An Elastic Distributed SDN Controller

UCSD CSE Computer Science and Engineering

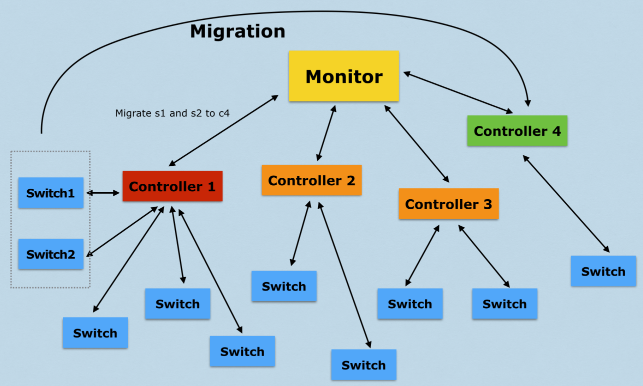
Guo Li, Xinyu Zhang, Liqiong Yang

Problem & Idea

Distributed SDN (distributed control plane) can provide with scalability and reliability

However, current mapping between a controller and a switch is statically configured

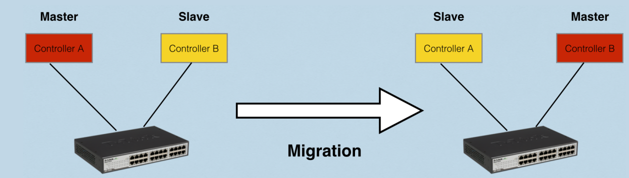
- **PROBLEM:** This can not adapt to traffic load variation
  - Controllers are not balanced, e.g. some are overloaded while others are idle
- **IDEA:** Real time monitoring & dynamic load balancing -> migration[1]
  - Monitoring logic: decide when to trigger the migration
  - Migration logic: migrate a switch from a heavily loaded controller to a lightly loaded one
- **Requirement:** No packet will get lost during the migration process

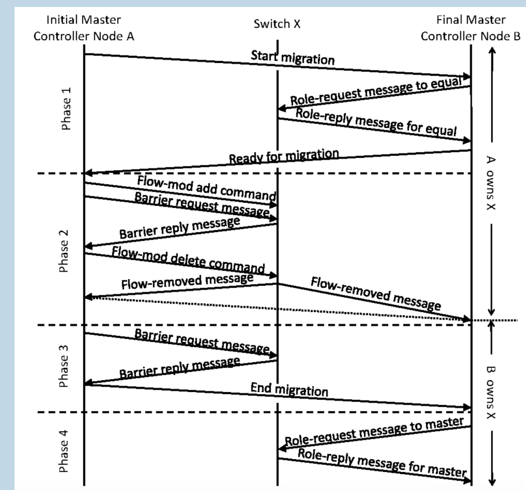


1. Dixit, Advait, et al. "Towards an elastic distributed SDN controller." *ACM SIGCOMM Computer Communication Review*. Vol. 43. No. 4. ACM, 2013.

Protocol

- **OpenFlow controller can be three different modes: Master, Equal, Slave.**
- **Migration of a switch is the process of changing its controllers' mode.**

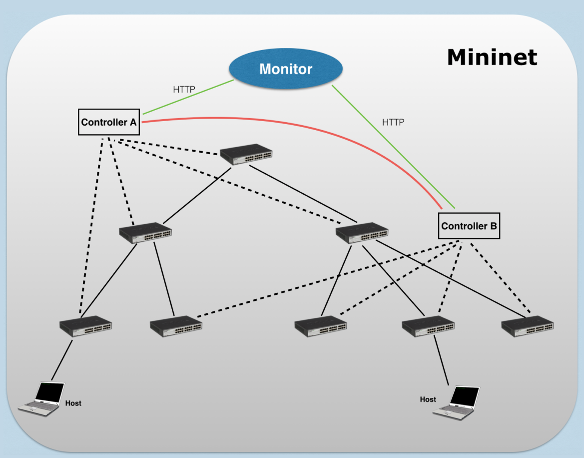




Protocol detail

- Phase 1: Change controller B from Slave to Equal
- Phase 2: Insert and remove a dummy flow
- Phase 3: Controller A finishes all the pending requests
- Phase 4: Controller B request to become Master

Implementation

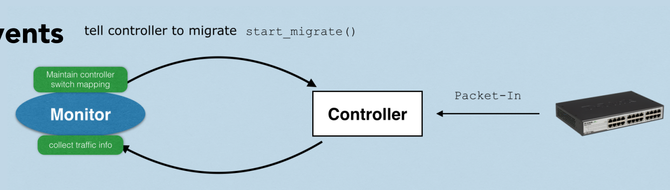


We implement the system on top of Mininet

- The network consists of 8 virtual switches, each linked to two hosts. There are two controllers in the system, each "connected" to 4 switches
- Monitor uses HTTP request to talk to controllers, controllers also use HTTP request to talk to each other
- Monitor only sends request to controllers, controllers will coordinate themselves to finish migration

Monitor maintain status and trigger events

- Controller periodically send traffic information to monitor
- Monitor collect and compare traffic information, and decide whether to trigger a migration event or not

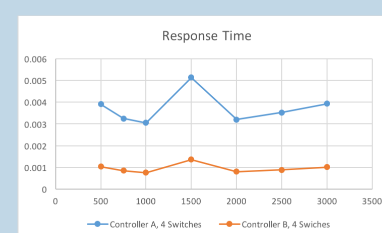


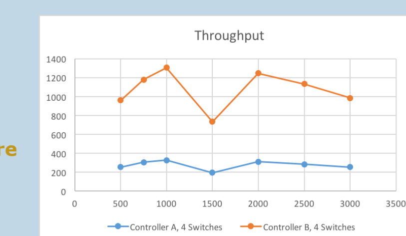
Evaluation

We test the Response Time and the Throughput of controllers

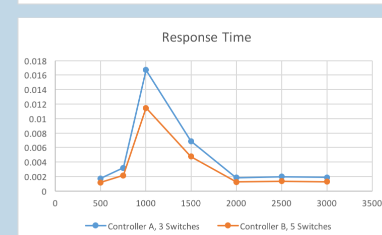
- We use `ping` to generate network traffic with different package rate, use WireShark to capture packet.
- We compare the differences in response time and throughputs of controllers before and after migration

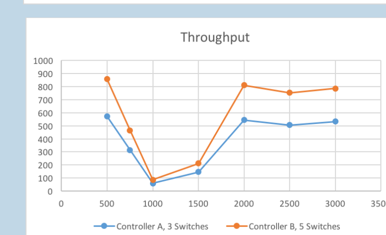
Before





After





After migration, not only does the load become more balanced between two controller, both throughput and response time have a clear improvement