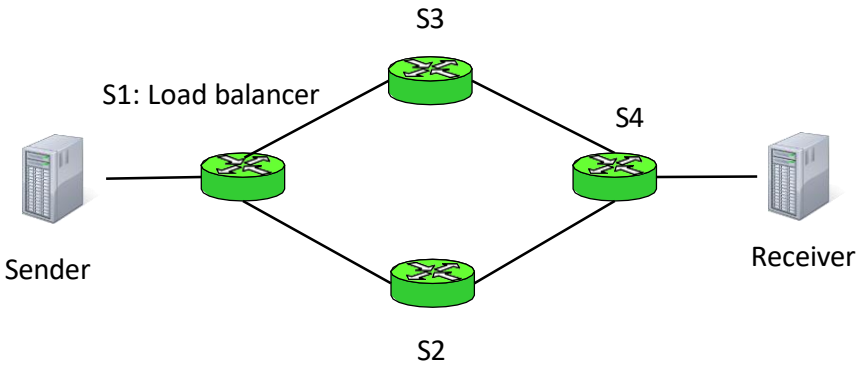
Homework Two: Programmable Networks

* 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 we used for the P4 hands-on lab in class. You can set it up following the instructions here (option b): https://github.com/jafingerhut/p4-guide/blob/master/bin/README-install-troubleshooting.md)



- **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. Per-packet load balancing here means for each packet, there is a chance to change the routing path. 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 a1, a2, …, ak, 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.

  Note 5: The high-level idea of flowlet switching is similar to ECMP, but it is on the basis of a flowlet. To do that, you will need to record the timestamp of a flowlet using P4 register arrays. Only when the time interval between two flowlets is larger than the path delay, you can change the routing path.

  Note 6: To test flowlet switch, you need to create flowlets in scapy, e.g., by adding delays between some packets using sleep.

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 .zip file. Provide a PDF file describing the results, and put them in the same .zip file. *This pdf file should include descriptions about every milestone and its tasks that you have accomplished, and which programs/configurations/.. belong to each milestone. Also describe the results you've obtained for each milestone.* For instance:

*** Example format of the result description:**

--Beginning of PDF file
* Milestone 1:
** Task 1: topology.json describes the topology.
** Task 2: ECMP.p4 implements the load balancing algorithm.
** Task 3: Lines XX—YY add the monitoring capability, and the commands C1, C2, and C3 can be used to read the data plane monitoring results.
** These tasks can be run by a command run_milestone1.sh

*Milestone 2, 3: …
--End of PDF file

Due date: 10/6/2025 11:59pm. **Late policy:** -20% of the scores for each late day.