OpenFlow Controller Architecture for Large-Scale SDN Networks

CHIBA Yasunobu, SUGYOU Kazushi

Abstract

When applying centralized network architecture such as OpenFlow to large-scale networks, the issue arises of how to ensure the scalability and reliability of the controller providing the network control functions. This paper proposes a new controller architecture that leverages design patterns widely used in client-server systems to perform network control. Moreover, by applying this architecture to an OpenFlow controller providing virtual layer 2 networks for data centers, it shows the feasibility of applying an OpenFlow controller to large-scale SDN networks.

Keywords

OpenFlow, OpenFlow controller, scalability, load balancer pattern, three-tier architecture

1. Introduction

As a component technology for enabling SDN (Software-Defined Networking), OpenFlow[1)] is equipped with a software-based OpenFlow controller to provide control functions. This allows a user to flexibly define the functions provided by the networks consists of OpenFlow switches.

Various problems arise when OpenFlow is applied to practical networks. The following points in particular occur as typical problems when it is applied to large-scale networks (**Fig. 1**).

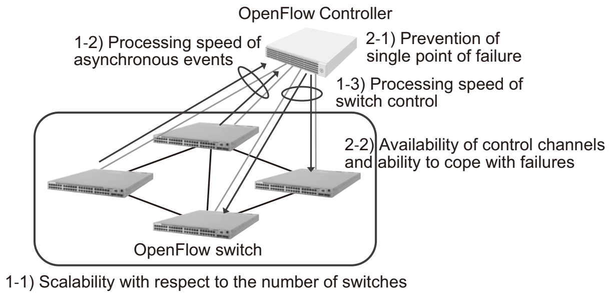
OpenFlow Controller
1-2) Processing speed of asynchronous events
2-1) Prevention of single point of failure
1-3) Processing speed of switch control
2-2) Availability of control channels and ability to cope with failures
OpenFlow switch
1-1) Scalability with respect to the number of switches

Fig. 1 Capabilities required when applying OpenFlow to large-scale networks.

(1) Controller performance and scalability

1) Scalability of the number of controllable switches
2) Processing speed of asynchronous events that are generated by switches and notified to the controller
3) Processing speed of switch control events that are proactively trigged by the controller

(2) Reliability and fault tolerance

1) Required to design a controller system that does not make the controller the single point of failure
2) Required to design a system that ensures the availability of control channels between the controller and switches and can continue to operate even during failure

This paper focuses mainly on performance and scalability among the above-mentioned problems and shows solutions to them. Concretely speaking, we attempted to solve these problems by applying design patterns widely used in client-server systems to the OpenFlow controller, while assuming the provisioning of virtual layer 2 networks for large-scale data centers. In the following section, we report their solution methods as well as performance evaluation results based on a proof-of-concept implementation.

2. Proposal Method

2.1 Prerequisites

The network functions provided by an OpenFlow controller and the operation of the controller to achieve the network functions in question can be freely defined by implementing the controller as a software application.

It is therefore impossible to generalize the assumed operation of the controller and decide on architecture applicable to all controllers. For this reason, this study assumes a controller that provides virtual layer 2 networks in a multi-tenant environment at large-scale data centers, while assuming the controller environment is as follows:

(1) Switch control is executed proactively and subjectively by the controller based on the settings and conditions defined other than the controller.

(2) Switch control triggered by asynchronous events generated by switches is rarer than (1).

(3) Switch control is highly independent of the switches and serial control of multiple switches accompanied by dependency rarely takes place.

In addition, this study made a qualitative and quantitative evaluation of the following points:

(1) Performance/scalability

The capability to ensure high performance and scalability for the above-mentioned assumed controller operations should be considered the minimum requirements.

(2) Design/implementation ease

Importance was attached to the capability to implement software easily, with low system complexity, and the ability to take advantage of existing software assets.

(3) Administrative ease

In addition to ease of software implementation, importance was also attached to aspects of system management. In addition, consideration was given to whether or not existing management know-how and systems could be reused.

(4) Reliability and fault tolerance

Although the central aim of this study was the assurance of performance and scalability, consideration was also given to the reliability and fault tolerance of the controller.

2.2 Approach

As a result of the evaluation, we designed OpenFlow controller architecture that leverages the load balancer pattern that is a design pattern for scaling out of client-server systems and the three-tier architecture which is one of the client-server architecture (**Fig. 2**).

By employing the load balancer pattern, the required controller performance can be obtained by increasing or de

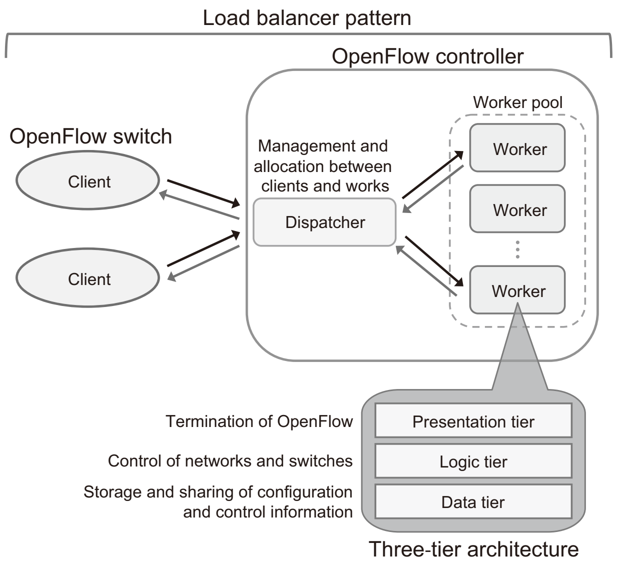

Fig. 2 Proposed architecture.

creasing the number of workers according to the size of the network. Thanks to its high ease of design and implementation and its frequent usage in Web systems, existing design and management know-how and software can be reused. The dispatcher accepts a request to establish a control channel from the OpenFlow switches and allocates their connections to the workers based on criteria defined in advance. The workers establish control channels according to the allocations from the dispatcher. Furthermore, they process the asynchronous events from the OpenFlow switches which are the clients in the load balancer pattern architecture, and then response as needed.

The workers are composed of architecture similar to the three-tier architecture. It consists of a presentation tier that terminates the OpenFlow protocol, a logic tier that executes control processing using the OpenFlow protocol and a data tier that stores control information and shares it among multiple workers. Sharing with other workers is all performed via the data tier. In the data tier, settings and control information given from the external system to determine the operation of the workers can be retained in addition to sharing their operation status.

The difference between the ordinary load balancer pattern/three-tier architecture and the architecture under investigation in this study is that workers sometimes control the switches proactively. This architecture is designed so that the workers corresponding to the switches execute proactive control for the switches once the corresponding relationship between switches and workers is established. The workers confirm the switch settings and control information stored in the data tier, whether or not an asynchronous event is generated by the OpenFlow switches corresponding to the client, and execute switch control as required.

3. Implementation

In order to evaluate the effectiveness of our proposed architecture, we implemented and evaluated an OpenFlow controller based on it.

3.1 Functional and Performance Requirements

The requirements for controller functions and performance as well as for the network to be managed and controlled are as follows (**Fig. 3**):

(1) Functional requirements

1) Conducting the provisioning and management of virtual layer 2 networks based on VXLAN[2]
2) Conducting management of mappings between OpenFlow switch ports and virtual networks
3) Conducting management of mappings between OpenFlow switch ports and the MAC addresses of the hosts connected to the ports
4) Providing an interface for the above-mentioned operations in the REST style

(2) Performance requirements

1) Capability of managing more than 1,000 OpenFlow switches
2) Capability of creating more than 10,000 virtual networks
3) Capability of connecting more than 10,000 end hosts to the virtual networks

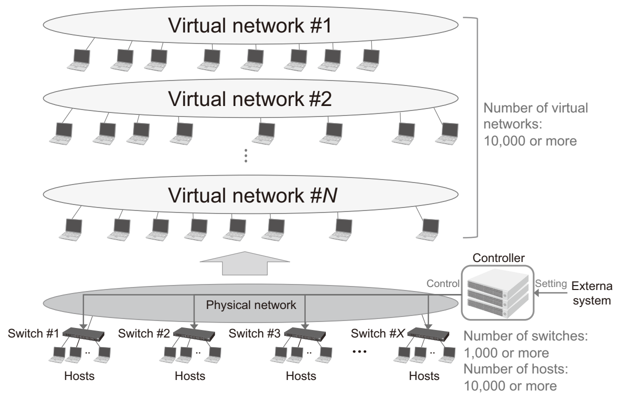

Fig. 3 Controller requirements.

3.2 Implementation

The design of the implemented controller is shown in **Fig. 4**.

The dispatcher was implemented by using the layer 4 load balance function of the LVS (Linux Virtual Server)[3]. Additionally, Keepalived[4] was used for the settings of the LVS and health checks of the workers. The workers were composed of Trema[5], which is an OpenFlow controller and switch development framework, our newly developed virtual network manager and a back-end database using MySQL[6]. Because an interface for configuring the virtual network needed to be provided, a function to store the settings in the back-end database was implemented as a configuration front-end.

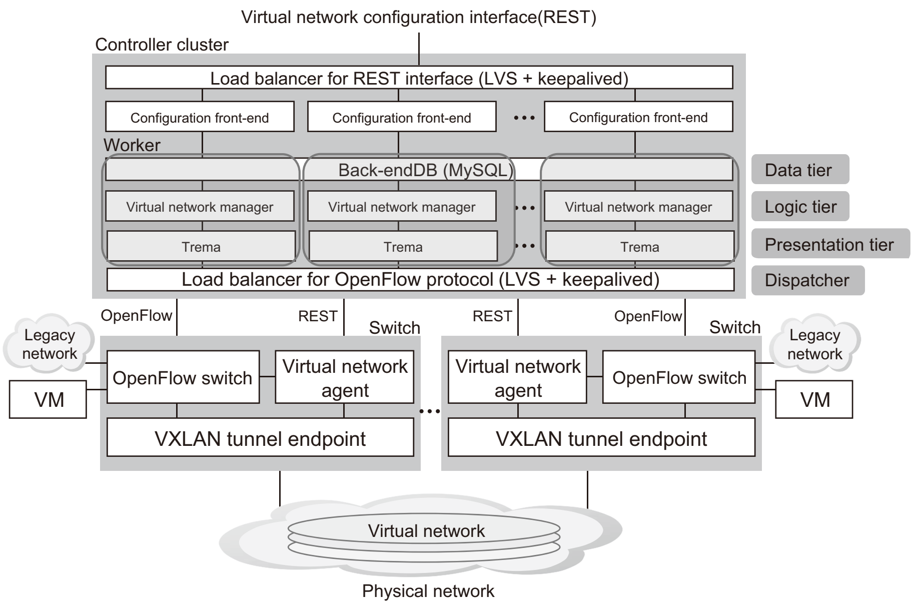

Fig. 4 Controller design.

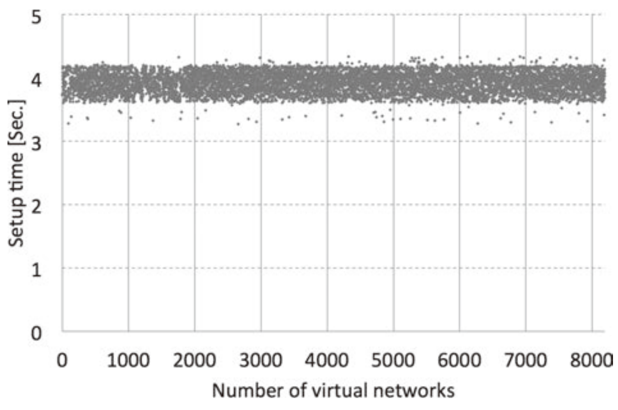

Fig. 5 Virtual network setting performance.

4. Evaluation

(1) Scalability with respect to the number of switches
Confirmation was made that approximately 410 switches could be allocated to a single worker and that control of all the allocated switches could be executed. Confirmation was also made that the number of controllable switches in a system increased in linear proportion to the number of workers.

(2) Scalability with respect to the number of virtual networks
Confirmation was made that more than 16,000 virtual networks could actually be created and managed under the condition in which there were 1,024 switches and 128 hosts (VMs) connected to each switch.

(3) Scalability of the settings and performance of the virtual networks
Confirmation was made that the time required to create and configure each virtual network was constant, independent of the total number of virtual networks (**Fig. 5**).

5. Conclusion

We have discussed OpenFlow controller architecture that enables large-scale SDN networks and have given an example of its implementation. The results of this study are applied to our commercial service at NEC BIGLOBE. We are committed to applying SDN to wider fields by continuing our efforts to investigate performance improvement and scalability assurance for controllers with diverse requirements.

* OpenFlow is a trademark or registered trademark of Open Networking Foundation.
* Linux is a registered trademark of Linus Torvalds in Japan and other countries.
* MySQL is a trademark of Oracle Corporation and/or its affiliates.

Reference

1) OpenFlow - Open Networking Foundation
https://www.opennetworking.org/sdn-resources/onf-specifications/openflow
2) VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks
http://tools.ietf.org/id/draft-mahalingam-dutt-dcops-vxlan-06.txt
3) The Linux Virtual Server Project
http://www.linuxvirtualserver.org/
4) Keepalived for Linux
http://www.keepalived.org/
5) Trema Full-Stack OpenFlow Framework in Ruby and C
http://trema.github.io/trema/
6) MySQL: The world's most popular open source database
http://www.mysql.com/

Authors' Profiles

CHIBA Yasunobu
Senior Research Engineer
Knowledge Discovery Research Laboratories

SUGYOU Kazushi
Principal Researcher
Knowledge Discovery Research Laboratories

The details about this paper can be seen at the following.

Related URL:

NEC BIGLOBE Virtualizes the BIGLOBE Data Center Using State-of-the-Art OpenFlow Technology
http://www.biglobe.co.jp/en/press/2013/0422.html

Vol.8 No.2 SDN and Its Impact on Advanced ICT Systems

Vol.8 No.2
April, 2014

Special Issue TOP