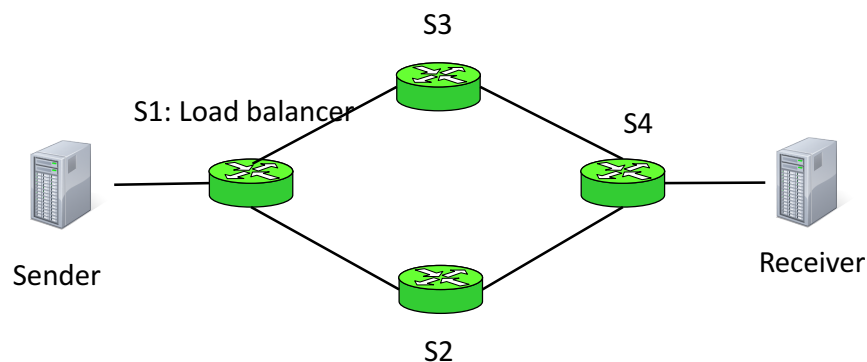


Homework Two: Programmable Networks

* Milestone 1: Build a topology for multipath routing

- **Task 1:** Building a topology with two servers and four switches in Mininet, where there are two alternative paths between the servers. (Please use the VM image provided by the TA)



- **Task 2:** Write a P4 program that performs ECMP load balancing at the first hop (S1), and destination-based forwarding for all other hops (S2-S4).

Note 1: You can use the same program for all switches. The sender can tag packets with a special header to indicate that they are on the first hop. Upon seeing this special header, S1 will perform load balancing; S1 can remove this header after processing, so S3—S4 will perform destination-based forwarding. (You can also load different programs to different switches, but this would involve a more complex configuration.)

Note 2: You can refer to the program examples in the P4 tutorial repo in the VM. However, please write your own programs and don't copy code from there.

- **Task 3:** Add monitoring capability to the P4 program, so that a switch can keep track of the number of bytes sent to each of its outgoing ports. Implement a special “query” packet (e.g., one with a special protocol field) to collect statistics from the data plane (e.g., after all packets have been processed by the P4 programs). (You can also write a control plane program to query these statistics, but this may be more complicated to implement.)

* Milestone 2: Understanding the limitations of ECMP

- **Task 1:** Generate random flows using scapy (as defined by the five tuples), with different sizes; and test how well ECMP load balances between the two paths (i.e., how many bytes are sent to the upper path vs. lower path). Report the total amount of traffic sent by the source, and the amount of traffic sent via each path.
- **Task 2:** Develop a per-packet load balancing mechanism in the P4 program, and test the load balance ratio again. Report the same set of numbers as Task 1.
- **Task 3:** Per-packet load balancing may result in out-of-order packet delivery. Quantify the amount of out-of-order packets per flow.

Note 3: There are multiple ways of doing this; for instance, you could embed a special counter in the packet header as the packet is sent at the source, and then collect the sequence of packets at the destination. Suppose you have a sequence a_1, a_2, \dots, a_k , then you can quantify the amount of reordering by computing the number of inversions in the sequence: <https://leetcode.com/problems/global-and-local-inversions/>

Note 4: You may need to configure different link latencies (e.g., for link S1-S2 vs. link S1-S3, so that they incur different amounts of forwarding delay) if you cannot observe reordering using the same link latency. Mininet is based on simulation, so there may be variances across runs and environments.

* Milestone 3: Implement flowlet switching (Only required for graduate students)

- **Task 1:** Implement flowlet switching on S1; other switches still use destination-based forwarding to their only next hops.
- **Task 2:** Quantify the amount of out-of-order packet delivery per flow. Compare the findings with those in per-packet load balancing.

Homework submission: Please directly compress the working directory for your P4 programs, topology files, command files, and other files that are necessary to build your programs as a .tgz file. Provide a PDF file describing the results, and put them in the same tarball. In your PDF file, please include a) your name, b) your Rice NetID, and c) your student ID (that starts with 'S'), and of course, descriptions for each milestone and results. Email your .tgz file to the instructor and all TAs.

Due date: October 6. **Late policy:** -20% of the scores for each late day.