A Model for Building SDN Control Plane with Multiple Controllers
draft-zhao-opsawg-model-sdn-multi-controller-00

Abstract

This document describes a reference model for building SDN control plane with multiple controllers. It defines the key components in the model and introduces an effective mechanism to balance the load between controllers. The design also considers the tradeoff between scalability and performance.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on December 16, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Introduction ................................................ 2

2. Terminology ................................................ 3

3. Multi-server Controller Framework ........................... 3

4. Switch-controller Interaction................................ 4

5. Design Issues ............................................... 5

5.1. Load balance ........................................... 5

5.2. Traffic Mapping ........................................ 5

6. Security Considerations...................................... 6

7. IANA Considerations ........................................ 6

8. References .................................................. 6

8.1. Normative References.................................... 6

8.2. Informative References................................. 6

1. Introduction

The centralized design of control plane in software defined networking(SDN) brings the benefit of network programmability. The design allows a logically centralized controller to control a number of network switches and thus enables more flexible control with global information. However, the centralized design also leads to scalability concerns. With the increase in flow number and network size, controllers will experience performance bottleneck. Therefore, it is critical to scale up the performance of controller for real-world SDN deployment.

The document defines a reference model and associated interconnection mechanism for building a controller using server clusters. The model integrates commodity servers and OpenFlow switches into a powerful logically centralized controller. On one hand, a number of OpenFlow switches are used to serve as a highly scalable front-end switch group, which can handle high-throughput data traffic. On the other hand, a number of commodity servers are deployed to serve as the controller cluster, which leverages the parallelism of cluster to balance the control plane load. Each server in the cluster thus deals with partial load from the control plane by a load balancer.
2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

SDN Software Defined Networking

3. Multi-server Controller Framework

With the increase in network scale and traffic flow, the control plane load is expected to increase significantly. In order to make the controller scalable, the design principle is to balance the control plane load across multiple OpenFlow controllers. A cluster of loosely or tightly connected servers that work together, can be regarded as one logically single controller. Each controller in the cluster deals with partial load from the control plane by a load balancer [2].

The multi-server controller reference model, as shown in figure 1, identifies the components that a controller cluster includes. Being an integrated architecture, the model takes the advantage of commodity server cluster and works as one logically centralized controller.

According to the need for scaling up the performance of controller, the multi-server controller reference architecture includes:

- **Server Cluster**
  A pool of OpenFlow controller units (commodity servers) are organized to share the controlling overhead of the logical controller.

- **Switch Group**
  A set of OpenFlow switches are employed at the front-end switch group, which servers as the I/O interface between the controller cluster and the network being managed.

- **Load balancer**
  To schedule the internal resources of the integrated controller, we introduce an load balancer. It manages both the Switch Group and the Controller Cluster. The load balancer actively monitors the CPU and I/O resources of the servers in Controller Cluster and the switches in Switch Group.
Global View

Though the servers in Controller Cluster work independently from each other, they share a common view of the control information, such as network topology, routing information, and the link status of the OpenFlow network. To synchronize the data among these servers, we introduce a network information database. Each server needs to report the latest control information to this database.

Figure 1 Controller cluster framework

The model takes the advantage of commodity PC server cluster and works a fully functional OpenFlow controller with high scalability. It can serve a number of client switches for a large-scale network.

4. Switch-controller Interaction

The typical interaction between the client switch and the multi-server controller is handling the packet-in message. The procedure works as follows.
1. According to the resource usage and the pre-defined resource allocation policy, the load balancer will first select a proper controller and redirect the packet to the controller.

2. When the packet reaches the given controller server in the cluster through switch group, the controller decodes the packet, and then processes it. The controller also needs to update the global view database accordingly.

3. If some feedback is generated, the controller will send it to the corresponding client switch.

5. Design Issues

5.1. Load balance

OpenFlow defines three roles for controller: master, slave and equal. Master and slave are usually used for fail-over or fast recovery. In the multi-server architecture design, controllers are configured to share the workload by setting the role as equal. With the role of equal, the controller can receive all the asynchronous messages from switches and send commands to switches.

The objective of a load balancer for OpenFlow network consists of the following aspects:

1. Balancing CPU workloads: The CPU-intensive applications, such as maintaining topology, querying the network configurations, and computing the minimum spanning tree or shortest path, may consume a large amount of CPU resources. High utilization can be achieved by allocating these CPU-intensive applications to a controller which has the lowest CPU usage.

2. Balancing I/O workloads: A controller generally involves periodical operations such as maintaining connection between controllers and switches, sending and receiving control commands for modifying switches. Though simple, these frequent I/O intensive operations will also result in a large amount of I/O resource consumption. In order to reduce the load on a certain controller, these I/O-intensive operations can be shared between different controllers by using the load-balancing structure.

5.2. Traffic Mapping

In the original OpenFlow design, each OpenFlow switch is connected to one single OpenFlow controller. In our design, cCluster consists of a number of controllers for scalability consideration. If a new
controller is added to the cluster, the load from client switches must be able to redirected to the new one for load balancing.

However, existing OpenFlow specifications do not support such feature. Therefore, a middle layer with the capability of flexible traffic mapping is needed in order to enable redirecting traffic from any OpenFlow switch to any controller in the cluster.

In the reference model, the middle layer is actually an internal SDN consists of two parts: Switch Group, and load balancer as shown in Figure 1. Switch group interconnects the controller cluster and the external client switches. With OpenFlow, the load balancer can measure the load of each OpenFlow controller, and then change the connection status adaptively in the middle layer.

6. Security Considerations

This document discusses a reference model for implementing SDN controller using multiple servers. The solution discussed may have security implications, such as control flow saturation attacks, unauthorized access to the global view. Framework described in this document is limited to the architecture design. Security considerations regarding specific attack can be addressed in their respective domains.

7. IANA Considerations

This memo does not have any IANA considerations.

8. References

8.1. Normative References


8.2. Informative References

Authors’ Addresses

Jin Zhao
Fudan University
825 Zhangheng Rd., Shanghai 201203, China
Email: jzhao@fudan.edu.cn

Yanwei Xu
Shanghai Engineering Research Center for Broadband Networks and Applications (BNC)
150 Honggu Rd., Shanghai 200336, China
Email: ywxu@bnc.org.cn