Network Troubleshooting with Shadow VNets

Andreas Wundsam Amir Mehmood Anja Feldmann Olaf Maennel
TU Berlin / Deutsche Telekom Laboratories, Germany

ABSTRACT

Troubleshooting a problem on today's Internet is a hard problem, as it is difficult to predict how a planned configuration change or software upgrade will behave in production. This demo demonstrates how the emerging concept of Network Virtualization can be leveraged to overcome such problems: When a problem occurs, an operator can pair his production network with a *Shadow VNet*, and then troubleshoot and upgrade the Shadow VNet in a safe way, without interfering with the production network. Only when convinced that the upgrade is beneficial and does not cause any unwanted side effects, the Shadow VNet is switched to production mode. With the help of the agility and isolation properties of the underlying virtualized infrastructure, the entire operation can be achieved without requiring changes to the physical or logical structure of the production network. With this core experiment from [5], we hope to demonstrate the significant potential of Network Virtualization for troubleshooting purposes to a wider audience outside of the core virtualization community.

1. INTRODUCTION

Debugging and troubleshooting networks is a job not for the faint of heart, especially when networks spawn multiple geographic and administrative domains, and provide many services with differing and sometimes outright contradicting requirements.

Predicting the potentially global implications of configuration changes or software updates in such a context is often difficult, and simulations or testbeds provide only limited insight. As such, these methods often do not suffice to catch real-life network problems that stem from complex interactions of many parties, especially if they only occur sporadically.

In this demonstration, we utilize the emerging concept of network virtualization to tackle network troubleshooting problems in a novel fashion. Network virtualization expands the existing concepts of host and link virtualization to the entire network. Indeed, a *virtual network* or *VNet* appears as a single coherent network, but may span multiple *physical network domains*, that may be *shared* with other VNets. Network virtualization frameworks enable VNets to be *dynamically provisioned* and *configured*, and provide *isolation* between the individual VNets. As a result, virtual networks can be quickly instantiated on demand, and each virtual network operates independently of the other virtual networks. Hardware support for advanced virtualization is currently rapidly improving, both for systems and networks, and links, with, e.g., OpenFlow [3] and Multi-Queue NICs increasing the flexibility and performance of Link-Layer virtualization.

We demonstrate how an operator can diagnose and troubleshoot a problem in his network with the help of *Shadow VNets*[1]. A Shadow VNet replicates not just VNets but also their input, safely. This offers network operators a new range of capabilities with low overhead, namely to safely evaluate and test new configurations or software components, at the scale of their production network, under real user traffic. As the new setup exists in parallel with the old one they can potentially be swapped in a near-atomic fashion. This enables the operator to switch only once he is convinced that the new setup is operational and offers the expected benefits.

Running both networks in isolation requires sufficient substrate resources. As we expect many VNets to be running in parallel on a substrate, each carrying but a fraction of the total traffic, we expect that Shadow VNets can be run for a subset of the productive VNets without overloading the substrate. If resources are sparse, production VNet traffic can be prioritized over shadow VNet traffic to ensure the productive service is not affected.

2. DEMONSTRATION OUTLINE

To demonstrate the benefit of a Shadow VNet we choose the following scenario: A VNet operator that offers both VoIP and Internet access across a best effort VNet, considers moving to a setup with service differentiation to offer better quality of service to its VoIP traffic. This move is motivated by customer complaints about their VoIP quality during certain times of the day.

[1]The name is inspired by the results of Alimi et al. [1], who implement *shadow configurations*. to improve the safety and smoothness of configuration updates.

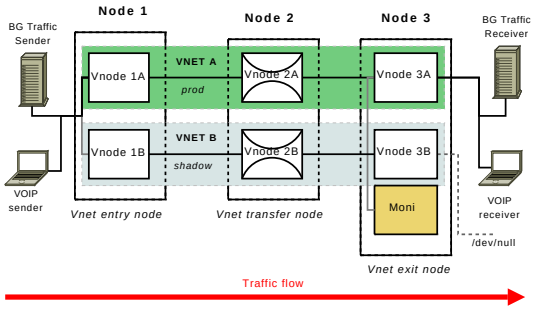

Figure 1: Shadow VNet experiment setup

Table 1: Demonstration phases

Phase	1	2	3	4	5	6
Production VNet	A	A	A	A	B	B
Active VNets	A	A	A&B	A&B	A&B	B
QoS enabled	-	-	-	B	B	B
Internet traffic intensity	L	L/H	H	H	H	H

For our demonstration, a VoIP call and background traffic of varying intensity is routed through a virtualized substrate (Fig. 1). The substrate network consists of three nodes. We now instantiate two parallel VNets, VNet A and B, each with a maximum bandwidth of 20 Mbit/s throughout the experiment, enforced by traffic shaping on Node 2. Moreover, we setup an additional virtual network for monitoring. On entry to the VNet, traffic is duplicated to both VNets A and B and forwarded within each via Node 2 to node 3 using separate virtual links (VLANs). On exit, when leaving Node 3, only output from one VNet is sent to the receivers. In addition, the monitoring VNet "Moni" receives a copy of the VoIP traffic from both VNets.

Metrics: For the monitoring , we measure at two points in the experiment: *Moni* records data for both VNets on exit of the VNet, while the Receiver records the quality as experienced by the user. We record the percentage of dropped packages on the VoIP call as a rough quality indicator, and calculate the *MoS* value as defined by the the E-model, an ITU-T standard for measuring the transmission quality of voice calls [2]

Setup: For the VoIP traffic we use an open source VoIP client based on SIP, generating traffic using the *G*.711 codec. A pool of servers is used to generate the background traffic, using Harpoon [4], with properties that are consistent with those observed in the Internet. To account for different intensities of the background traffic during different times during the day we use two different load levels: L/H corresponding to 20-25%, 60-86% average link utilization, respectively. All traffic sources are located on the left in Fig. 1.

The demonstration is conducted in six phases. Table 1 summarizes the configuration of each phase.

Demo outline: In phase 1, background traffic is running at low intensity. In the middle of phase 2 the intensity of the Internet traffic increases. This causes a problem in VoIP quality as measured by the MoS value, see Fig. 2 for a visualization. The perceived quality drops from a MoS score of 4.34 which corresponds to a "very satisfied" service level drops to 4.16 which corresponds to a level of "satisfied".

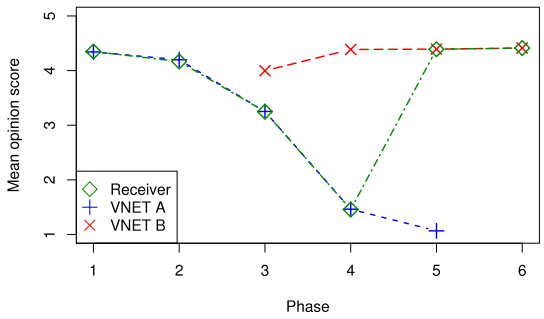

Figure 2: MoS results per phase

As such, the VNet operator instantiates a Shadow VNet at the beginning of phase 3. This means that all packets are now duplicated at Node 1 and are routed in both VNets A and B. However, the end user for VoIP service is still getting service through VNet A. This allows the operator to assess the impact of the degradation and to do root cause analysis in VNet B. In our scenario the operator decides to prioritize VoIP traffic to counter the bad performance. QoS is enabled at the start of phase 4. This reduces the loss rates within VNet B significantly and the MoS value increases again to 4.38. At the start of phase 5 the operator switches his production VNet from VNet A to VNet B. Therefore, the user is now getting the good performance provided by VNet B. With phase 6 the operator dismantles VNet A.

This demo visualizes the significant potential of Network Virtualization in general, and specifically Shadow VNets for troubleshooting and upgrading production networks.

3. REFERENCES

[1] R. Alimi, Y. Wang, and Y. R. Yang. Shadow configuration as a network management primitive. In *ACM Sigcomm*, 2008.

[2] The E-model, a Computational Model for Use in Transmission Planning, ITU-T Rec. G.107, 2005.

[3] 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 CCR*, 2008.

[4] J. Sommers and P. Barford. Self-configuring network traffic generation. In *ACM IMC*, 2004.

[5] A. Wundsam, A. Mehmood, A. Feldmann, and O. Maennel. Improving network troubleshooting using virtualization. Technical Report TU Berlin, Fak IV, No. 2009-12, 2009.