Link Colorization in SDN Simulation Animation for ns-3 Using NetAnim

Chuanji Zhang, Jared Ivey, George Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA, USA
{Jenny_Zhang, j.ivey, riley}@gatech.edu

1. INTRODUCTION

This work demonstrates a link colorization feature for network simulation animation in ns-3 with NetAnim applied to a software-defined network (SDN). Animation is an important tool for network simulation, and this new feature will improve the animation visualization and help distinguish the type of packets that are being transmitted. The currently available animators providing simulation visualization in ns-3 are PyViz and NetAnim. PyViz is a live simulation visualizer using no trace files [1]. It is mostly used for debugging purposes and cannot work without Python bindings. NetAnim [2] is an offline animator that can be used to display the topology and animate the packet flow between nodes. The packets being transmitted are expressed as arrows moving on the link between the nodes. However, the current version of NetAnim can only indicate the length of packet by varying the arrow length. There is no way to readily ascertain the packet types from the animation. It is difficult to determine which kind of packets are being transmitted during the simulation from the animation. Our work improves the functionality of NetAnim by enabling the link colorization based on the packet types.

2. BACKGROUND

2.1 NetAnim

NetAnim is based on the Qt4 GUI toolkit. It was first introduced to the ns-3 baseline for ns-3.6[2]. Enabling NetAnim visualization for a given ns-3 simulation script requires instantiation of a NetAnim `AnimationInterface` object as well as setting the position and mobility of the simulated nodes in the topology. An XML trace file will be generated when the ns-3 simulation script is executed. This trace file can be loaded into the NetAnim animator to visualize the traffic behaviors of the simulated topology. The simulated nodes are displayed optionally with their various address information, and their sizes and colors can be modified manually by the user. Traffic generation can be displayed in the context of the packets traversing the system. The user may also examine other tabs in the GUI that describe statistics and timing diagram analyses for each node in the simulated topology.

The functionality we added to ns-3 and Netanim is a link colorization mechanism. The type of packet traversing a particular link will determine which color it should be, e.g. the link color for ARP request/reply is green while the OpenFlow protocol message is purple. In order to realize this feature, we introduce three new tag classes: `RColorTag`, `GColorTag`, and `BColorTag`, containing the RGB color information of packets. The different color tags are created when packets are generated. Each time a node starts to send a packet, the animation interface will check to see if color tags are associated with this packet. If so, the color information will be written in the link description with the first-bit-transmitting time as its time stamp. NetAnim reads the XML file and if the color information is in the link description, it will extract the information and draw the link with the specified color. The link description containing the color information will not be displayed textually in the animation as other types of link description are. After the last bit of the packet finishes transmitting, the link color will return to the original black color. If no color information is associated with this packet, the link color will stay black.

2.2 Software-Defined Network

Software-defined networking (SDN) began as a means to provide centralized programmable control to the network and enable the decoupling of network control and the forwarding functions at the same time. The benefits brought by SDN, such as configuration enhancement, performance improvement, and innovation encouragement inspire the development of SDN in both academia and industry. The capabilities of SDN have been expanded beyond its initial definition. For example, controllers are able to oversee the whole network and provide a flexible and efficient platform to implement network applications and services. The logical centralized control with a global view of the whole network a feedback control can help with the network performance optimization problem.

An SDN network enables the network programmability by allowing some basic switches to examine the characteristics of the incoming traffic. These switches will perform actions, such as forward, drop, modify, and etc, based on the installed rules. These rules are generated, modified, and installed by a logically centralized controller. The controller will communicate with the switch using a standard protocol. The OpenFlow protocol is a popular standard protocol that

WNS3 2016, June 15, 2016, Seattle, Washington, USA

DOI: 10.1145/1235

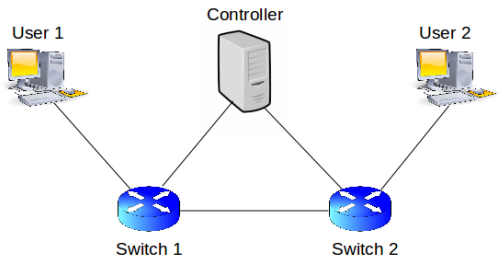

Figure 1: SDN Network Topology

enables the controller-switch communication. It is introduced as an attempt to address the issues of analyzing and testing new protocols realistically and scalably. An OpenFlow switch integrates a flow table, a secure channel, and the OpenFlow protocol. The flow table contains a set of flow entries that links different actions to different match qualifiers. The controller will communicate with the OpenFlow switch using the OpenFlow protocol in a secure channel to install and modify these entries based on the applications running on it. The Direct Code Execution (DCE), a subproject of ns-3 can be used to allow real-world controller applications to be introduced into ns-3, such as POX and Ryu. In this demo, we used a simple layer 2 learning switch controller application for testing. The layer 2 learning switch will inspect each packet it receives and learn the source-port mapping. Thus, every source MAC address it learns will be associated with the port from which the packet came in.

3. DEMONSTRATION

We demonstrate the link colorization feature in an SDN network simulation. The demonstration uses a dumbbell topology of the SDN network simulation to illustrate the link colorization functionality in NetAnim, which is depicted in figure 1. Switches 1 and 2 are controlled by the same controller and are each connected to one user. User1 will send packets to User2 through a TCP connection using the baseline ns-3 `OnOffApplication`. The link color will indicate the packet types transmitting during the whole process. Four types of packets are being transmitted during the simulation, namely messages controlling TCP state transitions, OpenFlow protocol messages, ARP request/reply, and packets containing application data. The link color for TCP control messages is red. For OpenFlow protocol messages, links will be purple. ARP request/reply messages color the links green. Payload packets cause the appropriate links to be colored blue. First, the TCP connections between the controller and switches are established by sending the three-way handshake packets. Then, the OpenFlow handshakes establish the connection between the controller and the switches. Before the TCP connection between User1 and User2 can be established, User1 and User2 should exchange ARP request and ARP reply to determine the MAC address of each other. Once the ARP reply from User2 is received, User1 initiates a three-way TCP handshake process to User2. After the TCP connection between User1 and User2 is established, the `ON_OFF` application will start to generate packets. As various packets are transmitted from User1 and User2, the OpenFlow switches will determine how to handle them based on layer 2 learning switch running on the controller. The TCP connection will be terminated once the application is done. The link color will change accordingly based on the type of packets being transmitted.

4. REFERENCES

[1] Pyviz. Website, 2015. https://www.nsnam.org/wiki/PyViz.
[2] Netanim. Website, 2016. https://www.nsnam.org/wiki/NetAnim.