SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 6, NO. 2
ISSN: 2054 -7420

End-to-End Service Delivery with QoS Guarantee in Software Defined Networks

Qiang Duan
Information Sciences & Technology Department, Pennsylvania State University, Abington USA
qduan@psu.edu

ABSTRACT

Software-Defined Network (SDN) is expected to have a significant impact on future networking. Although exciting progress has been made toward realizing SDN, application of this new networking paradigm in the future Internet to support end-to-end QoS provisioning faces some new challenges. The autonomous network domains coexisting in the Internet and the diverse user applications deployed upon the Internet call for a uniform Service Delivery Platform (SDP) that enables high-level network abstraction and inter-domain collaboration for end-to-end service provisioning. However, the currently available SDN technologies lack effective mechanisms for supporting such a platform. In this paper, we first present an SDP framework that applies the Network-as-a-Service (NaaS) principle to provide network abstraction and orchestration for end-to-end service provisioning in the SDN-based future Internet. Then we focus our study on two enabling technologies for such an SDP to achieve QoS guarantee; namely a network abstraction model and an end-to-end resource allocation scheme. Specifically, we propose a general model for abstracting the service capabilities offered by network domains and develop a technique for determining the required amounts of bandwidth in network domains for end-to-end service delivery with QoS guarantee. Both the analytical and numerical results obtained in this paper indicate that the NaaS-based SDP not only simplifies SDN service and resource management but also enhances bandwidth utilization for end-to-end QoS provisioning.

Keywords: Software-Defined Network (SDN), Service Delivery Platform (SDP), Network-as-a-Service (NaaS), QoS Provisioning.

1 Introduction

Software-Defined Network (SDN) is emerging network architecture that may have a significant impact on the development of future networking technologies. SDN architecture decouples network control and data forwarding functions; thus enabling network control to become directly programmable and underlying network infrastructure to be abstracted for applications [1]. Key features of SDN include separation between control plane and data plane, logically centralized network control, and programmability of the control plane. These features combined together gives SDN some great advantages in networking, including simplified and enhanced network configuration and operation, flexible and efficient network control and management, and improved network performance for meeting various application requirements. Therefore, SDN is expected to play a crucial role in the future Internet.

DOI: 10.14738/tnc.62.4373
Publication Date: 13th April 2018
URL: http://dx.doi.org/10.14738/tnc.62.4373

SDN architecture and its enabling technologies recently formed an important research area that has attracted extensive attention from both academia and industry. Active research topics in this area include SDN-enabled switching devices, SDN controllers, network operating systems, various network control/management applications, protocols between the data and control planes (southbound interface), and Application Programming Interfaces (APIs) for programming the control plane (northbound interface). Exciting progress has been made on SDN development and numerous research results have been reported in literature [2], [3], [4].

Although the SDN architecture has been successfully applied in some networking systems such as enterprise networks, data center networks, and inter-data center communications, adoption of this new networking paradigm in a large-scale internetworking scenario such as the future Internet faces new challenges that must be further investigated. One of the key issues lies in end-to-end service delivery across heterogeneous network domains with QoS guarantee for meeting diverse user requirements. In an enterprise or data center network, the user applications, network controller, and data forwarding devices all belong to the same administration domain; therefore, information of underlying network infrastructure can be made available to upper layer applications easily. However, in the Internet end users (computing applications) and network service providers often belong to different domains; therefore detailed information of network states may not be directly visible to applications. In addition, end-to-end communication paths in the Internet often traverse multiple autonomous systems operated by different organizations. End-to-end service provisioning in such a heterogeneous networking scenario requires a higher-level network abstraction for flexible interaction between users and service providers and loose-coupling collaboration among the involved autonomous systems. This calls for a service delivery platform that supports flexible and effective user-network interaction and inter-domain collaboration.

However, currently available SDN technologies lack an effective mechanism for building such a service delivery platform. Although a variety of SDN controllers have been developed, there is no standard yet for achieving interoperability between these controllers. What resulted is that no single vendor could deliver a standard-based northbound API for application development, or a standardized interface between controllers. In a large-scale inter-domain networking scenario, it is not feasible to require all autonomous network domains to adopt the same type of SDN controller. Therefore, lack of interoperability between SDN controllers prevents applications from functioning seamlessly across different controllers for inter-domain network service provisioning. Recent works on inter-domain networking in SDN mainly focused on distributed collaboration between SDN controllers for routing. End-to-end service delivery across heterogeneous SDN domains has not been sufficiently studied.

Recently, application of the service-orientation principle in SDN to address the challenging problem of end-to-end service delivery started attracting researchers' attention. The Service-Oriented Architecture (SOA) [5] offers an effective mechanism to enable flexible interactions among autonomous systems to meet diverse service requirements. SOA has been widely adopted in various areas, including Cloud computing and Web services, as the main model for service delivery. Application of the SOA principle in networking leads to a Network-as-a-Service (NaaS) paradigm, which enables networking resources and functionalities to be utilized by users as services through a standard abstract interface, much like computational resources are utilized as services in Cloud computing. NaaS enables abstraction of networking systems into network services that can be discovered, selected, and accessed by users; thus offering a flexible mechanism for user-network interaction. Network services can also be orchestrated for

end-to-end service provisioning. Network abstraction enabled by NaaS also allows flexible collaboration among autonomous network domains via loose-coupling service interactions. Therefore, NaaS may greatly facilitate end-to-end service delivery in the future Internet.

The research work reported in this paper tackles the challenging problem of end-to-end service delivery in SDN by exploiting the NaaS notion. We first present a framework of a Service Delivery Platform (SDP) that applies the NaaS paradigm in SDN to enable high-level network abstraction and inter-domain network service orchestration. Then we focus our study on two enabling technologies for the SDP to provide end-to-end QoS guarantee; namely an abstraction model for network service capabilities and an end-to-end resource allocation scheme for performance guarantee. Specifically, we propose a general model for abstracting service capabilities of network domains, which is then applied to composite network services for modeling capability of end-to-end service delivery. Based on the service capability model for network abstraction, we develop a technology that can be employed at the SDP to determine the required amount of bandwidth for achieving end-to-end QoS guarantee. Bandwidth utilization of the SDP is then analyzed and the obtained results show that such an SDP with a global network view may improve bandwidth utilization for end-to-end QoS provisioning.

SDN brings in potential benefits for enhancing future networking from at least two aspects: i) simplifying network control and management and ii) enhancing service provisioning for meeting diverse application requirements. Although the first aspect has been explored by many efforts, the second aspect has received less attention. The proposed SDP framework and the relevant technologies developed in this paper aim to address this issue in order to fully realize the potential of the emerging SDN paradigm in the future Internet.

The rest of the paper is organized as follows. In Section II we discuss challenges to end-to-end service delivery in SDN and review related works. A framework of a NaaS-based SDP for end-to-end service delivery in SDN is presented in Section III. We propose a high-level abstraction model for network service capabilities and apply the model to end-to-end network services in Section IV. Then in Section V we develop a technique for determining required bandwidth to achieve end-to-end QoS guarantee and analyze bandwidth utilization achieved by the SDP with this technique. Numerical results are provided in Section VI. We draw conclusions in Section VII.

2 End-to-end Service Delivery in SDN – Challenges and Solutions

2.1 Challenges to End-to-End Service Delivery in SDN

Recent rapid advancement in SDN research has yielded diverse technologies for realizing this new network architecture. Various SDN-enable switches have been developed. Although OpenFlow [6] has been widely adopted for controlling switches in the data plane, it is not the only southbound interface for

SDN. Possible protocols that may potentially play the same role include Forwarding and Control Element

Separation (ForCES) [7], Path Computation Element Communication Protocol (PCEP) [8], Protocol Oblivious Forwarding (POF) [9], and OpFlex [10]. Wide varieties of SDN controllers and network operating systems have also been developed. These include both centralized controllers such as NOX [11], Beacon [12], and Floodlight [13], and distributed network operating systems such as ONIX [14], ONOS [15], and HyperFlow [16].

Diversity in available SDN technologies brings in challenges to end-to-end service provisioning across multiple domains in SDN-based future Internet. Autonomous systems in the Internet should have the freedom to employ various SDN technologies, including switches, southbound protocols, and network controllers, that fit their particular networking needs. On the other hand, the objective of service provisioning is to deliver network services across the heterogeneous domains for meeting the diverse requirement of end users. Therefore, end-to-end service provisioning in the future Internet requires not only effective inter-domain collaboration but also flexible interaction between upper layer user applications and the underlying network domains.

However, the currently available SDN technologies lack sufficient capability of meeting this requirement for end-to-end service delivery. Development of network controllers often lacks consideration of interoperability with controllers from other vendors. Distributed network controllers mainly focus on cooperation among multiple homogeneous controllers in the same domain; thus are insufficient to handle heterogeneity of the controllers in multi-domain cases. Moreover, despite rapid development on standard southbound interface, currently there is no common standard for the northbound API between SDN controllers and network control/management applications. These applications are often developed based on the API provided by a particular type of controller; thus are tightly coupled with the controller design. Such tight coupling between applications and controllers significantly limits the capability of service provisioning over heterogeneous controllers in a multi-domain SDN environment.

Recently some study on inter-domain issues in SDN has been reported in the literature. SDNi [17] is a protocol recently proposed by IETF for coordinating operations and exchanging information between SDN controllers in different domains. The implementation of SDNi suggested in [17] is to extend BGP for information exchange. However, the hop-by-hop nature of BGP makes routing among domains in a decentralized manner without knowledge of end-to-end routes, which may not be able to achieve a global optimal path for end-to-end QoS provisioning. Research reported in [18] and [19] employs the SDN principle to address the inter-domain routing problem. Both works are based on BGP; thus are limited by its decentralized feature to fully realize the SDN benefit of centralized control with a global network view. The inter-AS routing proposed in [18] assumes that homogeneous controllers, specifically the NOX-OpenFlow controller, are used in all domains; thus may not be applicable to large-scale multi-domain scenarios. The multi-AS routing control platform proposed in [19] assumes the existence of a mechanism to communicate with SDN domain controllers without detailed discussion on the realization of such a mechanism.

In [20] the authors argue that BGP is a poor candidate for inter-domain routing in SDN and propose decoupling between routing and policy control to facilitate interoperability among SDN domains. The distributed control plan proposed in [21] employs a message-oriented communication bus for information exchange among SDN domain controllers. The aforementioned research focuses on controller collaboration for inter-domain routing in SDN. End-to-end service provisioning needs more than just routing across multiple domains. Flexible interaction between user applications and the SDN controllers in different domains of the underlying network infrastructure is another important aspect that so far has received little attention. It requires a high-level network abstraction, loose-coupling interaction between applications and controllers, and flexible collaboration among heterogeneous controllers.

2.2 Network-as-a-Service in SDN – a Promising Solution

The Service-Oriented Architecture (SOA) [5] offers a promising approach to addressing the challenges for end-to-end service provisioning in multi-domain SDN. The SOA can be described as architecture within which all functions are defined as independent services with invokable interfaces that can be called in defined sequences to form business processes. A *service* in SOA is a module that is self-contained (i.e., the service maintains its own states) and platform-independent (i.e., the interface to the service is independent with its implementation platform). Services can be described, published, located, orchestrated, and programmed through standard interfaces and messaging protocols. A key feature of SOA is "loose-coupling" interaction among heterogeneous systems, which allows entities to collaborate with each other while keep themselves independent. This feature makes SOA very effective architecture for coordinating heterogeneous systems to provide services that meet various application requirements.

Application of the service-orientation principle in networking provides a promising approach to addressing some challenges in the future Internet. Such a service-oriented networking paradigm is referred to as *Network-as-a-Service* (NaaS), in which networking resources are abstracted and utilized in form of SOA-compliant network services. In principle, a network service may represent any type of networking component at different levels, including an entire network domain, a single physical or virtual network, or an individual network node. Multiple network services can be combined into one composite inter-network service through a service orchestration mechanism.

Recently the NaaS paradigm has started attracting attention from the networking research community and interesting progress has been reported in the literature. Costa *et al.* proposed a NaaS model for data center networks in [22] for enabling Cloud tenants to have direct access to network infrastructure for improving service performance. Cloud-based network architecture that combines the Cloud service model with the network openness enabled by SDN was proposed in [23] in order to offer various network protocol services. An SDN control platform called Meridian was presented in [24], which provides a service-level network model with connectivity and policy abstractions for Cloud networking. Bueno and his colleagues developed a NaaS-based Network Control Layer (NCL) that provides an abstraction layer to obtain homogeneous control over heterogeneous network infrastructure [25].

The above works made interesting progress of applying NaaS in SDN for network service provisioning; however, they mainly focus on single-domain cases or assume homogeneous SDN controllers. The framework proposed in [22] assumes that applications can directly acquire detailed knowledge of underlying network infrastructure, which is reasonable in a single data center environment but not realistic for the large scale Internet with multiple autonomous domains. The prototype given in [23] for realizing the proposed network architecture used NOX controller and OpenFlow protocol for controlling all switches. Both Meridian platform and NCL were implemented based only on Floodlight controller. Cooperation between SDN domains with heterogeneous controllers for end-to-end service delivery is still an opening issue that has not been sufficiently addressed yet.

In order to address this important challenging issue, preliminary study of applying NaaS in SDN to support end-to-end QoS provisioning was presented in our previous work [26]. In this paper, we further develop the idea of NaaS-SDN integration to propose a framework of a NaaS-based Service Delivery Platform (SDP) for a multi-domain SDN environment. This platform provides a high-level abstraction of each SDN domain as a network service and enables network service orchestration for end-to-end service delivery. Then we

particularly investigate two key technologies for achieving end-to-end QoS guarantee through this SDP – an abstract model for network service capabilities and a technique for end-to-end bandwidth allocation. Our analysis results also indicate that end-to-end service delivery enabled by the SDP with a global network view improves resource utilization for QoS provisioning.

The research reported in [27] and [28] shares some similar ideas with the work presented in this paper. Zhu *et al.* proposed Software Service Defined Network (SSDN) architecture in [27], which employs SOA-based Enterprise Service Bus (EBS) to build a network software service layer that allows networking resources in multiple domains with different SDN controllers to be federated for end-to-end service delivery. However, some key technologies for achieving QoS guarantee with such a service layer, for example abstraction of service capabilities and cross-domain resource allocation, were not addressed in [27]. The authors of [28] developed a distributed QoS architecture for SDN, which employs a hierarchical control plane where a super controller coordinates the local SDN controllers in multiple domains to support end-to-end multimedia streaming. However, [28] did not give any specific mechanism for the super controller to coordinate heterogeneous SDN controllers in different network domains for service delivery. On the other hand, the research of [27], [28] and our work reported in this paper may complement each other. The ESB-based mechanism described in [27] may be applied to implement communications among the SDP, domain controllers, and end user applications in the framework proposed in this paper. Our SDP framework offers an approach to realizing the hierarchical control plane presented in [28].

3 A NaaS-Based Service Delivery Platform in SDN

The framework of a NaaS-based Service Delivery Platform (SDP) in a multi-domain SDN environment is shown in Figure 1. In this framework, each network domain may have its own choice of SDN technologies, including data plane switches, SDN controllers, and the southbound interface. A domain may also implement various control programs upon its own SDN controller to perform functions such as QoS routing and traffic engineering within the domain scope. Each network domain is abstracted as a network service through a NaaS interface, which provides a high-level abstraction of networking capabilities of the entire domain, including both forwarding and control functionalities, to the SDP. The NaaS interface also allows the SDP to specify its networking requests and policies to each domain. The NaaS-based network abstraction makes network infrastructure of each domain transparent to upper layer applications; thus enabling SDP to coordinate the resources provided by network domains for delivering network services to support diverse user applications.

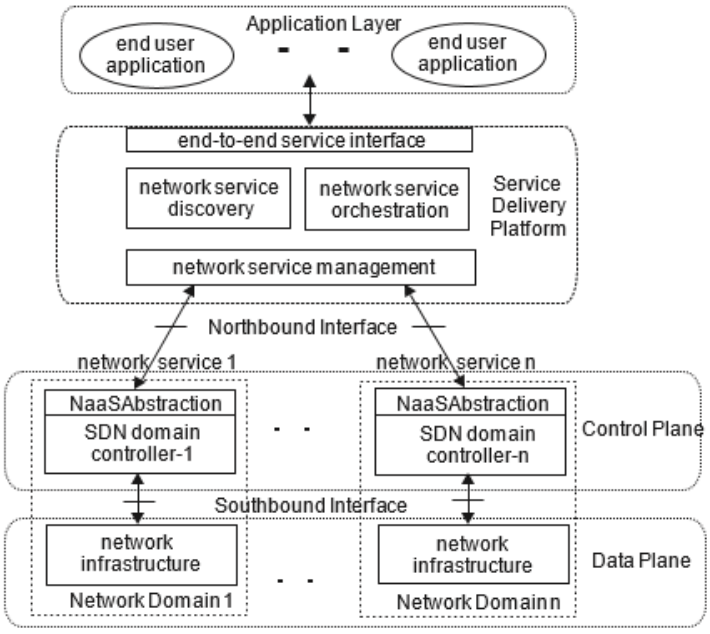

Figure 1. NaaS-based Service Delivery Platform in Software-Defined Network

The SDP serves as a middleware between upper layer user applications and the underlying network infrastructure consisting of heterogeneous domains. Key components of the SDP include a service interface and the modules for service management, service discovery, and service orchestration. The SDN controller of each network domain is responsible to publish and update an abstract model of the domain service capability at the service management module. The service interface allows upper layer user applications to specify their requests for end-to-end network services. Upon receiving a service request from an end user, the service discovery module searches the service registry maintained by the service management module to discover a network service for meeting the request. If no single network service provided by any individual domain can meet the requirement, the orchestration module will search for a service chain of multiple network services and orchestrate them for end-to-end service delivery. Then the service management module will send requests to the SDN controllers of all domains involved in service delivery for this user to allocate sufficient bandwidth for meeting user QoS requirement. In addition to these key components, the SDP may also perform some global network management functions, for example user authentication, service request authorization, end-to-end path computation, and traffic engineering.

The proposed SDP framework combines advantages of NaaS and SDN for improving end-to-end service provisioning in the future Internet. The separated data/control planes and logically centralized controlling enabled by SDN allows a global control mechanism over heterogeneous network infrastructure. NaaS provides a high-level abstraction of autonomous networking systems and enables loose-coupling collaboration among them. The proposed NaaS-based SDP offers a uniform platform upon which third party service providers can develop and deploy new end-to-end network services to meet various application requirements without knowing detailed implementations of underlying network infrastructure. Such a service delivery platform enables a new business model in which a service provider can lease networking resources from various domains and orchestrate the resources for end-to-end network service provisioning. Such a business model is similar to the model for Cloud service provisioning,

which allows service providers to lease computing resources from infrastructure providers for offering Cloud services to end users.

The proposed SDP also offers a promising approach toward federated management of networking and computing resources (such as CPU capacity and storage space in Clouds) to enable converged network and Cloud service provisioning in a Software Defined Environment. In such an environment, both networking and computing resources can be abstracted as services by following a uniform SOA-based mechanism, and then can be orchestrated to form composite network-Cloud services to end users. The NaaS-based

SDP also supports incremental SDN deployment in the Internet. Network domains implemented with non-SDN technologies can also be exposed as network services to the SDP, as long as they realize a NaaS interface for network abstraction, and then can be involved in end-to-end service delivery with SDN domains through service orchestration provided by the SDP.

Another important advantage of the proposed SDP is to realize the benefit of logically centralized control promised by the SDN paradigm in large-scale multi-domain networking environments. Such a centralized control with a global network view is particularly important for achieving end-to-end service delivery with QoS guarantee in the Internet consisting of various autonomous systems. Due to the heterogeneity of network protocols and technologies in these systems, exposure of networking capabilities to a central control unit without appropriate abstraction would lead to unmanageable complexity. The high-level abstraction enabled by the SDP addresses the diversity challenge; thus making centralized control for end-to-end QoS provisioning possible.

The presented framework gives functional architecture for a service delivery platform for inter-domain SDN, which may be realized with various implementations. Enabling technologies are required for implementing key functions in two categories: i) internal modules of the SDP, mainly including the service management, discovery, and orchestration modules; and ii) interfaces for the SDP to interact with user applications and network domains, including the service interface and the network abstraction interface. Recent research on NaaS has yielded various technologies for network service description, discovery, and composition. A summary of these technologies can be found in the survey paper [29]. These technologies form the foundation for implementing the key modules in the SDP. Standard interfaces for network and service abstractions form the other key aspect for realizing the SDP. From an end user's perspective, the SDP plays the role of a service broker in the SOA architecture; therefore, standard Web Service interfaces between service consumers and a service broker can be applied to realize the service interface between the SDP and the upper layer user applications. The network abstraction interface between the SDP and various network domains is essentially an SDN northbound interface. RESTful Web Service has been widely adopted for implementing a northbound interface in SDN. Application Layer Traffic Optimization (ALTO) [30] and Interface to Routing System (I2RS) [31], are two RESTful compatible protocols based on which a network abstraction interface may be realized.

A key for the NaaS-based SDP to achieve end-to-end service delivery with QoS guarantee in a multi-domain SDN environment is to discover, select, and orchestrate the appropriate network services with sufficient service capabilities that meet the performance requirements specified by end users. In order to achieve this objective, each network domain should provide the SDP with a high-level abstraction of its service capability information. In addition, the SDP should be able to determine the minimum service

capacity required for achieving end-to-end QoS guarantee. Therefore, an abstract model of network service capabilities and a method for determining required service capacity are two key enabling technologies for the proposed SDP, which are the focus of study for the rest of this article.

4 High-Level Abstraction Model for Network Service Capability

The SDP needs information about service capabilities of network domains in order to achieve end-to-end QoS provisioning. On the other hand, NaaS-based network abstraction requires hiding detailed information of network infrastructure. To balance the conflicting requirements from these two aspects, in this section we propose a high-level abstraction model of network service capabilities, which allows SDN domain controllers to provide the SDP with necessary information without exposing details of network infrastructure. Such a model should meet the following requirements in order to support end-to-end QoS provisioning in a large-scale multi-domain SDN environment: i) providing a high-level abstraction of topology and states of underlying network infrastructure, ii) presenting information about network capabilities required by the SDP for end-to-end QoS provisioning, iii) being agnostic to network implementations thus applicable to heterogeneous network domains, and iv) being extendable to model capabilities of composite network services for inter-domain service delivery.

4.1 Abstraction Model for Single Network Service Capability

We first consider modeling capabilities of single network services that virtualize the networking functionalities of individual network domains. In general, the service capability information about an individual network domain needed by the SDP for end-to-end QoS provisioning can be described at a high level from the following two aspects: *virtual connections* provided by the network service among the border nodes of the domain, and *capacity* of data transportation on each virtual connection. From a service provisioning perspective, topology of a network domain may be abstracted as a full mesh of virtual connections between any pair of border nodes of the domain. The SDP just needs to know if a network service provides a virtual connection from an ingress node to an egress node, and if so how much data transport capacity is available on the virtual connection. The actual path between the nodes is determined by the SDN controller in that domain, which has knowledge of the physical topology and network states of the entire network domain.

Therefore, for a network service that virtualizes a domain with n border nodes, a high-level abstraction of its service capability can be modeled by a matrix

$$\boldsymbol{C} = \begin{pmatrix} c_{1,1}\ c_{1,2} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{n,1}\ c_{n,2} & \cdots & c_{n,n} \end{pmatrix} \tag{1}$$

and each matrix element $c_{i,j}$ is defined as

$$c_{i,j} = \begin{cases} P_{i,j}, & \text{the network service provides virtual connection from node i to node j} \\ 0, & \text{otherwise} \end{cases}$$

where $P_{i,j}$ is called the *capacity profile* for the virtual connection from node i to node j, whose definition will be given later in this subsection. Each non-zero element $c_{i,j}$ in the matrix $\boldsymbol{C}$ indicates existence of a virtual connection from node i to node j and also describes the transport capacity available on the connection.

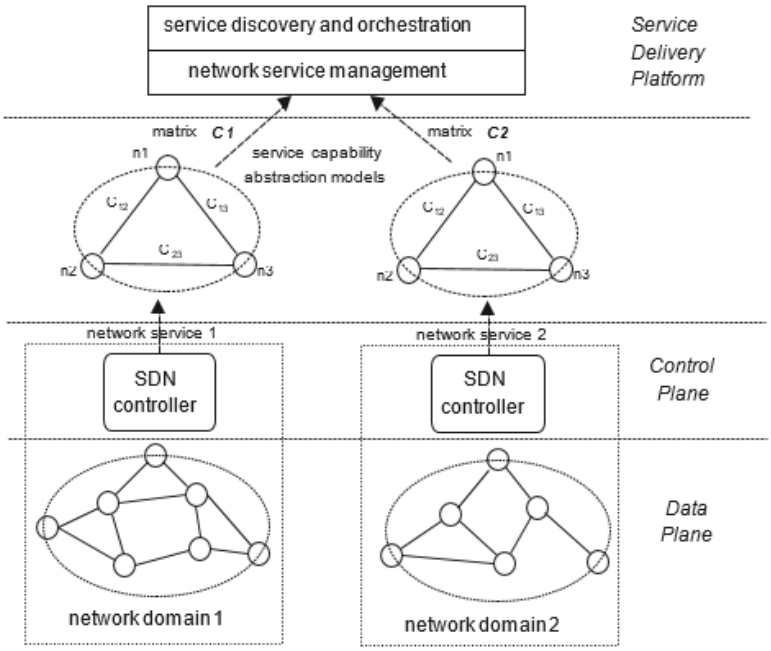

Figure 2. Abstraction of topology connectivity and transport capacity of network services

As illustrated by the example shown in Figure 2, physical topology of reach network domain is abstracted as a full mesh of virtual connections among border nodes of the domain. Service capability information of the domain is described by a matrix ***C*** presenting a set of virtual connections and the associated capacity profiles. Each SDN controller publishes the matrix ***C*** of its domain to the SDP service management module via the NaaS interface. Such a matrix exposes capability information of a network service to its potential users while keeping its implementation transparent to the users. Such a high-level abstraction of network domain internal topology and states is necessary for achieving scalability in disseminating, updating, and inquiring such information in a large scale inter-domain SDN environment; therefore meeting the requirement i) for an abstraction model.

In order to meet the requirement ii) for providing information needed for QoS provisioning, a profile $P_{i,j}$ is used as the value of each non-zero element $c_{i,j}$ of matrix ***C***. This profile describes the capacity that can be guaranteed by the network service for data transportation from node *i* to node *j*. Due to the wide variety of networking technologies employed in heterogeneous network domains, such a capacity profile must be independent to network implementations in order to meet the requirement iii). In addition, the profile should also be in a form that can be easily extended to describe the capacity of end-to-end service delivery across multiple domains; thus meeting the requirement iv). In order to develop a service capacity profile that meets all the above requirements, we employ the *service curve* concept from *network calculus* theory [32]. The service curve in network calculus is defined as follows.

Let $R(t)$ and $R^*(t)$ respectively be the accumulated amount of traffic that arrives at and departs from a system by time t. Given a non-negative, non-decreasing function, $S(\cdot)$, where $S(0) = 0$, we say that the system guarantees a *service curve* $S(\cdot)$ for the flow, if for any $t \geq 0$ in the busy period of the system,

$$R^*(t) \geq R(t) \otimes S(t) \tag{2}$$

where $\otimes$ denotes the convolution operator defined as $h(t) \otimes x(t) = \inf_s:0_{\leq s \leq t} \{h(t - s) + x(s)\}$.

Essentially a service curve of a networking system describes the minimum amount of service capacity guaranteed by the system. In our network abstraction model, we employ the service curve guaranteed by the network service for the virtual connection from node *i* to node *j* as the capacity profile $P_{i,j}$. Since a service curve is a general function for describing network service capacity, it is independent with network implementations thus applicable to model service capabilities of heterogeneous network domains.

In order to limit the overheads between domain controllers and the SDP for publishing and updating matrix $\boldsymbol{C}$, it is desirable to present a capacity profile with a simple data structure. Toward this end, we define a Latency-Rate capacity profile as follows. If a network service guarantees a virtual connection a service curve

$$\beta(r,\vartheta) = \max\{0, r(t-\vartheta)\} \tag{3}$$

then we say that the virtual connection has a Latency-Rate (LR) profile, where ϑ and r are respectively called the *latency* and *rate* parameters of the profile. A LR profile can serve as the capacity model for virtual connections provided by typical network domains. In order to achieve end-to-end QoS guarantee, the SDP requires each network domain involved in service delivery to provide a minimum bandwidth. Such a minimum bandwidth guarantee is described by the rate parameter r in the LR profile. Data transportation in a network domain experiences a fixed delay that is independent with traffic queuing behavior, for example signal propagation delay, link transmission delay, switch process delay, etc. The latency parameter ϑ of the LR profile is to characterize this part of fixed delay.

Please be advised that although the capacity profile is implementation agnostic, the profile for each virtual connection provided by a network domain is constructed by the SDN controller in the domain; therefore profile parameters are related to the implementation and control/management policies of the domain. For example, the physical path for a virtual connection from node *i* to node *j* is established by the SDN controller following the path computing policy of the domain. Then the available bandwidth on this path will be the service rate parameter $r_{i,j}$ in the capacity profile $P_{i,j}$. If there exists multiple physical paths from *i* to *j* and the domain policy allows parallel data delivery; then the controller may aggregate the available bandwidth on all the paths to get the service rate parameter $r_{i,j}$.

For a typical network domain where transport capacity of the virtual connection from any node *i* to any node *j* can be modeled by a LR profile $\beta(r_{i,j},\vartheta_{i,j})$, the matrix element $c_{i,j}$ can be presented by a simple data structure with two parameters $[r_{i,j},\vartheta_{i,j}]$. The abstraction model provides the key information of service capability needed by the SDP for QoS provisioning using a small set of parameters. Therefore, LR capacity profile reduces the communication overheads between domain controllers and the SDP for publishing and updating service capability information; thus improving system scalability.

4.2 Abstraction Model for Composite Network Service Capability

To achieve service delivery across network domains, the SDP orchestrates multiple network services to form a composite network service that provides an end-to-end virtual connection. Therefore the SDP also requires a model for abstracting end-to-end capabilities of composite network services. The proposed capability model for single network services can be extended for supporting network service composition.

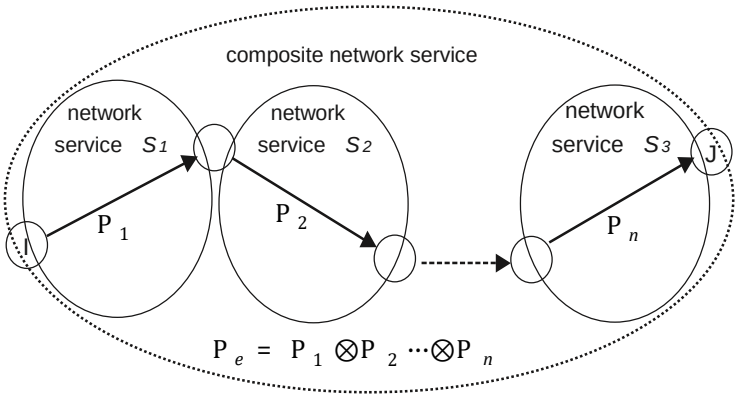

Figure 3. Capacity profile for a virtual connection provided by a composite network service

Known from network calculus, the service curve guaranteed by a series of tandem servers can be obtained through the convolution of all the service curves guaranteed by individual servers. Since the capacity profile of a virtual connection is essentially a service curve of the connection, the capacity profile of an end-to-end virtual connection traversing multiple domains can be determined by following the convolution theorem in network calculus. Suppose the source node *i* and the destination node *j* are in different domains, and the orchestration module selects *n* domains, which are abstracted by network services S_k, $k = 1,2,\cdots,n$ respectively, to form a composite service for providing a virtual connection from *i* to *j*, as shown in Figure 3. The connection from *i* to *j* consists of *n* virtual links, each is provided by a single network service. Suppose the capacity profile for the virtual link provided by service S_k is P_k, and the capability profile for the end-to-end virtual connection is denoted as P_e, then P_e can be determined as

$$P_e = P_1 \otimes P_2 \cdots \otimes P_n. \tag{4}$$

If each network service S_k guarantees a LR profile $\beta(r_k,\vartheta_k)$ for its virtual link, then it can be proved by following convolution theorem in network calculus that capacity profile for the end-to-end virtual connection is

$$P_e = \beta(r_e,\vartheta_e) = \beta(r_1,\vartheta_1) \otimes \cdots,\otimes\beta(r_n,\vartheta_n) \tag{5}$$

where $r_e = \min\{r_1,r_2,\cdots,r_n\}$ and $\vartheta_e = \vartheta_1 + \vartheta_2 + \cdots + \vartheta_n$.

Equation (5) implies that if the service capacity of each link of an end-to-end virtual connection can be described by a LR profile, then capacity of the virtual connection provided by a composite network service can also be modeled by a LR profile. The latency parameter of the end-to-end LR profile is equal to the summation of latency parameters of all links and the end-to-end service rate is limited by the bottleneck link with the least service rate value.

The proposed Matrix ***C*** and capacity profile provide a general abstraction model that can be used by SDN controllers in all network domains to publish service capability information of their network infrastructure at the SDP service management module. Publication and updating of the model could be implemented based on some available protocols, for example Application Layer Traffic Optimization (ALTO) [30]. Each SDN domain controller can implement an ALTO server that regularly disseminates a network-map and a cost-map to the SDP service management module, which can act as an ALTO client.

Matrix ***C*** and the associated capacity profiles can be presented as a network-map together with a cost map. The service management module then combines the network-maps and cost-maps of all domains

into a global network view that can be used by the service orchestration module for end-to-end service provisioning. ALTO supports RESTful Web service interface between servers and clients; thus supporting NaaS-based network abstraction interface between the SDP and SDN domain controllers.

5 Resource Allocation for End-to-end QoS Provisioning in SDN

End-to-end QoS provisioning in a multi-domain SDN environment requires allocation of sufficient networking resources in all the domains that are involved in service delivery for meeting user requirements. In the proposed SDP framework each domain is abstracted as a network service through a NaaS interface. Therefore, selecting and orchestrating the appropriate network services with sufficient data transport capacity for end-to-end service delivery is a key to QoS provisioning. The abstraction model developed in last section allows each domain to provide information about its service capability to the SDP, which forms the basis for network service selection and orchestration. For supporting QoS for an end user, the SDP also needs to determine the amount of service capacity required for meeting user performance requirement and assures that such capacity be allocated in each involved network domain. On the other hand, the SDP wants to minimize bandwidth consumption for each user in order to improve resource utilization in network infrastructure. Therefore, a method for determining the minimum amount of service capacity for meeting the end-to-end performance requirement specified by a user is an important technology needed by the SDP for QoS provisioning, which will be developed in this section.

5.1 Service Demand Profile

End users need to provide SDP with information about their networking demand in order for the SDP to select appropriate network services and determine required service capacity for meeting user requirements. In order to allow the wide variety of user applications to specify their diverse networking demands, we define a general demand profile $D\{C, Q, \mathrm{L}\}$. In this profile, element C gives the connectivity requirement, which can be specified by the addresses of source and destination for data transportation required by the application. The element Q in the profile is to specify QoS expectation for the service, which comprises a set of performance parameters such as the maximum delay and/or minimum throughput for data transportation. Since traffic flows with different load characteristics will require different amounts of service capacity for achieving a certain level of performance, we include a load descriptor L in the demand profile to characterize the traffic that a user application will load the network service. Considering the various user applications with diverse load characteristics, such a load descriptor must be general to support different types of traffic flows; while on the other hand be concise enough to be processed easily. The *arrival curve* concept in network calculus is employed here to develop a general load descriptor that meets such requirements.

Let $R(t)$ denote the accumulated amount of traffic arrives from a flow by time t. Given a non-decreasing, non-negative function, $\mathrm{L}(\cdot)$, the flow is said to have an arrival curve $\mathrm{L}(\cdot)$ if

$$R(t) - R(s) \leq \mathrm{L}(t - s) \qquad \forall\, 0 < s \leq t. \tag{6}$$

The arrival curve gives an upper bound for the amount of traffic that a user application can load a service delivery system; therefore is employed here as the load descriptor in a service demand profile. Since such a descriptor is defined as a general function of time, it can be used to describe the traffic load generated by any user application.

Currently most QoS-capable networks apply traffic regulation mechanisms at network boundaries to shape arrival traffic from end users. The traffic regulators most commonly used in practice are leaky buckets. A traffic flow constrained by a leaky bucket has a load profile

$$L(p,\rho,\sigma) = \min\{pt, \sigma + \rho t\} \quad (7)$$

where p, ρ, and σ are called respectively the peak rate, the sustained rate, and the burst size of the flow. Flow-based data forwarding in SDN data plane allows per-flow traffic regulation to be implemented easily at entry switches. Each user can specify its traffic load using a descriptor with p, ρ, and σ parameters, which may be enforced by a leaky bucket shaper at the SDN switch where user traffic enters the network. Therefore, it is reasonable to assume the availability of a leaky bucket load descriptor for the traffic flow of each user.

5.2 Minimum Capacity in Network Service for QoS Guarantee

Upon receiving the demand profile that a user submits with its service request, the SDP needs to determine the minimum service capacity required on a virtual connection for meeting the QoS expectation, which is the basis for network service selection and orchestration. In this subsection, we develop a technique for SDP to determine the minimum service capacity for meeting a given QoS requirement. We focus our analysis on the maximum delay and minimum throughput as performance parameters since they are required by most user applications with QoS expectation.

We first consider the case that a user application only requires the minimum throughput T_{req} as its QoS expectation; i.e., $Q = \{T_{req}\}$. Since throughput is the only QoS requirement, the minimum capacity C_{min} required on a virtual connection for supporting this user just needs to be T_{req}; that is, $C_{min} = T_{req}$. The capacity profile P of a virtual connection essentially gives a lower bound of the capacity that a network service guarantees to the connection. Following network calculus the minimum capacity available on the virtual connection can be determined as

$$b_{min} = \lim_{t\to\infty}[P/t] \quad (8)$$

Therefore, this virtual connection meets the user's requirement if $b_{min} \geq T_{req}$.

Suppose data transport capacity of a virtual connection can be modeled by a LR profile, i.e., $P = \beta(r,\vartheta)$, then

$$b_{min} = \lim_{t\to\infty}[r(t-\theta]/t = r. \quad (9)$$

If the virtual connection traverses n domains, then the end-to-end connection consists of n virtual links each modeled by a profile $\beta(r_i,\vartheta_i)$, $i = 1,2,\cdots,n$. According to (5) and (9), the end-to-end transport capacity $b_{min} = r_e = \min\{r1,r2,\cdots,r_n\}$. Therefore, the virtual connection meets the user's throughput requirement if $r_e \geq T_{req}$. Since throughput is the only QoS requirement, the minimum capacity C_{min} required on a virtual connection just needs to be T_{req}; that is, $C_{min} = T_{req}$.

Then we analyze the case that a user application only requires the maximum service delay D_{req} as its QoS expectation; i.e., $Q = \{D_{req}\}$. Consider a virtual connection selected for serving a user application, suppose the capacity profile of the connection is P and the traffic flow of this user has a load descriptor L, then network calculus shows that the maximum service delay guaranteed by the connection to this traffic flow is

$$d_{max} = sup_{t:t>0}\{inf\{\delta: \delta \geq 0\ L(t) \leq P(t+\delta)\}\}. \quad (10)$$

In order to determine the minimum service capacity C_{min} for meeting the requirement $d_{max} \leq D_{req}$, we apply the effective bandwidth concept in network calculus here. Considering a traffic flow with a cumulative arrival process $R(t)$ constrained by an arrival curve $L(t)$; for a fixed, but arbitrary delay requirement D_{req}, the effective bandwidth $R_e(D_{req})$ of the flow is defined as the minimum service rate required to serve the flow with $d_{max} \leq D_{req}$. Therefore, the effective bandwidth for the flow can be obtained as

$$R_e(D_{req}) = sup_{0\leq t_0\leq t}\left\{\frac{R(t)-R(t_0)}{t-t_0+D_{req}}\right\} = sup_{s\geq 0}\left\{\frac{L(s)}{s+D_{req}}\right\} \quad (11)$$

If the maximum delay is the only QoS expectation of a user, then the minimum service capacity required by the user is the effective bandwidth of its traffic flow; that is, $C_{min} = R_e(D_{req})$.

Suppose a traffic flow having a leaky-bucket load descriptor $L(p,\rho,\sigma)$ is served by a virtual connection with a LR capacity profile $\beta = (r,\vartheta)$, then following (10) and (11) the effective bandwidth for meeting a delay requirement D_{req} can be determined as

$$R_e(D_{req}) = \begin{cases} \rho & D_{req} \geq D_{max} \\ \frac{p\sigma}{[(D_{req}-\theta)(p-\rho)+\sigma]} & D_{min} \leq D_{req} \leq D_{max} \\ not\ available & D_{req} < D_{min} \end{cases} \quad (12)$$

where $D_{min} = \vartheta$ and $D_{max} = \vartheta + \sigma/\rho$.

Equation (12) shows that if the expected delay upper-bound is greater than a threshold D_{max}, then the required service capacity is equal to the sustained rate ρ of the arrival traffic. Actually, service capacity on a virtual connection should be no less than the sustained rate of a flow in order for the connection to guarantee any upper bounded service delay for the flow. On the other hand, no delay expectation that is tighter than D_{min} can be met by a virtual connection with finite service capacity. This is because the underlying network infrastructure always introduces a certain amount of latency for data delivery due to its physical properties. For any expected delay upper bound between D_{min} and D_{max}, the required minimum service capacity is a function of traffic parameters (p,ρ,σ), the latency parameter ϑ, and the delay requirement D_{req}.

For an end user that has both minimum throughput T_{req} and maximum delay D_{req} as QoS expectation, the minimum service capacity that must be available on a virtual connection for providing QoS guarantee to the user will be $C_{min} = \max\{T_{req}, R_e(D_{req})\}$.

5.3 Network Service Selection and Orchestration for End-to-End QoS Provisioning

The technique for determining the minimum required service capacity can be employed by the SDP to perform network service selection and orchestration for achieving end-to-end QoS guarantee. A general procedure of network service selection/orchestration is illustrated in Figure 4.

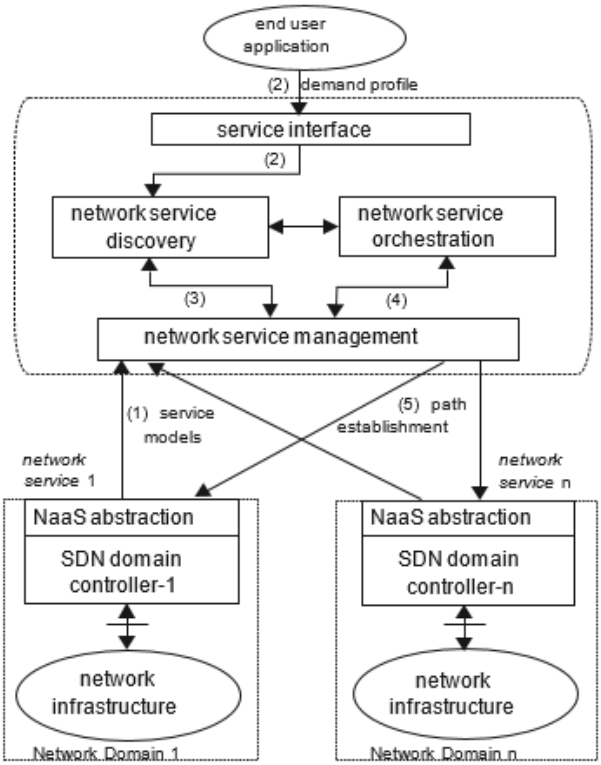

Figure 4. Network service selection and orchestration for end-to-end QoS provisioning

The SDN controller in each network domain is responsible for publishing the service capability model (the matrix ^{C}) of its domain at the SDP service management module. (step 1 in Figure 4). When an end user requests a network service from the SDP it submits a demand profile $D\{C, Q,$ L$\}$ through the service interface (step 2). The demand profile incudes a connectivity element C specifying the source and destination addresses (s,d), the Q element giving QoS requirements T_{req} and/or D_{req}, and a load descriptor L for the user's traffic flow. On receiving the request with the demand profile, the SDP service discovery module inquires the service management module for the capability models of all available network services. Connectivity and capacity information of single network services is first examined by the service discovery module to find a network service that can provide a virtual connection from s to d with sufficient capacity C_{min} for meeting QoS expectation (step 3). If a network service is found, the service management module will inform the SDN controller in the corresponding network domain for establishing a physical path and allocating bandwidth in network infrastructure (step 5). If no single network service can meet the requirements specified by the demand profile, the service orchestration module will search a chain of network services that can provide a virtual connection from s to d across multiple domains (step 4). The orchestration module determines the minimum capacity C_{min} required on the end-to-end virtual connection and only orchestrates network services with sufficient capacity for end-to-end service delivery. If such a service chain is found, the service management module will contact the SDN controller of each network domain involved in this service chain for establishing the physical path and allocating bandwidth in that domain (step 5).

The proposed SDP plays the role of a service broker for accepting end users' service requests, selecting appropriate network services for meeting users' requirements, and orchestrating multiple network services for inter-domain service delivery. Therefore, the SDP offers a platform that allows third party

providers to offer end-to-end network services by utilizing the services provided by infrastructure providers (who operate individual network domains). The SDP and domain controllers (representing infrastructure providers) establish a certain form of service contracts, which specify the service capabilities that the SDP expects from underlying network domains for supporting each virtual connection provided by the domains. The minimum service capacity that a domain must provide for a virtual connection, which can be determined by the technique developed in last subsection, is an important item included in the service contract for achieving QoS guarantee.

A centralized platform for service provisioning in a large scale networking environment may raise concerns about scalability issues. Interactions between the SDP and domain controllers cause overheads and delay that may limit scalability of the SDP for service management. NaaS-based network abstraction significantly simplifies information exchange between the SDN and domain controllers. The high-level abstraction model developed in this paper allows domain controllers to publish their service capability information with a relatively simple data structure; thus reducing overheads for network information exposure. Network service orchestration performed at the SDP searches available end-to-end paths based on a highly aggregated global virtual network topology; therefore does not need information dissemination among domain controllers as required by traditional distributed mechanisms for inter-domain routing. In order to establish paths for service delivery the SDP just needs to inform each domain controller about the required connectivity and capacity information. Such information can be described in a small set of parameters (the pair of source-destination border nodes for a virtual link and the minimum available bandwidth required on the virtual link); therefore limiting the control overheads between the SDP and domain controllers.

Please be noted that the proposed SDP framework is a logically centralized platform that may be realized by various implementations, which may have a distributed physical structure. The scalability issues associated with SDP-based service management shares a lot of similarity with the scalability of a centralized SDN controller controlling multiple switches in a large-scale network; therefore could be addressed by applying similar technical ideas. Although a thorough analysis on scalability of a NaaS-based SDP is an interesting and important problem, it is out of the scope of this paper and will be studied in our future work.

5.4 Bandwidth Utilization for End-to-End QoS Provisioning

The proposed SDP enables logically centralized service and resource management with a global network view for end-to-end QoS provisioning in a multi-domain SDN environment. Without such an SDP, the SDN controller in each domain provides a central control point but only within the scope of a single domain. No controller can obtain a purview of the entire path for service delivery across multiple domains; therefore, end-to-end QoS provisioning needs to be offered based on mutual collaboration between controllers in neighbor domains. With such a per-domain QoS mechanism, the end-to-end delay requirement is partitioned to a set of delay budgets, one for each domain involved in service delivery. Each domain has to determine and allocate sufficient amount of bandwidth in its own network infrastructure to guarantee its delay budget. With the proposed SDP, an end-to-end virtual path with QoS guarantee can be established through network service orchestration based on a global network view. The SDP determines the required service capacity on the path by viewing the entire path as if it belongs to one end-to-end virtual domain abstracted by a composite network service. The SDN controllers in all the

domains passed by the virtual path are required to allocate the same amount of effective bandwidth on the path, which is determined by the SDP. In this subsection, we study bandwidth utilization of the end-to-end QoS scheme enabled by the SDP and compare it with that of the per-domain QoS scheme without an SDP.

We consider a service delivery scenario in which a virtual path traverses n domains abstracted respectively by network services S_i, $i = 1,2,\cdots,n$, as shown in Figure 3. Denotes the virtual link provided by service S_i as l_i, and assume that the capacity profile of l_i is a LR profile $P_i = \beta(r_i,\vartheta_i)$, then the capacity profile of the end-to-end virtual path is $P_e = \beta(r_e,\vartheta_e)$, where $r_e = \min\{r1,\cdots,r_n\}$ and $\vartheta_e = \vartheta_1 + \vartheta_2 + \cdots + \vartheta_n$. Suppose the load descriptor for the traffic flow is $L(p,\rho,\sigma)$ and the expected end-to-end delay upper bound is D^e_{req}, then with the end-to-end QoS mechanism enabled by the SDP, the effective bandwidth that must be allocated to the virtual path for guaranteeing D^e_{req} can be determined by equation (12) as

$$R_e\left(D^e_{req}\right) = \frac{p\sigma}{\left(D^e_{req}-\theta\right)(p-\rho)+\sigma} \quad \left(D^e_{min} \le D^e_{req} \le D^e_{max}\right) \tag{13}$$

Where $D^e_{min} = \vartheta_e$ and $D^e_{max} = \vartheta_e + \sigma/\rho$. This is the amount of bandwidth that the SDP requests each domain controller to allocate for the virtual link provided by the domain.

Now we consider the case in which the per-domain QoS mechanism without a central SDP allocates effective bandwidth on a path passing the same set of n domains for meeting the same end-to-end delay requirement. Suppose the total delay requirement is partitioned to n delay budget D^i_{req}, $i = 1, 2, \ldots n$, one for each domain, then in the i-th domain the effective bandwidth that must be allocated on its virtual link l_i for meeting its delay budget will be

$$R_i\left(D^i_{req}\right) = \frac{p\sigma}{\left(D^i_{req}-\theta\right)(p-\rho)+\sigma} \quad \left(D^i_{min} \le D^i_{req} \le D^i_{max}\right) \tag{14}$$

where $D^i_{min} = \vartheta_i$ and $D^i_{max} = \vartheta_i + \sigma/\rho$.

Both R_e and R_i are the required amounts of bandwidth that need to be allocated in the domain S_i for meeting the same delay requirement, but the former is obtained by the SDP with a global network view while the latter is obtained by the local SDN controller in a single domain. To compare bandwidth utilization achieved by these two service management schemes, we defined U as the ratio between these two effective bandwidth values; that is,

$$U = \frac{R_i\left(D^i_{req}\right)}{R_e\left(D^e_{req}\right)} = \frac{\left(D^e_{req}-\theta_e\right))(p-\rho)+\sigma}{\left(D^i_{req}-\theta_i\right)(p-\rho)+\sigma}. \tag{15}$$

An observation we can have from (15) is that U = 1 when $p = \rho$. This implies that for a constant rate traffic flow ($p = \rho$) the end-to-end allocation scheme enabled by the SDP has the same level of bandwidth utilization as per-domain allocation does. This is because the effective bandwidth for a constant rate flow just needs to be the peak/sustained rate of the flow, and extra bandwidth allocation does not help improving delay performance; that is $R_e(D^e_{req}) = R_i(D^i_{req}) = p = \rho$.

We focus our analysis on the case of variable rate traffic flows; i.e. $p > \rho$. We define $\Delta d_e = D^e_{req} - \vartheta_e$ as an indicator to reflect how tight the end-to-end delay expectation is compared to the latency parameter of the service delivery path, which is the minimum delay that can be achieved on the path. A larger Δd_e value implies a relatively looser delay expectation. Similarly $\Delta d_i = D^i_{req} - \vartheta_i$ reflects the relative tightness of the delay budget for a single network domain S_i. Equation (15) shows that if $\Delta d_i \ge \Delta d_e$ then $U \le 1$; otherwise $U > 1$. This implies that if the delay budget for a single network domain, compared to the latency of the

virtual link in this domain, is relatively looser than the delay expectation for end-to-end service delivery, then the per-domain allocation scheme may actually require less amount of bandwidth than what is required by the SDP. However, given an end-to-end delay bound requirement, a loose delay budget for one network domain means tighter delay budgets thus more bandwidth consumption in other domains. Autonomous domains in the Internet are unlikely to sacrifice their own bandwidth resources for other domains' benefits.

Therefore we analyze a case that the end-to-end delay requirement is equally partitioned among all domains and assume the virtual links in all domains have an identical latency property; that is, $D^i_{req} = d$ and $\vartheta_i = \vartheta$ for $i = 1,2,\cdots,n$. Then $D^e_{req} = nd$ and from (5) we have $\vartheta_e = n\vartheta$.

Therefore,

$$R_e\left(D^e_{req}\right) = \frac{p\sigma}{n(d-\theta)(p-\rho)+\sigma}, \quad R_i\left(D^i_{req}\right) = \frac{p\sigma}{(d-\theta)(p-\rho)+\sigma}. \tag{16}$$

Then bandwidth ratio is

$$U = \frac{R_i(D^i_{req})}{R_e\left(D^e_{req}\right)} = \frac{n(d-\theta))(p-\rho)+\sigma}{(d-\theta)(p-\rho)+\sigma} = 1 + \frac{(n-1)(d-\theta)(p-\rho)}{(d-\theta)(p-\rho)+\sigma}. \tag{17}$$

Since we are considering variable rate flows ($p > \rho$) and the delay budget assigned to a network domain must be larger than the latency property of its network infrastructure ($d > \vartheta$), (17) shows that the bandwidth ratio U > 1 for end-to-end service delivery across domains ($n \geq 2$).

Equation (12) also gives a special case for loose delay expectation; that is, $R_e(D^e_{req}) = \rho$ when $D^e_{req} > D^e_{max} = \vartheta_e + \sigma/\rho$. Similarly, for per-domain allocation $R_i(D^i_{req}) = \rho$ when $D^i_{req} > D^i_{max} = \vartheta_i + \sigma/\rho$. Considering the above case in which $D^i_{req} = d = D^e_{req}/n$ and $\vartheta_i = \vartheta$ for $i = 1,2,\cdots,n$, then

$$\frac{D^e_{max}}{n} = \theta + \frac{\sigma}{n\rho} < D^i_{max} = \theta + \frac{\sigma}{\rho} \quad (n \geq 2) \tag{18}$$

Inequality (18) implies that for an end-to-end delay expectation that is looser than the maximum threshold; i.e., $D^e_{req} > D^e_{max}$, the effective bandwidth determined by the SDP will be $R_e(D^e_{req}) = \rho$. However, when dividing this delay expectation equally to obtain n delay budgets, one for each single domain, the obtained D^i_{req} might be tighter than the maximum delay threshold of its domain D^i_{max}; therefore the local controller may allocate more bandwidth; i.e. $R_i(D^i_{req}) > \rho$ in each domain.

The above analysis shows that in order to achieve the same level of delay performance guarantee in the considered scenarios, the per-domain QoS mechanism consumes more bandwidth in each individual network domain than the effective bandwidth determined by the SDP. This result indicates that the proposed SDP may not only simplify service management but also enhance bandwidth utilization for end-to-end QoS provisioning in a multi-domain SDN environment.

6 Numerical Results

Numerical results are given in this section to illustrate application of the developed techniques and obtained insights. We considered a networking scenario in which the SDP orchestrates the network services of three domains to provide an end-to-end virtual path for QoS provisioning. The path has been used to transport data for two applications, whose traffic flows, denoted as f_1 and f_2 respectively, are characterized by the following parameters: peak rate $p1$ = 60 Mbps, sustained rate $\rho 1$ = 1.5 Mbps, and

burst size $\sigma 1$ = 1.04 Mbits for f_1; and peak rate $\rho 2$ = 15 Mbps, sustained rate $\rho 2$ = 1.5 Mbps, and burst size $\sigma 2$ = 9.54 Mbits for f_2. These parameters are derived from traffic analysis reported in [33] and [34]. We assume that virtual links provided by all domains for the end-to-end path have a LR capacity profile.

Bandwidth allocation for achieving end-to-end delay performance guarantee is first analyzed. The amounts of effective bandwidth that the SDP must request from each domain to guarantee a set of delay requirements are determined and plotted in Figure 5, where effective bandwidth for f_1 and f_2 are denoted as R_e^1 and R_e^2 respectively. From this figure, we can see that the required amounts of bandwidth for both flows increase when the delay requirement value decreases. This means that more service capacity must be acquired by the SDP from the underlying network domains in order to provide a tighter end-to-end service delay guarantee.

Comparing the bandwidth curves in Figure 5 shows that R^2_e is greater than R^1_e for all delay requirements; that is, different amounts of bandwidth are required by these two flows for achieving the same delay performance on the same virtual path. This means that effective bandwidth is impacted by traffic load parameters as well as the delay requirement; thus verifying the necessity of including a load descriptor in a service demand profile in order for the SDP to achieve QoS guarantee. From Figure 5 we can also see that the R^2_e curve drops with increasing delay requirement value faster than the R^1_e curve does. This implies that for flows with different traffic load parameters, the same extent of improvement in delay performance requires different amounts of increment in effective bandwidth. Both the flows examined in our experiments have the same sustained rate ($\rho 1 = \rho 2$) but flow f_2 has greater burst size ($\sigma 2 > \sigma 1$). Such an observation we obtained from Figure 5 indicates that the parameter σ, which gives the maximum amount of traffic that an application can load on a virtual path continuously with its peak rate, has a strong impact on the required amount of bandwidth for achieving delay guarantee.

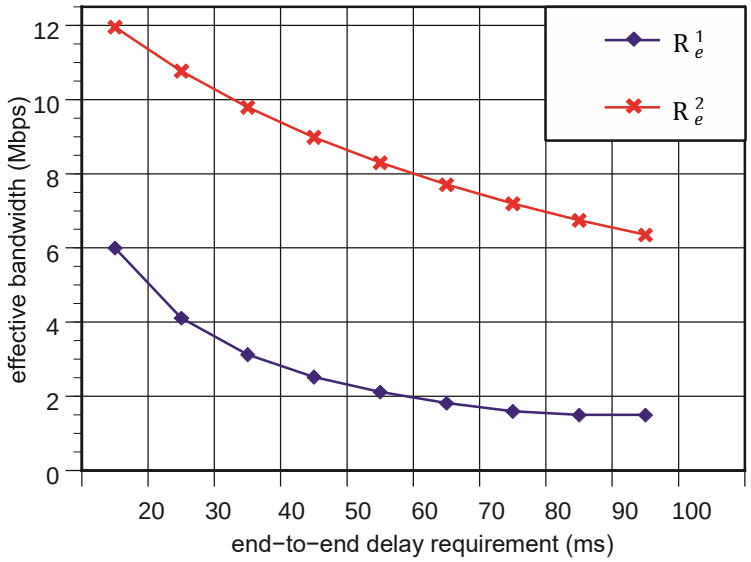

Figure 5. Effective bandwidth for end-to-end delay guarantee for flows f_1 and f_2

One can also notice from Figure 5 that the R^1_e curve becomes flat when the required delay bound is greater than a threshold (90 ms in this particular example) while the R^2_e curve keeps dropping with increasing delay bound value. This is because effective bandwidth is equal to the sustained rate of a flow when the required delay bound is looser than a threshold, as shown in equation (12) $R_e(D_{req}) = \rho$ when $D_{req} \geq D_{max}$. Also, the D_{max} threshold for a flow varies with the load parameters of the flow. In our experiment flow f_1

reaches such a threshold at around 90 ms where the R^1_e curve becomes flat; while f_2 does not reaches its threshold for all the delay bound values tested in the experiment; therefore the R^2_e curve keeps dropping with increasing delay requirement.

In order to evaluate bandwidth utilization achieved by the SDP with a global network view for QoS provisioning in a multi-domain SDN environment, we compare the end-to-end bandwidth allocation scheme performed by the SDP against the per-domain-based bandwidth allocation scheme discussed in Subsection V-D. We assume that the virtual links in the three network domains have identical LR capacity profiles and the end-to-end delay requirement is divided equally as the delay budgets for the three domains. We analyzed the amounts of effective bandwidth that will be determined by each individual SDN controller for meeting the delay budget in its domain. The obtained data for flows f_1 and f_2 are plotted in Figures 6 and 7, in which the per-domain allocation results for f_1 and f_2 are respectively denoted as R^1_d and R^2_d.

Figures 6 and 7 show that for a given flow, the amounts of effective bandwidth determined by the SDP with an end-to-end allocation scheme and by individual SDN controllers with per-domain allocation are both decreasing functions of the required delay bound. That is, more bandwidth is required by both schemes to achieve a tighter delay guarantee. Another important observation one can obtain from Figures 6 and 7 is that for both flows the SDP end-to-end allocation scheme always requires less amount of bandwidth than what per-domain allocation does in order to provide the same level of delay performance guarantee. The data shown in Figures 6 and 7 verify that the end-to-end bandwidth allocation enabled by the SDP with a global network view can achieve higher bandwidth utilization compared to the per-domain bandwidth allocation scheme of the conventional inter-domain QoS mechanism. This indicates that the SDP may realize the potential advantage of SDN logically centralized control vision to improve resource utilization for QoS provisioning in a multi-domain networking environment.

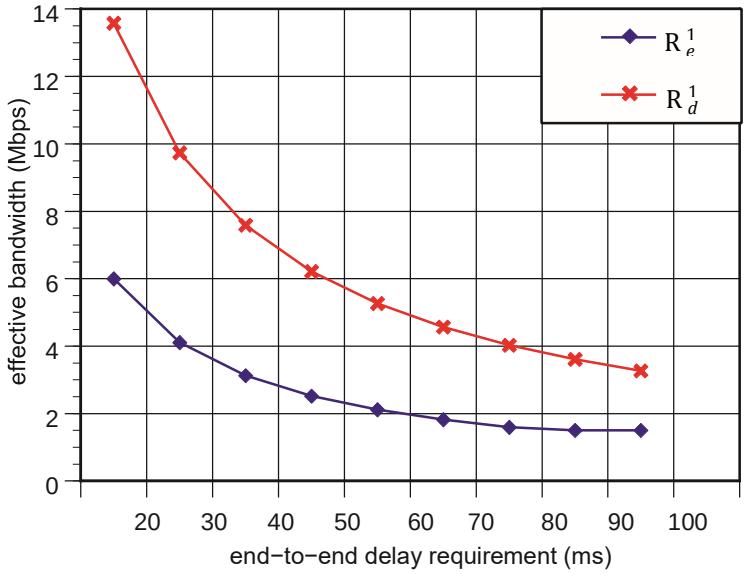

Figure 6. Comparison between effective bandwidth obtained by end-to-end and per-domain allocation schemes for flow f_1.

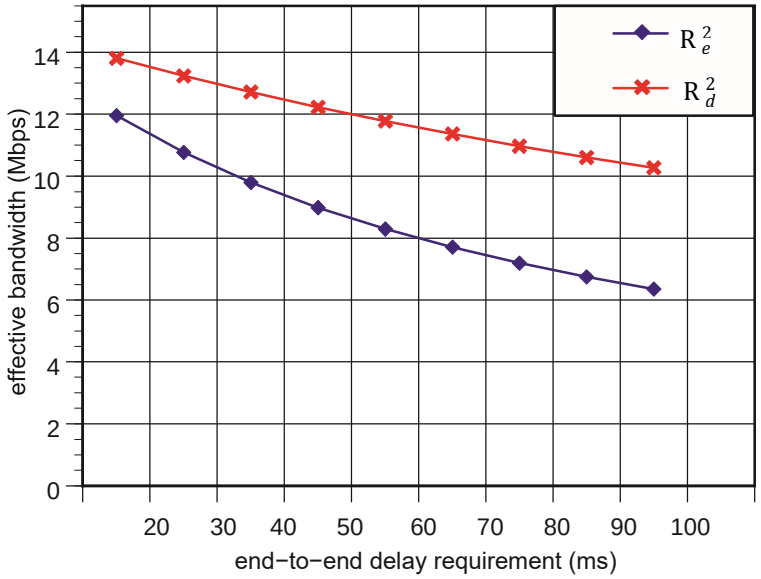

Figure 7. Comparison between effective bandwidth obtained by end-to-end and per-domain allocation schemes for flow f_2.

In order to examine the extent of improvement in bandwidth utilization introduced by the SDP, we also analyzed bandwidth ratios for flows f_1 and f_2, which are defined as $U_1 = R^1_d/R^1_e$ and $U_2 = R^2_d/R^2_e$ respectively. The obtained data are plotted in Figure 8. This figure shows that the bandwidth ratios of both flows are greater than 1; that is, end-to-end bandwidth allocation enabled by SDP achieves higher bandwidth utilization for providing delay performance guarantee. Comparison between the curves of U_1 and U_2 in Figure 8 shows that the two flows have different bandwidth ratio values for achieving the same delay requirement and $U_1 > U_2$ for all the delay bounds tested in our experiments. This implies that load parameters of a traffic flow also have an impact on the extent of improvement in bandwidth utilization introduced by the SDP to the flow.

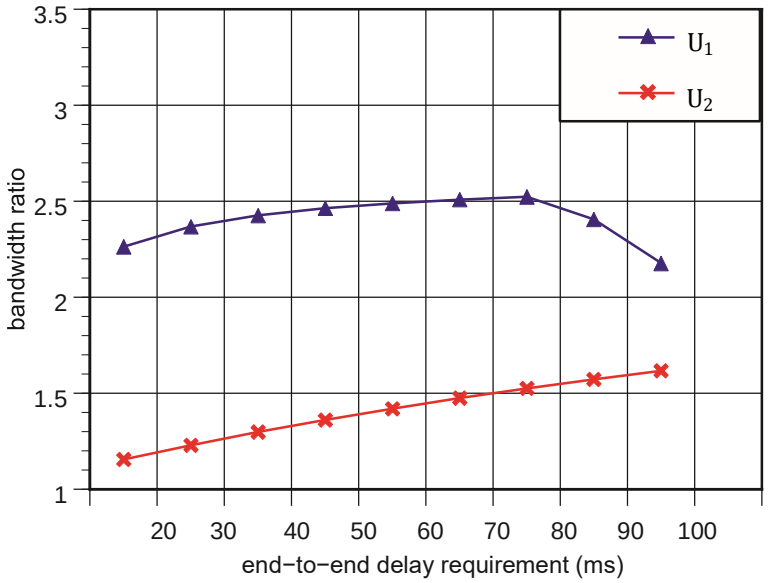

Figure 8. Bandwidth ratios for flows f_1 and f_2 for achieving different delay objectives.

We also noticed from Figure 8 that both U_1 and U_2 increase with the required delay bound in most cases except that U_1 drops when the delay requirement is greater than 80 ms. This implies that end-to-end bandwidth allocation enabled by the SDP typically achieves more improvement in bandwidth utilization for looser delay bounds than for tighter delay bounds. The exception happens flow f_1 when the delay bound is greater than the threshold (80 ms for in this experiment) beyond which end-to-end effective bandwidth is equal to the sustained rate of the flow; that is, the D_{max} beyond which $R_e(D_{req}) = \rho$ as shown in (12). Since the sustained rate is the minimum effective bandwidth that the SDP must request in each domain for achieving any delay bound guarantee, R^1_e stops decreasing for any looser delay bound requirement (as shown by the R^1_e curve in Figure 5); therefore will not further enhance bandwidth utilization. We noticed that even in this case U_1 is still well above 1; that is, end-to-end allocation saves bandwidth than per-domain-based allocation.

In order to evaluate the influence of the number of passed domains on improvement in bandwidth utilization, the bandwidth ratios of the two flows for guaranteeing a delay bound of 60 ms are tested with different numbers of domains passed by the end-to-end virtual path. The obtained results are plotted in Figure 9. This figure shows that both ratios increase with the number of domains, which implies that the more domains the virtual path traverses, the bigger is the difference between the amounts of effective bandwidth determined by the SDP and by individual domain controllers. We can also see from this figure that flows f_1 and f_2 have different bandwidth ratio values with the same number of domains, which reflects the influence of traffic load parameters on the bandwidth utilization improvement that can be achieved by the SDP. Comparing the two ratio curves in this figure shows that their increasing speeds with number of domains are quite different and U_1 increases much larger than U_2. This implies that the number of domains involved in service delivery has a stronger impact on bandwidth utilization to flow f_1 then to flow f_2, which again mainly due to the difference in the traffic load profiles of these two flows.

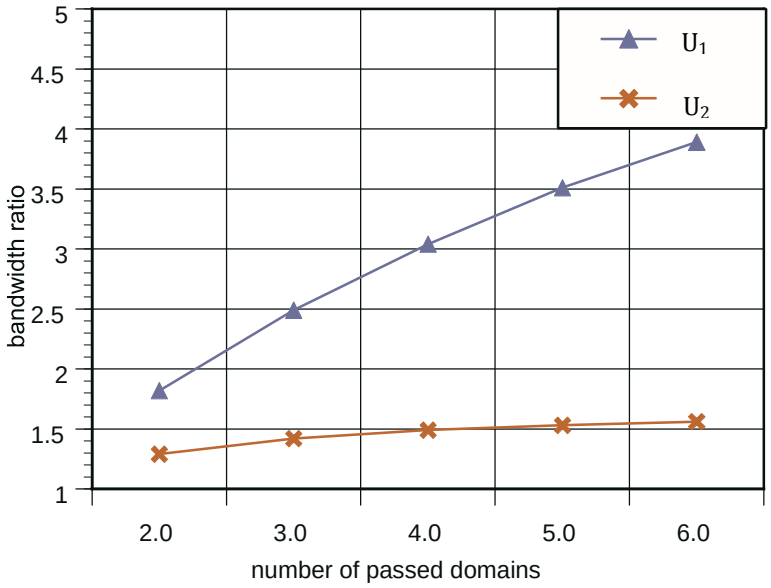

Figure 9. Bandwidth ratios for flows f_1 and f_2 passing different number of network domains.

7 Conclusions

In this paper, we studied the problem of end-to-end service delivery with QoS guarantee in the SDN-based future Internet. The autonomous network domains coexisting in the Internet and the diverse user applications deployed upon the Internet calls for a uniform Service Delivery Platform (SDP) that offers a high-level network abstraction and enables inter-domain collaboration for end-to-end service provisioning. However, the currently available SDN technologies lack effective mechanisms for supporting such a platform. In order to address this important and challenging issue, in this paper we first presented an SDP framework that employs the Network-as-a-Service (NaaS) principle to provide a high-level network abstraction and enables inter-domain collaboration through service orchestration for end-to-end service delivery. Then we focused our study on two key technologies, a network abstraction model and an end-to-end resource allocation scheme, for the SDP to achieve QoS guarantee. We proposed a general abstract model for characterizing the service capabilities offered by heterogeneous network domains and apply the model for abstracting end-to-end inter-domain network services. Then we developed a technique that can be used by the SDP to determine the minimum amount of bandwidth that must be allocated in each network domain involved in service delivery for achieving end-to-end QoS guarantee. We also examined bandwidth utilization of the SDP-based end-to-end resource allocation and compared it with per-domain based resource allocation. Both the analytical and numerical results obtained in this paper indicate that an SDP with the proposed network abstraction and resource allocation technologies not only simplifies service and resource management in SDN but also enhances bandwidth utilization for end-to-end QoS provisioning. Therefore, such an SDP framework offers a promising approach to fully realizing a key benefit promised by the SDN paradigm – logically centralized control for service provisioning to support diverse user applications – in a large-scale multi-domain networking environment.

REFERENCES

[1] ONF,“Open Networking Foundation Software-Defined Networking (SDN) Definition,” *https://www.opennetworking.org/sdn-resources/sdn-definition*, 2013.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A survey on Software-Defined Networking,” *IEEE Communications Surveys and Tutorials*, 2014.

[3] Y. Jarraya, T. Madi, and M. Debbabi, “A survey and a layered taxonomy of Software-Defined Networking,” *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, 2014.

[4] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] T. Erl, *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall, 2005.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[7] A. Doria, J. H. Salim, R. Hass, H. Khosravi, W. Wang, L. D. and R. Gopal, and J. Halpern, “Forwarding and control element separation (ForCSE) protocol,” *Internet Engineering Task Force Sepcification*, Mar 2010.

[8] J. Vasseur and J. L. Roux, "IETF RFC5440: Path Computation Element Communication Protocol (PCEP)," Mar. 2009.

[9] H. Song, "Protocol-Oblivious Forwarding: Unleash the power of SDN through a futur-proof forwarding plane," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, pp. 127–132, Jan. 2013.

[10] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "OpFlex control protocol," *Internet Research Task Force Internet-Draft*, April 2014.

[11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. Mckeown, and S. Shenker, "NOX: Toward an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[12] D. Erickson, "The Beacon OpenFlow controller," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, Jan. 2013.

[13] "Floodlight OpenFlow Controller." http://www.projectfloodlight.org/floodlight/.

[14] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "ONIX: a distirbuted cotnrol platform for large-scale production networks," in *Proc. of the 9th USENIX Conference on Operating Systems Design and Implementation*, Oct. 2010.

[15] U. Krishnaswamy, P. Berde, J. Hart, M. Kobayashi, P. Radoslavov, T. Lindberg, R. Sverdlov, and S. Zhang, "ONOS: An open source distributed SDN OS." available online: http://www.slideshare.net/ON-LAB/onos-open-network-operating-system-an-opensource-distributed-sdn-os.

[16] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow," in *Proc. of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, Apr. 2010.

[17] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A message exchange protocol for Software Defined Networks (SDNS) across multiple domains," *Internet Research Task Force Internet-Draft*, Jun 2012.

[18] R. Bennesby, P. Fonseca, E. Mota, and A. Passito, "An Inter-AS routing component for Software-Defined Networks," in *Proc. of the 2012 IEEE/IFIP Network Operations and Management Symposium (NOMS12)*, Aug. 2012.

[19] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better internet routing based on SDN principle," in *Proc. of the 11th ACM Workshop on Hot Topics in Networks (Hotnets'12)*, Oct. 2012.

[20] P. Thai and J. C. de Oliveira, "Decoupling policy from routing with software defined interdomain management: Interdomain routing for SDN-based networks," in *Proc. of the 2012 IEEE International Conference on Computer Communications and Networks (ICCCN'12)*, July 2012.

[21] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controller," in *arXiv preprint arXiv:1308.6138*, Aug. 2013.

[22] P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf, "NaaS: Network-as-a-Service in the Cloud," in *Proc. of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Apr. 2012.

[23] T. Feng, J. Bi, H. Hu, and H. Cao, "Networking-as-a-Service: a Cloud-based network architecture," *Journal of Networks*, vol. 6, pp. 1084–1090, July 2011.

[24] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: An SDN platform for Cloud network services,"*IEEE Communications Magazine*, vol. 51, pp. 120–127, Feb. 2013.

[25] I. Bueno, J. Aznar, E. E. J. Ferrer, and J. A. Garcia-Espin, "An OpenNaaS based SDN framework for dynamic QoS control," in *Proc. of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013.

[26] Q. Duan, "Network-as-a-Service in Software-Defined networks for end-to-end QoS provisioning," in *Proc. of the 2014 IEEE Wireless and Optical Communications Conference*, May 2014.

[27] J. Zhu, W. Xie, L. Li, M. Luo, and W. Chou, "Software service defined network: Centralized network information service,"in *Proc. of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013.

[28] H. E. Egilmez and a. M. Tekalp, "Distributed QoS architectures for multimedia streaming over software defined networks,"*IEEE Transactions on Multimedia*, vol. 5, no. 16, 2014.

[29] Q. Duan, Y. Yan, and A. V. Valisakos, "A survey on service-oriented network virtualization toward convergence of networking and Cloud computing," *IEEE Transactions on Network and Service Management*, vol. 9, pp. 373–392, Dec.2012.

[30] R. Alimi, R. Penno, and Y. Yang, "Internet-Draft: Application Layer Traffic Optimiation (ALTO) protocol," Mar. 2014. [31] A. Atlas, J. Halpern, S. Hares, and D. Ward, "Internet-Draft: An Archiecture of Interface to the Routing System," June 2013.

[32] J. L. Boudec and P. Thiran, *Network calculus: a theory of deterministic queueing systems for the Internet*. Springer Verlag, June 2001.

[33] J. W. Roberts, "Internet traffic, QoS, and Pricing," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1389–1399, 2004.

[34] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn, "Statistical service assurances for traffic scheduling algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2651–2664, Dec. 2000.

