Managing NFV using SDN and Control Theory

Nabeel Akhtar Ibrahim Matta Yuefeng Wang
Computer Science Department, Boston University
Boston, MA 02215
{nabeel, matta, wyf}@bu.edu

***Abstract*—Control theory and SDN (Software Defined Networking) are key components for NFV (Network Function Virtualization) deployment. However little has been done to use a control-theoretic approach for SDN and NFV management. In this demo, we describe a use case for NFV management using control theory and SDN. We use the management architecture of RINA (a clean-slate Recursive InterNetwork Architecture) to manage Virtual Network Function (VNF) instances over the GENI testbed. We deploy Snort, an Intrusion Detection System (IDS) as the VNF. Our network topology has source and destination hosts, multiple IDSes, an Open vSwitch (OVS) and an OpenFlow controller.**

A distributed management application running on RINA measures the state of the VNF instances and communicates this information to a Proportional Integral (PI) controller, which then provides load balancing information to the OpenFlow controller. The latter controller in turn updates traffic flow forwarding rules on the OVS switch, thus balancing load across the VNF instances.

This demo demonstrates the benefits of using such a control-theoretic load balancing approach and the RINA management architecture in virtualized environments for NFV management. It also illustrates that the GENI testbed can easily support a wide range of SDN and NFV related experiments.

I. System Overview

NFV elastic management includes tasks related to Virtual Network Function (VNF) stateful migration from one Virtual Machine (VM) to another, and adding or removing VNF instances depending on the load on the system [1], [2], [3]. NFV elastic management has recently received considerable attention in the research community [1], [2], [3]. However, most of this work focuses on VNF stateful migration. In this demo, we use a new internet architecture – the Recursive InterNetwork Architecture (RINA) [4] – to share VNF state information across the system and use a control-theoretic approach for managing load across VNF instances. To the best of our knowledge, this is the first work that uses a control-theoretic approach to NFV management.

Figure 1 shows an overview of the system. We deploy Snort [5], an Intrusion Detection System (IDS) as the VNF. There can be multiple source and destination hosts and all traffic directed from any source to any destination passes through Snort-IDS. VNF hosts run a distributed monitoring application (deployed over RINA) where each application instance shares the state of the VNF (*i.e.*, load information) with the central controller. The controller runs a control-theoretic Proportional Integral (PI) control algorithm that balances load across the VNF instances by providing the OVS controller with load balancing information, which is then used to update the flow forwarding rules on the OVS switch so new flows are directed to less loaded VNF instances.

We next explain the system in more detail.

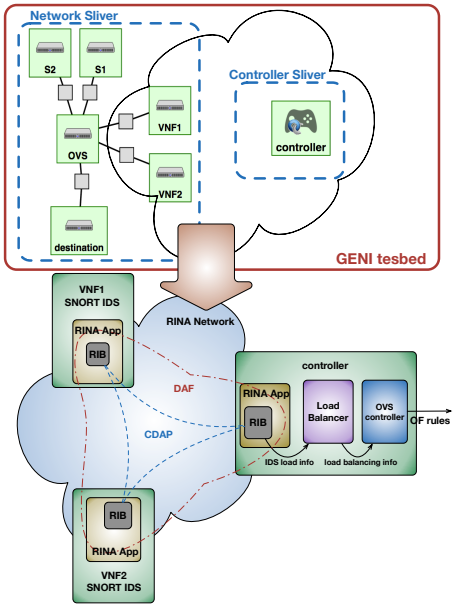


Fig. 1: System overview

1) GENI testbed: GENI (Global Environment for Network Innovations) [6] is a nationwide suite of infrastructure that enables research and education in networking and distributed systems. GENI supports large-scale experimentation with advanced protocols for data-centers, clouds, mobile and SDN networks, *etc*.

As shown in Figure 1, we reserved VMs on the GENI testbed for our use case.

2) Snort-IDS as VNF: Snort-IDS [5] is a widely deployed open-source network intrusion detection and prevention system (IDPS). It has the ability to perform real-time traffic analysis on IP networks. We used Snort as VNF in our use case, where all traffic is checked against Snort community rules for intrusion detection.

3) RINA: The Recursive InterNetwork Architecture (RINA) [4] is a clean-slate network architecture that overcomes inherent weaknesses of the current internet, *e.g.*, security and support for mobility and quality of service.

For our system, we created a distributed RINA monitoring

978-1-5090-0223-8/16/$31.00

application that monitors the VNF instances. The controller uses this monitoring application to get the state (load information) of the VNF instances. Each monitoring application process on the VNF VMs periodically publishes its VNF load information. The monitoring application process running on the controller VM subscribes to this information, and passes the average over the last few measurements to the PI controller for load balancing, as explained next.

4) PI Controller: The block diagram of the Proportional Integral (PI) controlled NFV system is shown in Figure 2. The RINA-based distributed monitoring application provides the VNF1 state (average CPU load) information $L(t)$ to the PI controller. The maximum capacity of a VNF instance is T. If the load on VNF1 exceeds T, new traffic flows are forwarded to a second VNF instance, VNF2. Assuming instantaneous feedback / measured load $L(t)$, the PI control equation is given by:

$$x(t) = \max[0, \min[1,\ x(t-1) + K(\frac{L(t)}{T} - 1)]]$$

where $x(t)$ is the ratio of traffic diverted to VNF2 at time t, and K is the controller's gain.

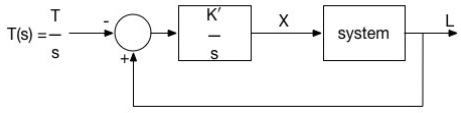


Fig. 2: Block diagram of the PI-controlled NFV system. System load L and target load $T(s) = \frac{T}{s}$ of VNF1 is used to compute X, *i.e.* ratio of traffic diverted to VNF2. $K' = \frac{K}{T}$.

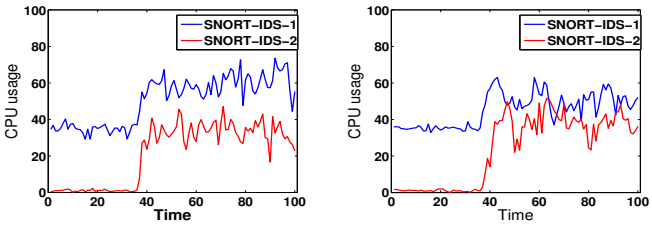
5) OVS Controller:

- *PI-based OVS controller:* Information about load balancing is provided to OVS by the PI controller. The control variable $x(t)$ provided by the PI controller is the ratio of traffic diverted to a second VNF (Snort) instance. The OVS controller updates the forwarding rules on the OVS switch based on this information to bring the system to a more balanced steady state.
- *Round Robin based OVS controller:* We also implemented an OVS controller based on a Round Robin (RR) load balancer. The RR balancer does not take load into account. The OVS controller running RR directs each new flow request to one of the two VNF instances in a round robin fashion.

II. Real-Time Graphs

We compare the PI-based load balancer with the traditional Round Robin (RR) load balancer using real-time load graphs for VNF1 and VNF2. The iPerf application [7] is used to generate TCP traffic. Figure 3a shows the instantaneous CPU load on each VNF instance under the simple RR load balancer. Since RR does not take into account load on the VNF instances, one VNF instance might become overloaded compared to the other one. The VNF host running Snort-IDS-1 initially has around 38% CPU usage because of other applications running on it. We generate traffic at time 39 using iPerf with similar flows that last for the same amount of time. We can see that RR directs iPerf traffic equally between IDS-1 and IDS-2. However this overloads VNF1 (IDS-1) since there are other applications running on it.

Figure 3b shows the CPU load on each VNF instance under the PI-based load balancer. The target load (T) on IDS-1 is set to 50%. When we generate iPerf traffic at time 39, the load on IDS-1 increases beyond 50% and so new flows get diverted to the second VNF instance (IDS-2). We can see that the system stabilizes at an average load on IDS-1 of 50% while the rest of the load is diverted to IDS-2.



(a) Simple Round Robin load balancing

(b) Load balancing based on PI control (T = 50%)

Fig. 3: Instantaneous CPU load under RR and PI control

III. Conclusion

This demo shows how control theory can be used to manage NFV using SDN. We also show that the management architecture of RINA provides several facilities that can be used for easy NFV elastic management.

The GENI testbed is used for experimentation. GENI provides state of the art experimentation facility and can support a wide range of experiments, including NFV and SDN related experiments. The widely used Snort IDS is deployed as the VNF and traffic is directed to different Snort IDS instances for processing. Benefits of using a control-theoretic load balancing approach over a traditional Round Robin based load balancer are highlighted in this work.

More details on this work can be found at [8].

References

[1] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14, 2014.

[2] Shriram, Rajagopalan, Dan, Williams, Hani, and Jamjoom, "Pico Replication: A High Availability Framework for Middleboxes," in *ACM Symposium on Cloud Computing (SoCC)*, Santa Clara, California, Oct 2013.

[3] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/Merge: System Support for Elastic Execution in Virtual Middleboxes," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13, 2013.

[4] Boston University RINA Lab, http://csr.bu.edu/rina/.

[5] SNORT, https://www.snort.org/.

[6] GENI, http://www.geni.net/.

[7] iPerf, https://iperf.fr/.

[8] N. Akhtar, I. Matta, and Y. Wang, "Managing NFV using SDN and Control Theory," CS Department, Boston University, Tech. Rep. BUCS-TR-2015-013, December 14 2015. [Online]. Available: http://www.cs.bu.edu/techreports/2015-013-nfv-sdn-control.ps.Z