An Adaptive Multipath Routing Algorithm for Maximizing Flow Throughputs

Yusuke Shinohara, Yasunobu Chiba, Hideyuki Shimonishi
System Platforms Research Laboratories, NEC Corporation
Kanagawa, Japan

Abstract— In datacenter networks, as the communication intensive middleware such as Hadoop being popular, the traffic among servers is heavily increasing and its traffic patterns vary from time to time. Conventional tree network topologies and routing mechanisms with single path routing will cause congestion along oversubscribed links. Thus, load-balancing path selections are needed to alleviate congestion and improve application performance. Multipath routing algorithms can distribute traffic over diverse paths optimally than simple solutions like ECMP. They, however, request considerable computations, i.e. frequently updating the path cost tree and dynamically optimizing path selections, thus they do not adapt large scale datacenter networks. Multinomial Logit Based (MLB) routing, a lightweight multipath routing algorithm, has been proposed as a countermeasure. However, it is not always efficient against various network topologies and varying traffic patterns unless its control parameter is adaptively chosen. In this paper, we present an automatic parameter tuning method, which estimates the path cost of individual flows in the network and dynamically tunes the traffic dispersion parameter to reduce the total path cost of all flows. In our experiments with real OpenFlow switches, we exhibit that our method optimally tunes the parameter and maximize the total throughput of the flows.

Keywords-component; Datacenter network; Multipath Routing;

I. INTRODUCTION

Datacenter networks need to provide enough communication capability among servers to maximize server utilization and thus total cost/power efficiency. Since the amount of traffic among servers is increasing due to recent cloud middleware like Hadoop, and their patterns vary time to time, thus improving robustness of datacenter network against the changes of traffic pattern is a challenging problem. to improve total datacenter efficiency.

A tree-based topology with routing protocols based on a single-shortest path method causes severe convergence of packets to the same links, and this convergence results in traffic congestion. Constructing datacenter networks with a topology that has multiple path candidates and multipath routing will be a key technology to solve this issue.

Some other methods have been developed to ease traffic congestion and to achieve load balancing for traffic. ECMP [1][2] is a well known technology to distribute traffic to multiple equal cost links. ECMP is used in the research of datacenter network architectures [3][4][5]. However, ECMP does not optimize the traffic distributions to equal and non-equal paths accounting varying traffic conditions Adapting to varying traffic conditions is important because the traffic pattern in the datacenter networks changes dynamically, for example, traffic demands in shuffle phase and map phase of Hadoop are quite different. K-Shortest Path (KSP) [6] can dynamically select paths with the periodically updated path cost tree, but its computational cost to construct the tree is huge. Specifically, its computational complexity is $O\left(N \times n^3\right)$, where N is the number of paths, and n is the number of switches. Hence, KSP cannot adapt to the network conditions in large datacenter networks.

We presented a new multipath routing method, called multinomial logit based (MLB) routing [7], and its application to flow-based networks [9]. MLB routing is based on a multinomial logit model (MNL) [8]. The MNL is a classical model of random utility theory, which is used to assess the individual's choosing behavior. MLB routing computes a weight matrix based on all link costs for all paths that can be accounted for. According to the weight matrix, MLB routing stochastically selects a path of traffic in a hop-by-hop manner. Therefore, MLB routing can statistically distribute the traffic in a flow-by-flow manner so that various network paths are efficiently utilized. Moreover, the weight matrix and the path selection probability of MLB routing can easily be constructed, so the system can adapt to network conditions in large scale networks. However, MLB routing has an important parameter that affects its performance because the parameter affects the path diversity of our method. Thus, MLB routing cannot always achieve efficient routing.

In this paper, we describe an automatic parameter tuning method that estimates the path cost of flows and that controls the traffic dispersion parameter to reduce the path cost.

The rest of the paper is organized as follows. In the next section, we describe the MLB routing model in detail and clarify the structure of the system. Section III describes our method for the parameter tuning. In section IV, we describe out implementation of the method on OpenFlow [10] switches and controller. Section V evaluates its performance through an experimental evaluation. Section VI discusses the overhead of out implementation. Finally, we conclude and mention future work in Section VII.

II. MULTINOMIAL LOGIT BASED ROUTING

In this section, we present the MLB routing method in detail. We first explain a brief summary of the multinomial logit model (MNL), which is well known in random utility theory. We then describe how to apply the MNL to the routing system.

A. Multinomial Logit Model

Let us suppose a situation where an individual is going to choose an alternative j from an alternative set $N \in \{1,2,...,J\}$. Furthermore, we assume that the individual recognizes the following utility when it chooses the alternative j and when the individual is supposed to choose the alternative to maximize its utility. The objective is to analyze which alternative the individual would choose to maximize the utility.

$$U_j \overset{def}{=} \{\text{Utility when he chooses the alternative } j\}. \quad (1)$$

In the MNL, random factors in the utilities are assumed to be independent random variables from the identical Gumbel distribution with parameters (α, γ). Note that α is the location parameter and that γ is the scale parameter. Therefore, the probability that the individual chooses the alternative j is as follows:

$$P_j = \frac{\exp[\gamma \cdot V_j]}{\sum_{j \in N} \exp[\gamma \cdot V_j]}, \quad (2)$$

where V_j is a function estimated by a researcher. V_j depends on the observed characteristics of the alternative j. When $\gamma = \infty$, only the best alternative is used, and if $\gamma = 0$, then all alternatives are used with equal possibility.

B. MLB Routing

To describe the MLB routing protocol, we first define a system model assumed in this study. We assume a network of nodes connected by directed links as a directed graph $G = \{N, E\}$. N is the set of nodes in the networks and E is the set of edges. Furthermore, the e_{ij} is a edge from vertex i to vertex j and the c_{ij} index is a cost metric on e_{ij}. We can arbitrary define c_{ij} as the sum of queuing delay and link delay, loss rate, utilization of the link. This paper uses link utilization as the link cost to maximize the total utilization of the datacenter networks. For the rest of this paper, we use the terms nodes (links) and vertices (edges) interchangeably. Each node v can act as a traffic source and/or sink. Finally, every node v has a set of $Z(v)$ neighbors denoted by $nbr(v)$.

Let us consider any packets p at an origin node o going to destination node d. There is not only the shortest path but also paths to reach destination d, and MLB routing can uses all the possible paths. However, this does not mean all routes paths are equally used; naturally, the shortest path (smallest cost path) should be used most often, and the longer higher the path cost of the path becomes, the less the path is used. To incorporate this feature, let us define the probability that packet p uses the r-th path P_r^{od} as follows:

$$P_r^{od} = \frac{\exp[-\gamma \cdot C_r^{od}]}{\sum_{r \in \Phi^{od}} \exp[-\gamma \cdot C_r^{od}]}, \quad (3)$$

where C_r^{od} denotes the cost of the r-th path between nodes o and d, and Φ^{od} is the alternative set of paths between these nodes.

C. Structure of MLB Routing

We herein describe the structure of the MLB routing system. First, we explain how to allocate packets to various paths.

Fortunately, the following equations demonstrate that a path generated by equation (3) is equivalent to the following Markov assignment flow:

$$p(j \mid i) = \exp[-\gamma \cdot c_{ij}] \cdot \frac{W_{jd}}{W_{id}}, \quad (4)$$

$$W_{id} \overset{def.}{=} \sum_{r \in \Phi^{id}} \exp[-\gamma \cdot C_r^{id}], \quad (5)$$

where $p(j \mid i)$ is a transition probability that a packet in node i selects the next hop j. This formulation satisfies the condition

$$\sum_{j \in nbr(i)} p(j \mid i) = \frac{\sum_{j \in nbr(i)} \exp[-\gamma \cdot c_{ij}] \cdot W_{jd}}{W_{id}} = 1, \quad (6)$$

that is required to be the conditional probabilities. Moreover, equations (3) and (4) cause the same results as proved in [7].

The existence of an equivalent Markov assignment implies that we can apply the simple rule to all nodes. In other words, if each node chooses the next node to pass through the packet following equation (4), its path consequently equals the assignment discussed in section II.B.

Note that $p(j \mid i)$ does not depend on the node of origin. No subscripted notations related were found to be related to the origin in equation (4). Thus, if each node follows the aforementioned Markov assignment, every packet that starts from an arbitrary node spontaneously is held by equation (3).

To determine the value of $p(j \mid i)$, we have to calculate the W_{id} that is included in $p(j \mid i)$. As in equation (5), all of the paths from node i to d have to be accounted to obtain in W_{id} deal with all of the paths from node i to d are accounted for, which is naturally difficult.

Now let us consider matrix **A** with N rows and N columns, whose $[i, j]$ element is given as follows:

$$a_{ij} = \begin{cases} \exp[-\gamma \cdot c_{ij}] & (\text{if } e_{ij} \text{ exists}) \\ 0 & (\text{Otherwise}) \end{cases}, \quad (7)$$

$$\mathbf{W} = [\mathbf{I} - \mathbf{A}]^{-1} - \mathbf{I}. \quad (8)$$

Note that matrix **A** must satisfy the Hawkins-Simon condition to use equation (8) and that [i, d] element in W is equivalent to W_{id}. **W** can be derived using an inverse matrix, which can be computed easily, and thus now we can easily derive W_{id} by just single inverse matrix operation.

MLB routing optimizes the multipath selections against varying traffic conditions by frequently measuring link cost c_{ij} and updating W_{id}. Then, the actual routing operation is quite easy, i.e., when a packet arrives at a node, the packet is just sent to the next node according to the probability W_{id}.

D. Flow-by-flow path selection

The above mentioned packet-by-packet path selection causes packet reordering in a flow and hence TCP throughput degradation, thus we applied MLB routing to flow-by-flow path selection [9]. Rather than routing a packet individually at a node, when a packet of a new flow comes to the network, its end-to-end path is calculated by applying MLB and flow table is set up at the nodes on the path so that the packets in the same flow go through the same path.

III. AUTOMATIC PARAMETER OPTIMIZATION

In MLB routing, we can compute **W** using an inverse matrix computation, and the weight matrix and packet forwarding table can be easily constructed. Thus, MLB routing can easily be applied to large networks. However, its traffic dispersion parameter, γ, has an impact on routing performance, and there is no indication to optimally choose the parameter. As we mentioned before, when $\gamma = \infty$, only the smallest cost path, is cshosen, and if $\gamma = 0$, then all alternative paths are used with equal likelihood. The optimum degree of traffic dispassion highly depends on network topology and traffic load, single parameter value cannot always optimize the routing performance.

We herein describe an automatic parameter tuning method to distribute traffic optimally. Our method dynamically updates γ with the following three steps every time interval I_γ.

1. Calculate a temporary lower limit of γ considering the path cost
2. If necessary, modify the lower limit of γ to satisfy the condition that the weight matrix is routable.
3. Calculate γ according to minimize the average path cost of flows and satisfy the lower limit of γ.

A. Setting Lower Limit of γ

As we described in II.A, MLB routing could select redundant paths with very small γ. We set the lower limit of γ to reduce the possibility.

Let's assume that the path cost from source node *o* to destination node *d* along the shortest path is C_{sp}^{od} and that the probability to select the shortest path is P_{sp}^{od}. To suppress the probability to select a path whose path cost is $k_p \cdot P_{sp}^{od}$ to $k_c \cdot C_{sp}^{od}$, following equations must be satisfied:

$$k_p \cdot P_{sp}^{sd} = kp \cdot \frac{\exp[-\gamma \cdot C_{sp}^{od}]}{\sum_{r \in \Phi^{od}} \exp[-\gamma \cdot C_r^{od}]} < \frac{\exp[-\gamma \cdot k_c \cdot C_{sp}^{od}]}{\sum_{r \in \Phi^{od}} \exp[-\gamma \cdot C_r^{od}]}$$

,

$$\gamma > \frac{-\log k_p}{(k_c - 1) \cdot C_{sp}^{od}}. \quad (9)$$

By applying the equation to all source-destination pairs, the following equation is attained:

$$\gamma > \frac{-\log k_p}{(k_c - 1) \cdot C_{sp}}, \quad (10)$$

where C_{sp} is the average path cost of all source-destination pairs. We can get the shortest path cost from the source node to the destination node by selecting the next hop switch that has the largest transition probability on switches from the source node to the destination node.

B. Modifying Lower Limit of γ Considering Weight Matrix

Our method modifies the lower limit of γ obtained from equation (10) so that the weight matrix is routable.

As we mentioned in section II.C, we can compute the weight matrix using equation (8), but matrix **A** must satisfy the Hawkins-Simon condition. To satisfy this condition, matrix **A** must satisfy the following Solow condition

$$\sum_{i=0}^{N} a_{ij} < 1, \quad j = 1,2,\cdots,N. \quad (11)$$

Then, the following equation must be satisfied.

$$\max_j \left[\sum_{i=0}^{N} a_{ij} \right] = \max_j \left[\sum_{i=0}^{N} \exp[-\gamma \cdot c_{ij}] \right] < 1. \quad (12)$$

Our method judges whether matrix **A** with the lower limit of γ satisfies equation (12) or not. If matrix **A** does not satisfy the equation, our method increases γ to satisfy the equation, and the nearest value to the lower limit of γ is determined in III.A as follows:

$$th < \max_j \left[\sum_{i=0}^{N} \exp[-\gamma \cdot c_{ij}] \right] < 1, \quad (13)$$

where *th* is the threshold for the traffic dispersion parameter. We can use binary search to satisfy equation (13). With this method, the lower limit of γ satisfies the Hawkins-Simon condition and can be set to the nearest value determined in III.A.

C. Decision of γ Considering Average Path Cost of Flows

Our method estimates the future average path cost, i.e. the path cost when the new weight matrix is applied, of flows and decides γ based on the path cost.

To estimate the future average path cost of flows, our method estimates the future link cost. The probability p_{odij} that

the link e_{ij} is used by a flow from source o to destination d is computed as follows:

$$p_{odij} = \frac{W_{oi} \cdot W_{id}}{W_{od}} \cdot p(j \mid i) = \exp[-\gamma \cdot c_{ij}] \cdot \frac{W_{oi} \cdot W_{jd}}{W_{od}}. \quad (14)$$

Moreover, the amount of traffic on e_{ij} used by the flow from source o to destination d, l_{ij}, can be estimated as follows:

$$l_{ij} = \sum_{o} \sum_{d} \left(p_{odij} \cdot T_{od} \right), \quad (15)$$

where T_{od} is a traffic matrix from source o to destination d. The link cost is defined as the link utilization, and the link utilization is computed by dividing the amount of traffic by the bandwidth of the link. Thus, our method can estimate the future link cost c'_{ij} of link e_{ij} as follows:

$$c'_{ij} = c_{ij} + l_{ij} / (bw_{ij} \cdot I_{\gamma}), \quad (16)$$

where bw_{ij} is the bandwidth of link e_{ij}.

Using the future link cost, our method estimates the future average path cost of flows. If traffic matrix **T** is input to the network, the future average path cost of flows from source o to destination d, C_{od}, can be derived as follows:

$$C_{od} = \sum_{i} \sum_{j} \left(p_{odij} \cdot c'_{ij} \right). \quad (17)$$

By applying equation (17) to all source-destination pairs, our method can estimate the future average path cost of flows C_{ave}. Using γ, which minimizes C_{ave}, can distribute traffic optimally and avoid redundant paths. The reason is that, if particular links are loaded, $p_{odij} * c'_{ij}$ of the link would increase and then C_{ave} would increase. However, if redundant paths are selected often, the average value of p_{odij} would increase. Our method decides γ using C_{ave}. Our method uses γ that is used at the point $\gamma \cdot (1+x)$ and $\gamma \cdot (1-x)$ as the candidates of γ. The candidate that computes the lowest value of C_{ave} is used as γ in the next term.

With the aforementioned update of γ, our method can reduce the path cost of flows.

IV. Implementation

In this section, we introduce our implementation of the proposed method on an OpenFlow controller. Note that original MLB routing has already been implemented on an OpenFlow controller [9]. An important additional function to the OpenFlow controller for our method is the computation of traffic matrix **T**.

To compute traffic matrix **T**, our method manages flow information that includes identifiers of ingress and egress switches for the flows. Our method obtains size of the flow by sending the flow statistics request message to the egress switches and receiving the flow statistics reply message from them. The sum of the difference between the size and the one obtained from the flows that passed the same ingress switch o and egress switch d are the element of traffic matrix, T_{od}.

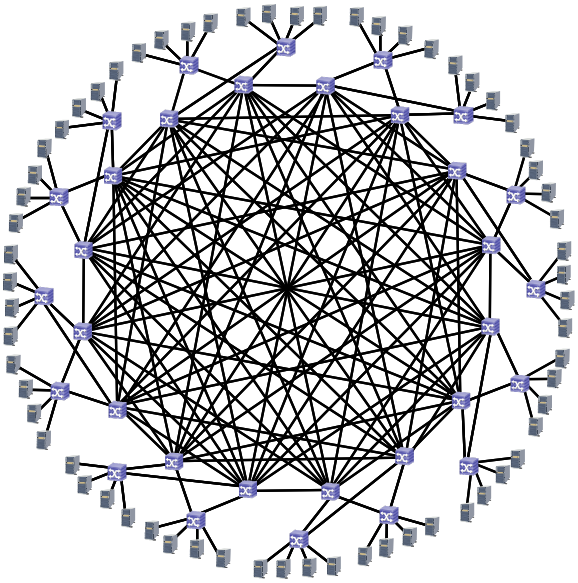

Figure 1. Enhanced four-dimensional hypercube topology

However, obtaining the size of all flows would not scale well. Because collecting a large number of flow statistics imposes high processing cost to the controller and switches. Thus, our method obtains the size of the F flows sampled on a round robin fashion from each egress switches every time interval I_{γ}. With the size of the flows, our method estimates traffic matrix from source o to destination d as following moving average:

$$T_{od} = \sum_{i=0}^{s} DS_{odi} \Big/ s. \quad (18)$$

Note that s is the shift parameter of the moving average and DS_{odi} is the sum of the difference between the size and before obtained size of the flow that pass ingress switch o and egress switch d at i-th previous sampling.

V. Performance Evaluation

In this section, we experimentally evaluated the performance of our method in a real network. Though we omitted the detail, we also confirmed its effectiveness on various topologies such as multi rooted fat-tree [3] by comparing with the other multipath routing algorithms such as ECMP and MLB routing with static γ through computer simulations.

A. Experimental Environment

We evaluated the performance of our method on an OpenFlow network. We have implemented the method onto our OpenFlow controller [11]. We used our own prototype OpenFlow switches based on NEC IP8800 Ethernet switch as core switches and Open vSwitches [12] as edge switches. We used virtual machines (VMs) for traffic source and destination. Each server had four VMs and one edge switch.

B. Experimental Conditions

In this experiment, we used enhanced four-dimensional hypercube topology, as shown in Fig. 1. In the original hypercube topology, core switch S connects to the switch

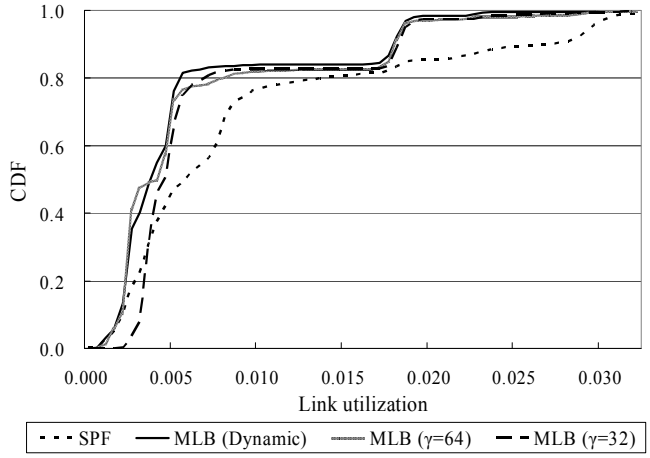

Figure 2. CDF of link utilization

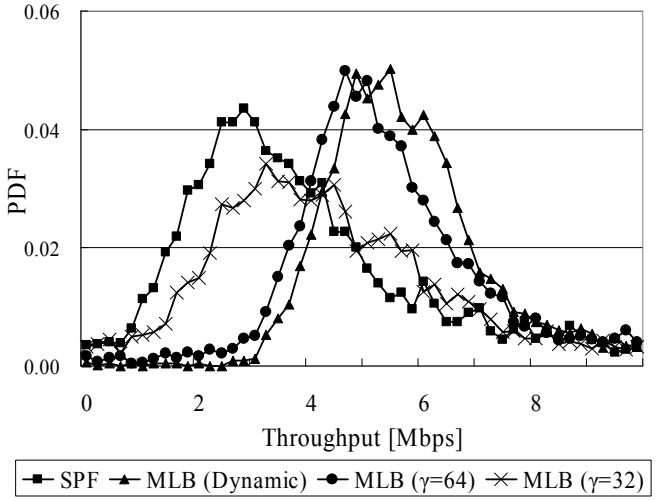

Figure 3. PDF of throughput of each thread

whose hamming distances from switch *S* is one. However, in the enhanced hypercube topology, core switch *S* connects to the switch whose hamming distances is one and two, to take advantage of high port density of the switches. An edge switch is connected to two core switches that are most distant each other in hamming distance to decrease the average hop counts. A core switch is connected to two edge switches. In this topology, we used sixteen edge switches, sixteen core switches, and sixty four VMs. All switches were connected with 1 GbE links. Each VM had eight virtual hosts, which randomly selects a destination host with randomly selected transmission time to *t* (*t* = 1 – 10s) using TCP. After finishing the transmission, each virtual host randomly selects the destination, transmission time, and sent TCP traffic again. Note that the packet size was set to 1.5 KB and that the experimental results were the average of 10 experiments whose duration was 600 s.

We compared our method with the original MLB and shortest path first (SPF) routing. From the several trial runs, we set the parameters as follows: the bias parameter B = 2.0, the update interval of the flow forwarding probability table I = 1.0[s] (original MLB routing and our method), the parameter of

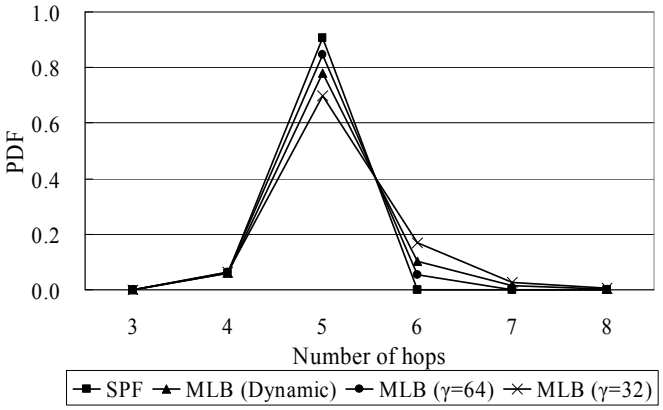

Figure 4. PDF of number of hops

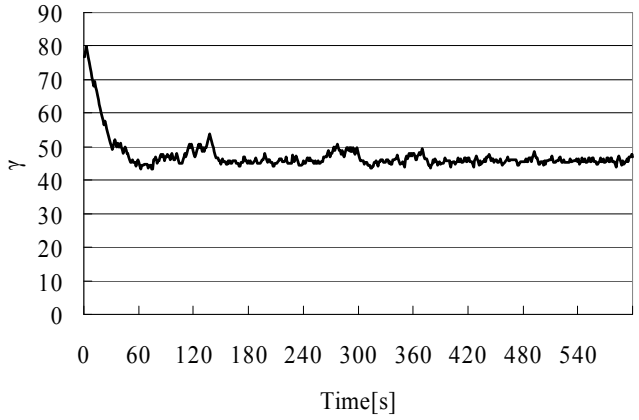

Figure 5. Change in γ over time

the Gumbel distribution $\gamma = 32$ and 64 (original MLB routing), an update interval of γ I_γ =1.0[s], (k_c, k_p) = (1.3, 0.001), *th*=0.99, x=0.05, *F*=10, *s*=10 (our method).

C. Experimental Results

Fig. 2 shows the cumulative distribution function (CDF) of the link utilization of all switches. Fig. 3 shows the probability density function (PDF) of throughput of all flows. Figure 4 shows the PDF of the number of hops. Fig. 5 shows the variation of γ of our method.

Fig. 2 shows that the original MLB routing and our method could distribute traffic. Fig. 3 shows that the original MLB routing and our method avoided to select bottleneck links and utilized less loaded links, thus more links were highly utilized than SPF did. Our method used the network efficiently. From Fig. 2, the original MLB routing achieved a lower average throughput than our method because the original MLB routing with large γ tends to select the shortest paths for flows. Meanwhile, our method with small γ selects redundant paths for flows. From Fig. 4, our method with small γ ($\gamma = 32$) selects the redundant paths for flows and increases the number of hops of flows. The flow whose path is too redundant degrades not only its throughput but also other flow's

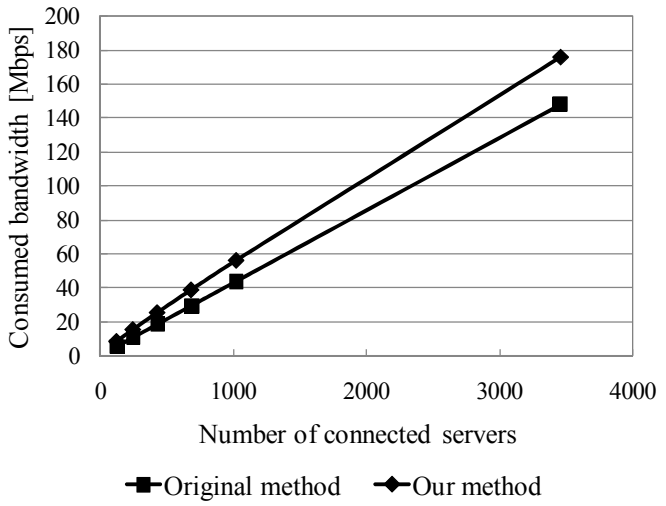

Figure 6. Consumed bandwidth by flow statistics messages

throughput because it consumes bandwidth unnecessarily. Our method did not select the redundant paths for flows. From Fig. 5, our method stabilized γ around 45. With the value of γ, which is larger than 32 (redundant) and smaller than 64 (congested), our method achieved both load balancing and avoiding the redundant paths. It should be noted that our method effectively avoided or even detoured the bottleneck links and thus the portion of flows having very low throughput was very small. In other words, it has good throughput fairness among flows. Indeed, Fig.3 shows that almost no flows experienced throughput less than 3Mbps in our method, whereas SPF made roughly half of the flows get smaller throughput than 3Mbps.

VI. DISCUSSION

In this section, we evaluated the overhead of implemented our method in terms of network cost. Comparing with original MLB routing, our method needs to compute by obtaining the size of flows.

Fig. 6 shows that the consumed bandwidth by control signals of original MLB routing and our method in multi rooted fat-tree topology. Note that control signals include packet-in, modify state and flow removed messages per flow and port statistics request and reply messages per link in orignal MLB routing. In addition to control signals in original MLB routing, control signals include flow statistics request and reply messages per edge switch in our method. The lengths of packet-in, modify state and flow removed, port statistics request, and port statistics reply, flow statistics request and flow statistics reply messages are 98, 210, 210, 82, 178, 194 and 210 bytes, respectively. We set the parameters as follows: an update interval of the flow forwarding probability table I = 1.0[s], an update interval of γ I_γ=1.0[s], F=30, s=10. Moreover, we assumed that each host generates 2 flows in one second.

From Fig. 6, the difference between bandwidth consumed by original MLB routing and our method gets smaller in larger network. In the most largest network, the difference is only 15%. This is because the implemented our method estimates traffic matrix by obtaining the number of bytes for F flows on a round robin fashion from each egress switches every I_γ.

VII. CONCLUSIONS

In this paper, we explained the needs for effective multipath routing algorithms, as well as MLB routing algorithm. MLB routing has an important parameter, the traffic dispersion parameter γ, which seriously affects the performance of MLB routing. We designed the automatic parameter tuning method to achieve efficient routing at any time. Numerical results show that our method can achieve load balanced path selection and higher throughput than shortest path forwarding.

Our future work includes experiments using other multipath routing algorithms in real cloud testbeds and an evaluation of its scalability.

ACKNOWLEDGMENT

This work was partly supported by the Ministry of Internal Affairs and Communications (MIC), Japan.

REFERENCES

[1] J. Moy, OSPF version 2, RFC2328, 1988.

[2] Y, Rekhter and T. Li, "A Border Gateway Protocol version 4," RFC1771, 1995.

[3] A. Greenberg, J. R. Hamilton and N. Jain, "VL2: A Scalable and Flexible Data Center Network," In *Proceedings of SIGCOMM 2009*, 2009.

[4] Chuanxiong Guo et al., "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," In *Proceedings of SIGCOMM 2009*, 2009.

[5] Radhika Niranjan Mysore, et al., PortLand, "A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," In *Proceedings of SIGCOMM 2009*, 2009.

[6] R. Bellman and R. Kalaba, "On kth Best Policies," In Journal of SIAM, vol. 8, no. 4, pp. 582-588, 1999.

[7] Y. Honma, M. Aida, H. Shimonishi and A. Iwata, "A New Multi-path Routing Methodology Based on Logit Type Assignment," In FutureNet II, 2009.

[8] K. Train, "Discrete Choice Methods with Simulation," Cambridge University Press, 2003.

[9] Y. Shinohara, Y. Chiba, H. Shimonishi, Y. Honma and M. Aida, "An Efficient Multipath Routing Algorithm for Datacenter Networks and its Implementation on OpenFlow-based Network," IEICE, vol. 109, no. 448, NS2009-164, pp. 13-18, March 2010.

[10] Flow Switch Specification Version 1.1.0, http://www.openflowswitch.org/documents/openflow-spec-v1.1.0.pdf, February 2011.

[11] H. Shimonishi and S. Ishii, "A network OS for integrated control plane and its application to OpenFlow controller," IEICE, vol. 109, no. 448, NS2009-162, pp. 1-6, March 2010.

[12] Open vSwitch 1.0, http://openvswitch.org/, May 2010.