Automatic Custom Generation of Topologies and Configuration of Routing protocols in SDN

Apoorv Shukla
TU Berlin
apoorv@inet.tu-berlin.de

Mengchen Shi
TU Berlin
mengchen.shi@campus.tu-berlin.de

Anja Feldmann
TU Berlin
anja@inet.tu-berlin.de

ABSTRACT
Software-Defined Networks (SDNs) has been an area of interest among researchers from academia and industry. SDNs, however, also introduce new challenges, for example researchers work under strict time constraints and need to conduct frequent experiments to verify their ideas on scalable simulation of real-life topologies. The challenge is two-fold: first, the researchers need to manually generate the topologies and second, manually configure the devices in the generated topologies to enable routing protocols. We demonstrate two novel tools, namely, Topology Generator and Enhanced Automatic Configuration RouteFlow (EACRF), which automatically generate the custom scalable topologies at the SDN data plane and configure routing protocols like BGP and OSPF, at the SDN control plane, in a seamless fashion in quick time. EACRF is an enhancement of RouteFlow which can be used in conjunction with Topology Generator or independently.

CCS CONCEPTS
• Networks → Programmable networks; Network experimentation;

KEYWORDS
Reproducible Research; SDN; Mininet; Quagga; OpenFlow

INTRODUCTION
Researchers grapple with the evaluation of their prototypes on Software-Defined Networks (SDNs), primarily due to the time required for manual generation and configuration on SDN switches. The time consumed in creating and configuring custom topologies is directly proportional to the scale and complexity of the topologies and configurations. Moreover, manual generation and configuration is error-prone since misconnections and misconfigurations occur and thus, troubleshooting a wrongly-configured flow-rule consumes even more time and resources. Topology Generator is a novel tool that automatically generates custom topologies. In OpenFlow networks, RouteFlow [5] provides a manual way to run routing control platforms (e.g. Quagga). It executes the control logic of the OvS [3] switches (S1, S2, S3 and S4 in Fig. 1) through virtual machines (VM1, VM2, VM3 and VM4 in Fig. 1) which mirror a physical topology. Each virtual machine (VM) runs a routing control platform (e.g. Quagga) and is dynamically interconnected with other VMs. In the same figure, the four switches connect with RFProxy (SDN controller) by OpenFlow API. Four corresponding VMs run OSPF protocol and generate the routing tables. Then, RFClient collects the information and sends to RFProxy which translates these forwarding decisions into corresponding flow tables of the four switches. Automatic Configuration RouteFlow (ACRF) [6] was a solution presented only for automatic configuration but the code lacked BGP support and the performance was bottlenecked by FlowVisor and topology discovery module leading to less throughput and more execution time with the increase in network scale. Moreover, FlowVisor only supports OpenFlow 1.0 standard. If the RouteFlow project is updated to version 1.3 or 1.4, the FlowVisor could not be used anymore. Therefore, we have developed Enhanced Automatic Configuration RouteFlow (EACRF) to improve ACRF.

AUTOMATIC CUSTOM TOPOLOGY GENERATION
We have implemented data plane tool, Topology Generator, which consists of two main components: MininetParser and a Text-Based User Interface (TUI). Inspired from Auto-Mininet [4], MininetParser uses Mininet API and a python package called NetworkX [2] to produce various topologies with one line command. The central idea is using NetworkX to generate or translate a network graph, then the graph will be translated into the Mininet structure. Moreover, NetworkX provides a topology plot with the customized label for the nodes. To make the creation of a topology convenient, we developed a TUI to simplify the process. With inputs such as topology type and controller IP address to TUI, a Mininet script is ready to execute and can be reused. In TUI, due to the error checking and input revocation, the incorrect input is prevented. For example, IP...
shows the total time taken by both of the tools:

shows the execution time of EACRF to configure in four scenarios, namely, OSPF, Full-Mesh iBGP, eBGP and BGP (iBGP + eBGP)

Figure 2: Left: Internet 2 Topology generated by Topology Generator. Right: Execution Time of the EACRF for configurations in four scenarios, namely, OSPF, Full-Mesh iBGP, eBGP and BGP (iBGP + eBGP)

address should lie in 255.255.255.255 range, and the port number should be an integer in the range of [1 - 65535]. Users save time to retract and change their input options without restarting TUI. Topology Generator allows generation of topologies like Ring, Star, Binary Tree, Full Mesh, 2-D mesh and any custom backbone topology from Internet Topology Zoo [1]. The left image in Fig. 2 shows the Internet2 topology created automatically by Topology Generator which uses Internet2 topology in GraphML syntax from Topology Zoo and takes 30 seconds for the topology creation. Topology Generator can be used to customize the addition of Autonomous Systems (ASes) and allocate different switches to the respective ASes along with the choice of routing protocol (OSPF or BGP or both) to be configured later by EACRF. For e.g., in the left image of Fig. 2, we generated two Autonomous Systems (ASes): AS65001 and AS65002 and allocated different switches to the respective ASes. The plot to the left in Fig. 4 shows that the time taken to generate linear topologies by Topology Generator (in red) is significantly less than the manual generation (in blue).

3 AUTOMATIC CONFIGURATION OF ROUTING PROTOCOLS

Inspired from ACRF [6] and based on RouteFlow [5], we implemented control plane tool, Enhanced Automatic Configuration RouteFlow (EACRF). EACRF has two flavours: without FlowVisor as the performance is better, with FlowVisor for multiple SDN controllers. Currently, we can configure OSPF, BGP (iBGP and eBGP). For better execution time, performance and compatibility, we have changed the architecture of ACRF by removing FlowVisor and Topology Discovery module. As illustrated in the architecture of EACRF shown in Fig. 3, we employ an active method to inform the VM controller about the information of the network. After generating the Mininet topology, we call the API: topo.l1nlks(withInfo = True) to get the topology information and then send the information to the VMServer by RPC client application. Execution Time is a key metric for evaluation of EACRF. We measured the execution time in the VMServer which creates routing configuration and VMs from the moment it receives the first RPC message, to the time that all of the nodes in the network are reachable by each other. There are three stages which are as follows:

- **Stage1:** Creates the VM nodes and basic routing configuration.
- **Stage2:** Configures the links and routing protocol between VMs, restart the Quagga once it is done with the setting up of all ports on one VM.
- **Stage3:** Generates routing tables, read and install them into data plane switches.

Total time is the sum of the time of the three stages. The plot to the right of Fig. 2 shows the execution time of EACRF to configure the routing protocols in the Internet2 topology generated by Topology Generator for configuring four scenarios, namely, OSPF only, Full-Mesh iBGP, eBGP and BGP (iBGP + eBGP). The plot to the right of Fig. 4 shows the total time taken by both of the tools: Topology Generator and EACRF (in red) to generate and configure the linear topologies is considerably less as compared to the manual generation and configuration (in blue) of linear topologies.

Source code along with documentation (Readme with how to install steps) available at: https://bitbucket.org/Apoorv1986/automatic-custom-topology-generation-and-configuration

4 ACKNOWLEDGMENTS

This work was supported by Leibniz Prize project funds of DFG - German Research Foundation: Gottfried Wilhelm Leibniz-Preis 2011 (FKZ FE 570/4-1).

---

1The link to the video of the two tools: Topology Generator and EACRF is at https://tinyurl.com/mr740e3
REFERENCES