SDN-Based ECMP Algorithm for Data Center Networks

Hailong Zhang
School of Information Engineering,
Communication University of China
Beijing, China, 100024
zhlcuc@163.com

Xiao Guo, Jinyao Yan, Bo Liu, Qianjun Shuai
Computer and Network Center,
Communication University of China
Beijing, China, 100024
{xguo, jyan, lb2008, sqj}@cuc.edu.cn

Abstract—**With the limitation of port densities in the highest-end commercial switches, topologies of data center network often take the form of a multi-rooted tree with higher-speed links. In order to achieve multipath transmission, data center uses Equal Cost Multipath (ECMP) algorithm to forward flows. ECMP algorithm is lack of dynamic scheduling mechanism. In this paper, we completed a full implementation of three scheduling algorithm in the data center network of SDN architecture. One is based on OSPF algorithm. The second is based on ECMP algorithm. In the third algorithm, we optimized the ECMP algorithm through OpenFlow protocol and made it possible to dynamically adjust the flow forwarding link according to the bandwidth utilization of core links. The evaluation results demonstrate that the third scheduling algorithm can improve performance of throughput in data center networks.**

Keywords—traffic engineering; sdn; ecmp; data center

I. INTRODUCTION

With the rapid development of big data, cloud computing and Internet, large organizations, such as universities, research labs and companies, are building enormous data centers to support their applications. Those applications, such as scientific computing, financial analysis, data analysis, warehousing and large-scale network services, require substantial intra-cluster bandwidth. As those applications continue to expand, scaling the capacity and processing capability of data centers has become a challenge problem. In order to adapt to existing network conditions, make efficient use of network resources, researchers do traffic engineering to control flows' forwarding path [5].

Today's data centers constitute by tens of thousands of machines with significant aggregate bandwidth demands. The traditional Internet communication are controllable through a modest number of given host pairs. There are particularly a small amount of paths between communication host pairs. But the loads of data center network are constantly changing over both time and space. Those properties inherent in the data center makes the traditional network architecture and network protocol no longer applicable. Compared with the traditional network architecture, typical topology of data center rely on the path multiplicity to achieve horizontal scaling of hosts. A typical architecture usually consist of three level trees of switches or routers. For example, a three-tiered topology has a core layer in the root of the tree, an aggregation layer in the middle and an edge layer at the leaves of the tree. Those architectures have multiple paths advantages, which can deliver full bandwidth between arbitrary hosts. To reduce costs, researchers proposed an improved architecture, FatTree [14] topology.

Most large IP networks operate Open Shortest Path First (OSPF) to compute a single shortest path for each pair of communication host. But if we put OSPF protocol in typical architecture of data center, it cannot significantly utilize multiple paths advantages of this architecture. In data center networks, Equal Cost Multipath [2, 8] (ECMP) is normally used to statically stripe flows across available paths. According to the flow's hash value, different ranges of flow are forwarded to different path. This defined mapping of flows to paths does not account for bandwidth utilization of current network. For the lack of dynamic scheduling capabilities, ECMP can't realize fair distribution of link's bandwidth. To solve this problem, we use OpenFlow [1, 10] protocol to optimize the ECMP algorithm.

In this paper, we built a data center network that based on Software Defined Network [13] (SDN) architecture, compare the bandwidth utilization of two experiments to show the performance of three scheduling algorithms. In the first experiment, we use traditional OSPF algorithm and classical ECMP algorithm to forward flow and compare their performance by calculating the bandwidth utilization of core links. In the second experiment, we using classical ECMP algorithm and optimized dynamic ECMP algorithm to forward flow and compare their performance by calculating the bandwidth utilization of core links also.

The remainder of this paper is structured as follows. In Section II, we give an overview of primary components in our date center networks, including topology, routing protocol and scheduling algorithm. Section III introduced our scheme in more detail. The results about our simulation are described in Section IV. We summarize the main contribution of this paper in Section V.

II. DATA CENTER NETWORK

Most data center topologies have been organized as FatTree topology. These multi-rooted trees have many equal-cost paths between all host pairs. A key challenge is to dynamically forward flows along these equal-cost paths to

978-1-4799-4811-6/14/$31.00 ©2014 IEEE

improve the throughput of data center networks. ECMP has been widely used in data center networks to statically forward flows across multiple equal-cost paths. This static mapping of flows to paths does not account for either current network utilization or flow size, with resulting some links bandwidth utilization exceed threshold while other still idle. To solve this problem, we proposed SDN-based ECMP algorithm to realize dynamic flow scheduling according to links bandwidth utilization. SDN is a new network architecture that allows network administrators to control network traffic flexible. There is less research work in data center network of SDN architecture. Therefore, we investigated how to use the flexibility of SDN technology to improve the throughput of data center networks.

From a high-level perspective, there are three main points for a SDN-based data center architecture. They are physical topology, routing protocol and dynamic scheduling algorithm. They are not independent, that is, the performance of one will be influenced by the choices of others.

A. Topology

Bandwidth is becoming the bottleneck for scalability of large-scale clusters. Existing solutions for solving this bottleneck mainly use hierarchies of switches. With the expensive, non-commodity switches at the top of the hierarches protocol, one problem is that the port density of high-end switches limits overall cluster size. And besides, it incurs high cost. Recent approaches such as FatTree and VL2 [7] are Clos topologies that use multiple core switches to provide full bandwidth between any pair of hosts in the network. Those topologies are interconnected by two or three layers of switches to avoid defects in port densities which exists in commercial switches. Recent research advocates the horizontal expansion of data center networks [3, 7]. These research present a data center communication architecture that leverages commodity Ethernet switches to deliver scalable bandwidth for large-scale clusters. The results of their experiments show that this topology is able to deliver scalable bandwidth at significantly lower cost than existing techniques.

B. Routing

The current scheme uses ECMP or multiple static VLANs to take advantage of the properties of multiple equal-cost path in FatTree topology. ECMP or multiple VLANs provide the basis for randomized load balancing as the default path selection method. Based on a hash of 10-tuple in each flow packet, classical ECMP can forward this flow to corresponding path. In this way, a flow's packets all take the same path, and their arrival order is maintained.

First we should calculate the equal-cost path of the data center network. Then we statically forward flows across available paths using flow hashing. This static mapping of flows to paths does not account for either current network bandwidth utilization or flow packets size, which resulting collisions of data flows and degrading overall throughput.

A key limitation of ECMP is that if the path of two or more large long-lived flows transport through one or more same core links, the bandwidth utilization of this core link maybe exceed the threshold. This is because ECMP can't dynamically select path according to the bandwidth utilization of core links. To address this problem, a centralized flow scheduler has been proposed.

C. Software Defined Network

SDN is a new approach for computer networking that allows network administrators to manage network services through abstraction of lower level functionality. SDN requires southbound interface for the control plane to communicate with the data plane. One of such southbound interface, OpenFlow protocol, is the most suitable standard until now. OpenFlow was first proposed in [1] as a way to enable researchers to conduct experiments in production networks. However, its advantages may lead to its use beyond this range.

OpenFlow provides an open protocol to program the flow-table in different switches and routers. A network administrator can modify the flow-table through controller to control their own flows – by choosing the routes their packets follow and the processing they receive. In this way, researchers can implement firewalls, Network Address Translation, Quality of Service, and can collect networks' operation status statistics as well.

The OpenFlow switch itself holds a flow table which stores flow entries. A flow entry consist of three fields: (1) A packet header that defines the flow, (2) The action, which defines how the packets should be processed, and (3) Statistics, which keep track of the number of packets and bytes for each flow, and the time since the last packet matched the flow (to help with the removal of inactive flows). The handing process of incoming packets in an OpenFlow switch is visualized in Fig. 1.

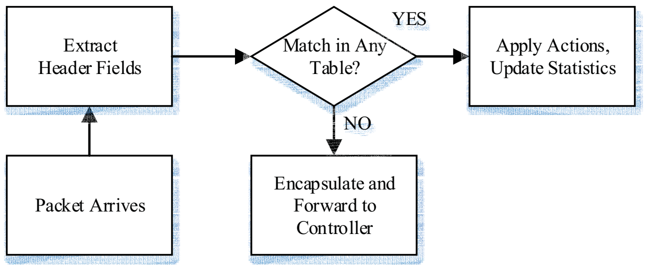
Extract Header Fields
Match in Any Table?
YES
Apply Actions, Update Statistics
NO
Packet Arrives
Encapsulate and Forward to Controller

Figure 1. Handling of incoming packets in an OpenFlow switch.

In our scheme, we use OpenFlow protocol to implement ECMP algorithm and SDN-based ECMP algorithm in data center network of SDN architecture.

D. Scheduling Algorithm

As mentioned above, using a central scheduler can improve the performance of data center network. The scheduler can periodically measurement the bandwidth utilization of core links. Thus, the scheduler can assign flows to reasonable paths using dynamic scheduling algorithm with the value monitoring.

Traditionally, it is difficult for dynamic path selection

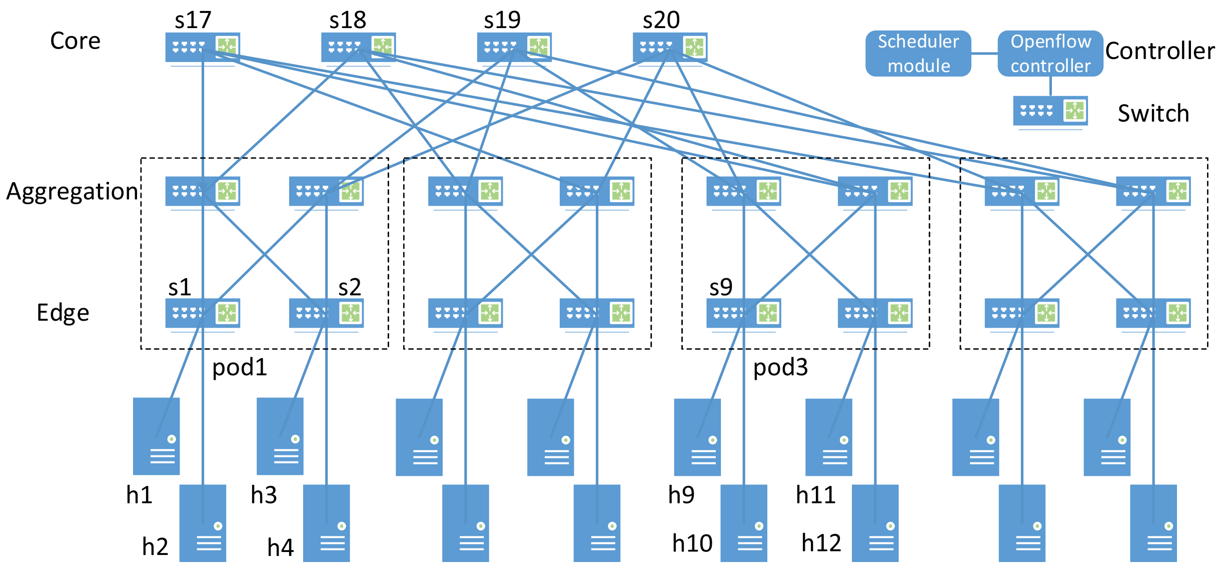

Figure 2. Network Topology. It is a typical FatTree topology. All switches connect to Floodlight, a control platform.

according to the link status of the whole network. The SDN network architecture makes this possible. The switches in the SDN-based network all run OpenFlow protocol. All of these switches connect to a central controller. This controller can query, insert, modify flow entries, or perform many other actions. Once a flow makes the bandwidth utilization of some core links beyond the specified threshold, the controller can modify the flow entry for that flow to redirect it along a newly chosen path.

III. SCHEME DESIGN

The goal of this paper is to implement the SDN-based ECMP algorithm through OpenFlow protocol and to prove that this algorithm is outperformed the ECMP algorithm in data center networks. Described at a high-level, our scheme includes a control loop of three basic parts. In the first part, create a virtual data center network topology of SDN architecture in a virtualization platform. In the second part, use central controller to develop a scheduler modules which will forward flow according to different scheduling algorithm. In the last part, calculate the bandwidth utilization of specified core links in different scheduling algorithm to compare their performance of throughput in data center networks.

A. Realization of Topology :

Mininet [11] is a network simulator that can conveniently create a network topology which composed of many virtual OpenFlow switch. At the beginning of our scheme, we create a virtual data center network topology using Mininet. Fig. 2 shows this network topology that based on a FatTree architecture topology. This topology has been presented in [3] and proved to have a better performance in throughput and cost. This network built from k-port switches, there are k pods, each consisting of two layers: lower pod switches (edge switches), and the upper pod switches (aggregation switches). Each edge switch manages (k/2) hosts. The k pods are interconnected by $(k/2)^2$ core switches. There are $(k/2)^2$ equal-cost paths between any two communicating edge switches, each belongs to different pods. In our scheme, k is 4. Obviously, those direct links (core links) of core switch take numerous network traffic load. Its traffic load is key to the network performance.

For example, there are four equal-cost paths between the host h1 and host h9 as shown in Fig. 2. Each of the equal-cost path contains a core switch. In the scheme, we measure the bandwidth utilization of four core links to compare the performance of different scheduling algorithms.

B. Controller and Scheduler Module:

In this step, we choose Floodlight [9] as the central controller. Floodlight is Java-based and intended to run with standard jdk tools and can optionally be run in Eclipse. The Floodlight controller realizes a set of common functionalities to control and inquire an OpenFlow network, while applications on top of it realize different features to solve different user needs over the network. In our scheme, we developed a scheduler module based on Floodlight to achieve the optimization goal of our scheme. When Floodlight is running, it will connect to the virtual network topology we created above. Then the scheduler module of Floodlight will start a thread which can measure the number of transmitted bytes of the specified switch port every 10 seconds. Through those statistics we can calculate the bandwidth utilization of core links.

The scheduler module has three scheduling algorithms. First is the OSPF algorithm that flow from the same pod will be transmitted through the same path, looks like OSPF protocol operates in the traditional network. Second is the ECMP algorithm that implements ECMP through Floodlight controller and can forward flows to different equal-cost path according to the calculated hashing value of data packets. This algorithm does not consider the core links bandwidth utilization. The controller uses ECMP to statically stripe flows across available equal-cost path. The third algorithm is SDN-based ECMP algorithm. In this algorithm, we set a threshold of bandwidth utilization for all core links. When one of core link bandwidth utilization of a core switch exceed the specified threshold, the scheduler module will forward some flows to other links which have a lower bandwidth utilization. By this dynamic scheduling, it can be predict that the throughput of the date center network will be improved. Fig. 3 is the flow chart of SDN-based ECMP algorithm.

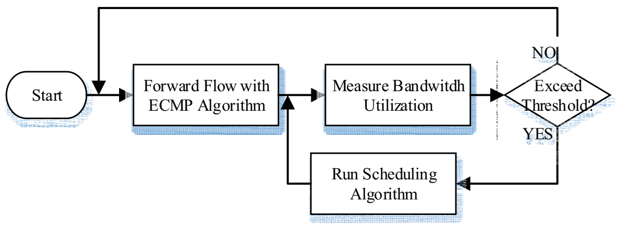
Start
Forward Flow with ECMP Algorithm
Measure Bandwitdh Utilization
Exceed Threshold?
NO
YES
Run Scheduling Algorithm

Figure 3. Flow chart of the SDN-based ECMP algorithm.

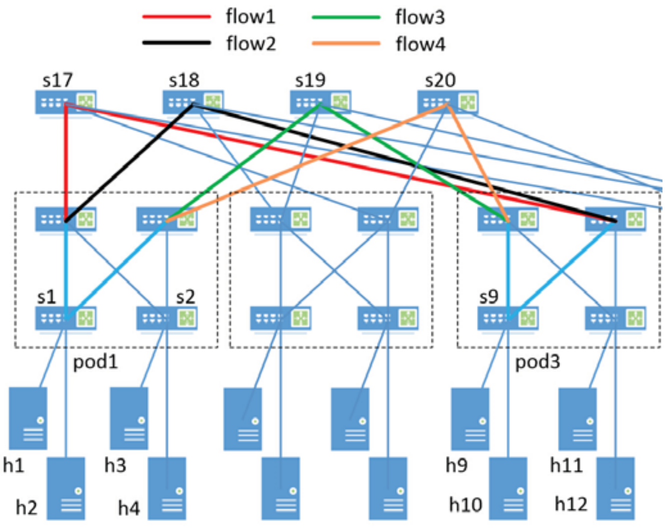

Figure 4. Four equal-cost paths between s1 and s9.

C. Compare the Performance of Algorithms:

We select a number of host pairs belong to different edge switches for data transmission. According to different algorithms and measured bandwidth utilization, those data flows will be transmitted through different paths.

As illustrated in Fig.4, there are four equal-cost paths between every two edge switches, such as s1 and s9. Every edge switch connect to two host. In our scheme, the four communication host pairs (h1-h9, h1-h10, h2-h9, h2-h10) are able to communicate through one of four equal-cost paths. There are four equal-cost paths between s2 and s9, which are the same four paths between s1 and s9. For example, host h1 may communicate with h9 through s17 while host h3 may communicate with h10 through s17 also. So when we calculate the core link bandwidth utilization of s17, it is necessary to calculate the sum of both. In Section IV, we transmit data at a constant speed between these host pairs (h1-h9, h1-h10, h2-h9, h2-h10, h3-h9, h3-h10, h4-h9, h4-h10) and calculate the core link bandwidth utilization of all four core switches to compare the performance of different scheduling algorithms.

IV. Simulation and Evaluation

In this section, we first describe the experimental design. We conduct tests with Mininet on Linux host with the Floodlight controller. Then we measure the network bandwidth utilization under different scheduling algorithm to compare these algorithms' performance.

A. Experimental Design

1) Network Emulator

We use Mininet to model data center network topology, as shown in Fig. 2. This fat-tree network topology has 20 switches and 16 hosts.

2) Central Controller

We choose Floodlight as the central controller and developed a scheduler module. The controller queries the transmitted bytes on each port from core switches every 10 seconds. We only query the upward traffic of core switches.

3) Traffic Design and Measurement

We use Iperf to generate UDP traffic in the data center network. Host of h1, h2, h3, h4 is set to Iperf client mode as h9, h10 is set to Iperf server mode. The Iperf clients transmit data flows to Iperf servers. The traffic rate of each flow is in the specified range of the overall core link bandwidth. Based on the pre-set access speed, we calculated the possible bandwidth utilization before testing. These possible bandwidth utilization are shown in Table I and Table II.

In our two experiments, the threshold of bandwidth utilization is 0.7. Each algorithm is simulated for 60 seconds and get six groups of transmitted bytes data. Based on these data, we calculate the bandwidth utilization and compare the performance of three scheduling algorithms.

4) Algorithm Evaluation

We designed two experiments to evaluate the performance of the three scheduling algorithms. In the first experiment, we compare the different bandwidth utilization of corresponding core links when scheduler module operate OSPF algorithm and ECMP algorithm. Like the first experiment, we compare the difference when scheduler module operate OSPF algorithm and SDN-based ECMP algorithm in the second experiment. Before the test, we specify the access speed of each flow and then calculate the possible bandwidth utilization of core links that transmit these flows. For example, host h1 communicate with host h9 at a constant speed. Their data flow will go through s18, so we can calculate a possible bandwidth utilization of s18's core link. This value should be roughly equal to the bandwidth utilization that calculated by central controller. We have designed different access speed of the eight host pairs for different experiments. Thus the possible bandwidth utilization is different in different experiments.

In the first experiment, we first calculated the possible bandwidth utilization as shown in Table I. When the first scheduling algorithm is applied, eight flows (h1-h9, h1-h10, h2-h9, h2-h10, h3-h9, h3-h10, h4-h9, h4-h10) will be transmitted through one core switch s18. But if the second scheduling algorithm is applied, four flows (h1-h9, h1-h10, h2-h9, h2-h10) will be transmitted through the four equal-cost paths. Thus one core switch will transmit one data flow. Another four flows (h3-h9, h3-h10, h4-h9, h4-h10) will be transmitted through the same equal-cost paths. So the core links bandwidth utilization of four core switches is the sum of two flows.

In the second experiment, we calculated the possible bandwidth utilization as shown in Table II. When the second scheduling algorithm is applied, the forwarding path of different flows will be the same as in the first experiment. If the third scheduling algorithm applied, the forwarding path of different flows will be the same as the second scheduling algorithm when the core links bandwidth utilization of four core switches not exceed the threshold. But when the bandwidth utilization of some core links exceed the threshold, this scheduling algorithm will dynamic forward corresponding data flow to a path of lowest bandwidth utilization. Fig. 5 is the flow chart of our two experiments.

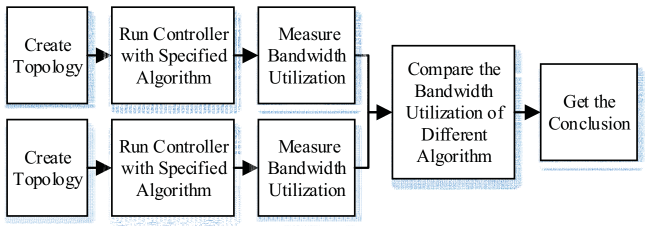
Create Topology
Run Controller with Specified Algorithm
Measure Bandwidth Utilization
Create Topology
Run Controller with Specified Algorithm
Measure Bandwidth Utilization
Compare the Bandwidth Utilization of Different Algorithm
Get the Conclusion

Figure 5. Flow chart of our two experiments.

B. Results

1) First Experiment

Depending on the pre-set access speed of each host, we calculate the possible bandwidth utilization of eight flows (h1-h9, h1-h10, h2-h9, h2-h10, h3-h9, h3-h10, h4-h9, h4-h10), as shown in Table I. These values represent the communication speed of the host pairs.

TABLE I. POSSIBLE BANDWIDTH UTILIZATION OF EACH ACCESS FLOW

Data Flow	***h1 → h9***	***h1 → h10***	***h2 → h9***	***h2 → h10***	***h3 → h9***	***h3 → h10***	***h4 → h9***	***h4 → h10***
Possible Bandwidth Utilization	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

The simulation results shown in Fig. 6 and Fig. 7.

Fig. 6 shows the bandwidth utilization of the OSPF algorithm. In this scenario, all of the access flows was forwarded to switch s17. The core links bandwidth utilization of s17 exceeded threshold while other switches' core links bandwidth utilization close to zero.

Fig. 7 shows the bandwidth utilization of the ECMP algorithm. As shown that bandwidth utilization of all core switches almost equal. This is because the use of ECMP algorithm makes different flows can be forwarded to equal-cost paths evenly.

Through comparison, we find that the ECMP algorithm can statically load balance the core links bandwidth utilization and improve the throughput of the data center network.

2) Second Experiment

We calculate the possible bandwidth utilization of eight flows (h1-h9, h1-h10, h2-h9, h2-h10, h3-h9, h3-h10, h4-h9, h4-h10) and show the results in Table II.

TABLE II. POSSIBLE BANDWIDTH UTILIZATION OF EACH ACCESS FLOW

Data Flow	***h1 → h9***	***h1 → h10***	***h2 → h9***	***h2 → h10***	***h3 → h9***	***h3 → h10***	***h4 → h9***	***h4 → h10***
Possible Bandwidth Utilization	0.4	0.2	0.4	0.2	0.4	0.1	0.4	0.1

The simulation results shown in Fig. 8 and Fig. 9.

Fig. 8 shows the bandwidth utilization of the ECMP algorithm. Because the scheduler cannot select forwarding path according to the link status, two core switches bandwidth utilization of core links exceed threshold. For example, flow from h1 to h9 and flow from h3 to h9 were forwarded through the same equal-cost path. Thus the total bandwidth utilization of s17's core link exceed threshold. This limitation will surely reduce the throughput of data center network.

Fig. 9 shows the bandwidth utilization of the SDN-based ECMP algorithm. The simulation results show that the bandwidth utilization of corresponding core links is very high and none of the links' bandwidth utilization exceeds threshold. This is because the total bandwidth utilization of s17's core link exceed threshold. The controller will dynamic adjust the transmission path. By dynamically adjusting, for example, flow from h3 to h9 will be forwarded to the path that flow from h1 to h10 are using while flow from h3 to h10 will be forwarded to the path that flow from h1 to h9 are using.

From Fig. 8 and Fig. 9, we find that when SDN-based ECMP algorithm is used, the throughput of data center network has been improved. The reason is that SDN-based ECMP algorithm could dynamically adjust the flow forwarding path according to the bandwidth utilization of core links. This also proves that topology of SDN architecture is very suitable for data center networks.

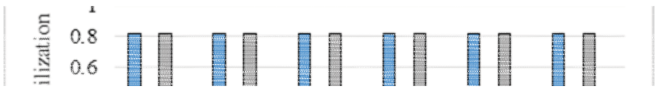

Figure 9. Second Experiment with SDN-Based ECMP Algorithm

V. CONCLUSIONS

FatTree network architecture is a suitable solution for addressing limitations of traditional data center topology. With the use of ECMP algorithm, the throughput of date center network that adopting FatTree architecture will be greatly improved. But ECMP algorithm is lack of flexibility. With the rapid development of SDN, we find a way to solve the flexibility problem of ECMP algorithm.

This work proposed a SDN-based ECMP algorithm by applying a SDN central scheduler with global statistics of links bandwidth utilization to dynamically assign traffic loads. The simulation results show that our proposed algorithm can significantly outperform the traditional ECMP algorithm and other algorithms in data center networks. We believe that the SDN-based data center network solutions are the future trends.

REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2 , 2008, pp. 69-74.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data centernetworks," in 7th USENIX Symposium on Networked System Design and Implementation (NSDI), San Jose, CA, USA, April 2010, pp. 281–296.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proceedings of the ACM SIGCOMM, New York, NY, USA, August 2008, pp. 63-74.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," in Proceedings of the Seventh Conference on emerging Networking EXperiments and Technologies, New York, NY, USA, December 2011.

[5] M. Chiesa, G. Kindler, and M. Schapira, "Traffic Engineering with Equal-Cost-MultiPath: An Algorithmic Perspective," in INFOCOM, 2014. Toronto, ON, CAN, April 2014, pp. 1-9.

[6] C. Raiciu, C. Pluntke, S. Barre, A. Greenhalgh, D. Wischik, and M. Handley, "Data center networking with multipath TCP," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Monterey, CA, USA, October 2010.

[7] A. Greenberg el al., "VL2: A Scalable and Flexible Data Center Network," in Proceedings of the ACM SIGCOMM, Barcelona, Spain, August 2009, pp. 51-62.

[8] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe, "Resilient routing using MPLS and ECMP," in Proceedings of IEEE Workshop on High Performance Switching and Routing, Phoenix, AZ, USA, 2004.

[9] "Project Floodlight," http://www.projectfloodlight.org/floodlight/, last accessed on 2014.06.30.

[10] "OpenFlow," http://archive.openflow.org/, last accessed on 2014.06.15.

[11] "Mininet," http://mininet.org/, last accessed on 2014.05.30.

[12] "Floodlight-developers," https://groups.google.com/a/openflowhub.or g/forum/#!forum/floodlight-dev, last accessed on 2014.07.01.

[13] D. Nadeau, and K. Gray, SDN: Software Defined Networks, O'Reilly Media, Inc, USA, 2013.

[14] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," IEEE Transactions on Computers, vol. 34, no. 10, October 1985, pp. 892-901.