Computer Networks 96 (2016) 69–78

Contents lists available at ScienceDirect

Computer Networks

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

Jointly optimized QoS-aware virtualization and routing in software defined networks

Shih-Chun Lin [a,1,*], Pu Wang [b,2], Min Luo [c,3]

[a] Broadband Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b] Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS 67260, USA
[c] Shannon Lab, Huawei Technologies Co. Ltd., Santa Clara, USA

ARTICLE INFO

Article history:
Received 10 July 2015
Revised 13 August 2015
Accepted 15 August 2015
Available online 2 September 2015

Keywords:
Controller management tool
Joint virtualization and routing decision
Fine-grained network virtualization
Dynamic flow allocation
End-to-end QoS provisioning
Software defined networks

ABSTRACT

Software Defined Networks (SDNs) have been recognized as the next-generation networking paradigm that decouples the network control plane from the data forwarding plane. A logically centralized controller is responsible for all the control decisions and communication among the forwarding switches. However, current traffic engineering techniques and state-of-the-art routing algorithms do not effectively use the merits of SDNs, such as global centralized visibility, control and data plane decoupling, network management simplification and great computation capability. In this paper, a multi-tenancy management framework is proposed to enable the jointly optimized design of quality-of-services (QoSs)-aware virtualization and routing by tenant isolation and prioritization as well as flow allocation, fulfilling QoS requirements of tenants' applications. Specifically, a fine-grained network virtualization is first proposed to isolate and prioritize tenants through the design of network and switch hypervisors. Furthermore, a QoS-aware dynamic flow allocation is proposed to enable optimal flow routes selection upon the given network slicing with QoS provisioning. Finally, an adaptive feedback management tool, called QoS-aware Virtualization-enabled Routing (QVR), is proposed to combine virtualization with flow allocation and supports reliable and efficient transmissions with regards of time-varying QoS requirements, network topologies, and traffic statistics. Simulations confirm that QVR achieves much less shared edges, congestion latency, and traffic delay for multiple tenants, thus facilitating virtualization-enabled traffic engineering for multi-tenancy SDNs.

1. Introduction

During the past decade, the increment and complexity of tenants' application demands and requirements motivate the reconsideration of better traffic engineering solutions. In particular, as applications dramatically increase with various types and thus become more challenging to address, network operators in commercial clouds and data centers have been trying to improve network performance while fulfilling application requirements. However, this objective is almost impossible to accomplish with the current *closed/fixed* network architectures [1]. Recently emerged Software Defined Network (SDN) [2,3] decouples control and data planes, simplifies network management and control, and provides global visibility and direct control of the underlying forwarding devices via *open software-based* implementations. The new system architecture allows the separation of routing

* Corresponding author. Tel.: +1 7654048029.
E-mail addresses: slin88@ece.gatech.edu (S.-C. Lin), pu.wang@wichita.edu (P. Wang), min.ch.luo@huawei.com (M. Luo).
[1] Student member, IEEE,
[2] Member, IEEE,
[3] Senior member, IEEE.

http://dx.doi.org/10.1016/j.comnet.2015.08.003
1389-1286/

decisions from the forwarding switches so that the decisions can be easily modified, reconfigured, and optimized by the centralized controller. In particular, instead of using traditional switches where the forwarding rules are defined and installed by different vendors, open switches, e.g., OpenFlow enabled switches [4], are introduced that only support flow tables whose and are programmable following controller's policy. Thus, this decoupling architecture provides the global and real-time network status to the centralized controller, allowing the implementation of promising traffic engineering that is infeasible in the closed architectures.

Exploiting the novel SDN architecture by cloud computing and data centers, a multi-tenant and resource sharing scheme is widely considered due to its infrastructure and maintenance cost-effectiveness, simplification, lower system requirements and flexibility. Specifically, in single tenant solutions, tenants' applications have their own dedicated resource and nearly 45% of these resources is idle for most of the time. On the other hand, in multi-tenant solutions, resources are shared among tenants, which implies the efficient resource utilizations. However, under such multi-tenancy scenarios, the resource assignments might be overlapped due to the sharing, and high-demanded tenants can monopolize all the shared resources, thus greatly affecting other tenants' operations [5,6]. In that case, while SDN's global network view might allow the supervision of tenants' resource consumption to detect the high-demanded tenants, there is still a need of virtualization mechanism that isolates and prioritizes tenants' resource usages from each other, thus allowing customized performance and security level.

As mentioned previously, the routing decisions also need to be reevaluated in these multi-tenancy SDNs, from a fully distributed computation of path-selection towards a central administration of the path calculation supported and managed by the logically centralized controller. In particular, the improvement of network performance should be explored with respect to latency, throughput, and reliability in the designed routing. These service demands can be further associated with the specific quantitative quality-of-services (QoSs) of application flows, e.g., delay bounds, throughput restrictions, and packet losses. Taking into consideration of all these flow requirements introduces a new challenge in routing decision. Moreover, regarding various types of tenants' applications upon the same SDN infrastructure, best-effort traffic, real-time traffic, multimedia like video and voice, data applications, etc. should all be supported at the same time, where more careful considerations are needed. For example, real-time traffic does not have the elasticity to adapt the packet transmission rate due to network congestion as elastic traffic does and is tolerable with packet delay. Therefore, the concurrent presence of different types of traffic in the same network brings the following undesired situations: real-time traffic will eventually squeeze the service out of non-real-time traffic due to its strictly service demands; elastic traffic could arrive with starvation in network congestion, as real-time flows won't hold back and share resources fairly [7,8]. It implies that the service guarantees for a variety of traffic types also introduces another great challenge in routing decision. However, the existing work of traffic engineering in SDNs [1,9] neither consider multi-tenancy scenarios with virtualization scheme nor examine the QoS provisioning in flow allocation with regards of various tenants' applications.

Leveraging SDN's new system architecture, it is possible to develop a routing framework in multi-tenancy environments with the provisioning of QoS-aware flow allocation and tenant isolation and prioritization, which is significant for cloud computing and enterprise data centers [10]. Therefore, in this paper, a jointly optimized design of virtualization and routing is addressed and an adaptive solution is proposed to react to time-varying QoS requirements, network topologies, and traffic statistics in a real-time manner. Specifically, a fine-grained network virtualization is first proposed to slice the physical network infrastructure into several isolated subnets for multiple tenants. Ideally, the infrastructure slicing should support 100% independence among multi-tenants, i.e., no shared edges among tenants' subnets, in such a way each tenant's preference can be satisfied. Towards this, the network and switch hypervisors are introduced to efficiently give a feasible solution to the NP-complete problem of network graph partitioning [11], supporting network slicing for tenants isolation and prioritization. Furthermore, a QoS-aware dynamic flow allocation is proposed to enable fast flow allocation with regards of traffic variations and end-to-end QoS requirements. The management tool of QoS-aware Virtualization-enabled Routing (QVR) algorithm is further proposed to facilitate an adaptive feedback control of network virtualization and flow allocation and to improve service performance for achieving tenant's satisfaction, thus providing a customized solution for multi-tenancy SDNs. The key features of our solution are summarized as follows:

- Fine-grained network virtualization: QVR provides topological and bandwidth virtualization(s) to realize resource slicing in terms of network topology and link capacity.
- Dynamic flow allocation: QVR supports traffic-awareness allocation that fast manages dynamic flows with low computational complexity.
- End-to-end QoS provisioning: QVR enables end-to-end QoS guarantees within the adaptive feedback design of virtualization-enabled traffic engineering.

Performance evaluation confirms QVR outperforms the conventional approaches with less shared edges to ensure resource isolation of infrastructure usages, congestion latency, and traffic delay for multi-tenants, successfully introducing virtualization-enabled traffic engineering with reliable and efficient transmissions in practical implementations of multi-tenancy SDNs.

To the best of our knowledge, this work is the first to provide a joint consideration of network virtualization and routing decision with QoS provisioning in centralized controller in SDNs. The rest of the paper is organized as follows. Section 2 introduces the system model and Section 3 presents the multi-objective optimization problem for jointly virtualization and routing. Sections 4 and 5 further provide the proposed solutions of fine-grained network virtualization and QVR algorithm, respectively. Section 6 gives the performance evaluation of QVR and Section 7 concludes the paper.

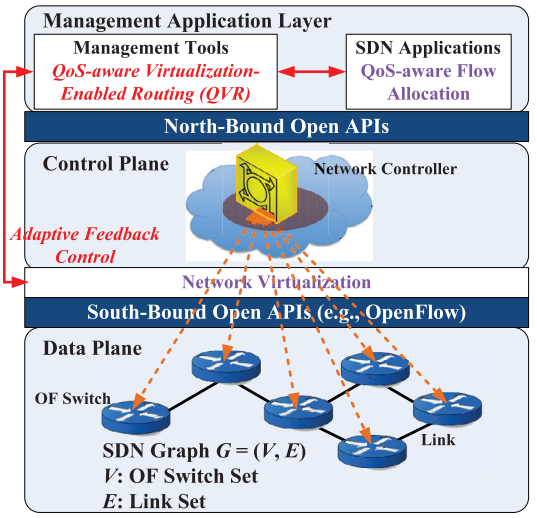


Fig. 1. SDN system architecture.

Table 1
Important notations utilized in this paper.

Notation	*Description*
$G=(V,E)$; $\bar{G}=(V,\bar{E})$	A SDN graph G and its complement $\bar{G}$
$n \in N$	Tenant set with total $\|N\|$ tenants
$G_n=(V_n,E_n)$	The sliced subnet for tenant n
w_n^{max}	The largest allowable path weight of subnet G_n
$W(G_n)$	The weight supported by the subnet G_n
$f \in F^n$	Application flow set of tenant n
$X_{f,i,j}^n$	Achievable packet data rate of link $(i,j) \in E_n^f$, $f \in F^n$
λ_f^n	Packet arrival rate from source node s_f^n with destination node d_f^n
c_{ij}	Link capacity of link $(i,j) \in E$
$D_{n,f}^{max}$	Total delay bound for flow $f \in F^n$
$J_{n,f}$	Jitter bound for flow $f \in F^n$
$p_{n,f}^{max}$	Packet loss bound for flow $f \in F^n$
$D_{f,i}^{q,n}$	Queuing delay at node i of flow $f \in F^n$
$p_{f,i,j}^n$	Packet loss of link $(i,j) \in E_n^f$
$\mathcal{A}_{i,j}^c, \mathcal{A}_{i,j} \subseteq N$	Inactive and active subnets over link $(i,j) \in E$
$Q_{f,i,j}^n$	Queue length of flow $f \in F^n$ over link $(i,j) \in E$
$t_{i,j}^n$	Allocated time portion of link (i,j) to tenant $n \in N$
$\hat{t}_{i,j}^n$	Reallocated time portion from switch hypervisor

2. System model

2.1. SDN architecture and operations

Indicated by [1,3], the system architecture of SDNs consists of three layers (planes), including *management application*, *control*, and *data planes*, as shown in Fig. 1. Specifically, the data plane includes OpenFlow enabled switches (i.e., OF switches) that have multi-flow tables and serve as the packet forwarding devices. This data plane communicates with logically centralized control plane via south-bound open APIs, e.g., OpenFlow [4]. Furthermore, the control plane consists of centralized controller(s) that assign the forwarding rules to switches and enable ubiquitous regulation over the entire network. Similarly, this control plane communicates with management application layer through north-bound open APIs. Moreover, the management application layer consists of several SDN applications and management tools to fulfill various QoS requirements from multiple tenants' specific applications. For example, real-time game applications could have more strict QoS demands than applications with best-effort traffic.

Our objective is to develop a routing algorithm that provides complete isolation of resource usages by multi-tenants upon the same SDN infrastructure, in such a way QoS requirements of all tenants' applications are fulfilled at the same time. Towards this, we propose a two-phase design (i.e., network virtualization with QoS-aware flow allocation) and a feedback management tool (i.e., QVR algorithm). In particular, the first phase of network virtualization sites between data and control planes, and divides the network resources and infrastructure into isolated portions to separate application flows from different tenants. This virtualization maintains the isolated resource portions unchanged regardless of the new flow arrivals. Furthermore, for the second phase of flow allocation, it sites at application layer, allocates flows according to their respective QoS demands, and calculates and adjusts flow paths for every new flow. Moreover, the QVR management tool combines these two phases to determine the suitable paths for new flows as well as to decide the optimal resource allocation according to current network status. In addition, if the flow allocation phase fails to provide the required QoS performance, QVR will further feedback the operations to virtualization phase in order to enable a better network slicing. Therefore, this joint design indeed facilitates the jointly optimized virtualization and routing in SDNs. Note that important notations in this paper are summarized in Table 1.

2.2. Network model

The SDN is modeled by an undirected network graph $G=(V,E)$ as shown in Fig. 1, where V is the set of OF switches with total $|V|$ switches (more complicated models can be considered through the assumption that each node in V can be an aggregated node grouping multiple physical switches) and E is the set of links with total $|E|$ links. Let N denote the set of tenants that shares the same SDN with total $|N|$ tenants. While our objective aims to virtualize the physical network into multiple separated subnets for isolated resource usages among tenants, we introduce some useful notations and definitions as follows. A graph $G'=(V',E')$ is a subgraph (subnet) of graph (network) G, i.e., $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. A connected graph G is a nontrivial graph that any two vertices in graph can be connected by at least one path in G. Moreover, the complement of a graph G, denoted as $\bar{G}=(V,\bar{E})$, has the same node set as G and the edges that are not included in G. A minimum spanning tree of a connected undirected graph G is a tree that includes every node using the minimal set of edges in terms of the given weight function. Note that as it is very unlikely that there always exists a direct physical link between each source–destination pair, it is assumed that there are more than one feasible routes for each pair. Note that we use edge and link interchangeably through the paper.

3. Multi-objective optimization problem for joint virtualization and routing decision

We aim to bring a joint design scheme that decides the completed isolation of resource usages and the optimal flow allocation among multi-tenants' applications at the same time. In the following, we explain each design ingredient in detail and formulate this problem as a multi-objective optimization.

3.1. Network resource slicing

Given multiple tenants sharing a SDN infrastructure, the goal of network resource slicing is to divide the topological resources among multi-tenants wisely, in such a way the resource usages can be isolated and each tenant's preference can be satisfied. Specifically, for $|N|$ tenants upon a SDN G, the subnets for each tenant $G_n = (V_n, E_n), \forall n \in N$, where $V_n \subseteq V$ and $E_n \subseteq E$, should be decided to minimize the overlapped edges (i.e., $|E_n \cap E_m| \to 0$, $\forall 1 \le n < m \le |N|$) and tenants' satisfactions in terms of weights $w_n^{max}, \forall n \in N$ are achieved. These weight parameters can be considered either from graph-centric perspective (e.g., the subnet diameter or size) or QoS-centric perspective (e.g., the largest allowable path delay or packet loss). In particular, we consider

$$W(G_n) < w_n^{max}, \quad \forall n \in N \tag{1}$$

where w_n^{max} denotes the largest allowable path weight of the subnet n and $W(G_n)$ denotes the weight supported by the subnet with respect to all existing paths. Eq. (1) indicates the constraints of weight satisfactions for all tenants. Note that each subnet must be a connected graph, since every pair of nodes in the subnet should be able to communicate with each other.

To provide complete resource isolation among tenants, each subnet should better be the subset of the complement of the other $|N| - 1$ union graph, i.e., $G_n \subseteq \cap_{k=1,k\neq n}^{N} \bar{G}_k$. In general, this edge-disjoint requirement is hard to achieve, especially when there are many tenants sharing few edges of a SDN. Hence, a practical approach is to allow a certain degree of dependent resource utilization by limiting the number of *shared edges* among tenants. In particular, an appropriate network slicing should support the following:

$$\min_{\{G_n;\forall n\in N\}} \sum_{1\le n<m\le |N|} |E_n \cap E_m|. \tag{2}$$

Regarding these shared edges, a flow allocation scheme, e.g., the one explained later in Section 3.2, should be proposed to ensure each dependent tenant's application requirements are fulfilled. Specifically, as mentioned in Section 2.1, the centralized controller of SDN has the capability to monitor and regulate tenants' traffic by implementing routing and resource allocation decision made by the proposed management tools; thus it can adaptively rearrange the shared edges and link capacities accordingly among dependent tenants. Moreover, to further provide the serving differentiation among multiple tenants, it is assumed that tenants' demands are less imperative as the order of tenant $n \in [1, N]$ grows. It implies tenant 1 has the highest serving priority while tenant N has the lowest.

3.2. Flow allocation

To characterize the packet forwarding of application flows upon the given network slicing of multi-tenants, $X_{f,i,j}^n$ denotes the achievable packet data rate of link $(i, j) \in E_n^f$ for the flow f in subnet n, and λ_f^n denotes the packet arrival rate to the source node s_f^n, where $f \in F^n$ and $n \in N$. Therefore, the routing constraint becomes

$$X_{f,i,j}^n = 0, \forall (i, j) \notin E_n^f, f \in F^n, n \in N. \tag{3}$$

It implies that if a link is not sliced to a flow in subnet, the link packet rate should be set to zero for that specific flow. Moreover, given the link capacity c_{ij} for link $(i, j) \in E$, the corresponding capacity constraint is provided as

$$\sum_{n\in N} \sum_{f\in F^n} X_{f,i,j}^n \le c_{ij}. \tag{4}$$

Also, the flow conservation can be considered as the following:

$$\sum_{j;(i,j)\in E_n^f} X_{f,i,j}^n - \sum_{j;(j,i)\in E_n^f} X_{f,j,i}^n = \lambda_f^n \mathbb{I}_{\{i=s_f^n\}}, \quad \forall i \neq d_f^n, f \in F^n, n \in N, \tag{5}$$

where d_f^n denotes the destination of flow f in subnet n and $\mathbb{I}_{\{\cdot\}}$ is an indicator function that gives 1 when the event occurs and 0 otherwise. This implies that the outgoing flows of node should be equal to the incoming flows from neighbors plus flow arrival rate, if the source node is concerned. In addition to the above constraints, the objective of flow allocation is to decide an effective resource utilization for multi-tenants' applications with respect to the given link capacities. Specifically inspired by the work in [9,12,13], for each tenant, the minimum arrival rate among flows should be maximized from the flow allocation; thus,

$$\sum_{n\in N} \max_{\{X_{f,i,j}^n;\forall (i,j)\in E_n^f, f\in F^n\}} \min_{f\in F^n} \lambda_f^n \tag{6}$$

provides the total achievable data rate accordingly.

3.3. End-to-end QoS provisioning

Aiming at supporting a great variety of QoS requirements for tenants' applications, the four major end-to-end QoS are considered as follows. First, given the maximum tolerable delay, jitter, and packet loss for a specific flow (i.e., $D_{n,f}^{max}, J_{n,f}, p_{n,f}^{max} \forall f \in F^n, n \in N$), the QoS constraints are provided respectively as $\forall f \in F^n, n \in N$,

$$D_f^n = \sum_{(i,j)\in E_n^f} \left(\frac{1}{X_{f,i,j}^n} + D_{f,i}^{q,n} \right) < D_{n,f}^{max}; \tag{7}$$

$$var(D_f^n) < J_{n,f}; \tag{8}$$

$$\prod_{(i,j)\in E_n^f} p_{f,i,j}^n < p_{n,f}^{max}, \tag{9}$$

where $1/X_{f,i,j}^n$ and $D_{f,i}^{q,n}$ denote the transmission and queuing delay, respectively, $var(\cdot)$ gives the variance of the delay, and $p_{f,i,j}^n$ denotes the link packet loss. Moreover, to ensure

Table 2
Formulation for jointly virtualization and routing.

Objective:	
$\min_{\{G_n;\forall n\in N\}} \sum_{1\le n<m\le \lvert N\rvert} \lvert E_n \cap E_m\rvert$	(2)
$\sum_{n\in N} \max_{\{X^n_{f,i,j};\forall (i,j)\in E^f_n, f\in F^n\}} \min_{f\in F^n} \lambda^n_f$	(6)
Subject to:	
Subnet weight constraint:	
$W(G_n) < w^{max}_n, \forall n \in N$	(1)
Routing constraint:	
$X^n_{f,i,j} = 0, \forall (i,j) \notin E^f_n, f \in F^n, n \in N$	(3)
Link capacity constraint:	
$\sum_{n\in N}\sum_{f\in F^n} X^n_{f,i,j} \le c_{ij}, \forall (i,j) \in L$	(4)
Flow conservation constraint:	
$\sum_{j;(i,j)\in E^f_n} X^n_{f,i,j} - \sum_{j;(j,i)\in E^f_n} X^n_{f,j,i} = \lambda^n_f \mathbb{I}_{\{i=s^n_f\}}, \forall i \ne d^n_f, f \in F^n, n \in N$	(5)
End-to-end delay constraint:	
$D^n_f = \sum_{(i,j)\in E^f_n} (\frac{l^n_f}{X^{q,n}_{f,i,j}} + D^{q,n}_{f,i}) < D^{max}_{n,f}, \forall f \in F^n, n \in N$	(7)
Jitter constraint:	
$var(D^n_f) < J_{n,f}, \forall f \in F^n, n \in N$	(8)
Packet loss constraint:	
$\prod_{(i,j)\in E^f_n} p^n_{f,i,j} < p^{max}_{n,f}, \forall f \in F^n, n \in N$	(9)
Data rate constraint:	
$X^n_{f,i,j} > \lambda^n_f, \forall (i,j) \in E^f_n, f \in F^n, n \in N$	(10)

sources' arrival rates are supportable by the links of forwarding paths, it implies that $\forall (i,j) \in E^f_n, f \in F^n, n \in N$,

$$X^n_{f,i,j} > \lambda^n_f. \tag{10}$$

Note that considering various tenants' applications, Eqs. (7)–(10) may give different degrees of QoS requirements according to the applications. For example, in interactive applications, the latency is more effective to the need of real-time responses, and thus the delay and jitter constraints are more stringent than the loss constraint. On the other hand, for web browsing or emails, the jitter constraint is not applicable due to its little performance impact, whereas the throughput and loss constraints are of considerable significance. Therefore, the entire formulation for jointly virtualization and routing is well-established and Table 2 summarizes all the details. In the following, we first propose a fine-grained network virtualization to address the network slicing issue in Section 4, and then provide a completed solution for the joint design in Section 5.

4. Fine-grained network virtualization for network resource management

In this section, a fine-grained network virtualization is proposed to manage topological resources among multiple tenants. In particular, the high-level resource management and low-level resource scheduling are introduced by the designated network hypervisor and switch hypervisor, respectively.

4.1. Network hypervisor

As mentioned in Section 3.1, the network slicing should be an efficient resource management that aims to minimize the number of shared edges among tenants, i.e., $|E_n \cap E_m| \to 0$. It thus provides isolated resource usages and minimizes the interference for the multi-tenants' applications. Towards this, the proposed network hypervisor in Algorithm 1 perfectly suits the need of network virtualization, and the details are explained as follows. Given the serving priorities among tenants, i.e., tenant 1 (N) has the highest (lowest), Algorithm 1first deals with tenant 1 subnet, then tenant N subnet, and finally the remaining tenant subnets. Specifically, first of all, for tenant 1, a minimum spanning tree is searched within the SDN graph G using Kruskal's algorithm [14], and the found edges are added to the highest priority subnet G_1. Then, tenant 1's satisfaction is evaluated for Eq. 1 such that all source-destination pairs of application flows should have at least one path with path weight less than w^{max}_1. This path weight can be determined through a

Algorithm 1: Network hypervisor.

Input : Topology G; subnet weights $w^{max}_1, \ldots, w^{max}_N$
Output: Subnets $G_1, \ldots, G_N$
Tenant 1 *subnet*
Run Kruskal's alg. over G % Obtain the min. spanning tree over G
Add found edges into G_1
Find source-destination pairs (u,v) with path weight larger than w^{max}_1
for *each discovered pair* (u,v) **do**
  while *path weight between* $(u,v) > w^{max}_1$ **do**
    Run Dijkstra alg. over G_1 for pair (u,v)
    Add new found edges into G_1
  end
end
Tenant N subnet
Run Kruskal's alg. over $\bar{G}$ % Obtain the max. spanning tree over G
Add found edges into G_N
Find source-destination pairs (u,v) with path weight larger than w^{max}_N
for *each discovered pair* (u,v) **do**
  while *path weight between* $(u,v) > w^{max}_N$ **do**
    Run Dijkstra alg. over G_N for pair (u,v)
    Add new found edges into G_N
  end
end
Tenant $2, \ldots, N-1$ *subnet*
for *each tenant* $n \in \{2, \ldots, N-1\}$ **do**
  Run Kruskal's alg. over $G_n = G \setminus (\cup_{k=1}^{n-1} G_k \cup G_N)$
  Find isolated nodes
  Add edges from isolated nodes into G_n % Transform G_n into a connected graph
  Find source-destination pairs (u,v) with path weight larger than w^{max}_n
  for *each discovered pair* (u,v) **do**
    while *path weight between* $(u,v) > w^{max}_n$ **do**
      Run Dijkstra alg. over G_n for pair (u,v)
      Add new found edges into G_n
    end
  end
end

shortest path algorithm, e.g., Dijkstra's algorithm [15]. In particular, all source-destination pairs that do not satisfy the w_1^{max} constraint are first found. For each discovered pair, Dijkstra's algorithm is run over G_1, and new edges are added to guarantee the corresponding route has a weight less than w_1^{max}. The iteration continues until w_1^{max} is fulfilled in all source-destination pairs inside subnet G_1. Next, for tenant N, similar procedures are executed, except that the maximum spanning tree is considered. In particular, Kruskal's algorithm is applied over the graph $\bar{G}$ for a maximum spanning tree and the found edges are added to the lowest priority subnet G_N. Then, tenant N's satisfaction is evaluated with respect to w_N^{max}. Finally, the remaining subnets G_n, where $2 \le n \le N-1$ are constructed. In particular, Kruskal's algorithm is run over the graph $G \setminus (\cup_{k=1}^{n-1} G_k \cup G_N)$, the isolated nodes are found, and the edges from these isolated nodes to subnet G_n are added. If more than one possible edge for an isolated node is found, network hypervisor will choose the edge with less weight. Then, tenant's satisfaction is evaluated with w_n^{max}. Note that there might be some edges that are already included in the higher priority subnets, and these edges become *shared edges* among tenants' subnets. Therefore, through Algorithm 1, a single SDN can be well-sliced into multiple subnets while minimizing the number of shared edges.

4.2. Switch hypervisor

While network hypervisor provides the sophisticated resource management among tenants, there is a need of an executor to fulfill such a management. Towards this, switch hypervisor is proposed to perform the resource scheduling and consists of two elements, i.e., queue-length based generalized processor sharing (GPS) and Flowvisor [16]. First, while the conventional GPS can easily enable each subnet to operate independently, it does not provide throughput efficiency in reallocating unused network resources (i.e., links) when some subnets are temporally idle, i.e., having all flow queues empty. This is because the sharing time calculation of conventional GPS is insensitive to the queue-length conditions. Instead, we propose a queue-length based GPS that assigns time portions proportional to queue lengths as follows. Consider for link $(i,j) \in E$, there are sets of $\mathcal{A}^c_{i,j}$, $\mathcal{A}_{i,j} \subseteq N$, i.e., inactive and active subnets respectively, and there are F^n application flows with $Q^n_{f,i,j}$ as the queue length of flow f in subnet n. Then, the sharing time of the subnet $n \in \mathcal{A}_{i,j}$ over link $(i,j) \in E$ is obtained by

$$\hat{t}^n_{i,j} = t^n_{i,j} + \sum_{k \in \mathcal{A}^c_{i,j}} \frac{\sum_{f \in F^n} X^n_{f,i,j} Q^n_{f,i,j} \mathbb{I}_{\{(i,j) \in E^f_n\}}}{\sum_{l \in \mathcal{A}_{i,j}} \sum_{f \in F^l} X^l_{f,i,j} Q^l_{f,i,j} \mathbb{I}_{\{(i,j) \in E^f_l\}}} t^k_{i,j}, \tag{11}$$

where $t^k_{i,j}, \forall k \in \mathcal{A}^c_{i,j}$ indicate unused time portions of inactive subnets. Next, a well-known software program, Flowvisor [16], is exploited to virtualize the network according to our designated of network and switch hypervisors. In particular, there are three modes of Flowvisor that can be utilized such as slicing by host IP or MAC address, by port number, and by the protocol type. An example of slicing by IP address is shown in Fig. 2 with the original physical infrastructure and two subnets. The key idea is to determine the source

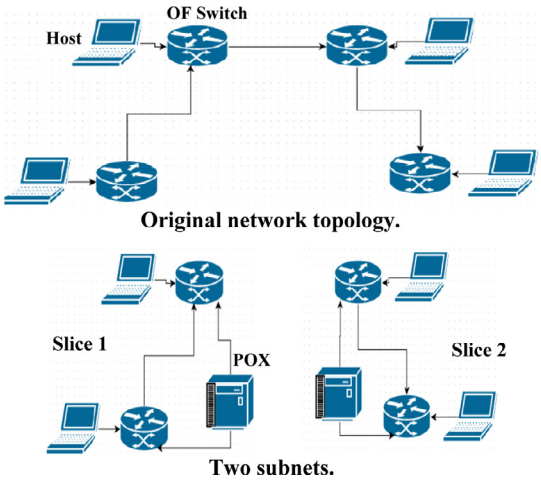


Fig. 2. An example of network slicing by host IP address with two subnets.

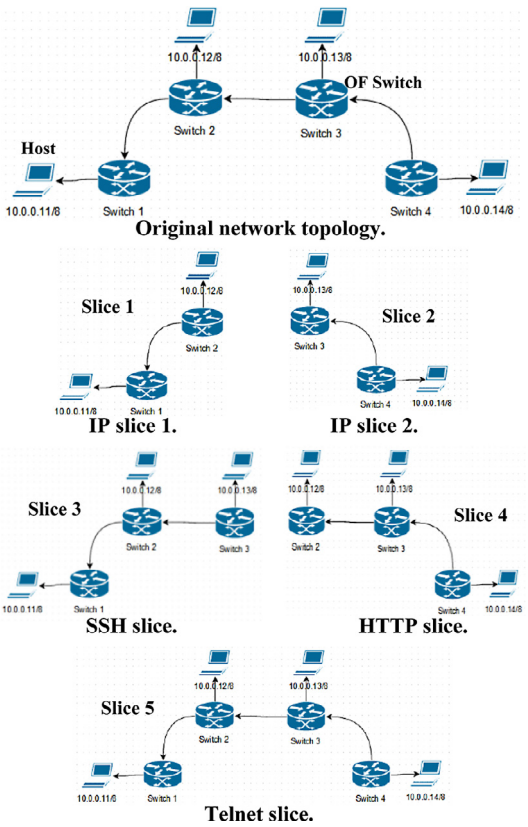


Fig. 3. An example of combined multi-slicing with five subnets.

and destination IP address for each shared switch. Moreover, regarding the better design flexibility, Fig. 3 shows a more complicated example for a combined multi-slicing with five respective subnets. In particular, the original network is first sliced twice according to IP address of hosts connected to the switches, and then is sliced according to protocols such as SSH, HTTP, and Telnet. Hence, we have successfully provided a fine-grained network virtualization that allows slices of multiple subnets to be defined and configured

independently, facilitating the resource isolation and serving priority among tenants.

5. QoS-aware virtualization-enabled routing (QVR) in SDNs

In this section, a management tool of QoS-aware virtualization-enabled routing (QVR) is proposed. In particular, based on the results of network virtualization in Section 4, a dynamic flow allocation is first introduced to guide packet flows in such a way the flow allocation and end-to-end QoS provisioning are supported. Moreover, QVR algorithm is then proposed to combine the designated network hypervisor and QoS-aware dynamic flow allocation, thus enabling an adaptive control for a completed solution for multi-objective optimization problem in Table 2.

5.1. Dynamic allocation framework with QoS provisioning

QoS-aware Dynamic Flow Allocation in Algorithm 2 provides a dynamic solution to maximize the minimum flow

Algorithm 2: QoS-aware dynamic flow allocation.

Input : $G_1, \ldots, G_N$; link capacity $\{c_{ij}\}$; QoS indices $\{D^{max}_{n,f}, J_{n,f}, p^{max}_{n,f}\}$
Output: Total data rate λ^*; flow allocation $\{X^n_{f,i,j}\}$

1 **for** *each subnet* G_n **do**
2 **for** *each flow* $f \in F^n$ **do**
3 **Find** all paths between (s^n_f, d^n_f)
4 **for** *each path* k **do**
5 **Initialize** $\lambda^n_{f,k} = \infty$
6 **while** ***NSQP***(*Eq. (7)-(10)*)$,\lambda^n_{f,k}$ **do**
7 $\lambda^n_{f,k} \leftarrow \lambda^n_{f,k} - 1$
8 **Find** $\{X^n_{f,i,j}\}$ for path k satisfying Eq. (4)-(6)
9 **end**
10 **end**
11 $\lambda^n_f = \max_k \lambda^n_{f,k}$
12 **end**
13 **Find** $\lambda^n = \min_{f \in F^n} \lambda^n_f$
14 **end**
15 $\lambda^* = \sum_{n \in N} \lambda^n$

arrival rates for multiple tenants. First of all, for each application flow in subnets, all available paths between flow source and destination are found. Next, with respect to these paths, the constraints of flow allocation in Section 3.2 and end-to-end QoS provisioning in Section 3.3 are evaluated, and the maximum achievable arrival rates are yielded accordingly. In particular, **NSQP** function in line 6 in Algorithm 2 enables the QoS constraint evaluations that gives the true value if the QoS provisioning is not satisfied with the given arrival rate in the current round. In that case, the algorithm enables the successive round to reduce the arrival rate and finds the suitable flow allocation accordingly via line 8; otherwise, the algorithm stops and the optimal solution is reached. Finally, the maximum rate among available paths of a flow is selected as the flow arrival rate, and the minimum flow rate in a subnet and the total rates over multi-tenants can be obtained accordingly. Therefore, due to the simple operations of Algorithm 2, it provides a feasible solution in a timely manner, facilitating dynamic flow allocation for all tenants' applications.

5.2. QoS-aware virtualization-enabled routing (QVR)

QVR algorithm in Algorithm 3 provides an adaptive feed-

Algorithm 3: QoS-aware virtualization-enabled routing (QVR)

Input : G; $\{w^{max}_n\}$; $\{c_{ij}\}$; $\{D^{max}_{n,f}, J_{n,f}, p^{max}_{n,f}\}$
Output: $\{G_n\}$; λ^*; $\{X^n_{f,i,j}\}$

1 **while** $\lambda^* == 0$ **do**
  Network Virtualization Phase
2 $\{G_n\} \leftarrow$**Algorithm 1**$(G, \{w^{max}_n\})$
  Flow Allocation Phase
3 $(\lambda^*; \{X^n_{f,i,j}\}) \leftarrow$**Algorithm 2**$(\{G_n\}, \cdot)$
4 **end**

back control between network hypervisor in Algorithm 1and dynamic flow allocation in Algorithm 2, yielding a completed solution for jointly optimized virtualization and routing decision. In particular, network hypervisor is first executed to partition the SDN infrastructure into several subnets for multi-tenants. Once the network slicing is finished, dynamic flow allocation is applied to provide the optimal network throughput and the corresponding flow assignments. Normally, dynamic flow allocation can resolve all impacts from time-varying QoS requirements, network topologies, and link capacities by re-running the allocation accordingly, and network hypervisor only needs to run once and for all. However, if the allocation cannot provide a feasible solution, i.e., λ^* remains zero, QVR will feedback to network hypervisor and enable a better network slicing as well as the achievable flow allocation. Therefore, upon this stage, we have successfully presented an adaptive solution that completely solve the joint virtualization and routing problem in a timely manner, facilitating the practical implementations of multi-tenants' applications in SDNs.

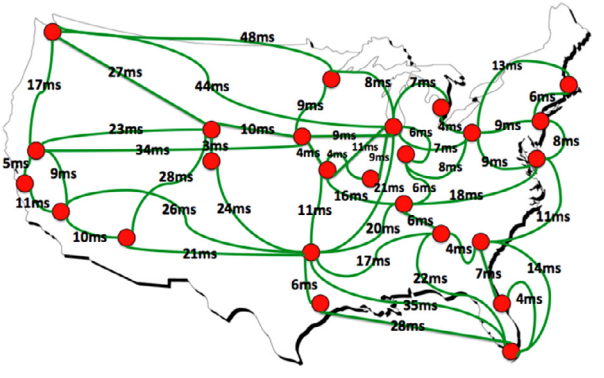
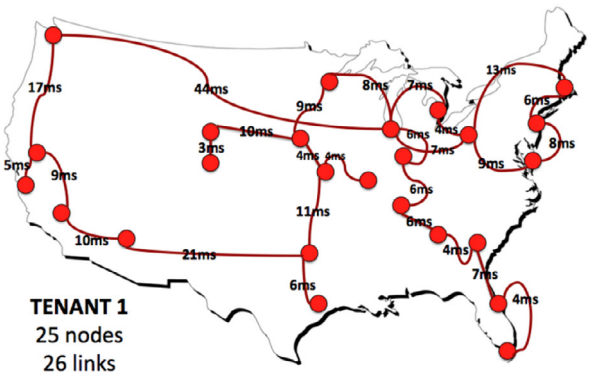
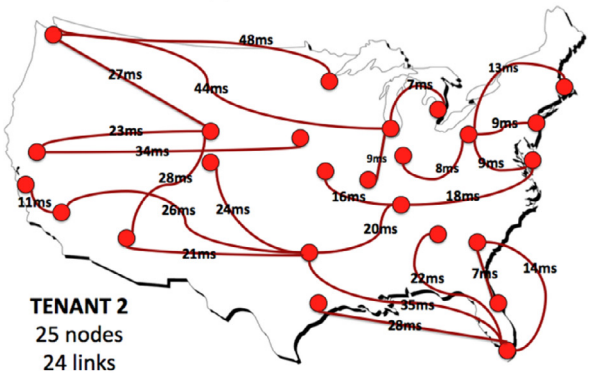


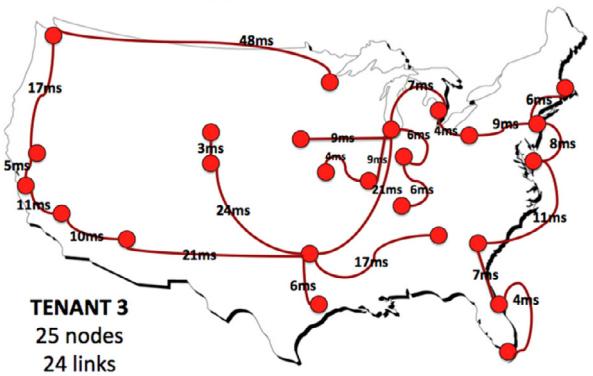
Fig. 4. Network topology of real Sprint GIP network [17] over North America.



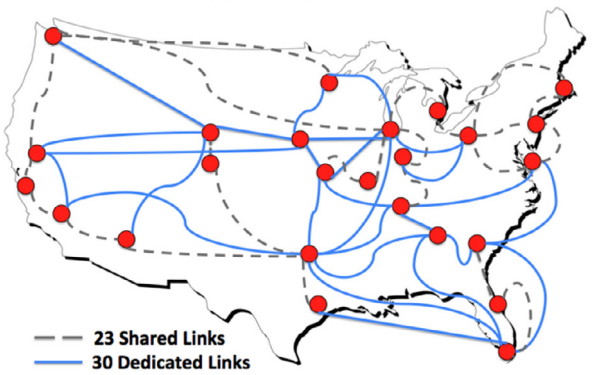
(a) Tenant 1 subnet.



(b) Tenant 2 subnet.

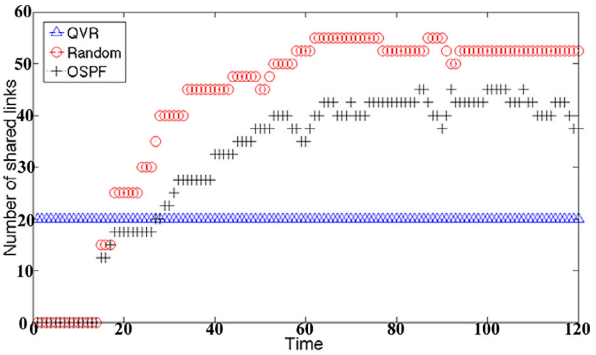


(c) Tenant 3 subnet.

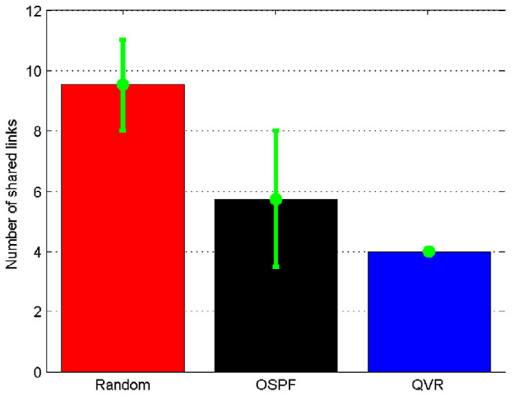


(d) Shared and dedicated links.

Fig. 5. Tenants' subnets and shared links. (a) Tenant 1 subnet. (b) Tenant 2 subnet. (c) Tenant 3 subnet. (d) Shared and dedicated links.



(a) Time evolution of a single evaluation.



(b) Results of 20 experiments.

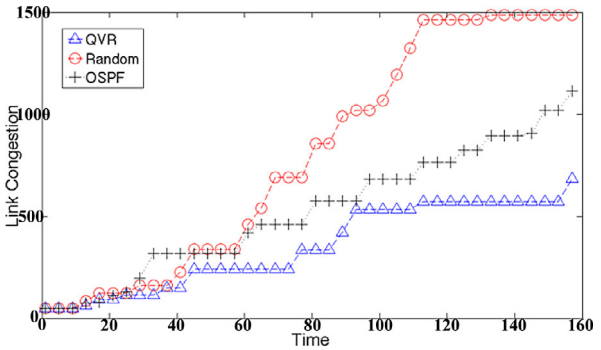
Fig. 6. Number of shared links with respect to different flow allocations. (a) Time evolution of a single evaluation. (b) Results of 20 experiments.

6. Performance evaluation

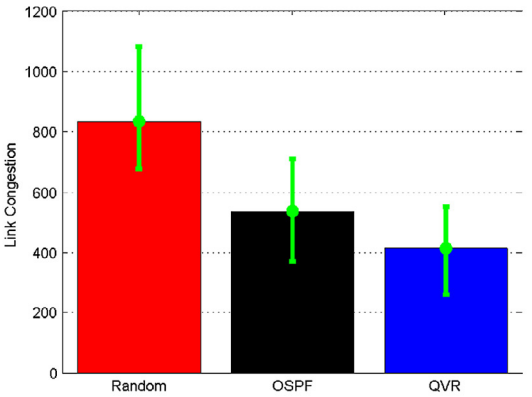
To evaluate the proposed QVR algorithm in Algorithm 3, a Sprint network topology [17] with 25 nodes and 53 links is considered as the SDN infrastructure shown in Fig. 4. It is assumed that there are three tenants, operating in the infrastructure. In particular, tenant 1 generates traffic from real-time applications, which is modeled with Pareto arrivals due to traffic burstiness; tenant 2 and tenant 3 generate traffic from non-real-time applications, which are modeled with exponential arrivals. A random variable $X \in \mathcal{PAR}(\alpha, x_m)$ if it follows Pareto distribution with parameters α and x_m, i.e., $P(X > x) = (x_m/x)^{\alpha}$. A random variable $X \in EXP(\lambda)$ if it follows exponential distribution with parameter λ, i.e., $P(X > x) = e^{-\lambda x}$. Towards this, we have $\mathcal{PAR}(1.5, 10)$ for tenant 1, $EXP(0.2)$ for tenant 2, and $EXP(0.4)$ for tenant 3. The link service time follows an exponential distribution. Moreover, the highest allowable path delay is selected as the maximum path weight, and the values for three tenants are set as 100 ms, 400 ms, and 500 ms, respectively, according to their specific applications. In the following, we first evaluate the network slicing capability of QVR, and then examine the QVR performance in detail.

6.1. Network slicing

Fig. 5 shows the tenant isolation by QVR algorithm, particular from network hypervisor. Specifically, Fig. 5(a)–(c)



(a) Time evolution of a single evaluation.



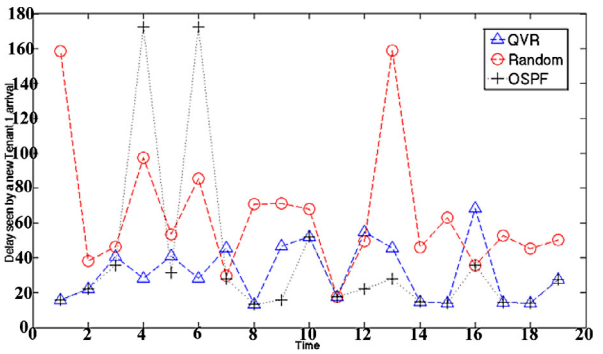
(b) Results of 20 experiments.

Fig. 7. Number of congested links with respect to different flow allocations. (a) Time evolution of a single evaluation. (b) Results of 20 experiments.

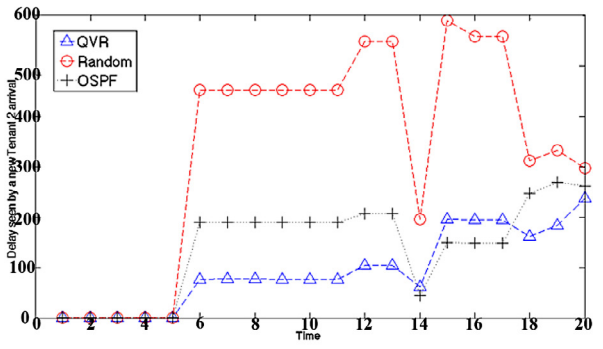
provide the subnets for tenant 1–3, respectively, and Fig. 5(d) further provides the superposition of three tenants, where the shared links are highlighted by the dotted lines. The results imply that after executing the network hypervisor, three subnets are obtained from the original physical network, and the subnets are built to have the minimum shared links. Regarding these few shared links, the slicing of link capacity among sharing subnets will be taken care by the dynamic flow allocation, fulfilling fine-grained network virtualization. Given the subnets of three tenants in Fig. 5, in the following we compare the performance of QVR upon this network slicing with the conventional flow allocation solutions.

6.2. Performance of QVR

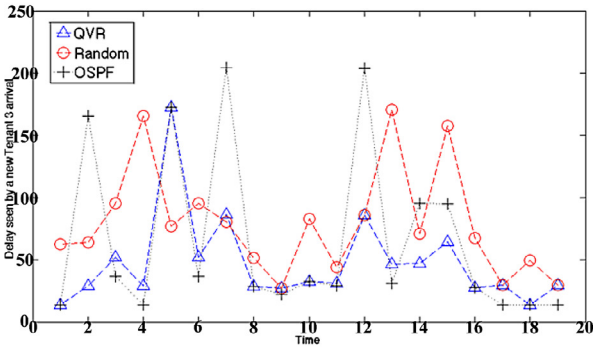
The number of shared links and the latency from congested links are first examined through the perspectives of time evolution and multiple experiments for three approaches: OSPF [18], random flow allocation, and QVR. In particular, OSPF decides the flow allocation and corresponding network slicing through shortest-path algorithm, whereas random flow allocation determines those through random path selections. Fig. 6 shows the results of shared links, and indicates that under both OSPF and random approach, the link number reaches the maximum value in an early stage. However, QVR can maintain the number of shared link as a



(a) Tenant 1 subnet.



(b) Tenant 2 subnet.



(c) Tenant 3 subnet.

Fig. 8. End-to-end packet delay with respect to three tenants. (a) Tenant 1 subnet. (b) Tenant 2 subnet. (c) Tenant 3 subnet.

constant value due to its network hypervisor, thus achieving better tenant isolation. Moreover, Fig. 7 shows the congestion latency, and indicates that QVR performs much better with less latency than OSPF and random solution. The reason for QVR's superiority comes from the optimized route selections that wisely utilizes the link capacity for current flows and thus can provide more bandwidths upon bottleneck links for future flow arrivals. In short, the above results show that QVR accomplishes tenant isolation while improves the congestion latency, allowing more incoming flows from tenants.

The delay perceived by new flow arrivals of three tenants are provided in Fig. 8. In particular, tenant 1 considers real-time applications that brings more bursty traffic and requires much less delay than the non-real-time traffic of tenant 2 and tenant 3. The results show that OSPF and random solution

cannot fulfill tenant 1's requirements due to the large and highly fluctuated delay, and provides much greater delay for both tenants 2 and 3. On the other hand, QVR provides little delay with less fluctuation for all three tenants, thus confirming its efficacy. In summary, QVR meets the need of joint virtualization and routing, facilitating reliable and efficient transmissions for multi-tenants' applications in SDNs.

7. Conclusion

In this paper a joint design of optimized QoS-aware virtualization and routing is addressed as a multi-objective optimization problem. This complex optimization is completely solved in a timely manner through the proposed management tool of QVR in SDN controller. Specifically, with the aid of incorporation between network virtualization and flow allocation, QVR enables an adaptive solution with respect to time-varying QoS requirements, network topologies, and traffic statistics that slices the network resource among multi-tenants' applications and decides the optimal flow routes to fulfill the QoS guarantees for applications. Performance evaluation confirms QVR outperforms the conventional approaches with less shared edges, congestion latency, and traffic delay for multi-tenants. Therefore, we have presented a novel paradigm to facilitate virtualization-enabled traffic engineering for centralized controller in practical SDN implementations.

References

[1] I.F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, A roadmap for traffic engineering in SDN-OpenFlow networks, Comput. Netw. 71 (2014) 1–30.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.

[3] I.F. Akyildiz, P. Wang, S.C. Lin, Softair: a software defined networking architecture for 5G wireless systems, Comput. Netw. 85 (2015) 1–18.

[4] ONF, OpenFlow Switch Specification, version 1.4.0 [online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[5] D. Shue, M.J. Freedman, A. Shaikh, Performance isolation and fairness for multi-tenant cloud storage, in: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation, 2012, pp. 349–362.

[6] W.-T. Tsai, Q. Shao, J. Elston, Prioritizing service requests on cloud with multi-tenancy, in: Proceedings of the IEEE 7th International Conference on e-Business Engineering (ICEBE), 2010, pp. 117–124.

[7] S. Shenker, Fundamental design issues for the future internet, IEEE J. Sel. Areas Commun. 13 (7) (1995) 1176–1188.

[8] S. Shenker, L. Zhang, D. Clark, A scheduling service model and a scheduling architecture for an integrated services packet networks ps, Xerox Parc, Tech. Rep., 1993 [online]. Available: ftp://ftp.parc.xerox.com/pub/archfin.

[9] S. Agarwal, M. Kodialam, T.V. Lakshman, Traffic engineering in software defined networks, in: Proceedings of the 2013 IEEE INFOCOM, 2013, pp. 2211–2219.

[10] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, A. Vahdat, B4: experience with a globally-deployed software defined WAN, in: Proceedings of the ACM SIGCOMM, 2013, pp. 3–14.

[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of Np-Completeness, W. H. Freeman & Co., New York, NY, USA, 1979.

[12] B. Fortz, M. Thorup, Optimizing OSPF/IS-IS weights in a changing world, IEEE J. Sel. Areas Commun. 20 (4) (2002) 756–767.

[13] J. Chu, C.-T. Lea, Optimal link weights for IP-based networks supporting hose-model VPNs, IEEE/ACM Trans. Netw. 17 (3) (2009) 778–788.

[14] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Am. Math. Soc. 17 (1) (1956) 48–50.

[15] E.W. Dijkstra, A note on two problems in connection with graphs, Numer. Math 1 (1) (1959) 269–271.

[16] R. Sherwood, G. Gibby, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, Flowvisor: A Network Virtualization Layer, 2009, Deutsche Telekom Inc. R&D Lab., Stanford University, Nicira Networks, Tech. Rep. OPENFLOW-TR-2009-1.

[17] Sprint, Overland Park, KS, Sprint IP network performance, 2011 [online]. Available: http://www.sprint.net/performance.

[18] J. Moy, OSPF version 2, 1998, IETF RFC 2328.

Shih-Chun Lin (S'08) received the B.S. degree in electrical engineering and the M.S. degree in communication engineering from National Taiwan University in 2008 and 2010, respectively. He is a graduate research assistant in the Broadband Wireless Networking Laboratory (BWN Lab), School of Electrical and Computer Engineering, Georgia Institute of Technology. Currently, he is working toward the Ph.D. degree in electrical and computer engineering under the supervision of Prof. Ian F. Akyildiz. His research interests include wireless underground sensor networks, software defined networking, large machine-to-machine communication, cognitive radio networks, and statistical scheduling in wireless systems.

Pu Wang (M'05) received the B.S. degree in electrical engineering from the Beijing Institute of Technology, Beijing, China, in 2003, the M.Eng. degree in computer engineering from the Memorial University of Newfoundland, St. Johns, NL, Canada, in 2008, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 2013. He is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS. Dr. Wang received the BWN Lab Researcher of the Year award in 2012, Georgia Institute of Technology. He received the TPC top ranked paper award of IEEE DySPAN 2011. He was also named Fellow of the School of Graduate Studies, 2008, Memorial University of Newfoundland. His research interests include wireless sensor networks, cognitive radio networks, software defined networks, nanonetworks, multimedia communications, wireless communications in challenged environment, and cyber-physical systems.

Min Luo received the Ph.D. degree in Electrical Engineering from Georgia Institute of Technology, Atlanta, GA USA, in 1992. He also held the B.S. and M.S. degrees in 1982 and 1987, respectively in Computer Science. Currently, he is the Head and Chief Architect of the Advanced Networking at Huawei's Shannon (IT) Lab, leading the research and development in Software Defined Networking (SDN) and other future networking initiatives. He served as Chief/Executive Architect for IBM SWG's Strategy and Technology, Global Business Solution CenterGCG, Industry Solutions, and Center of Excellence for Enterprise Architecture and SOA for more than 11 years. He also worked as Senior Operations Research Analyst, Senior Manager and Director of Transportation Network Planning and Technologies for two Fortune 500 companies for seven years. He's certified and awarded as the Distinguished Lead/Chief Architect from Open Group in 2008. He is an established expert in the field of next negation software defined networking (SDN), enterprise architecture and information systems, whole life cycle software application and product development, business intelligence, and business process optimization. He is also a pioneer and one of the recognized leading experts and educators in Service-oriented architecture (SOA), Model/business-driven architecture and development (MDA-D), and component/object-oriented technologies. He coauthored two books, including the pioneering Patterns: Service Oriented Architecture and Web Services in 2004, and published over 20 research papers. As a senior member of IEEE, he has been serving on the organizing committee for IEEEs ICWS, SCC and CC (Cloud Computing) Conferences, chaired sessions, presented several tutorials on SOA and Enterprise Architecture and their best practices and gave lectures at the Service University. He has served as adjunct professors in several USA and Chinese universities since 1996.