SDN Interactive Manager: An OpenFlow-Based SDN Manager

Pedro Heleno Isolani*, Juliano Araujo Wickboldt*, Cristiano Bonato Both†,
Juergen Rochol*, and Lisandro Zambenedetti Granville*
*Federal University of Rio Grande do Sul, Porto Alegre, Brasil
†University of Santa Cruz do Sul, Santa Cruz do Sul, Brasil
Email: {phisolani, jwickboldt, juergen, granville}@inf.ufrgs.br, cboth@unisc.br

I. Related Work

Currently, many investigations addressed SDN management considering using monitoring information for different purposes. Zhang [1] used monitoring information to develop algorithms for anomaly detection and traffic engineering. Jose *et al*. [2] used the same monitoring information to propose mechanisms for online measurement of large traffic aggregates. Yu *et al*. [3] proposed FlowSense, which is aimed to keep the lowest control channel overhead and the highest information accuracy as possible in OpenFlow monitoring. Chowdhury *et al*. addressed SDN monitoring from a different perspective. The authors proposed Payless [4] framework that is able to deal with monitoring considering the polling frequency and data granularity. Payless relies on OpenTM [5] to select only important switches to be monitored, also aiming to reduce the overhead imposed in the control channel. Thus, this framework allows to adjust the polling frequency parameter to balance the control channel overhead, imposed by monitoring messages, and accuracy of monitored information.

The aforementioned proposals are focused on the use of monitoring information to automate tasks, such as reducing control traffic overhead and protecting the network. No previous investigation aims to employ monitoring information to help the administrator understanding the network behavior nor interact with it. To the best of our knowledge, there are no solutions that leverage OpenFlow monitoring messages to create network visualizations and address the interactive configuration of SDN-related parameters. Before presenting how we approached such a problem, we present our motivation and all objectives to archieve with our propototype, focusing on didactly detach important central points of our solution.

II. Motivation and Objectives

In summary, SDN management activities are considerably different from traditional networks, thus deserving proper attention. The SDN controller customizations might impact in terms of resource consumption and traffic forwarding performance. As a consequence, such impact is difficult to asses because traditional network management solutions were not designed to cope in the context of SDN. Moreover, several solutions deal with SDN monitoring, visualization, or configuration for automated tasks (*e.g.*, reduce control traffic or network protection) and they not aim to help the administrator to understand the network behavior and interact with it. Therefore, our objectives in building our SDN management prototype are the following:

- Perform SDN monitoring with configurable polling intervals specifically focusing on inspect resource usage and control channel load metrics.
- Present monitoring information obtained in interactive visualizations that emphasize these metrics.
- Support the administrator interaction by configuring and reconfiguring SDN-related parameters.

III. SDN Interactive Manager

In this section, we describe our prototype to manage SDN through monitoring, visualization, and configuration. Our prototype is an Web application named *SDN Interactive Manager* that allows the administrator to understand and control the network behavior through a *Graphical User Interface* (GUI).

Figure 1 shows the SDN architecture along with our prototype. Overall, SDN introduces an architecture with four planes: *management*, *application*, *control*, and *forwarding*. All SDN planes communicate with each other through interfaces. For example, the *management* plane uses a set of *Management Interfaces* (MI) to exchange information and to control elements in other planes. In addition, an *northbound* API interface establishes bidirectional communication between *application* and *control* planes, while the *southbound* API does the same for *control* and *forwarding* planes.

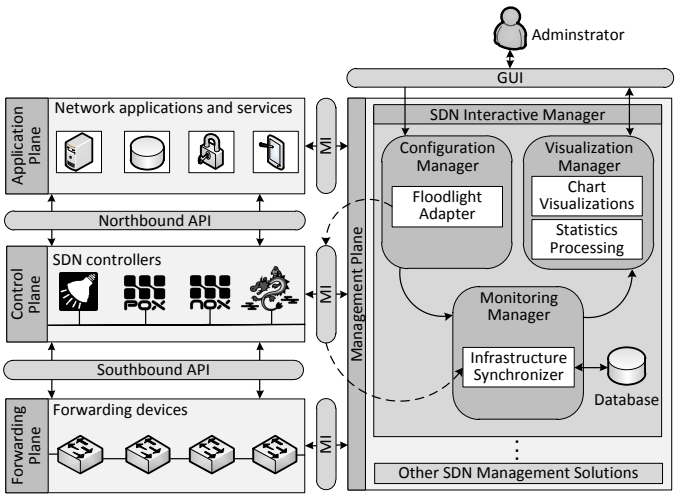

Fig. 1. Conceptual Architecture

Our prototype includes three main components: *Monitoring Manager*, *Visualization Manager*, and *Configuration Manager*. The *SDN Interactive Manager* sits in the *management* plane of SDN alongside other existing or yet to be developed solutions. Since our approach to SDN management comprises interactive network management, we also depict in the architecture the *Administrator* who interacts primarily with the *Visualization Manager* and the *Configuration Manager* components through an GUI (shown in Figure 2). Therefore, our solution creates a loop of activities by integrating these components with the *Administrator* interactions. Each of these components performs independent tasks described as follows.

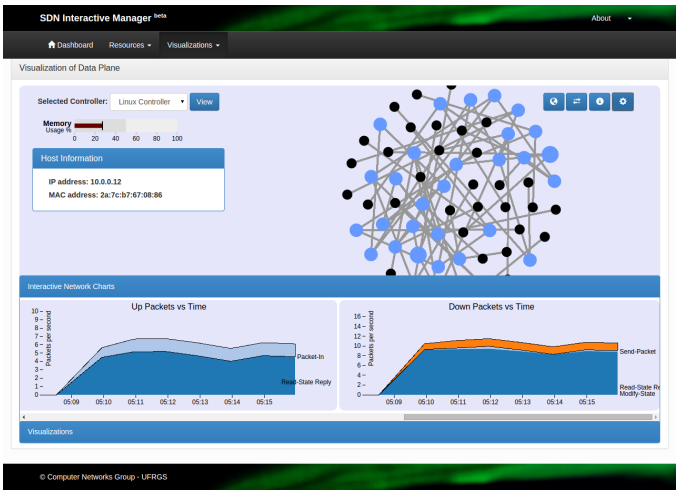

Fig. 2. SDN Interactive Manager GUI

Monitoring Manager – Responsible for retrieving updated information about the network and storing it in a local *Database*. The *Infrastructure Synchronizer* module collects information, such as traffic statistics, network topology, and device data, by accessing an MI to one or more controllers situated in the *control* plane. In this case, we used the RESTful API provided by the Floodlight controller to access information about the physical topology, including links, switches, and hosts. Moreover, the *Monitoring Manager* also gathers data traffic counters of every rule installed on each switch in the network. However, by default, the Floodlight controller does not address control traffic counters. Therefore, we developed a module for Floodlight controller, which we called *Control Statistics Aggregator*, to gather these counters and extended the RESTful API to report them. The *Control Statistics Aggregator* module monitors control channel connections to inspect and count OpenFlow messages generated either by the controller or switches.

Visualization Manager – Comprises the *Statistics Processing* and *Chart Visualizations* modules. With information stored in the *Database*, the *Statistics Processing* module is able to aggregate data per host, switch, controller, or even the entire network to be used by the *Chart Visualizations* module. Furthermore, the *Statistics Processing* module is also able to identify which rules are active and which are idle on forwarding devices. To build interactive visualizations that can be analyzed by the *Administrator*, the *Chart Visualizations* module uses a rich library of interface components (*e.g.*, charts and diagrams) that enables data updates in real time. This component is implemented as an interactive Web based application relying mainly on the D3.js library. Thus, as the network is periodically monitored by the *Monitoring Manager* component, visualizations are updated in the same pace. The integration between these two components allows our prototype to display network visualizations, such as: topology view with intuitive hints on where resource bottlenecks might be occurring, charts of idle and active rules installed on forwarding devices, amount of OpenFlow messages flowing in control channels, and the traffic rate these messages generate.

Configuration Manager – Allows the *Administrator* to check and configure SDN-related parameters on network controllers through the MI. The *Floodlight Adapter* module permits setting up the polling interval for monitoring network devices through a friendly GUI. Moreover, the *Administrator* can trigger the *Floodlight Adapter* module to set the *idle timeout* to be configured on rule installation process. All SDN-related parameter configurations on the controller are performed using an extension of the *northbound* API implementation that is embedded in MI between *management* and *control* planes. We developed *Floodlight Adapter* module to support the configuration of the forwarding rule *idle timeout* parameter, but the implementation can be easily extended to support others. Another parameter that can be set is the polling interval of the *Infrastructure Synchronizer* module. This parameter affects the frequency of reading information from counters of network devices and also impacts on the control channel load generated in both directions.

Our prototype was designed following the Model-View-Template (MVT) design pattern and we chose Python 2.7 with Django 1.6 as development framework. The *Database* used to store network devices information as well as traffic statistics is PostgreSQL 9.3.5 and has been implemented as Django Models. Each of monitoring, visualization, and configuration functionalities were developed as Views and the GUI as Templates. As a consequence of our unusual requirements, we had to implement a module for the Floodlight controller to support our advanced management features, such as reporting control traffic statistics and dynamically configuring the *idle timeout* of forwarding rules.

REFERENCES

[1] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in *Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, December 2013, pp. 25–30.

[2] L. Jose, M. Yu, and J. Rexford, "Online Measurement of Large Traffic Aggregates on Commodity Switches," in *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, October 2011, pp. 13–13.

[3] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring Network Utilization with Zero Measurement Cost," in *Proceedings of the 14th International Conference on Passive and Active Measurement (PAM)*, March 2013, pp. 31–41.

[4] S. Chowdhury, M. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for Software Defined Networks," in *Proceedings of the 14th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[5] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic Matrix Estimator for OpenFlow Networks," in *Proceedings of the 11th International Conference on Passive and Active Measurement (PAM)*, April 2010, pp. 201–210.