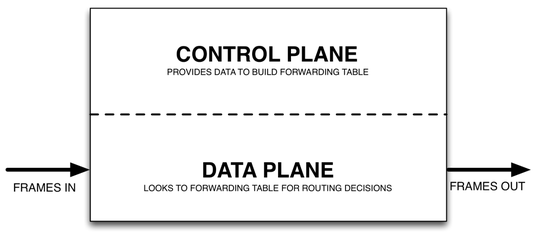
Migration of OpenFlow Enabled Switches Between SDN Controllers

Anuradha Kannan, Prarabdh Joshi
Department of Electrical Engineering, San Jose State University, San Jose, California 95192

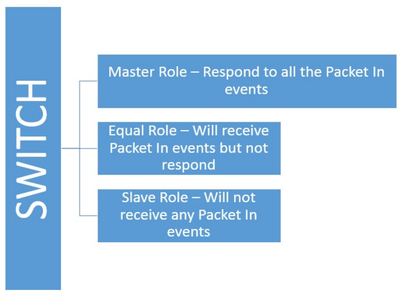
Introduction

Software Defined Networking is becoming quite popular among the researchers because it enables dynamic network programmability and ease of management. This reduces the development cycle and augments innovation.



The idea behind Software Defined Networks is to decouple the control plane and the data plane. It assumes a centralized control plane responsible for the orchestration of the entire network. Obviously, having a single controller in your network raises scalability and redundancy concerns as a single controller can't handle more than a certain amount of load and becomes a single point of failure for the entire network. So it becomes clear that we need a distributed control plane in order to make SDN scalable. Also, the traffic in the network depends on both time and location. So a need arises to dynamically shrink and expand the controller pool along with the ability to migrate switches between different controllers for load balancing purposes [1]. In this project, we have attempted to develop a controller-controller communication mechanism which will initiate/co-ordinate the migration and ultimately achieve seamless migration of a OpenFlow enabled switch between different controllers.

Roles of Controller



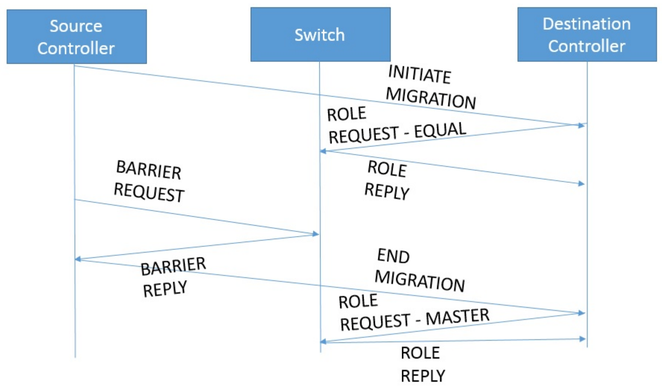
OpenFlow 1.3 introduces the concept of role of a controller with respect to an OpenFlow [1] enabled switch which has become crucial for migration.

Master Role – There can be only one master for a switch which will be responsible for all the events corresponding to that switch.

Equal Role – There can be more than one controllers with an Equal Role for a switch. These controllers will receive but not respond to any of the events.

Slave Role – There can be more than one controllers with this role but they will not receive any events (except Echo Request and Echo Reply)

Switch Migration Mechanism



Key Points

- Role Request message can be used to change the controller's role for a switch
- Barrier Request message only receives a reply when all the previous messages have been processed

Switch Migration Mechanism:

Source Controller: The controller who initiates the migration or is asked to initiate the migration by another layer of abstraction above it. This controller will be the master for the switch prior to migration.

Destination Controller: The controller with which the switch will be mapped to after the migration has successfully completed. This controller will be the master for the switch after migration.

PHASE 1 : INITATE MIGRATION
The destination controller will receive an INITIATE MIGRATION [2] message along-with the information needed for the migration to proceed such as the datapath id of the switch. As soon as the controller receives this message, it sends a ROLE REQUEST message to the switch asking the switch to change its role to an Equal. The switch responds with the ROLE REPLY message and from this point on the destination controller starts receiving all the events for that switch.

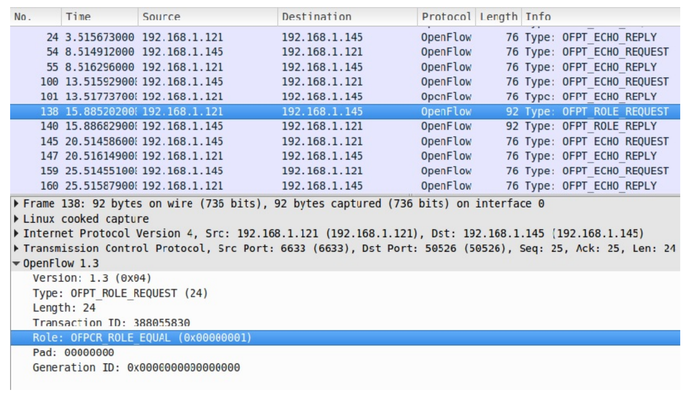
PHASE 2 : BARRIER REQUEST
After sending the, initiate migration message, the source controller will send the switch a BARRIER REQUEST [2] message. The functionality of this message is crucial to this mechanism as after this message is received by the switch, it will not send a BARRIER REPLY unless it has received a response for all the pending packets that it has sent to the controller.

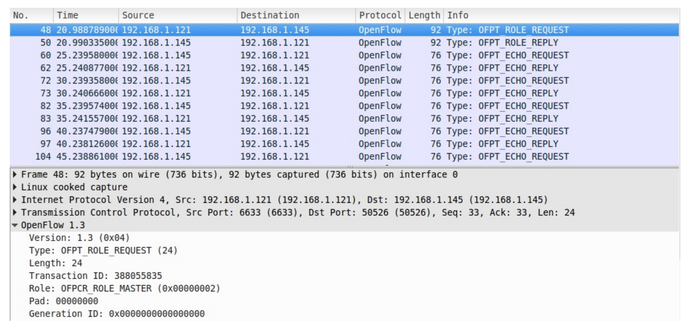
PHASE 3 : END MIGRATION
After receiving the BARRIER REPLY, the source controller will send the END MIGRATION message to the destination controller indicating that it is safe to change the mapping of the switch now. Upon receiving this message, the destination controller will send a ROLE REQUEST message to the switch asking the switch to change its role to master.

Implementation

In order to implement the migration mechanism, we have made use of the Ryu controller [3] platform and Mininet. We have embedded a web server inside a RYU controller which will basically keep listening for incoming HTTP messages on a particular port number. We assume that all the controllers know this static port number prior to migration. Our partial implementation provides all the functionalities that a destination controller needs to perform for the migration.

In order to verify the implementation, we have used a basic curl utility to send the HTTP messages to the destination controller. As you can see below, the destination controller successfully sends the corresponding ROLE REQUEST messages upon receiving INITIATE MIGRATION and END MIGRATION commands.





Conclusion

We have shown one of the ways to achieve seamless migration for an OpenFlow enabled switch between controllers in a distributed controller plane. We believe that optimizing the migration process is a key factor needed for SDN to be accepted as an industry standard in future.

Future Scope

- Secure Communication between controllers
- Design an API for initiating and monitoring the migration for higher level orchestration tools such as OpenStack

Key References

[1] Advait Dixit, Fang Hao, Sarit Mukherjee, T.V. Lakshman, RamanaKompella, "Towards An Elastic Distributed SDN Controller, " in HotSDN, 2013

[2] A. Valdivieso, L. Barona, and L. Villalba, "Evolution and challenges of software defined networking," in Proceedings of the 2013 Workshop on Software Defined Networks for Future Networks and Services, pp. 61–67, IEEE, November 2013

[3] Z. Cai, A. L. Cox, and T. S. E. Ng, "Maestro: A system for scalable OpenFlow control," Tech. Rep. TR10-11, CS Department, Rice University, Dec. 2010

[4] Open Networking Foundation, "OpenFlow Switch Specification (Version 1.3.0)," June 2012

Acknowledgments

We would like to thank Dr. Tarng for his encouragement and support during the course of this project.

For further information

Please contact *prarabdh9909@gmail.com,anuradhakannan7@gmail.com*