Assignment 3: ECMP Routing

Deadline:14/05/2018

1 Introduction

CISCO white paper on ECMP [1] describes it as following. Equal-cost multi-path routing (ECMP) is a routing strategy where next-hop packet forwarding to a single destination can occur over multiple "best paths" which tie for top place in routing metric calculations. Multi-path routing can be used in conjunction with most routing protocols, because it is a per-hop decision limited to a single router. It can substantially increase bandwidth by load-balancing traffic over multiple paths; however, there may be significant problems in deploying it in practice.

ECMP Per-Flow Load Balancing distributes packets across multiple links based on Layer 3 routing information. If the router discovers multiple paths to a destination, the routing table is updated with multiple entries for that destination.Per-flow load balancing allows the router to use multiple pathsto achieve load sharing across multiple source-destination host pairs. Packets for a given source-destination host pair are guaranteed to take the same path, even if multiple paths are available. Traffic streams destined for different pairs tend to take different paths.

Consider the example tutorial in Figure 1. Using ECMP, the first flow from node A to B will follow path $A \rightarrow S1 \rightarrow S0 \rightarrow S5 \rightarrow B$, the second and all later flows will follow the same path since there is only one shortest path. Communications from A to C will be different: the first flow from A to C would be randomly chosen from the two equal-length path, let us assume that $A \rightarrow S1 \rightarrow S0 \rightarrow S5 \rightarrow S4 \rightarrow C$ is selected. The second flow will follow the other path $A \rightarrow S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \rightarrow C$. The third flow will again use $A \rightarrow S1 \rightarrow S0 \rightarrow S5 \rightarrow S4 \rightarrow C$ and so forth.

2 Instructions

In this assignment you will use the NetworkX Python library [2] for generating a random graph, and for finding shortest paths. You should use the Erdös-Rényi model for generation a random graph $G_{n,p}$. A random graph $G_{n,p} = (V, E)$, is a graph with the vertices

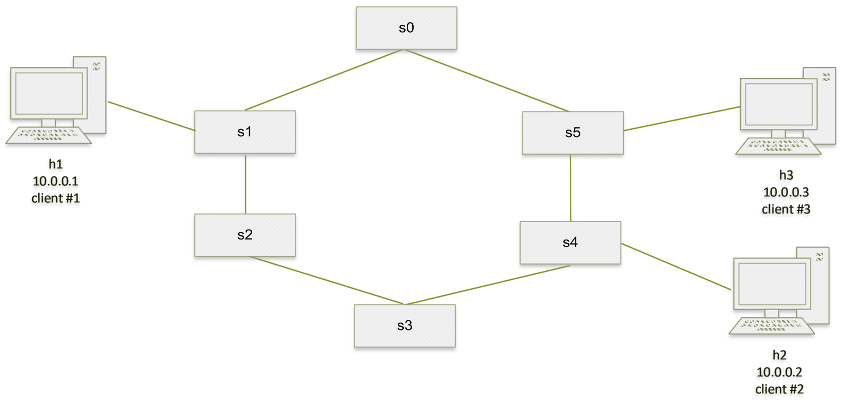

Figure 1: ECMP Example

set V of size n, and an edge set E of size m, such that each possible edge has probability p.

Use the NetworkX package in order to create a random graph based network topology. Choose at random at most 50% of switches to have a host connected to it.

Create an ECMP controller that will utilize the ECMP routing for each new IP flow. For example, assume that h_1 sends a flow (TCP destination port 80, for example) to h_{10}. Let us assume for the sake of example that h_1 is connected to s_1 and h_{10} is connected to s_{10}. The switch s_{10} will send a *PacketIn* message to the controller. The controller will calculate a shortest path to h_{10}. If h_1 initializes a new flow to h_{10} (TCP destination port 22, for example), the controller will select a different shortest path.

3 Running Your Application

Write your application as a .py file(s) under ~/pox/ext (name it ECMPBalancer.py). You may assume that the controller is fully aware of the topology. For accomplishing this, it is recommended to serialize your topology into a formatted file (e..g, JSON, XML, binary, etc.) which the controller can open and parse.

The recommended way for showing that load balancing is working, is to use iperf [3] for generating TCP, UDP, or SCTP flows for several times, and showing that different paths were selected for each new flow.

4 Submission Instructions

Please submit your assignment **BEFORE 23:50 May 14th, 2018**. Submissions after the deadline will not be graded.

Attach a submission comment file which includes short description of how to run your code and full names with IDs of your team members. Specify, who from the team is a corresponding member if a correspondence is required.

ZIP all your code, files, and documents (if any) into a single file named:

```
assignment2_sdn_%MEMBER_ID%.zip
```

where %MEMBER_ID% is the ID of the corresponding member.

Submissions in any other name formats will be penalized. You are strongly encouraged to submit your first draft of your assignment as soon as possible. Later (but before the deadline) you can submit the final version of your assignment. Only the last submission will be graded.

This script should work exactly as expected to get full grade. Points will be subtracted for missing or incorrect functionality. The penalty for late submission is 10% per day. The submitted code will be tested for plagiarism. Any attempt to plagiarize will be accordingly punished.

Before contact, please make sure that you have formulated your question clearly and that you have already studied the POX wiki, the Mininet documentation, and the OpenFlow tutorial thoroughly.

GOOD LUCK!

References

[1] CISCO Documentation "ECMP Load Balancing", Available online

[2] "NetworkX:Software for complex networks", https://networkx.github.io/

[3] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP", https://iperf.fr/