4.5.4 Content Storage

Some applications require storing content such as payload or whole packets for future use. Simple examples are web cache, which stores HTTP responses, and spam filter, which quarantines messages for future inspection. The OpenBox Storage Server (OBS) stores content received from OBIs, making it available for OpenBox applications and for OBIs. Multiple instances of OBS can exist in a network, usually one per each OBI or a set of OBIs. Data is stored separately for different applications for security purposes, and can be retrieved by the OpenBox Application that stored it or by an OBI on behalf of that application, using a key (e.g., a URL). OBS instances can be placed on the same physical machine that runs their corresponding OBI or in a remote location, thus data is sent to and from it over the network.

5. INITIAL IMPLEMENTATION

We have began implementation of an OpenBox framework that consists of an OpenBox controller named *Moonlight*, implemented in Java (sources: https://github.com/DeepnessLab/moonlight). OBIs are implemented in Python and C (sources: https://github.com/DeepnessLab/obsi). In addition, we implemented an OpenFlow 1.3 based traffic steering application (TSA) that manages and enforces service chains, as an OpenDaylight [6] bundle. We emulate a network in Mininet [15] where OBIs and OBC run as hosts.

The Moonlight controller defines the following processing modules: *header lookup*, *session analyzer*, *protocol analyzer*, *ingress hasher*, *payload handler*, and *decision maker*. Each module has some settings that can be applied to it. For example, an OpenBox application can tell the header lookup module which fields it needs for future processing and thus should be saved in metadata map. The payload handler can be set to use decompression and normalization modules.

On top of the Moonlight controller we have implemented three sample OpenBox applications: a NAT, a NIPS, and a L7 load balancer. Their code is available under the Moonlight github repository. The NAT implementation, which provides fully-functional NAT capabilities, takes about 100 lines of code.

6. CONCLUSIONS AND FUTURE WORK

In this paper we present OpenBox - an open framework for development of advanced network control applications on top of a network that contains general purpose service instances. The OpenBox framework can replace legacy middleboxes and provide more flexible and scalable development and deployment of applications with the same roles, and allow innovative solutions to be easily created. The framework can be augmented and extended by any vendor by adding more sophisticated functionalities to its data plane.

Our research now focuses on finishing the definition of the OpenBox protocol, and the implementation of the OpenBox service instances and the Moonlight OpenBox controller. Future research may focus on interesting problems of the suggested framework, such as smart allocation of OBIs, smart allocation of tasks for OBIs to avoid overloading, and questions related to traffic engineering in a network with OpenBox. Of course, the framework invites innovative development of applications for enhanced network security and performance.

Acknowledgments

This research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement n° 259085, the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and the Neptune Consortium, administered by the Office of the Chief Scientist of the Israeli ministry of Industry, Trade, and Labor.

7. REFERENCES

[1] 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.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, Jul 2014.

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

[4] ECMA. The JSON data interchange format, October 2013. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[5] ETSI. Network functions virtualisation - introductory white paper, 2012. http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[6] L. Foundation. Opendaylight. http://www.opendaylight.org/.

[7] O. N. Foundation. Openflow switch specification version 1.4.0, October 2013. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[8] 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.

[9] 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.

[10] V. Heorhiadi, M. K. Reiter, and V. Sekar. New opportunities for load balancing in network-wide intrusion detection systems. In *CoNEXT*, pages 361–372, 2012.

[11] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM*, pages 51–62, 2008.

[12] L. Kekely, V. Pus, and J. Korenek. Software defined monitoring of application protocols. In *INFOCOM*, pages 1725–1733, 2014.

[13] 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.

[14] P. Kothari. Network Service Header support for OVS. OVS Code Patch, September 2013. http://openvswitch.org/pipermail/dev/2013-September/032036.html.

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

[16] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *SIGCOMM*, pages 27–38, 2013.

[17] P. Quinn, P. Agarwal, R. Manur, R. Fernando, J. Guichard, S. Kumar, A. Chauhan, M. Smith, N. Yadav, and B. McConnell. Network service header. IETF Internet-Draft, February 2014. https://datatracker.ietf.org/doc/draft-quinn-sfc-nsh.

[18] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *NSDI*, pages 323–336, 2012.

[19] N. Shah. Cisco vPath technology enabling best-in-class cloud network services, August 2013. http://bit.ly/1F6bbMH.

[20] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. In *SIGCOMM*, pages 13–24, 2012.

[21] R. Wang, D. Butnariu, and J. Rexford. OpenFlow-based server load balancing gone wild. In *Hot-ICE'11*, pages 12–12, 2011.