Computer Communications 45 (2014) 1–10

Contents lists available at ScienceDirect

Computer Communications

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

# Design and analysis of techniques for mapping virtual networks to software-defined network substrates

Mehmet Demirci [*], Mostafa Ammar

*School of Computer Science, Georgia Institute of Technology, Atlanta, Georgia 30332, United States*

ARTICLE INFO

*Article history:*
Received 30 December 2013
Received in revised form 26 February 2014
Accepted 12 March 2014
Available online 20 March 2014

*Keywords:*
Network virtualization
Virtual network embedding
Software-defined networking

ABSTRACT

Software-defined networking has emerged as a powerful approach to improve the customizability and flexibility of networks. In this work, we focus on virtualization in the realm of software-defined networking, and study how to embed virtual networks in this environment. Virtual network embedding is an important problem because intelligent embedding can lead to better performance and a more efficient allocation of network resources compared to random mapping. In software-defined networking, the presence of a central controller is a complicating factor, and customizable routing and differences in resource sharing present new opportunities and challenges. We identify two aspects of virtual network embedding in software-defined networks: virtual node and link mapping, and controller placement. We tackle these problems together, developing techniques to perform embedding with two goals: balancing the load on the substrate network and minimizing controller-to-switch delays. We evaluate our techniques with simulation and Mininet emulation, and show that they are able to optimize for one of the above objectives while keeping the other within reasonable bounds.

## 1. Introduction

Over the past few years, software-defined networking (SDN) has emerged as a promising technology. By separating the control plane from the data plane, SDN facilitates network experimentation and allows for optimizing switching/routing policies. The forwarding planes in software-defined networks are managed by remote processes called *controllers*. OpenFlow [1] was proposed and has been extensively used as a uniform interface between the control and data planes in SDN.

Multiple tenants can share the underlying SDN infrastructure through virtualization. One method of virtualization in SDN is FlowVisor [2], a special purpose OpenFlow controller that can create *slices* of network resources and place each slice under the control of a different OpenFlow controller. FlowVisor slices the network along multiple dimensions: topology, bandwidth, switch CPU, and the set of traffic under the control of the slice, i.e., its *flowspace*. Each slice has a *slice policy* defining its resources and the controller associated with it. A more recent solution for virtualization in SDN is FlowN [3], which will be discussed in Section 2. In this work, we assume FlowVisor as the method of virtualization since it is well-documented, in widespread use in experimental environments such as GENI [4], and available in emulation environments such as Mininet [5].

FlowVisor is interjected between multiple OpenFlow slice controllers and OpenFlow switches acting as the data plane. It intercepts messages between the control and data planes in both directions, and rewrites them in compliance with the slice policy of the corresponding slice. FlowVisor uses a variety of mechanisms (details are in [2]) for different network resources to ensure isolation between slices. Switch CPU is identified as the most critical resource as it is the most easily exhausted, but there are methods to divide and isolate link bandwidth and flow entry space as well.

Fig. 1 illustrates two slices sharing the same network substrate. Each slice has its own view of the topology, which we call the *virtual topology* (VT) of the slice, depending on the switches it contains: Alice's VT is ABDE, and Bob's VT is ABCD. Switches that are added into both slices (like A, B, and D) need to allocate CPU power and flow entries between these slices. Similarly, if a link is being shared between multiple slices (like AB and BD), its bandwidth must be divided to accommodate the flows belonging to each slice. Note that the dashed lines showing the controller-to-switch communications constitute a conceptual representation. The exact paths for control traffic will be determined by the physical locations of the controllers.

Slices of the network can be used for many different purposes depending on what the owners are trying to accomplish. Services or experiments that run on these slices may display a wide variety

* Corresponding author. Tel.: +1 (404) 3129897.
*E-mail addresses:* mdemirci@gatech.edu (M. Demirci), ammar@cc.gatech.edu (M. Ammar).

http://dx.doi.org/10.1016/j.comcom.2014.03.008
0140-3664/© 2014 Elsevier B.V. All rights reserved.

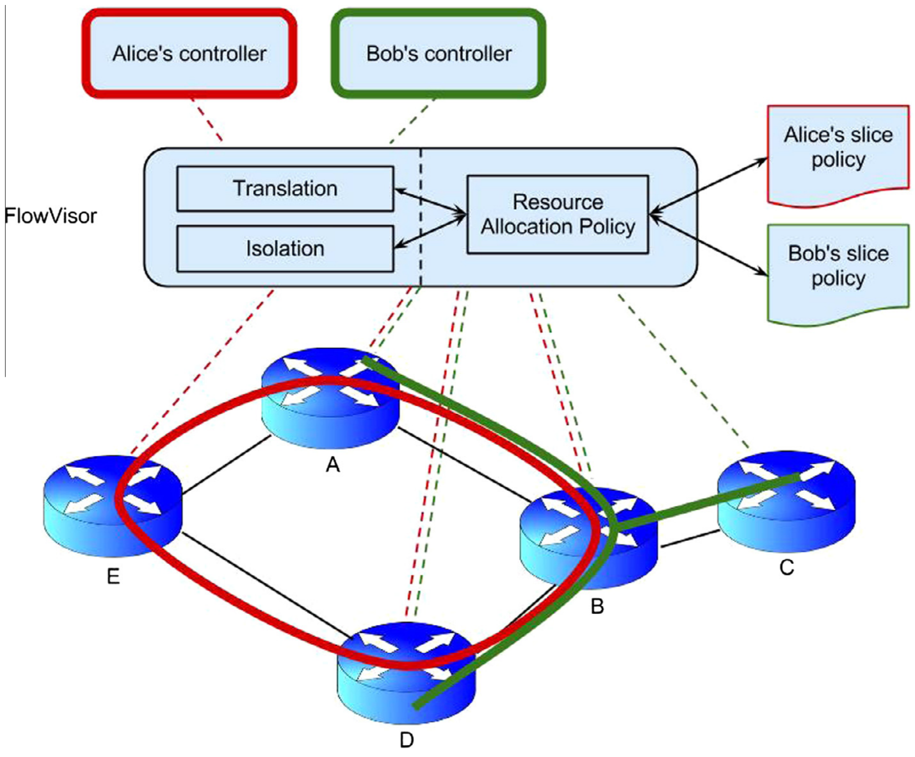

**Fig. 1.** Two FlowVisor slices sharing a substrate of OpenFlow-enabled switches.

of resource requirements. We can think of a slice along with its controller as a virtual network (VN) in SDN. As VNs get bigger and the number of VNs increases, resource contention may become a problem.

VN embedding[1] within a *network virtualization environment*, where substrate network providers are decoupled from VN providers that deploy and operate the VNs [6,7], is a well-known problem. It is an important NP-hard problem because suboptimal mappings can cause bad performance and/or higher operating cost. Solutions that strive for well-balanced and resource-efficient VN mappings have been proposed by researchers [8–10].

In this work, we tackle the problem of designing embedding techniques for VNs in the SDN environment. While some of the previous VN embedding solutions may be applicable to SDN virtualization, some differences inherent in SDN call for different definitions and approaches. Firstly, each VN on an SDN-based substrate possesses its own controller, and there are requirements and considerations that come with the existence of this controller. The controller is responsible for organizing the operation of the VN by sending routing updates, traffic engineering policies etc. and needs to react quickly to faults in the network. Therefore, it must be able to communicate effectively to all the switches that are part of the VN, so any VN embedding effort needs to make sure all controller-to-switch channels avoid congestion and high delays. Furthermore, the ease of customizing packet routes in SDN presents an additional degree of freedom in VN embedding. For these reasons, studying VN embedding in software-defined networks presents different challenges from other network virtualization environments:

i. The placement of the controller is important for the above stated reasons, so it should be treated as a special node by the VN embedding method.
ii. We have more control over the routes in the substrate, so VN embedding can take advantage of this flexibility to fine-tune solutions.
iii. As we will explain in Section 3, differences in SDN resource sharing from that in traditional networks call for modifications to previous VN embedding efforts.

The systems that we investigate in this work are those with many tenants running a number of VNs with varying sizes, functions and requirements. The mapping of the virtual components to substrate components and the placement of the controllers are both important variables influencing the performance of the VNs in such a system. Efficient utilization of networks resources, low risk of congestion, reliability and fast response are all desirable properties. To this end, we identify two simple goals that guide our design: balancing the load on substrate nodes and links, and maintaining low delay between controllers and switches in all VNs.

The contributions of this work can be summarized as follows.

i. We identified virtualization properties unique to the SDN environment and discussed how these properties should affect VN embedding efforts. Specifically, we argued that the placement of the controller for each VN should be a part of embedding due to the critical mission of the controller and the necessity for the controller to have a fast communication channel to the switches that it controls.
ii. We developed two embedding techniques with different priorities: the first method focuses on balancing the stress on substrate components while keeping the delays between controllers and switches within bounds. The second method strives to minimize controller-to-switch delays while limiting the stress on substrate components.
iii. Our stress-balancing embedding technique is derived from previous work [8], but we modified the definitions of node and link stress to account for the presence of control traffic, and made adjustments to treat the controller as a special node.

[1] *VN embedding* and *VN mapping* are being used interchangeably throughout this paper.