Bandwidth Scheduling for Maximizing Resource Utilization in Software-Defined Networks

Poonam Dharam
Department of Computer Science
Saginaw Valley State University
Saginaw, MI, USA
Email: pdharam@svsu.edu

***Abstract*—Software-Defined Networks (SDNs) enable the network control plane to be decoupled from the data plane and assign the control to a programmable software unit, i.e., controller. With such a logically centralized control, SDNs provide the capability of bandwidth reservations that meets user requirements. In SDNs with multiple logically centralized controllers, it is challenging to maintain accurate link-state information at every controller in a consistent manner. Bandwidth scheduling in presence of inaccuracy may lead to rejection of reservation requests in turn affecting the Quality of Service (QoS) provided by the Network Service provider. In order to minimize the service disruption caused by lack of uniform global network view (GNV) at controllers, we propose a routing scheme based on multiple weights' distribution such that the number of requests dropped or rejected during bandwidth reservation setup phase is minimized. The performance superiority of the proposed bandwidth reservation solutions is illustrated by extensive simulations in comparison with existing methods.**

Keywords*: *Software-Defined Networks; bandwidth reservation; bandwidth scheduling.

I. INTRODUCTION

Next-generation e-science applications in various domains generate colossal amounts of simulation, experimental, or observational data, on the order of terabyte at present and petabyte or exabyte down the road, now frequently termed as "Big Data", which must be transferred to remote sites for collaborative processing, analysis, and storage. The data transfer at such scales necessitates the development of high-performance networks that are capable of provisioning dedicated channels with reserved network resources through either circuit/lambda-switching or Multi-Protocol Label Switching (MPLS) tunneling techniques.

The decoupling of control and data planes in SDNs gives network designers freedom to re-factor the network control plane, allowing the network control logic to be designed and operated as though it were a centralized application, rather than a distributed system. SDNs with logically centralized control and distributed flow tables enable various networking services, such as bandwidth reservations to support Big Data transfer. Such reservations not only ensure the confirmed availability of bandwidth during the entire course of data transfer, but also yield a higher level of resource utilization in the network. In addition to Big Data transfer, some real-time applications, such as Video-on-Demand (VoD) or virtual collaboration might require an immediate allocation of bandwidth for an indefinite duration [1].

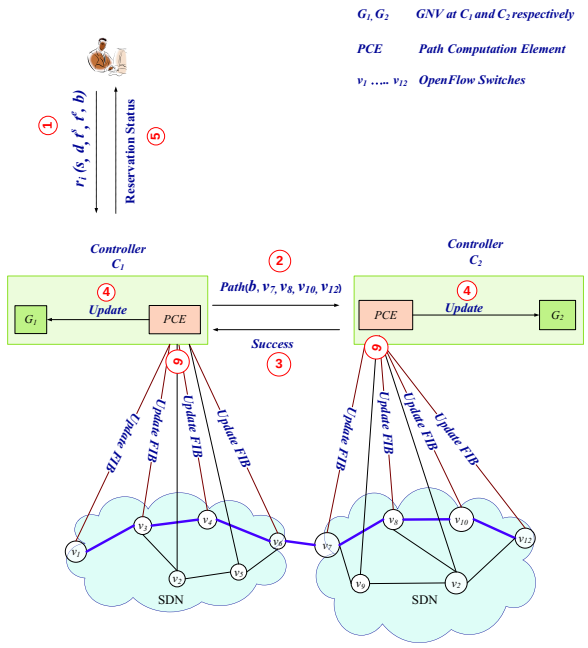

Fig. 1. Bandwidth scheduling in Software-Defined Networks.

We now describe the process of scheduling bandwidth reservation requests in SDNs using a simple example shown in Figure 1. The bottom layer consists of OpenFlow switches, v_1, v_2, ..., v_{12}, grouped into one or more separate domains, responsible for forwarding incoming packets based on the information stored in their Forward Information Base (FIB) consisting of flow entries and next-hop information. C_1 and C_2 are controllers consisting of Global Network Views (GNVs) G_1 and G_2, respectively, consisting of upto-date link-state information of the underlying network. The Path Computation Element (PCE) is responsible for processing the incoming requests for path setup and updating the flow entries in FIB. The controllers exchange their link-state information among themselves to maintain a consistent GNV across every controller.

When a bandwidth reservation request $r_i(s, d, t^s, t^e, b)$ arrives at controller C_1, the controller follows a two-phase process. In Phase 1, the controller checks its GNV to verify if b units of bandwidth are available. If the network has sufficient

 ISBN: 978-1-61208-633-0

bandwidth, controller C_1 computes a path $p(r_i)$ that connects source s and destination d and meets the bandwidth requirement of b units for the time duration [t^s, t^e], according to the advertised link-state information. From the perspective of the controller, if there is sufficient network resource (bandwidth) to accommodate the request, it launches a signalling/setup process to ensure that b units of bandwidth are indeed available on every link $l \in p(r_i)$ during [t^s, t^e]. As the signalling message traverses all the component links along the selected path $p(r_i)$, each responsible controller performs an admission test to check if a component link in its domain can actually support the request. If the link has sufficient resource, the controller reserves bandwidth on behalf of the new connection before forwarding the setup message to the controller servicing the next component link along the path; otherwise, it rejects the connection request immediately. Upon the confirmation of the user request, the controllers employ the state distribution method to distribute their state information to other controllers in an attempt to achieve a consistent GNV across the network.

When the link-state information in the GNV is not up-to-date, i.e., it does not represent the current picture of the network state, the routing process might select a path that is unable to support the call requirements. Consequently, a reservation request that is initially scheduled on a particular path would be rejected in the path setup process, which is termed as a false positive. Out-of-date information may also cause controllers to make a wrong decision on path selection; some paths may be overused (since the GNV is updated periodically, the routing scheme tend to select the same path for an extended period of time [2]) and hence cannot meet the required QoS while there exist other underutilized paths. This is widely recognized as a routing inaccuracy problem. The false positive resulted from inaccuracy jeopardizes users' satisfaction [3]. This situation deteriorates as the level of inaccuracy/inconsistency increases.

The setup failure in a blocked or rejected request incurs extra overhead to reserve resources along the path and may delay the establishment of other connections. In addition, a failed connection temporarily holds resources on its upstream links, which may block other connections in the interim [4] [5]. Also, the performance of a QoS routing algorithm can be significantly undermined by inaccurate link-state information. Thus, it is critical to minimize the setup failures caused by inaccurate state information for efficient network utilization and improved QoS. A good routing algorithm must incorporate mechanisms to account for the inaccuracy in the GNV and make effective routing decisions in the presence of such inaccuracy [6].

We propose a weight-based routing scheme, called MinBlock-Routing, to schedule incoming bandwidth reservation requests such that the total number of reservation requests blocked due to inaccurate GNV is minimized. When multiple requests with similar requirements (source, destination, bandwidth, start-time, and end-time) arrive at controllers simultaneously, our routing scheme assigns multiple weights to links connecting the same pair of OpenFlow switches, but belonging to different controllers, thus choosing different paths for similar requests.

The rest of the paper is organized as follows. Section II conducts a brief survey of related work. Section III constructs the cost models and formulates the bandwidth reservation problem under study. Section IV details the scheduling algorithm design. Section V evaluates the scheduling performance in simulated networks.

II. Related Work

Most of the work in this field has been focused on QoS routing whose goal is to find a path that satisfies multiple QoS constraints while maximizing network utilization. The existing work on this subject can be broadly classified into single-path routing and multi-path routing.

In single-path routing, only a single path is considered between a source-destination pair. Some work in this category only considers bandwidth in path computation. In [7], Guerin *et al.* proposed source routing algorithms with inaccurate link-state information by exploring the impact of information inaccuracy in the context of QoS routing. In [2], Apostolopoulos *et al.* proposed a new routing mechanism, Safety Based Routing (SBR), to address the routing inaccuracy issues when computing explicit paths with bandwidth constraints. SBR incorporates a new link attribute, safety (S), which represents the effect of the routing inaccuracy in the link-state reliability, in the path selection process. In [4] [8] Bruin *et al.* proposed a QoS routing mechanism, BYPASS Based Routing (BBR), which bypasses those links along the selected path that potentially cannot satisfy the traffic requirements.

There exists some other work that considers both bandwidth and delay in path computation. In [9], Korkmaz *et al.* addressed the path computation problem under bandwidth and delay constraints with inaccurate link-state information. They adopted a probabilistic approach where the state parameters are characterized by random variables to find the most-probable bandwidth-delay-constrained path (MP-BDCP). In [10], Zhang *et al.* studied QoS routing in the presence of inaccurate state information, formulated as the Most-Probable Two Additive-Constrained Path (MP-TACP) problem. They proposed a routing algorithm that uses pre- and on-demand computation along with both linear and nonlinear search techniques that take inaccuracies into consideration. In most existing bandwidth scheduling algorithms, reservation requests with similar requirements (the same source/destination and overlapped start/end time) are typically routed along the same path, hence resulting in unbalanced traffic. This situation becomes worse in the presence of inconsistent GNVs in SDNs.

Multi-path routing, on the other hand, probes multiple feasible paths simultaneously, and if more than one path can satisfy the QoS requirement of a request, the shortest one is typically selected. In [11], Jia *et al.* proposed a routing strategy, in which connection requests with specific bandwidth demands can be assigned to one of several alternative paths. They introduced a collection of k-shortest path routing schemes and investigated

 ISBN: 978-1-61208-633-0

the performance under a variety of traffic conditions and network configurations. The idea of randomized routing [2], is to compute a set of feasible paths and then randomly select one for a connection request. Every time when the link-state is updated, k paths with the largest bandwidths are selected as the candidates to be used until the next link-state update. When a request arrives, it is randomly routed among the k paths. Since the routing process does not always select the best path, the requests can be distributed across multiple available paths for better load balance. In [12], Dharam *et al.* proposed a randomization-based scheme for path computation by assigning random weights to links during path computation, in turn distributing the incoming requests across multiple path Existing schemes probe multiple paths either concurrently, which introduces much higher overhead than their single-path counterparts, or sequentially, which incurs a longer path setup time. Similar to multi-path routing, randomized path selection also requires computing a set of candidate paths to the destination, which incurs overhead for maintaining the information of all the computed paths.

Our proposed routing scheme distributes incoming reservation requests with similar requirements, in terms of source and destination, across multiple paths such that the time taken to update the GNVs does not affect the ongoing routing decisions.

III. Cost Models and Problem Formulation

In this section, we discuss the cost models by using a graph to represent the SDN and formulate our problem based on the assumptions made.

A. Cost Models

We use a network graph $G_j[V, E, B]$ to represent the latest GNV at controller C_j, where V is a set of OpenFlow switches, E is a set of network links, and B is a set of advertised bandwidths $b_l^{ad}(t)$ for each link l in the network at a future time point t.

For a bandwidth reservation request $r_i(s_i, d_i, t_i^s, t_i^e, b_i)$ arriving at a controller C_j, the controller needs to compute a path $p(r_i)$ that connects source s_i and destination d_i and meets the requirement of bandwidth b_i for a time duration $[t_i^s, t_i^e]$, according to the advertised link-state information. From the perspective of the controller, if there is sufficient network resource (bandwidth) to accommodate the request, it launches a signalling/setup process such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) to ensure that b_i units of bandwidth are indeed available on every link $l \in p(r_i)$ during $[t_i^s, t_i^e]$. As the signalling message traverses all the component links along the selected path $p(r_i)$, each responsible controller performs an admission test to check if a component link in its domain can actually support the request. If the link has sufficient resource, the controller reserves bandwidth on behalf of the new connection before forwarding the setup message to the controller servicing the next component link along the path; otherwise, it rejects the connection request immediately. Upon the confirmation of the user request, the controllers employ the state distribution method to distribute their state information to other controllers in an attempt to achieve a consistent GNV across the network.

A link without sufficient available bandwidth along the path computed based on the advertised bandwidth would lead to a setup failure, and the inaccuracy in GNV is one major cause for such setup failures.

B. Problem Formulation

We formulate a Bandwidth Reservation (MinBlock-Routing) problem as follows:

Definition 1: **MinBlock-Routing** Given an SDN with multiple controllers, a graph $G_j[V, E, B]$, which represents the latest GNV at controller C_j and a set $R_j = \{r_1, ..., r_i, r_{i+1}, ..., r_n\}$ of n bandwidth reservation requests at C_j, where each request $r_i \in R_j$ specifies source s_i, destination d_i, start time t_i^s, end time t_i^e, and required bandwidth b_i, we propose a bandwidth scheduling scheme with the goal of minimizing the total number of requests blocked at the time of resource reservation, due to inconsistent GNVs.

IV. Bandwidth Scheduling Algorithm

We first generate a new network topology G' from G_j by pruning the links from G_j without sufficient bandwidth during the time interval $[t_i^s, t_i^e]$ to accommodate the incoming request r_i due to the existing bandwidth reservations within that interval. Multiple weights are assigned to links that connect OpenFlow switches belonging to same pair of controllers. For example, let link l_1 connect two OpenFlow switches sw_{11} and sw_{21} such that sw_{11} and sw_{21} belong to controllers C_1 and C_2 respectively. Also, let link l_2 connect two OpenFlow switches sw_{12} and sw_{22} such that sw_{12} and sw_{22} belong to C_1 and C_2 respectively. Thus, there are two possible ways of connecting C_1 and C_2. By forcing C_1 to use l_1 and C_2 to use l_2 to move traffic between the two SDN networks, the GNV inconsistency due to inaccurate state information can be minimized. Both C_1 and C_2 agree upon randomly generated weights for l_1 and l_2 as follows: C_1 assigns a lower weight to l_1 and higher weight to l_2 where as C_2 does the reverse i.e., assigns a higher weight to l_1 and lower to C_2. For a path from C_1 to C_2, l_1 is chosen, whereas for a path from C_2 to C_1, l_2 is chosen. Thus, by assigning multiple weights, controllers tend to choose multiple paths between the same set of networks. This in turn gives time for the controllers using same path to update their GNVs, thus reducing the inconsistent view. If a valid path exists for a reservation request, then bandwidth is reserved along the path and the GNV is updated with the new bandwidth values; otherwise, if no valid path is found for a reservation request, then the reservation request is rejected.

This algorithm is executed by a top-level controller with the network and request information in each network domain. If such a global controller does not exist, the algorithm could be executed in a distributed manner on each controller with its local information and the remote information sent by its peers.

V. Performance Evaluation

The simulations are executed in a set of randomly generated networks comprised of 10 Gbps links with varying sizes from small to large scales and a set of hundreds of reservation

 ISBN: 978-1-61208-633-0

TABLE I
THE SIZES OF NETWORKS USED IN THE SIMULATIONS.

Index of problem sizes	# of nodes	# of links
1	5	12
2	10	20
3	15	34
4	20	42
5	25	51
6	30	65
7	35	74
8	40	88
9	45	91
10	50	102
11	55	120
12	60	134

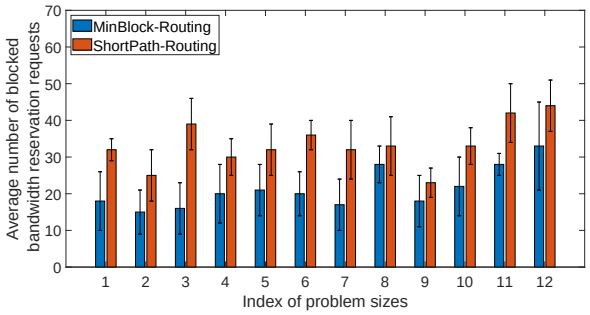

Fig. 2. The average number of blocked reservation requests with the standard deviations resulted from ShortPath-Routing and MinBlock-Routing.

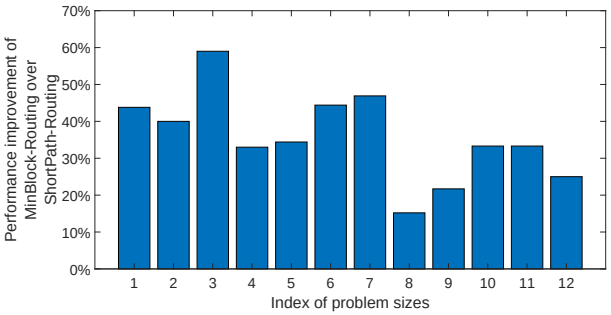

Fig. 3. The performance improvement of MinBlock-Routing over ShortPath-Routing in terms of percentage reduction in blocked bandwidth reservation requests.

requests under Poisson distribution with random input parameters chosen within appropriate ranges.

We consider 12 network sizes, indexed from 1 to 12, each with a different number of nodes and links as tabulated in Table I. For each network size, we generate 10 random problem instances of different network topologies. We run the proposed MinBlock-Routing algorithm and also a traditional scheduling shortest path routing algorithm (ShortPath-Routing), for comparison as it has been widely used for bandwidth reservation in real-life high-performance networks.

We evaluate the global routing performance in large-scale multi-domain networks with multiple controllers, each of which processes around 100 bandwidth reservation requests. The proposed MinBlock-Routing algorithm is able to balance the loads of incoming requests by distributing them across various alternative paths, and hence significantly reduce the number of blocked requests. Figure 2 shows the average number of blocked reservation requests with the standard deviations resulted from ShortPath-Routing and MinBlock-Routing. For a visual comparison, we further plot in Figure 3 the performance improvement of MinBlock-Routing over ShortPath-Routing in terms of the percentage decrease in blocking of incoming reservation requests, calculated as: $\frac{B(ShortPath)-B(MinBlock)}{B(ShortPath)} \cdot 100\%$. We observe that the proposed MinBlock-Routing algorithm significantly outperforms ShortPath-Routing. Our proposed Routing algorithm is able to route the incoming reservation requests with similar requirements across multiple paths in presence of inaccuracies in GNVs.

VI. CONCLUSION

We formulated and solved a bandwidth scheduling problem for minimizing the number of reservation requests blocked in software-defined networks with multiple controllers and inconsistent GNVs. In the current work, we consider scheduling incoming reservation requests with similar requirements across multiple paths thus minimizing the blocking ratio of user requests for bandwidth reservation.

REFERENCES

[1] I. Ahmad, J. Kamruzzaman, and S. Aswathanarayaniah, “A book-ahead routing scheme to reduce instantaneous request call blocking and preemption rate,” in Proc. of IEEE ICON, vol. 1, Nov. 2005, p. 6.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, “Improving qos routing performance under inaccurate link state information,” in Proceedings of the 16th International Teletraffic Congress, 1999.

[3] N. Ansari, G. Cheng, and N. Wang, “Routing-oriented update scheme (rose) for link state updating,” in IEEE Transactions on Communication, 2008.

[4] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, and J. Domingo-Pascual, “Optimizing routing decisions under inaccurate network state information,” in Lecture Notes in Computer Science No.3375: Quality of Service in Multiservice IP Networks, 2005, pp. 445–455.

[5] A. Shaikh, J. Rexford, and K. G. Shin, “Evaluating the impact of stale link state on quality-of-service routing,” IEEE/ACM Transactions On Networking, vol. 9, 2001.

[6] X. Yuan, W. Zheng, and S. Ding, “A comapritive study of qos routing schemes that tolerate imprecise state information,” in Intl. Conf. on Computer Comm. and Netw., Oct. 2002, pp. 230 – 235.

[7] R. Guerin and A. Orda, “Qos routing in networks with inaccurate information: Theory and algorithms,” IEEE/ACM Trans. on Networking, vol. 7, no. 3, pp. 350–364, 1999.

[8] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, and J. Domingo-Pascual, “Qos routing algorithms under inaccurate routing information for bandwidth constrained applications,” in Proc. of Int. Conf. on Communications, vol. 3, 2003, pp. 1743 – 1748.

[9] T. Korkmaz and M. Krunz, “Bandwidth-delay constrained path selection under inaccurate state information,” IEEE/ACM Transactions On Networking, vol. 11, no. 3, 2003.

[10] Y. Zheng and T. Korkmaz, “Two additive-constrained path selection in the presence of inaccurate state information,”J. Comp. Comm., vol. 30, pp. 2096 – 2112, Jun. 2007.

[11] Y. Jia, I. Nikolaidis, and P. Gburzynski, “On the effectiveness of alternative paths in qos routing: Research articlesemphIntl. Journal of Communication Systems, vol. 17, 2004.

[12] P. Dharam, C. Q. Wu, and N. S. V. Rao, “Advance bandwidth scheduling in software-defined networks,” in Globecom, 2015.

 ISBN: 978-1-61208-633-0