Automatic resolution of dynamic resource conflicts between SDN Applications

Yadi Ma, Alvin AuYoung, Sujata Banerjee, Jeongkeun Lee, Puneet Sharma
HP Labs, Palo Alto, CA
{firstname.lastname}@hp.com

1. INTRODUCTION

As SDN deployment increases, a network administrator can cherry pick from a variety of different apps in the SDN marketplace to deploy a network customized to his/her unique requirements. While this SDN ecosystem has the potential to disrupt the market, the freedom to use a combination of in-house and third-party apps also introduces a critical technical challenge: how can you ensure your network remains well-behaved when running such loosely coupled apps? Similarly, how do we automatically guide the network to an operating point that attempts to simultaneously satisfy as many of the diverse goals of the *network administrator* and the apps, as possible?

To address these questions, we developed Athens, a system for *automatic detection and resolution of dynamic resource conflicts* between *black-box* SDN and cloud apps. The system comprises a programming framework for SDN apps, and a configurable coordinator that detects and mediates these conflicts on behalf of the SDN controller. The Athens coordinator sits between apps and the SDN controller, requiring apps to use the Athens API to make changes to the underlying infrastructure.

2. DEMO GOALS AND BASIC IDEA

Our demo consists of two loosely coupled traffic engineering apps running on top of the same SDN controller. We show how using Athens can detect potential conflicts between these two apps, and in the event of a conflict, resolve them by balancing the goals of each app.

Description: Consider a scenario where two SDN apps, a server load balancer (SLB) and network load balancer (NLB), are deployed, aiming to achieve both goals of balancing server load and network load, without having to refactor either of the two apps. Given a topology as shown in Figure 1, we assume traffic arrives via switch S1. SLB tries to balance server load by distributing the incoming traffic uniformly among four servers connected to switches S3 and S4. Simultaneously, NLB aims to balance network (switch) traffic by minimize the maximum load on any link. We demonstrate how these two apps interact with and without Athens.

Without Athens: A common way to manage app conflicts is to manually assign a (static) priority to each deployed app. With two apps, we have two choices. First, if SLB is given higher priority, the links (S1 to S3 and S1 to S4) on the shortest paths to the four servers will have heavier load, since absent any other goals, SLB would setup the default shortest paths to each of the four servers. In this scenario, the goal of server load balancing is met, but, the goal of network load balancing is not. On the other hand, if NLB is given higher priority, this app monitors network load and moves flows by overriding the flowtable rules installed by the SLB app. Network load balance might be achieved, but with larger overhead. Without explicit coordination between the changes made by these apps, one would have to carefully plan how to set priorities to achieve a reasonable operating point. While this is a simple example, a scenario involving more apps, each with non-obvious interactions, would be difficult to coordinate manually.

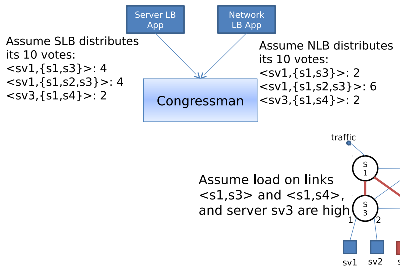

Figure 1: Demo with two SDN apps.

With Athens: When using the Athens API, SLB is encouraged to propose multiple possible paths (i.e., it has no preference for the default or more load-balanced path). Moreover, it can propose multiple target servers (i.e., in the evaluation phase, it can express its preference for more load-balanced assignments). Upon receiving proposals, Athens checks for conflicts and sends conflicting proposals to apps that are registered for the path resource (SLB and NLB in this case) for evaluation. Without loss of generality, assume that SLB proposes server sv1 and sv3 as destination servers, and three different paths for the incoming flow (Figure 1). In the evaluation round, both SLB and NLB have an opportunity to express a relative value for the various flow-path assignments. In this example, the second proposal with the lightly loaded server (sv1) and the lightly loaded path (s1,s2,s3) gets the largest cumulative votes from the two apps and is chosen to be deployed on the network. By encouraging apps to increase the possible allocation space (via multiple proposals), and express their relative value for each, Athens makes it more likely that a dynamic resource conflict between multiple applications will result in a jointly (near) optimal allocation.

Equipment and URL: We need space for laptops, and need wireless Internet access. This work is based on Athens: `http://conferences2.sigcomm.org/co-next/2014/CoNEXT_papers/p391.pdf`.