Open Box Dashboard

Roy Franco

Project proposal

IDC Msc

August 2017

Goals

The goal of this project is to create a web-based graphical user interface for an administrator of the OpenBox controller.
The dashboard will be used for demonstration purposes, and will allow users to play around with the system and get familiar with it, analyze and monitor the different components of the network, create and deploy OBAs and more.
The goal is to further extend the OpenBox SDK to rapid the adoption of the OpenBox framework on real production networks.

Introduction

Most modern networks nowadays contain a massive amount of appliances,
each appliance typically executes one network function (NF) (eg. Firewall). Each such appliance is bought, configured and administered separately. Most NFs perform some kind of Deep Packet Inspection (DPI). OpenBox provides a framework for network-wide deployment and management of NFs which decouples the NFs control plane from NFs data plane. OpenBox consists of three logic components. First, user-defined OpenBox
Applications that provide NF specifications. Second, a logically-centralized
OpenBox Controller (OBC) which serves as the control plane. Finally,
OpenBox Instances (OBI) constitute OpenBox's data plane.

In January 2017, in his Msc final project, Dan Shmidt had presented the design and implementation for the user facing interface of the OpenBox Controller, which allows network administrators to efficiently create and manage their network functions.
The implementation supplies users with a framework from which they can build and experiment with NFs, as well as a functioning OpenBox Controller which loads NFs and manages the OpenBox control plane.

The OpenBox Dashboard project will further extend Dan's work on the OpenBox controller, and will include a REST API for the OpenBox controller which will be utilized by the dashboard.

Related Work

In recent years, middleboxes and network functions have been major topics of interest. In this section we discuss and compare the state-of-the-art works that are directly related to this paper.

OpenBox allows easier adoption of hardware accelerators for packet processing. Very few works have addressed hardware acceleration in an NFV environment [9], and those that have focused on the hypervisor level [10,11]. Such ideas can be used in the OpenBox data plane by the OBIs, and thus provide additional hardware acceleration support.

CoMb [12] focuses on consolidating multiple virtual middleboxes into a single physical data plane location, thus improving the performance of the network in the common case where not all the middleboxes have peak load at the same time. E2 [13] is a scheduling framework for composition of multiple virtual NFs. It targets a very specific hardware infrastructure, and manages both the servers on which NFs are running and the virtual network switches that interconnect them. Unlike OpenBox, CoMb and E2 only decompose NFs to provide I/O optimizations such as zero-copy and TCP reconstruction, but not to reuse core processing blocks such as classifiers and modifiers. Specifically, the CoMb paper has left for future research the exploration of the choice of an optimal set of reusable modules [12, Section 6.3]. We view our paper as another step forward in this direction. xOMB [14] presents a specific software platform for running middleboxes on general purpose servers. However, it does not consolidate multiple applications to the same processing pipeline. ClickOS [15] is a runtime platform for virtual NFs based on the Click modular router [16] as the underlying packet processor. ClickOS provides I/O optimizations for NFs and reduced latency for packets that traverse multiple NFs in the same physical location. ClickOS does not have a network-wide centralized control, and it does not merge multiple NFs, but only chains them and optimizes their I/O. Commercial solutions such as OpenStack [17], OpenMANO [18] OpNFV [19], and UNIFY [20] are focused on the orchestration problem. They all assume each NF is a monolithic VM, and try to improve scaling,

placement, provisioning, and migration. Stratos [21] also provides a solution for NFV orchestration, including placement, scaling, provisioning, and traffic steering. OpenNF [22] proposes a centralized control plane for sharing information between software NF applications, in cases of NF replication and migration. However, their work focuses only on the state sharing and on the forwarding problems that arise with replication and migration, so in a sense it is orthogonal to our work. OpenState [23] and SNAP [24] are programming language for stateful SDN switches. OpenState makes it possible to apply finite automata rules to switches, rather than match-action rules only. SNAP takes a networkwide approach where programs are written for “one big switch” and the exact local policies are determined by the compiler. Both these works are focused on headerbased processing, but such ideas

could be useful to create programming languages on top of the OpenBox framework. To the best of our knowledge, Slick [25] is the only work to identify the potential in core processing step reuse across multiple NFs. They present a framework with centralized control that lets NF applications be programmed on top of it, and use Slick machines in the data plane to realize the logic of these applications. The Slick framework is mostly focused on the placement problem, and the API it provides is much more limited than the OpenBox northbound API. Slick does not share its elements across multiple applications and the paper does not propose a general communication protocol between data plane units and their controller. Unlike our OBIs, Slick only support software data plane units; these units cannot be extended. The solutions to the placement problems presented in [25] can be implemented in the OpenBox control plane.

In Their 2015 workshop paper[6], Anat Bremler-Barr, Yotam Harchol, and David Hay described the proposed architecture but presented a very limited framework that uses a unified processing pipeline for merging multiple middleboxes. The proposed unified pipeline was very restrictive. In their 2016 paper[7], they present a much more flexible NF programming model, including an algorithm to merge multiple applications given this flexible model.

OpenBox is not only easy to deploy and to program but also improves network performance. We envision that frameworks such as OpenBox will pave the way for further advances in network function virtualization (NFV) with respect to NF programming, deployment, and easier management, while maintaining and improving performance. The flexible support for hardware accelerators for packet processing makes OpenBox even more appealing as today most NFV frameworks assume completely virtual environments and do not support any hardware accelerators.

Proposal

To the best of our knowledge, OpenBox is the first general framework for software-defined NFs, which includes specifications for a logically-centralized controller and its northbound API for NF application development, for an extensible data plane instance (OBI), and for a communication protocol between the two. OpenBox also propose a novel algorithm for merging the core logic of multiple NF applications in the control plane, such that computationally-intensive procedures (e.g., packet classification or DPI) are only performed once.

In order to make another step towards an OpenBox industry adoption, we will create the OpenBox dashboard, a web based application that will utilized the moonlight controller's REST api to configure, analyze and monitor an OpenBox network.

Implementation Details

OpenBox Controller

The OpenBox controller would be extended by the following

- **Network Information REST service**
  Report on the deployed network components – topology, OBAs, OBIs etc.

- **OBA Controller REST service**
  - Receive read/write requests and send them through the SouthBoundAPI
  - Deploy/Remove OBA

- **Profiling component**
  Collect the data sent through the northbound API and the southbound API and store it in a database.
  A worker will periodically execute all possible read requests to the deployed OBAs and save the responses in a database

Technology Stack

- Jersey web services
- MySQL database (for now)

OpenBox Dashboard

The OpenBox dashboard requires both client side and server side development.

- **Client Side**
  The client side includes the views for the various application components as well as an OpenBox controller API client that communicates with the dashboard's backend.

- **Server Side**
  The OpenBox dashboard's backend is responsible for providing all data required by the client side views through a REST API as well as exposing an API for performing actions on a network.

  The backend will query OpenBox Controller's "Network Information REST service" in order to obtain information on the network.

  The backend will send requests to the OpenBox Controller's "OBA Controller REST service" in order to make changes in the network.

  The backend will query the database populated by the OpenBox controller in order to obtain the network's event log

Technology Stack

- Frontend framework - angular2
- Graphs – D3 framework - http://jsfiddle.net/MetalMonkey/JnNwu/
  - Display processing graphs component

Validating Project Success

The OpenBox Dashboard is another step for OpenBox, and will allow users to more easily get familiar and experiment with OpenBox.
We would want that the dashboard would help to grow the OpenBox community by doing so.

We will measure the OpenBox community growth by four means:
1. Github contributors/forks.
2. Downloads and citations of the OpenBox papers.
3. Demonstration initiatives by the OpenBox team to expose key players to OpenBox using the dashboard.

Project Planning

Stage	Task	Start Date	End Date
Dashboard Mock Data Preparation	Design Network Information REST service	W3 September 2017	W2 October 2017
	Design Profiling component	W3 October 2017	W3 November 2017
	Prepare Mock data	W4 November	W4 November
Dashboard Development	Processing Block Component	W1 December 2017	W2 December 2017
	Processing Graph Component	W3 December 2017	W4 December 2017
	Topology view	W1 January 2017	W4 January 2017
	OBA view	W1 February 2018	W4 February 2018
	OBI view	W1 March 2018	W4 March 2018
Moonlight Controller Development - Read Only services	Network Information REST service	W1 MarAprilch 2018	W4 April 2018
	Profiling component	W1 May 2018	W4 May 2018
Dashboard+Controller Integration - Read Only	Replace mock services with the controller for read-only mode	W1 June 2018	W4 June 2018
Packaging	Bundle the application to make it out of the box demo ready	W1 July 2018	W3 July 2018
OBA Controller - interactive mode REST services	Send read/write requests	W4 July 2018	W2 August 2018
	Deploy a new OBA and load it dynamically	W3 August 2018	W4 August 2018
	Remove OBAs	W1 September 2018	W2 September 2018
Dashboard Development - interactive services	Send read/write requests	W3 September 2018	W2 October 2018
	Deploy a new OBA and load it dynamically	W3 October 2018	W2 November 2018
	Remove OBAs	W3 November 2018	W4 November 2018

References

[1] Mininet. http://mininet.org/.

[2] Openbox controller implementation. https://github.com/

OpenBoxProject/moonlight.

[3] Openbox firewall implementation. https://github.com/

OpenBoxProject/MoonlightFirewall.

[4] Openbox protocol specification. http://www.deepness-lab.org/pubs/

OpenBoxSpecification1.1.0.pdf.

[5] Openbox snort ids implementation. https://github.com/

OpenBoxProject/MoonlightSnort.

[6] Anat Bremler-Barr, Yotam Harchol, and David Hay. Openbox: Enabling innovation in middlebox applications. In *Proceedings of the 2015*

ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network

Function Virtualization, pages 67–72. ACM, 2015.

[7] Anat Bremler-Barr, Yotam Harchol, and David Hay. Openbox: A

software-defined framework for developing, deploying, and managing network functions. ACM, SIGCOMM, 2016.

[8] Pavel Lazar. Design and implementation of a data plane for the openbox framework, 2016.

[9] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. IEEE Comm. Surveys Tutorials, 18(1):236–262, 2016.

[10] Z. Bronstein, E. Roch, J. Xia, and A. Molkho. Uniform handling and abstraction of NFV hardware accelerators. IEEE Network, 29(3):22–29, 2015.

[11] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu. OpenANFV: Accelerating network function virtualization with a consolidated framework in openstack. In SIGCOMM, pages 353–354, 2014.

[12] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated m

[13] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker. E2: a framework for NFV applications. In SOSP, pages 121–136, 2015.

[14] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat. xOMB: extensible open middleboxes with commodity servers. In ANCS, pages 49–60, 2012.

[15] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. In NSDI, pages 459–473, 2014.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. ACM Trans. Comput. Syst., 18(3):263–297, Aug 2000.

[17] OpenStack open source cloud computing software. https://www.openstack.org/.

[18] OpenMANO. https://github.com/nfvlabs/openmano.

[19] OpNFV. https://www.opnfv.org/

[20] W. John, C. Meirosu, B. Pechenot, P. Skoldstrom, P. Kreuger, and R. Steinert. Scalable Software Defined Monitoring for Serv

[21] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar. Stratos: A network-aware orchestration layer for middleboxes in the cloud. CoRR, abs/1305.0209, 2013.

[22] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. OpenNF: enabling innovation in network function control. In SIGCOMM, pages 163–174, 2014.

[23] G. Bianchi, M. Bonola, A. Capone, and C. Cascone. OpenState: Programming platform-independent stateful OpenFlow applications inside the switch. SIGCOMM Comput. Commun. Rev., 44(2):44–51, Apr 2014.

[24] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker. SNAP: Stateful Network-Wide Abstractions for Packet Processing. In SIGCOMM, 2016.

[25] B. Anwer, T. Benson, N. Feamster, and D. Levin. Programming Slick Network Functions. In SOSR, pages 14:1–14:13, 2015.

[26] A. Bremler-Barr, Y. Harchol, and D. Hay. OpenBox: Enabling Innovation in Middlebox Applications. In HotMiddlebox, pages 67–72, 2015.

[27] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral. Deep packet inspection as a service. In CoNEXT, pages 271–282, 2014.