$, there are two new nodes $w_{uv}$ and $w_{vu}$, and four directed edges $\langle v'', w_{vu}\rangle$, $\langle w_{vu}, u'\rangle$, $\langle u'', w_{uv}\rangle$, and $\langle w_{uv}, v'\rangle$ added to $V'$ and $E'$, respectively, i.e., $V' = \{v', v'' \mid v \in V\} \cup \{w_{uv}, w_{vu} \mid (u,v) \in E\}$ and $E' = \{\langle v', v''\rangle \mid v \in V\} \cup \{\langle v'', w_{vu}\rangle, \langle w_{vu}, u'\rangle, \langle u'', w_{uv}\rangle, \langle w_{uv}, v'\rangle \mid (u,v) \in E\}$. Fig. 2 illustrates the construction of $G'$ from $G$.

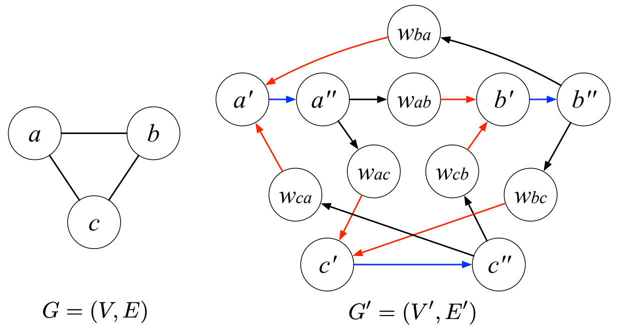


Fig. 2. The construction of $G'(V', E'; \omega)$ from an SDN $G(V,E)$.

Denote by $E'_v = \{\langle v'', w_{vu}\rangle \mid (v,u) \in E\}$ and $E'_e = \{\langle w_{vu}, u'\rangle \mid (v,u) \in E\}$ the sets of *node-derived edges* and *link-derived edges* in $G'$, respectively. The weight assigned to each edge $e \in E'$ is defined as follows.

$$\omega_e(k) = \begin{cases} 0, & \text{if } e \in \{\langle v', v''\rangle \mid v \in V\} \\ \alpha^{1-\frac{L_v(k)}{L_v}} - 1, & \text{if } e = \langle v'', w_{vu}\rangle \in E'_v \\ \beta^{1-\frac{B_{(v,u)}(k)}{B_{(u,v)}}} - 1, & \text{if } e = \langle w_{vu}, u'\rangle \in E'_e, \end{cases}$$

where the weight of each node-derived edge reflects the node resource consumption on one of its branches on its corresponding switch node while the weight of each link-derived edge reflects the bandwidth resource consumption on its corresponding link in the multicast tree.

Now, given a multicast request $k$ with $(s_k, D_k; b_k)$, the problem is to find a multicast tree $T(k)$ in $G'$ rooted at $s'_k$ and spanning all nodes in $D'_k = \{u' \mid u \in D_k\}$ such that the weighted sum of all edges in $T(k)$ is minimized, which is a classic *directed Steiner tree problem* that is NP-hard. As it is very unlikely to find an exact solution for it in polynomial time, an approximate solution instead suffices, by applying the approximation algorithm in [4]. The approximation ratio of the approximate solution is $|D_k|^\epsilon$, and its running time is a polynomial function of $n$ and $\frac{1}{\epsilon}$, where $\epsilon$ is a fixed value with $0 < \epsilon \le 1$.

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 $k$ is admitted only if

(i) $\sum_{e\in T(k)\cap E'_v} \omega_e(k) \le \sigma_v$, and
(ii) $\sum_{e\in T(k)\cap E'_e} \omega_e(k) \le \sigma_e$,

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.

### B. *Competitive ratio analysis*

Let $d_{\max}$ be the maximum degree of nodes in $G$, i.e., $d_{\max} = \max\{d_v \mid v \in V\}$, and it usually is a small constant,

**Algorithm 2** Online routing algorithm for multicast requests

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

**Output:** Admit or reject multicast request $k$. If admitted, a routing multicast tree for the request will be found.

```
1: for each node $v \in V$ do
2:     if the available table size at node $v$ is less than $d_v$, i.e., $L_v(k) < d_v$, then
3:         Remove $v$ and its incident edges from $G$;
4:     end if;
5: end for;
6: Construct an auxiliary edge-weighted, directed graph $G' = (V', E'; \omega)$ from the resulting graph $G(V, E)$ /* ensure that the switch table of each node $v$ has at least $d_v$ available entries */;
7: Find an approximate multicast tree $T(k)$ in $G'$ rooted at $s'_k$ and spanning all nodes in $D'_k$, by applying the algorithm in [4];
8: if $T(k)$ does not exist then
9:     Reject multicast request $k$;
10: else
11:     If $(\sum_{e \in T(k) \cap E'_v} \omega_e(k) \leq \sigma_v)$ and $(\sum_{e \in T(k) \cap E'_e} \omega_e(k) \leq \sigma_e)$, then admit request $k$ with $T(k)$; otherwise, reject $k$.
12: end if.
```

i.e., $d_{\max} \ll L_{\min}$. Recall that $L_{\min}$ is the minimum table 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 $\epsilon$ is a fixed value with $0 < \epsilon \leq 1$. We now analyze the competitive ratio of Algorithm 2 for the network capacity maximization problem for online multicasting. 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 k. Let $\alpha$ and $\beta$ be given values with $2|V| \leq \alpha \leq 2^{L_{\min}/d_{\max}}$ and $2|V| \leq \beta \leq 2^{B_{\min}/b_{\max}}$. Then, the cost sums of nodes and of links of G when multicast request k arrives are*

$$\sum_{v \in V} c_v(k) \leq |\mathcal{S}(k)|(\sigma_v + n - 1) d_{\max} \log \alpha, \tag{15}$$

*and*

$$\sum_{e \in E} c_e(k) \leq \mathbb{B}(k)(\sigma_e + n - 1) \log \beta, \tag{16}$$

*respectively.*

*Proof:* Consider an admitted multicast request $k' \in \mathcal{S}(k)$ by the online algorithm. Let $d_v^{T(k')}$ be the outgoing degree of a node $v''$ in the multicast tree $T(k')$ delivered by the algorithm. Notice that $d_v^{T(k')}$ is a small constant, and $d_v^{T(k')} \leq d_v \leq d_{\max} \ll L_v$. 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 $k'$ is

$$\begin{aligned}
&\sum\nolimits_{v \in V} (c_v(k'+1) - c_v(k')) \\
&= \sum\nolimits_{v \in T(k')} \left(L_v \alpha^{1 - \frac{L_v(k')}{L_v}} \left(\alpha^{\frac{L_v(k') - L_v(k'+1)}{L_v}} - 1\right)\right) \\
&\leq \sum\nolimits_{v \in T(k')} \left(L_v \alpha^{1 - \frac{L_v(k')}{L_v}} \left(\alpha^{\frac{d_{\max}}{L_v}} - 1\right)\right) \qquad (17) \\
&\leq (\sigma_v + (n-1)) d_{\max} \log \alpha, \qquad (18)
\end{aligned}$$

where Inequality (17) follows since the degree of $v$ in multicast tree $T(k)$ cannot exceed the degree of $v$ in $G$, and Inequality (18) follows since request $k'$ 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 the requests in $\mathcal{S}(k)$ is

$$\sum_{v \in V} c_v(k) \leq |\mathcal{S}(k)|(\sigma_v + (n-1)) d_{\max} \log \alpha.$$

The proof of Equation (16) is similar to the proof of Equation (9) in the unicast case, and thus omitted. ■

We then show the lower bound of the cost sum of node-derived and link-derived edges of the multicast tree for multicast request $k'$ admitted by an optimal offline algorithm but rejected by the online algorithm, by the following lemma.

**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 k, and let $T_{opt}(k')$ be the multicast tree in $G'$ found by the optimal offline algorithm for request $k' \in \mathcal{R}(k)$. Assume that $\alpha$ and $\beta$ are given values with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}/d_{\max}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$. Then, for each multicast request $k' \in \mathcal{R}(k)$, we have*

$$\sum_{e \in T_{opt}(k')} \omega_e(k') \geq \frac{\min\{\sigma_v, \sigma_e\}}{K^\epsilon}.$$

Due to the space limit, the full proof is omitted. The basic idea is similar to that in Lemma 2.

Having Lemmas 3 and 4, we show 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 $(s_k, D_k; b_k)$ arriving one by one without the knowledge of future arrivals with $D_k \subseteq V$. There is an online algorithm, Algorithm 2, for the network capacity maximization problem online multicasting with the competitive ratio of $2K^\epsilon(d_{\max}\gamma \log \alpha + \log \beta) + 1$, if both $\alpha$ and $\beta$ are given with $2|V| = 2n \leq \alpha \leq 2^{L_{\min}/d_{\max}}$ and $2|V| = 2n \leq \beta \leq 2^{B_{\min}/b_{\max}}$, where $\gamma = B_{\min}/\log \beta$.*

*Proof:* Let $\mathbb{B}_{opt}(k)$ be the total bandwidth of the multicast requests admitted by an optimal offline algorithm. Combining Lemma 3 and Lemma 4, we have

$$\begin{aligned}
&\frac{(n-1)}{K^\epsilon}(\mathbb{B}_{opt}(k) - \mathbb{B}(k)) \leq \gamma \sum_{v \in V} c_v(k) + \sum_{e \in E} c_e(k) \\
&\leq \gamma |S(k)|(\sigma_v + (n-1)) d_{\max} \log \alpha + \mathbb{B}(k)(\sigma_e + (n-1)) \log \beta \\
&= 2(n-1)(\gamma |S(k)| d_{\max} \log \alpha + \mathbb{B}(k) \log \beta) \qquad (19)
\end{aligned}$$

From Inequality (19), we have

$$\frac{\mathbb{B}_{opt}(k)}{\mathbb{B}(k)} \leq 2K^\epsilon\left(\frac{|\mathcal{S}(k)|}{\mathbb{B}(k)} d_{\max} \gamma \log \alpha + \log \beta\right) + 1.$$

■

When $\alpha = \beta = 2n$, the approximation ratio of Algorithm 2 is $2K^{\epsilon}(d_{\max}B_{\min} + \log(2n)) + 1 = O(K^{\epsilon}\log n)$.

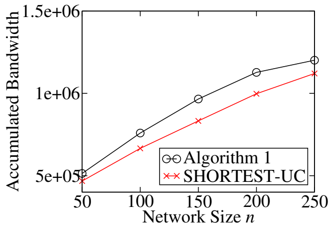
## VII. Performance Evaluation

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

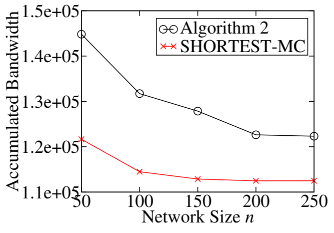
### A. Environment settings

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 [6]. The size of the forwarding table $L_v$ of each switch node $v \in V$ is from 500 to $5,000$, and the bandwidth capacity $B_e$ of each link $e \in E$ varies from $1,000$ Mbps to $10,000$ Mbps [10], [13], [16]. The bandwidth demand $b_k$ of each unicast or multicast request $k$ is randomly assigned between 1 Mbps and 50 Mbps. The number of terminals in a multicast request is 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 $30,000$ multicast requests. Notice that the bandwidth requirement of each request is consistent with the range given by Theorem 2, i.e., $b_k \leq \frac{B_{\min}}{\log(2n)}$, where $B_{\min} = \min\{B_{k'} \mid 1 \leq k' \leq k\} = 1,000$.

To evaluate the performance of the proposed algorithms against benchmarks, we here propose two heuristic algorithms `SHORTEST-UC` and `SHORTEST-MC` for online unicast and multicasting, respectively. Specifically, for an incoming request $k$, the heuristic algorithms first remove the links and nodes from the network that do not have enough capacities to support the admission of request $k$, and then assign each link the same weight. Algorithm `SHORTEST-UC` finds a shortest path with the minimal number of links from the source to the destination of request $k$, while algorithm `SHORTEST-MC` finds a single-source shortest path tree spanning all terminals.



(a) Accumulated bandwidth by Algorithms 1 and `SHORTEST-UC`

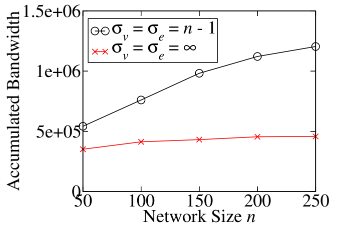


(b) Accumulated bandwidth by Algorithms 2 and `SHORTEST-MC`

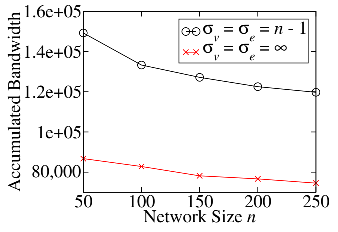
Fig. 3. The accumulated bandwidth delivered by different algorithms through varying $n$, while keeping other parameters fixed, i.e., $\alpha = \beta = 2n$ and $\sigma_v = \sigma_e = n - 1$, assuming that there are 50,000 unicast requests and 30,000 multicast requests.

### B. Performance evaluation of different algorithms

We first evaluate the proposed online algorithms 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. 3 plots the performance curves of different algorithms, from which it can be seen that the proposed algorithms, Algorithm 1 and Algorithm 2, outperform `SHORTEST-UC` and `SHORTEST-MC` in all cases. Specifically, in online unicasting, Algorithm 1 outperforms algorithm `SHORTEST-UC`, as Fig.3(a) and Fig. 3(b) clearly indicate that Algorithm 1 delivers



(a) Accumulated bandwidth by Algorithm 1 with and without thresholds



(b) Accumulated bandwidth by Algorithm 2 with and without thresholds

Fig. 4. 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 thresholds $\sigma_v = \sigma_e = \infty$ for a monitoring period consisting of 50,000 unicast requests or 30,000 multicast requests when $\alpha = \beta = 2n$.

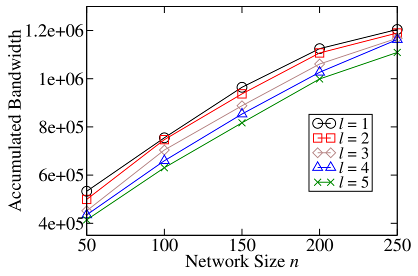
$10\%$ more accumulated bandwidth and admits $9\%$ more requests than algorithm `SHORTEST-UC` does. In addition, with the growth of network size $n$, the performance curves of both algorithms in comparison go up. In online multicasting, Algorithm 2 delivers more accumulated bandwidth and admits more requests than algorithm `SHORTEST-MC` does. In particular, the accumulated bandwidth delivered by Algorithm 2 is $20\%$ more than that by algorithm `SHORTEST-MC` when $n = 50$. However, Algorithm 2 still delivers $8\%$ more accumulated bandwidth compared with algorithm `SHORTEST-MC` when the network size is 250. The reason is that with the growth of the network size $n$, the number of terminals in each multicast request increases, requiring more node and link resources.

### C. Impact of parameters on the performance of algorithms

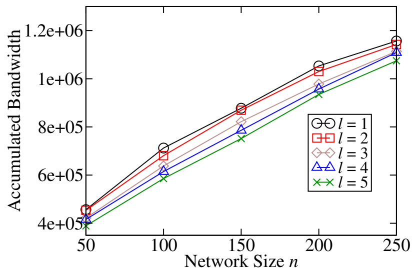
We first evaluate the impact of the admission control threshold parameters $\sigma_v$ and $\sigma_e$ on the performance of Algorithms 1 and 2. Admission control aims to prevent the admission of requests with large costs relative to thresholds $\sigma_v$ and $\sigma_e$. When $\sigma_v < \infty$ and $\sigma_e < \infty$ are bounded, this implies that a request is very likely to be rejected even if there are sufficient available node and link resources for its admission.

Fig. 4 shows the performance curves of Algorithm 1 and of Algorithm 2 with and without thresholds, respectively, from which it can be seen that Algorithm 1 and Algorithm 2 with admission control significantly outperform themselves without admission control. Specifically, for the online unicasting, the performance gap of Algorithm 1 with thresholds and without thresholds becomes larger and larger, which can be seen in Fig. 4(a) with the increase in network size $n$. For example, the ratio of the accumulated bandwidth delivered by Algorithm 1 with and without the admission control thresholds grows from 1.25 when $n = 25$ to 2.5 when $n = 250$, see Fig. 4(a). For the online multicasting, the performance gap of Algorithm 2 with and without the admission control thresholds is stable, as shown in Fig. 4(b). The difference in the accumulated bandwidth only drops from $500,000$ to $450,000$ for a monitoring period when the network size increases from 100 to 250.

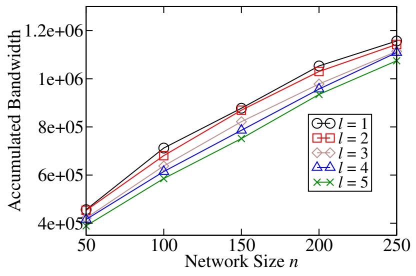
We then investigate the impact of parameters $\alpha$ and $\beta$ on the performance of the proposed algorithms, by varying $\alpha$ and $\beta$ from $2^1 n$ to $2^5 n$ while setting $\sigma_v = \sigma_e = n - 1$. Fig. 5 and Fig. 6 plot the performance curves of Algorithms 1 and Algorithm 2, by varying the value of either $\alpha$ or $\beta$ while fixing the value of the other. It can be seen from Fig. 5(a) to Fig. 5(c) that when $\alpha$ is fixed, the larger the value of $\beta$, the less the accumulated bandwidth delivered by Algorithm 1 for different network sizes $n$, 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



(a) The performance of Algorithm 1 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^1 n$

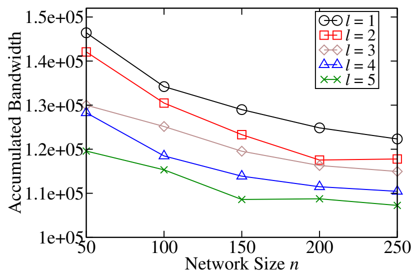


(b) The performance of Algorithm 1 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^3 n$

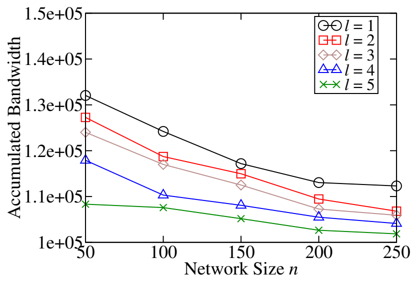


(c) The performance of Algorithm 1 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^5 n$

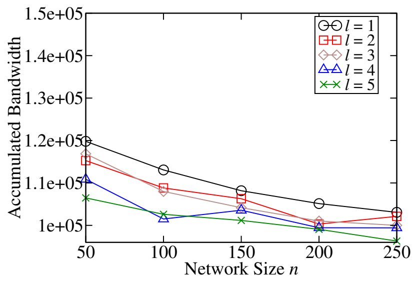
Fig. 5. The performance of Algorithm 1 for online unicasting by varying $\alpha$ and $\beta$, when $\sigma_v = \sigma_e = n - 1$.



(a) The performance of Algorithm 2 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^1 n$



(b) The performance of Algorithm 2 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^3 n$



(c) The performance of Algorithm 2 by varying $\beta = 2^l n$ with $1 \le l \le 5$, when $\alpha = 2^5 n$

Fig. 6. The performance of Algorithm 2 for online multicasting by varying $\alpha$ and $\beta$ when $\sigma_v = \sigma_e = n - 1$.

also can been seen that the performance gap of Algorithm 1 under different values of $\alpha$ and $\beta$ is stable with the increase of network size $n$. Similarly, Fig. 6 draws the performance curves of Algorithm 2 by varying the values of exactly one of $\alpha$ and $\beta$ each time, from which it can be seen when the value of $\alpha$ is fixed, the larger $\beta$, the less the accumulated bandwidth delivered by Algorithm 2, and vice versa. The performance gap of Algorithm 2 with different $\beta$ is also stable for different network sizes.

## VIII. Conclusion

In this paper we studied dynamic unicast and multicast routing in Software-Defined Networks under both switch node and link capacities and request bandwidth demands constraints. We first proposed a novel cost model to model the usage costs of node and link resources. We then devised efficient online algorithms for online unicast and multicast capacity maximization problems. We also performed analytical analysis on the competitive ratios of the proposed algorithms. We finally evaluated the performance of the proposed algorithms through experimental simulation. The simulation results indicate that the proposed algorithms are very promising.

## References

[1] S. Agarwal, M. Kodialam, and T. V. Lakshman. Traffic engineering in software defined networks. *Proc. INFOCOM*, IEEE, 2013.

[2] J. Aspnes, Y. A. Yossi, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, vol. 44, no. 3, pp. 486–504, May 1997.

[3] Z. Cao, M. Kodialam, and T. V. Lakshman. Traffic steering in software defined networks: planning and online routing. *Proc. ACM SIGCOMM Workshop Distributed Cloud Computing (DCC)*, 2014.

[4] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *Proc. 9th ACM-SIAM Symp. Discrete Algorithms (SODA)*, 1998.

[5] R. Cohen, L. Eytan, J. Naor, and D. Raz. On the effect of forwarding table size on SDN network utilization. *Proc. IEEE INFOCOM*, 2014.

[6] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[7] H. Huang, S. Guo, P. Li, and W. Liang. Cost minimization for rule caching in software defined networking To appear in *IEEE Trans. Parallel and Distributed Systems*.

[8] L. Huang, H. Hung, C. Lin, and D. Yang. Scalable steiner tree for multicast communications in software-defined networking. *Computing Research Repository (CoRR)*, vol. abs/1404.3454, 2014.

[9] S. Jain et al. B4: experience with a globally-deployed software defined WAN. *Proc. ACM SIGCOMM*, 2013.

[10] Y. Kanizo, D. Hay, and I. Keslassy. Palette: distributing tables in software-defined networks. *Proc. IEEE INFOCOM*, 2013.

[11] N. Katta, J. Rexford, and D. Walker. Infinite cacheflow in software-defined networks. Technical Report TR–966–13, Department of Computer Science, Princeton University, 2013.

[12] K. Kar, M. Kodialam, T. V. Lakshman, and L. Tassiulas. Routing for network capacity maximization in energy-constrained ad hoc networks. *Proc. INFOCOM*, 2003.

[13] D. Kreutz, F. M. V. Ramos, P. Verissimo et al. Software-defined networking: a comprehensive survey. *Proc. IEEE*, vol. 103, pp. 14–76, 2015.

[14] W. Liang and X. Guo. On-line multicasting for network capacity maximization in energy-constrained ad hoc networks. *IEEE Trans. Mobile Computing*, vol. 5, pp. 1215–1227, 2006.

[15] W. Liang and Y. Liu. On-line data gathering for maximizing network lifetime in sensor networks. *IEEE Trans. Mobile Computing*, vol. 6, pp. 2–11, 2007.

[16] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: past, present, and future of programmable networks. *Communications Surveys & Tutorials*, vol. 16, pp. 1617–1634, 2014.

[17] NTT Communications. NTT communications and jba's sdn project wins ibc 2013 innovation award. https://www.ntt.com/aboutus_e/news/data/20130918.html, NTT, 2013.

[18] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *J. Selected Areas in Communications*, 1995.

[19] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. *Proc. ACM SIGCOMM*, 2013.

[20] E. Spitznagel, D. Taylor, and J. Turner. Packet classification using extended TCAMs. *Proc. IEEE ICNP*, 2003.