Furthering Network Switch Probe Capacity of Tango, an SDN Programming System

Shenil Dodhia
Advisor: Y. Richard Yang

Motivation

Software-defined networking (SDN) offers the advantage of simplifying programming and managing complex networks through a centralized network control program. While SDN through open standards such as OpenFlow offer the benefit of providing a single unified interface to control the behavior of diverse network devices, the added abstraction of SDN could potentially lead to inefficiency due to differences in switch capabilities and performance between the various network switches.

Thus, it would be desirable to probe characteristics of network devices dynamically to learn about their capabilities and ultimately use that information to optimize network performance of SDN.

Background

Tango is an SDN programming system that is beginning to address the issues of understanding and utilizing diverse network switches. To assess each network switch’s performance, Tango uses a probing engine that makes real measurements on the switch. It accomplishes probing via Tango patterns, which are a sequence of OpenFlow flow modification commands. Specifically, Tango applies a Tango pattern to a switch, makes pre-defined measurements, and then stores the results in a central Tango Store database.

Currently, Tango has the capability to probe for TCAM size, processing throughput, and cache replacement algorithm. TCAM size is probed through looking at round-trip times (RTT) as rules are added, looking for a sharp increase in the RTT. For the cache replacement algorithm probing, Tango makes three assumptions: the algorithm depends on a subset of the parameters time since insertion, time since last use, traffic count, and rule priority; comparisons between values of a specific attribute are monotonic; flows are ordered lexicographically under a permutation of aforementioned attributes.

Probing involves using the control plane to insert dummy flow entries to probe, and then using the data plane to assess how long packet forwarding takes for every installed flow. A weakness of Tango currently is that it utilizes two extra machines besides the controller to generate the data plane traffic, which would not be ideal/realistic for real usage.
Outline of Project Proposed

This project’s chief goal is to optimize and expand the current network switch probing capabilities of Tango.

The specific improvements to be attempted include probing from just the controller. Currently, probing requires two extra machines to generate traffic through the data plane. In order to bypass the current usage of these two extra machines, a few alternative strategies exist that will be pursued. One, the network switches themselves may be used to generate the traffic, so for example, if switches A and B are connected to C, then they could be used to probe C. In this case, strategies will need to be pursued to efficiently select the switches to generate the traffic such that the probing does not have too high an overhead on the networks or dramatically impedes its performance. However, this method would have the advantage of utilizing already existing architecture and not requiring the controller to join the data plane.

Alternatively, the controller could generate the traffic itself if it were part of the data plane. This would also require novel algorithms to perform efficiently. It may simply be a constraint of the system that the controller would be required to be part of the data plane, but this is not necessarily an ideal solution.

In addition, the project will seek probing to infer the existence of multiple tables and their priorities. This includes user space software tables, kernel software tables, and the TCAM hardware tables. While the TCAM size is probed by looking at RTT as rules are added, one possibility for probing multiple tables could involve a more detailed analysis of RTT as rules are added and using a more diverse set of rules to add, some which may be more prone to end up in one type of table than another.

Yet another modification of Tango’s probing engine to be explored would be trying to infer cache replacement algorithms with fewer assumptions than are currently used. In particular, one assumption to try to not assume would be that flows are organized lexicographically under a subset of the attributes. One alternative model would be where each of the parameters used is given a weight. Hence, rather than a lexicographic ordering, a weighted score could be used to order flows. Other possible models may also exist, and while the network switches themselves are not open source, various documentation can be consulted to determine whether any other possibilities are plausible, and then to reflect the probing assumptions accordingly.

These different subgoals all tie into the bigger goal of improving the probing engine of Tango. As some of these subgoals may end up being more feasible than others, the time will be split amongst the different tasks depending on challenges that come up during the programming and design process. A common theme between the different goals is the desire to improve the thoroughness of probing—meaning probing for more detailed and accurate information. Another common theme between the goals is efficiency; as the probing engine increases in complexity, it will be increasingly important to be careful that the probing itself does not significantly hinder the network performance itself. This means that benchmarking will also be an important evaluation task to be performed, specifically benchmarking of rule installation time and delay.
**Deliverables**
- Source code, which is primarily the modifications to Tango’s probing engine, but also includes any other modifications to Tango as well as the benchmarking code
- Benchmark results of the installation times, delays
- Written final report

**References**

Yang, R. “Tango: Simplifying SDN Programming with Automatic Switch Behavior Inference, Abstraction, and Optimization”


McKeown, Nick. "Software-defined networking."
