A Weighted ECMP Load Balancing Scheme for Data Centers Using P4 Switches

Jin-Li Ye[1] and Chien Chen[12]
[1]Department of Computer Science and [2]Information Technology Service Center
National Chiao Tung University, HsinChu, Taiwan
dannyxx001@gmail.com, chienchen@cs.nctu.edu.tw

Yu Huang Chu
Chunghwa Telecom Laboratories
Chunghwa Telecom Co., Ltd.
Taoyuan, Taiwan
yhchu@cht.com.tw

Abstract—**This paper proposes a weighted Equal-cost multi-path (W-ECMP) scheme in datacenters using a Programmable Data plane. Nowadays, data center networks face ongoing challenges for higher performance and greater efficiency. Equal-cost multi-path (ECMP) routing, the most common load-balancing scheme used on data center networks, tries to balance the traffic without awareness of the network congestion. To overcome this problem, CONGA and HULA are proposed successively. CONGA has implemented customized ASIC as a part of the switch fabric to collect congestion information distributively. Nevertheless, ASICs solution shows a lack of scalability and flexibility. However, although HULA solves some problems of CONGA by using p4 switch, best path selection strategy in HULA only selects and records the best path, which makes it easy to congest the best path, and HULA uses probes that waste bandwidth and deteriorate the performance. To solve the problems of HULA, W-ECMP chooses a path with a weighted probability to avoid congesting a path quickly. W-ECMP encapsulates the path congestion information into the normal traffic similar to the concept of Inband Network Telemetry (INT) in P4, which increases updating speed as the network loading increases. From experimental results, W-ECMP performs better than HULA does in average flow completion time (FCT).**

Keywords—data center load balancing, network congestion, programmable data plane, weighted-ecmp

I. Introduction

Nowadays, data center networks are expected to deliver messages with low latency and fully utilize the potential capacity of network devices to provide high throughput for cloud providers or enterprises. Multi-rooted tree-based topologies like Fat-tree [1] and Clos [2] are generally used to build up a data center. These topologies are well-known for providing large bisection bandwidth through transferring the traffic among multiple paths. The most commonly used scheme for load balancing in data centers is equal-cost multi-path (ECMP), which usually separates the traffic flows randomly into different paths using a hash or round-robin scheduling without considering the congestion status of the network. Due to hash collisions or a lack of congestion information, ECMP may assign more than one large flow to the same path, which leads to inefficient usage of bisection bandwidth in data center. To overcome the problems of ECMP, many kinds of research have been proposed from different aspects.

Some approaches based on software-defined networking (SDN) use a centralized controller to balance the flows [3][4]. Although centralized schemes can achieve nearly optimal solutions, they are too slow to react to the bursts of traffic in data center networks. Therefore, the distributed mechanisms implemented on the data plane are presented. CONGA [5] is the first example that implements an efficient global congestion-aware load balancing mechanism in the network data plane. Nevertheless, CONGA still has some shortcomings in its implementation. First, CONGA uses too much memory space to store path information. Second, getting feedback from remote switches may not be real-time enough to react to the current state. Third, CONGA is implemented in customized ASICs, which is time-consuming and lacks flexibility. Therefore, HULA [6] is proposed to solve some problems of CONGA by using a programmable data-plane platform P4-enable switches [7]. The results of HULA shows that it achieves better performance than CONGA does in the same average flow completion time (FCT) [9].

However, HULA still has the following derivative problems. First, HULA makes each switch only select and record the best path with least congestion to the destination, which results in congesting the best path rapidly. In HULA, the updating speed of best path is determined by probe frequency. Second, because of using additional probes to collect the path's congestion information, HULA wastes the link bandwidth and intensely influences the performance if the link bandwidth is limited and probe frequency is very high.

To overcome the major problem that is congesting a path quickly in HULA, this paper proposes a weighted equal-cost multi-path load-balancing (W-ECMP) scheme in data center networks. We use P4 to program W-ECMP into the data-plane. P4 is characterized by protocol-independence, reconfiguration and not limited by any protocol or hardware, so it is suitable to fulfill our distributed architecture and quick to adapt to our algorithm. In W-ECMP, each leaf switch maintains a table to store the utilization of all paths to the other leaf switches, and the traffic is handled at the granularity of flowlets [11], which can split a flow without causing packet reordering. Through transforming the utilization to weight, W-ECMP chooses a path with a weighted probability for a flowlet to avoid congesting the least congested path quickly. Flowlet-based granularity uses the burst nature of packets to split the traffic. When the inter-arrival time between two consecutive packets in the same flow is greater than a threshold, the following packets can be considered a new flowlet. Furthermore, instead of using probes we encapsulate the path's congestion information into regular traffic like the concept of Inband Network Telemetry (INT) [12],

which increases status update speed as the network loading increases.

II. W-ECMP SCHEME

We propose a weighted equal cost multi-path (W-ECMP). W-ECMP uses the path's utilization as the probability of choosing a path other than the best path, thus it is not as sensitive to the frequency of state update compared to CONGA and HULA. In the following sections, we will show the detailed design of a W-ECMP load balancing scheme in data center using P4.

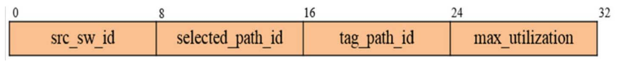


Fig. 1. W-ECMP Header Format

A. W-ECMP Header Format

This paper designs a self-defined header of 4 bytes that is called the W-ECMP header as shown in Fig. 1. The W-ECMP header is not limited by any kinds of protocol, which means any traffic flows can be leveraged in the data center to carry the information. In this paper, TCP traffics are used to encapsulate the W-ECMP header to collect the following messages:

- *src_sw_id* (8bits): The id of the source leaf switch is used to identify where the flow comes from.
- *selected_path_id* (8bits): Each leaf switch uses the selected path id to specify the whole path where the traffic is forwarded. The switches on the selected path identify the next hop through this field.
- *tag_path_id* (8bits): The switches on the selected path tag one bit of the field when the packet goes through them. The bits of this field will be marked from the right side to the left mapping to the sequence of the switches along the packet's path.
- *max_utilization* (8bits): This field is the maximum utilization of a path, and is used by the switches along the packet's path to convey the congestion information in the opposite direction.

The W-ECMP header carries four important information in order to achieve two goals. One is the forwarding path determined from one leaf switch to another leaf switch in the data center network. The other is the collection of congestion information of the forwarding paths in reverse direction.

B. Reverse Collection for Congestion

To calculate the link utilization, a simple module is added in bmv2. This module is driven by a thread, which is independent from the pipeline threads. It directly computes the transmitting rates of each port every second. Assuming that each switch has inherent link bandwidth information mapping to all its ports, the utilization can be estimated based on the ratio of the transmitting rates to the bandwidth to denote the level of congestion. Besides, W-ECMP uses a simple weight to quantify the outgoing congestion of a port, so each switch converts link utilization into a weight based on TABLE 1. Furthermore, the middle switches in the same layer are numbered with a tag id, which is used to tag the W-ECMP header for the destination leaf switch identifying the path where the packet goes through.

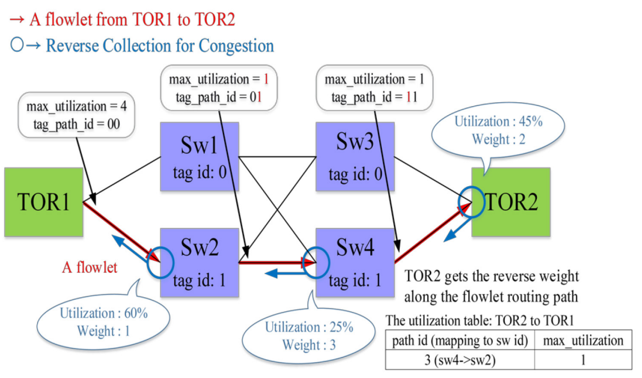


Fig. 2. Reverse Collection for Congestion

TABLE I. UTILIZATION-TO-WEIGHT MAPPING

utilization	**<= 10%**	**<= 25%**	**<= 50%**	**<= 75%**	**otherwise**
weight	4	3	2	1	0

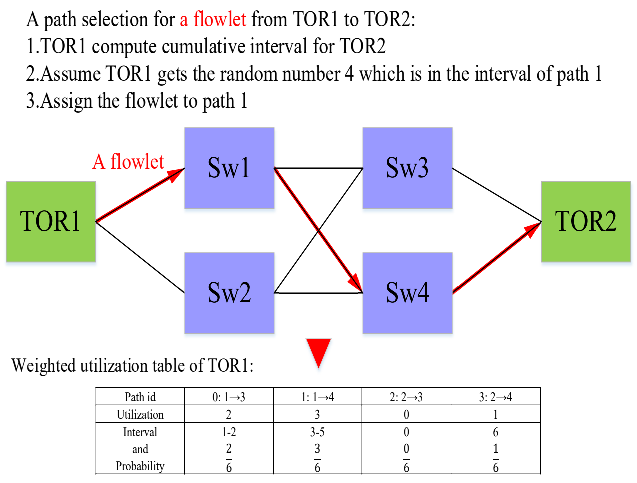
W-ECMP uses regular traffic to carry the reverse path's weight along the traveling traffic as shown in Fig. 2. The middle switches between any pair of two leaf switches will process the *max_utilization* and *tag_path_id* fields of the W-ECMP header using the following steps:

- Each switch prepares the metadata including the weight of ingress port and tag id.
- Each switch marks the *tag_path_id* based on the order the packet passes. The *tag_path_id* will be tagged from the right side to the leaf. Take Fig. 2 as an example, when a packet arrives Sw2, the first bit from the right side of the tag_path_id will be tagged as 1, when a packet arrives Sw4, the second bit from the right side of the tag_path_id will be tagged as 1, and so on. At last, when this packet arrives TOR2, the tag_path_id will be binary 11, which means the index of path id is 3.
- If the *max_utilization* field of the W-ECMP header is greater than the weight of ingress port, the switch will replace the field with the weight of ingress port. Otherwise it will keep the field the same. In the example of Fig. 2, *max_utilization* of path id 3 from TOR2 to TOR1 is 1.

When the traffic arrives at the destination leaf switch, the switch will update the status of the path, which is from the destination leaf switch to the source leaf switch. In this way, any leaf switch has sufficient real-time path information to make the probabilistic decision for load balancing.

C. Flowlet Probabilistic Decision with Weight

For the purpose of avoiding packet reordering in TCP flow, W-ECMP splits the traffic at the granularity of flowlets to do load-balancing. As aforementioned, how a switch detects a new


A path selection for a flowlet from TOR1 to TOR2:
1.TOR1 compute cumulative interval for TOR2
2.Assume TOR1 gets the random number 4 which is in the interval of path 1
3.Assign the flowlet to path 1

Weighted utilization table of TOR1:

Path id	0: 1→3	1: 1→4	2: 2→3	3: 2→4
Utilization	2	3	0	1
Interval and Probability	1-2 $\frac{2}{6}$	3-5 $\frac{3}{6}$	0 $\frac{0}{6}$	6 $\frac{1}{6}$

Fig. 3. W-ECMP Mechanism Example

flowlet is by determining if the inter-packet gap is greater than a flowlet threshold T_{fl}. Generally, the threshold T_{fl} is set to the order of network round trip time (RTT) which is enough to absorb the delay caused by congested paths. In this paper, a hash function is used to hash five tuples (*src IP*, *dst IP*, *protocol*, *src port*, *dst port*) as the index to access the flowlet table. The flowlet table is used to record two pieces of information: the last time when the flowlet arrives, and the selected path id for the flowlet. The initial values of the two fields of the flowlet table are set to zero. When the data center network is started up, there must be a latency before the traffics begin to be transferred to the network. Typically, the latency is greater than RTT, so it does not need to worry that the initial flow will not be identified as a new flowlet. The concept of flowlet detection will be conducted by the following steps:

- When a packet comes, the leaf switch calculates the hash of the flow's five tuples to get the index of the flowlet table.
- It accesses the previous timestamp of the flow, and calculate the subtraction of the current timestamp and the previous timestamp.
- If the inter-packet gap is greater than the threshold T_{fl}, it takes the packet as a new flowlet and uses the weighted utilization table to choose a new path for the flowlet probabilistically. At last, it will update the selected path id of the flowlet table by the hash index.
- Otherwise, it will use the path id recorded in the flowlet table.

In this paper, each leaf switch stores the weight of all the paths to the other leaf switches in the weighted utilization table. When a leaf switch needs to choose a new path for a new flowlet, it will accumulate the weight in the order of path id of destination leaf switch and randomly select a number from one to the total accumulated weight. Fig. 3 shows an example of how to use weighted utilization table as the probability to choose a path. The total accumulate weight in this example is 6. The 2/6, 3/6, 0/6, and 1/6 are the probabilities to select the path 0, 1, 2, and 3 respectively. When the random number is in the certain interval such as between 3 and 4, the corresponding path id (i.e. 1) will be selected. Otherwise, if all the weights are zero, a path will be randomly chosen like the original ECMP. Additionally, note that the switch id mapping to the path id and the cumulated weight are not stored in the weighted utilization table.

III. Performance Evaluation

In this section, the performance of our W-ECMP scheme is compared with ECMP and HULA. Because HULA has proven that their method outperforms CONGA, W-ECMP doesn't redundantly compare with CONGA. In this experiment, the mechanisms of ECMP, HULA, and W-ECMP all run in the same evaluation environment. They are implemented with different P4 programs.

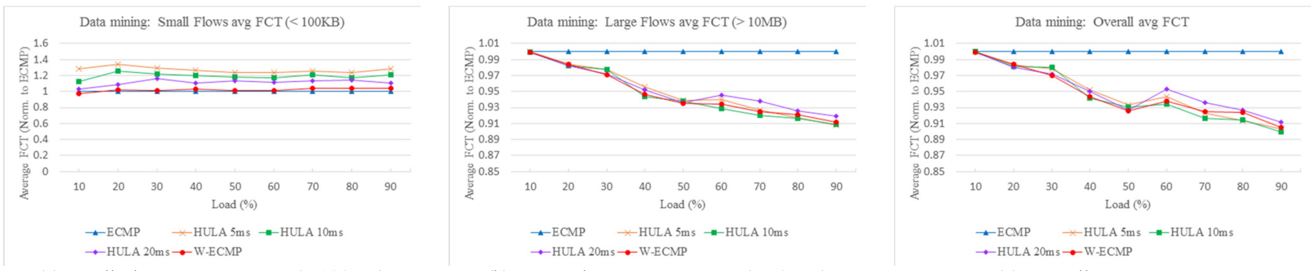
A. *Experiment Setting*

The environment is built up by the following software tools: Mininet, Iperf, and P4 Behavioral-model. Mininet is used to generate the topology with three layers: leaf, spine, and core. Due to the inherently poor performance of P4 bmv2 switch[13][14], the link bandwidth capacity is reduced. The link bandwidth between a pair of switches is set to 20Mbps, and the link bandwidth between a host and a leaf switch is set to 5Mbps. The realistic flow size distributions derived from [6] are used to generate the traffic for testing. The distributions are heavy-tailed, which means most flows are small and a small portion of flows occupy most data of total traffic. One of those distributions we used in our experiment is data-mining workload, whose 80% of flow sizes are less than 10KB, and only 5% of flow sizes are greater than 10MB.

Each host connected to a leaf switch uses Iperf to run both client and server processes at the same time. Each client is made to randomly select a remote server that is in a different pod to make sure that all the traffic must pass through the core switch. In general cases, each client generates a flow according to the Poisson process, which means the inter-arrival time between two consecutive flows is taken from an exponential distribution. And the mean of the exponential distribution is adjusted to achieve the desired loading varying from 0.1 to 0.9 on the network. Similar to prior works [5][6][9], the average flow completion time is calculated as performance metrics to compare with other methods and additionally separate the results of small and large flows to give a detailed explanation. In this experiment, nine random seeds are run for each loading per workload and we measure the average of the nine runs.

B. *Analysis and Discussion*

In this section, W-ECMO, ECMP, and HULA mechanisms are compared using data mining workload. The flowlet threshold is set to 20ms for all three methods. HULA is also tested with three probing frequencies, 5ms, 10ms, and 20ms, to see the impact on the updating of the best path. All the values of the results are normalized to those of ECMP. The results of the data-mining workload are shown in Fig. 4(a)-(c). For small flows, the average FCT in Figure 4(a) shows that W-ECMP



(a) Small Flows Average FCT (< 100KB) (b) Large Flows Average FCT (> 10MB) (c) Overall Average FCT

Fig. 4. Results of Data-Mining Workload

performs almost the same as ECMP does. However, no matter ECMP and W-ECMP. With the increase of probe frequency, the small flows' average FCT of HULA is getting worse. For large flows, the average FCT in Figure 4(b) shows that ECMP performs the worst, and both HULA and W-ECMP methods make improvements on FCT relative to ECMP. Besides, in microscopic comparison, W-ECMP achieves a similar improvement as HULA with probe frequency 5ms. Hence, the probe frequency results in a large overhead for extremely small flows, but it shows almost no overhead for extremely large flows. That is why the performance of small flow and large flow is different in these two workloads for HULA. For overall FCT as shown in Figure 4(c), it is the average of the FCTs of all flows including FCTs of the middle size flows, whose size is between small flows and large flows. The result is almost the same as the average FCT of large flows. Because large flows contribute the most data in the data center network, which means they give the largest percentage of FCT, the result of overall FCT is close to the FCT of large flows. As a result, W-ECMP improves FCT for large flows by as much as ~10% compared with ECMP. In summary, W-ECMP does not cause large overhead in the FCT of small flows, and it can achieve slightly better performance than HULA in the FCT of large flows. W-ECMP encapsulates the information in traffic, which is more efficient for bandwidth usage. Compared from many kinds of aspects, W-ECMP outperforms HULA. In overall average FCT, W-ECMP achieves a ~10% improvement in data-mining workload compared to ECMP. In the best case, it also achieves a ~0.7% improvement in data-mining workload compared to HULA.

IV. Conclusions

In this paper, we present a scalable and efficient W-ECMP load-balancing scheme in data center using p4 programmable data planes. Because of the flexibility of P4, our W-ECMP uses a maximum utilization table as weights to determine the routing probability for each path. It applies this probabilistic selection to avoid always choosing the best path with the least congestion state, which could cause congesting of the best path in a short period of time if the congestion status could not be renewed in time. For real-time congestion information, our scheme encapsulates a self-defined header in regular traffic, which increases update frequency under higher network loading. Without using an independent probe, the bandwidth of the link can be used more efficiently. In addition, W-ECMP reversely collects the data to reduce the memory space that is stored for feedback. The experimental results verify that the proposed mechanism is effective in load balancing against other schemes. It is simple enough to be implemented and deployed on a data center using programmable data planes.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," Proceedings of the ACM SIGCOMM Computer Communication Review, vol 38 no 4, pp. 63-74, 2008

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network." Proceedings of the ACM SIGCOMM Computer Communication Review, vol 39 no 4, pp. 51-62, 2009

[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," Proceedings of 7th USENIX Conf. Netw. Syst. Design Implement., pp. 19-19, 2010

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined WAN," Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 3-14, 2013

[5] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters." Proceedings of the ACM SIGCOMM Computer Communication Review, vol 44 no 4, pp. 503-514, 2014

[6] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," Proceedings of the Symposium on SDN Research, pp. 51-62, 2016.

[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, and G. Varghese, "P4: Programming protocol-independent packet processors," Proceedings of the ACM

[8] SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87-95, 2014

[9] N. Dukkipati, and N. McKeown, "Why flow-completion time is the right metric for congestion control," Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 36, no. 1, pp. 59-62, 2006

[10] "The P4 Language Specification, Version 1.0.4," https://p4lang.github.io/p4-spec/p4-14/v1.0.4/tex/p4.pdf, 2017

[11] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," Proceedings of the ACM SIGCOMM Computer Communication Review, vol. 37, no. 2, pp. 51-62, 2007

[12] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," Proceedings of the Symposium on SDN Research, 2015

[13] Y. Iozzelli, L. Rizzo, and G. Lettieri, "Performance improvements on the P4 software switch," https://core.ac.uk/download/pdf/79622350.pdf 2016

[14] H. T. Dang, H. Wang, T. Jepsen, G. Brebner, C. Kim, J. Rexford, R. Soulé, and H. Weatherspoon, "Whippersnapper: A P4 Language Benchmark Suite," Proceedings of the Symposium on SDN Research, pp. 95-101, 2017