Distributed Multipath Routing for Data Center Networks based on Stochastic Traffic Modeling

Omair Fatmi and Deng Pan
School of Computing and Information Sciences
Florida International University
Miami, Florida

***Abstract*—Modern data center networks often adopt multipath topologies for greater bisection bandwidth and better fault tolerance. However, traditional distributed routing algorithms make routing decisions based on only packet destinations, and cannot readily utilize the multipath feature. In this paper, we study distributed multipath routing for data center networks. First, to capture the time varying and non-deterministic nature of data center network traffic, we present a stochastic traffic model based on the log normal distribution. Then, we formulate the stochastic load-balanced multipath routing problem, and prove that it is NP hard for typical data center network topologies, including the fat tree, VL2, DCell, and BCube. Next, we propose our distributed multipath routing algorithm, which balances traffic among multiple links by minimizing the probability of each link to face congestion. Finally, we implement the proposed algorithm in the NS2 simulator, and provide simulation results to demonstrate the effectiveness of our design.**

I. Introduction

Data centers of today contain no less than several hundreds and thousands of servers [4]. To connect such a large number of servers, data center networks are usually built using multi-rooted hierarchical topologies. For example, data centers based on the fat tree topology [3] contain a layer of hosts, edge switches, aggregation switches, and core switches, respectively, as shown in Fig. 1. Such topologies provide multiple possible paths between a pair of hosts. For example, in the 4-pod fat tree network in Fig. 1, there are four different paths between hosts 1 and 2. Hence, an efficient routing strategy is necessary to utilize the multipath feature of modern data center networks.

However, traditional routing algorithms such as distance vector and link state do not support multipath routing well. The reason is that they make routing decisions based on only packet destinations, and hence all packets to the same destination take the same path. The inability to support multipath may greatly reduce the routing efficiency of a data center network, as many links would remain under-loaded while traffic would be directed to a small group of links, resulting in congested hot spots. Therefore, in order to fully utilize the network capacity, the routing algorithm should efficiently balance the traffic among all available paths while fulfilling the traffic demand.

There exist some multipath routing solutions in the literature [2], [11], [17], [22]. Equal Cost MultiPath (ECMP) [17] provides multipath forwarding by performing static load splitting among the flows. Each node running ECMP has multiple next hops configured for a given destination. When a packet arrives, the node forwards the packet according to the hash of selected fields in the packet header. ECMP can split traffic to a destination across multiple paths, but such traffic splitting is restricted to cases where equal cost paths exist. Adaptive Multipath Routing (AMP) [11] introduces an efficient signaling architecture by reducing the view of the network for each node to a local scope, i.e. each node is informed about the congestion on the links that are in its immediate neighborhood. The information is distributed with the help of a unique signaling mechanism known as the backpressure mechanism, in which a node experiencing congestion sends out the congestion information to its immediate neighbors. As a result, the neighbors offload some of their paths containing that congested link by selecting different paths. The neighboring nodes eventually send the congestion information to their adjacent nodes, in proportion to their contribution in the congestion. Although AMP achieves multipath traffic engineering utilizing the link information, it supports only congestion alleviation, but not congestion prevention at the beginning of path selection. Hedera [2] is a scalable and dynamic flow scheduling system for data centers. It takes a centralized approach in which a single scheduler has global knowledge of all the flows in the network. It schedules the flows over different available paths in the network, and its scheduling decisions are based on large flows as they are presumably responsible for much of the traffic sent across the network. By comparison, our approach is a distributed one and thus more scalable. Maximum Alternative Routing Algorithms (MARA) [22] construct directed acyclic graphs (DAGs) that include all edges present in the network, so that the maximum number of alternate paths should be available between a pair of nodes. Similarly, MARA works in a centralized mode and needs the entire network topology.

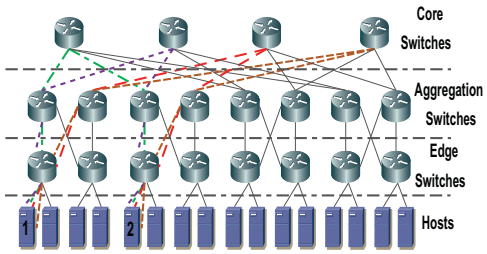


Fig. 1. Hierarchical 4-pod fat tree topology.

Recent studies have shown that traffic patterns in data center networks are not deterministic [4]. Instead, they are stochastic in nature and are usually modeled by heavy-tailed distributions [5], based on how packet inter-arrival times and flow sizes are distributed [19], [23]. It is due to the fact that although most of the traffic inside a data center network have flows that are reasonably small, some flows are relatively large because of sudden data bursts.

Considering the above mentioned fact, our solution takes into account the stochastic nature of traffic in typical data center networks, and makes routing decisions based on those traffic patterns. Our algorithm takes a fully distributed approach, which means that instead of a centrally controlled path selection mechanism such as that in Hedera [2], each node in the path of a flow autonomously selects an optimal link. Based on the stochastic traffic model, the optimal link is determined by comparing the probabilities of the links to become overloaded, and selecting the link with the lowest probability. The use of the above distributed routing approach can prevent centralized control limitations, such as a single point of failure or performance bottleneck [4]. Also, our algorithm avoids the division of a single flow into multiple paths, because 99% of the traffic flows in data centers are TCP flows [12], which suffer performance degradation when packet delivery is not in order [16].

Our main contributions in this paper can be summarized as follows. First, we present a stochastic traffic model that uses the log normal distribution to capture the non-deterministic nature of data center network traffic. Second, we formulate the load-balanced multipath routing problem, and prove that it is NP hard for typical data center network topologies, including the fat tree [3], VL2 [13], DCell [15], and BCube [14]. Third, we propose a novel distributed multipath routing algorithm, in which each node makes independent decisions to balance traffic based on the stochastic traffic model. Finally, we conduct extensive simulations in different topologies including the fat tree and VL2. Our results show substantial improvement in the end-to-end packet delay and bandwidth utilization, not to mention a highly load-balanced network as compared to other routing algorithms.

The remaining of this paper is organized as follows. Section II presents the formal definition of the load-balanced multipathing problem, and proves that it is NP hard for typical topologies. Section III describes the stochastic traffic model. Section IV proposes the distributed multipath routing algorithm. Section V shows the simulation results. Finally, Section VI concludes the paper.

II. Problem Formulation

In this section, we describe our stochastic traffic model, formulate the stochastic load-balanced multipath routing (SLMR) problem, and prove that it is NP-hard for typical data center network topologies.

A. Stochastic Traffic Model

Because recent studies indicate that data center traffic can be best modeled by heavy-tailed distributions [5], we use the log normal distribution [24] to represent the dynamic bandwidth demand of a flow. Specifically, for a flow f_i, its bandwidth demand d_i follows a log normal distribution, i.e. $d_i \sim \ln\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i is the mean and σ_i is the standard deviation.

Further, the aggregated demand of multiple individual flows can also be approximated by a log normal distribution [9]. Assuming that a link l has n flows $f_1, \ldots, f_n$ traversing it, denote the aggregated demand of all the flows as $d_l = \sum_i d_i$, and $d_l \sim \ln\mathcal{N}(\mu_l, \sigma_l^2)$, in which the calculation of μ_l and σ_l is explained in the following [9].

$$\mu_l = \mu_1 + \mu_2 + \ldots + \mu_n \tag{1}$$

$$\sigma_l^2 = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_n^2 \tag{2}$$

Note that $\ln(d_l)$ follows a standard normal distribution [24], i.e. $\ln(d_l) \sim \mathcal{N}(M_l, D_l^2)$, in which the mean M_l and stand deviation D_l are calculated as follows.

$$D_l = \sqrt{\ln(\frac{\sigma_l^2}{\mu_l^2} + 1)} \tag{3}$$

$$M_l = \ln\mu_l - \frac{D_l^2}{2} \tag{4}$$

Knowing the log normal distribution of the aggregate flow of a link, we can easily calculate its probability of oversubscription. Oversubscription happens if the aggregated bandwidth demand is greater than the product of the link capacity and a threshold defined in the service level agreement (SLA). Denote the capacity of link l as c_l and the oversubscription percentage threshold as T. Utilizing the cumulative distribution function of the standard normal distribution by its relationship with the log normal distribution [24], we can calculate the oversubscription probability as follows.

$$\Pr\{d_l > Tc_l\} = 1 - \Phi\Big(\frac{\ln(Tc_l) - M_l}{D_l}\Big) \tag{5}$$

where $\Phi()$ is the cumulative distribution function of the standard normal distribution.

B. Formulation of Stochastic Load-Balanced Multipath Routing

Model a data center network as a graph $G = (V, E)$. The vertex set $V = H \cup S$ includes a set of hosts $H = \{h_i\}$ and a set of switches $S = \{s_j\}$. The edge set $E = \{(a_x, a_y)\}$ includes links (a_x, a_y), where a_x and a_y are either two switches or a switch and a host. Each link (a_x, a_y) has a non-negative capacity $c_{(a_x,a_y)} \geq 0$ indicating its available bandwidth.

Consider a set of flows $F = \{f_1, \ldots, f_n\}$. Each flow is defined as a four-tuple $f_k = (u_k, v_k, \mu_k, \sigma_k)$, where $u_k \in H$ is the source host, $v_k \in H$ is the destination host, and $\ln\mathcal{N}(\mu_k, \sigma_k)$ is the log normal distribution describing the bandwidth demand of the flow. Use $r_k(a_x, a_y)$ to indicate whether flow f_k is routed via link (a_x, a_y). To avoid TCP performance degradation [16], we do not allow a single flow to be split among multiple routes, and thus $r_k(a_x, a_y)$ is either 1 or 0. However, different flows between the same pair of hosts are allowed to take different routes. For link (a_x, a_y),

use $d_{(a_x,a_y)}$ to represent its aggregated bandwidth demand, i.e. $d_{(a_x,a_y)} = \sum_{f_k} d_k \cdot r_k(a_x, a_y)$.

To achieve load balancing, we define two objectives. First, there should be a small probability of oversubscription, as it may result in packet loss. Second, the aggregated bandwidth demand on each link should have a small mean value, so that the average packet delay is small. Since there are two objectives, we assign each a weight factor [7], and minimize the maximum weighted sum of all the links. Therefore, the stochastic load-balanced multipath routing problem can be formulated as:

$$\textbf{minimize } maxWS$$

subject to the following constraints:

$$\forall (a_x, a_y) \in E, \alpha \Pr\{d_{(a_x,a_y)} > Tc_{(a_x,a_y)}\} + \beta \mu_{(a_x,a_y)} \leq maxWS \quad (6)$$

$$\forall k, \forall a_x \in V \setminus \{v_k, u_k\}, \sum_{a_y \in V} r_k(a_x, a_y) = \sum_{a_y \in V} r_k(a_y, a_x) \quad (7)$$

$$\forall k, \sum_{a_x \in V} r_k(v_k, a_x) = \sum_{a_x \in V} r_k(a_x, u_k) = 1 \quad (8)$$

Equation (6) defines the weight sum, in which the oversubscription probability $\Pr\{d_{(a_x,a_y)} > Tc(a_x, a_y)\}$ and the mean of the aggregated bandwidth demand $\mu_{(a_x,a_y)}$ can be calculated as in Section II-A. α is the weight for the former and β is the weight for the latter. Without loss of generality, assume that α and β are integers. Equation (7) states the flow conservation constraint, i.e. the flow values do not change at intermediate nodes. Equation (8) states the demand satisfaction constraint, i.e. the flow value at the source and destination is equal to the demand of the flow.

C. NP-hardness Proof

We now show that the above formulated problem is NP-hard for typical data center network topologies by reduction from the integer partition problem. In the following theorem, we first look at the popular fat tree topology.

Theorem 1. *The stochastic load-balanced multipath routing problem is NP-hard for the fat tree [3] topology.*

Proof: We prove the theorem by reduction from the integer partition problem [20]. An integer partition problem decides whether a set of integers $I = \{i_1, \ldots, i_n\}$ has a subset I_s whose sum is half of $\sum_{i_k \in I} i_k$, i.e.

$$\exists I_s \subseteq I, \sum_{i_k \in I_s} i_k = \sum_{i_k \in I \setminus I_s} i_k \quad (9)$$

To reduce the load balancing problem from the above integer partition problem, consider an instance of a partition problem with set I. From that, we construct an instance of the load-balanced multipath routing problem. First, set up a 4-pod fat tree network $G = (V, E)$, with the capacity of each link being infinite, i.e. $\forall (a_x, a_y) \in E, c(a_x, a_y) = \infty$. Fig. 2 illustrates the detail of one pod. Next, set up n flows

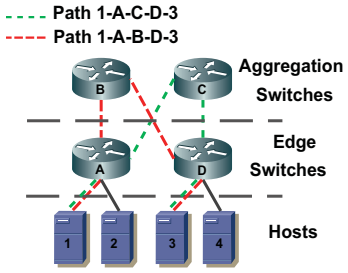


Fig. 2. Single pod from a 4-pod fat tree topology.

$f_1, \ldots, f_n$. For each flow f_k, its source host u_k is host 1, its destination host v_k is host 3, the demand mean $\mu_k = i_k$ where $i_k \in I$, and the demand standard deviation $\sigma_k = 0$.

As can be seen, there are only two paths between hosts 1 and 3: 1-A-B-D-3 and 1-A-C-D-3, as shown in Fig. 2. Further, since each link has infinite capacity, the oversubscription probability is zero.

Suppose that the integer partition problem I has a solution I_s as in Equation 9, then we can construct a solution for the load-balanced multipath routing problem to achieve an optimized objective of $maxWS = \beta \sum_{i_k \in I} i_k/2$ as follows. For each $i_k \in I_s$, we assign the corresponding flow f_k to the path 1-A-B-D-3; otherwise, if $i_k \in I \setminus I_S$, we assign the corresponding flow f_k to the alternative path 1-A-C-D-3. It can be easily verified that such a route assignment satisfies the flow conservation and demand satisfaction constraints as well.

Conversely, suppose that the load-balanced multipath routing problem has an optimized objective value of $maxWS = \beta \sum_{i_k \in I} i_k/2$. Note that the optimized objective must be an integer, since we do not allow flow splitting and $\sigma_k = 0$. Thus, we can construct a solution I_s for the integer partition problem as follows. For any flow f_k taking the path 1-A-B-D-3, assign the corresponding integer i_k to I_s. ■

Theorem 2. *The load-balanced multipath routing problem is NP-hard for the VL2 [13], BCube [14], and DCell [15] topologies.*

Proof: The proof is similar to that for the fat tree topology. The key is to find a network with only two disjoint paths between two selected hosts, and map the two sets of integers to the two sets of flows taking different paths. Due to space limitations, the detail proof is omitted. ■

III. Distributed Multipath Routing Algorithm

In this section, we present the stochastic load-balanced multipath routing (SLMR) algorithm. The design of SLMR is based on the fact that the traffic demand in data centers shows a stochastic pattern. Hence each link in the network has an oversubscription probability, based on the behavior of traffic traveling through it. Using this unique property, SLMR selects optimal paths by obtaining and comparing the oversubscription probabilities of the candidate links. Because data center networks are usually organized in a hierarchical structure, candidate links are defined as the links that lead

to the next hierarchical layer to the destination node. SLMR selects the candidate link with the minimum oversubscription probability. After a link is being selected, it is added to the optimal route and the packet will be sent to the next hierarchical layer until it reaches the destination. Finally, a route consists of a sequence of such locally selected links with small oversubscription probabilities, and all the following packets of the same flow will use this route. Due to this distributed approach of SLMR, each node only needs the information of the candidate links connecting its neighboring layer, instead of the global information of the network. In the following, we discuss the operation and implementation details of the proposed algorithm.

A. Algorithm Description

The operation of SLMR consists of two steps. The first step is to determine the routing direction of a flow and the set of candidate links. The direction can be either upstream or downstream, based on the network topology and the destination address, as hosts in data center networks usually have addresses based on their topological locations [6]. Such network topology information can be computed in advance during network initialization, and stored in each switch. For example, for a data center network implementing the fat tree topology, hosts in the same pod share the same sub-net address [3]. So, when the packet of a new flow arrives at one of the switches, the switch checks the destination address of the packet to see if it belongs to the same pod as the source or a different one. If it is destined to a different pod, then all the upstream links of that layer are considered as the candidate links. If the destination pod is same as the source pod, the algorithm identifies the candidate links using the destination and sub-net address of the pod.

The second step is to determine the oversubscription probabilities of all the candidate links, and assigning the link with the smallest probability to the flow. In this step, for all the candidate links, the current link load value is obtained and their oversubscription probabilities are calculated, using Equation (5). The algorithm then checks whether the obtained probabilities are within the oversubscription percentage threshold T. The threshold in our case is 99% [1]. This is to ensure that SLMR does not include oversubscribed links in the calculation of the optimal path. From all the calculated probabilities, the algorithm then finds the link with the smallest probability, by comparing the probabilities of all the candidate links. The link with the smallest oversubscription probability is then selected and added to the route. Finally, the packet moves to the next layer where the above process is repeated until the packet reaches the destination. At this point, the path between the source and destination is successfully created, after which all the remaining packets of the flow follow the same path.

B. Implementation

Here we describe the implementation details of the SLMR algorithm. This includes the implementation of flow table and application profiles by the switches in the network. We also describe the behavior of SLMR under different scenarios, in cases when new and existing flows are seen by the switches, and also when the links in the network have high oversubscription probabilites.

In order to take optimal routing decisions for each flow and to achieve distributed control over the network, each switch maintains a flow table. An entry in the flow table consists of source address and port number, destination address and port number and the outgoing link on which the flow is assigned [18]. Whenever a new flow arrives at a switch, SLMR records an entry in the flow table for the new flow. After a link is selected for the flow, it is added to the entry in the flow table as an outgoing link for that particular flow.

In addition to the flow table, each switch also maintains a profile of applications that are generating traffic inside the data center. This profile contains the port numbers and stochastic traffic parameters (μ_k and σ_k, as discussed in Section II-A) of the flows generated by different applications. SLMR learns the stochastic traffic parameters of the new flow by checking its entry in the flow table and matching the port numbers of the new entry with the port numbers in the profile. For the flows succeeded by the first one, SLMR checks the originating host and destination addresses, and using the recent history of profiles used for the host pair, obtains the stochastic parameters from the profile.

In case of an existing flow, each switch along the path already has an entry in its flow table. The switch determines if it is an existing flow by comparing the four-tuple information (source address and port number, and destination address and port number) provided by the packets of the flow. After this determination, the switch forwards the packet to the link which is present in the flow table for that particular entry.

If there are no links available on a particular layer, i.e. all the links have probabilities exceeding oversubscription percentage threshold T, the current switch sends the packet to the previous switch. This means that the packet is backtracked to the previous layer. The previous hop switch then repeats the process in section III-A, and sends the packet to the next link with a greater oversubscription probability than the link where the backtracked packet is received. If there is no more available link, the packet is backtracked again, until either a route is successfully found or there is no more backtracking possible, which means a route failure. Note that since the entire network is load balanced, a route failure should happen with a small chance.

From the above explanation we can easily deduce that our proposed algorithm is fulfilling the design objectives. Firstly, with the help of link availability searching and backtracking, it finds a path if there exists one. Secondly, with the help of link load probabilities, it guarantees an optimal load balanced network.

IV. Simulation Results

This section presents the simulation results used to validate the efficiency of SLMR over traditional routing algorithms, such as Distance Vector and Link state. We also perform SLMR comparison with and without considering the stochastic traffic behavior to show the effectiveness of our traffic model.

Our simulation scope covers the average end to end delay and also average normalized throughput per host in the network.

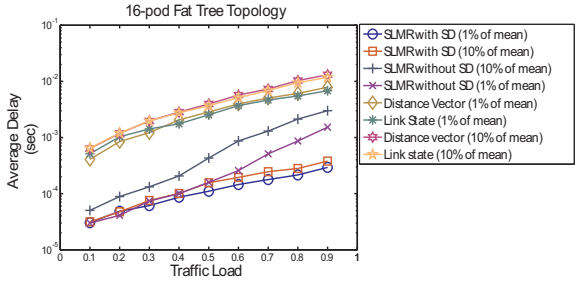
A. Simulation Settings

We use Network Simulator 2 ($NS2$) [8] to simulate the operation of all the aforementioned algorithms and to compare their performance under both the fat tree and VL2 topologies. We consider a 16-pod fat tree topology with 1024 hosts, 128 edge switches, 128 aggregation switches and 64 core switches. Similarly for the VL2 topology we consider 1024 hosts, 128 edge switches, 64 aggregation switches and 32 core switches. Each link in both the topologies is bidirectional and has a bandwidth of 1 Gbps for each direction. Each flow has a bandwidth demand that follows log normal distribution, with a maximum demand of 5 Mbps. During traffic generation, the values are obtained from the profile of applications. For each profile, we have different μ_k values and for each flow f_k, we have two σ_k values, which are 1% and 10% of μ_k. Although experiments were done with other σ_k values, the above mentioned values most clearly portray the simulation based traffic deviation. For both the topologies, we consider both uniform and non-uniform traffic distribution. For uniform distribution, the flow destination of a host is uniformly distributed among all the remaining hosts. For non-uniform distribution, 70% of the traffic generated is destined to the hosts situated in the same pod (or connected through one of the two aggregate switches for a particular source edge switch, in case of the VL2 topology) [21]. Each link has a propagation delay of 1 μs. The packet length is uniformly distributed between 40 and 1,400 bytes [5] [10], and packet arrival follows the log normal random process [5]. Each simulation run has a total duration of 30 seconds. As 80% of the flows inside a typical data center network last no more than 10 seconds [5], therefore a simulation duration of 30 seconds is sufficient to capture the behavior of SLMR in typical data center networks. Traffic is generated separately for each value of σ_k and behavior of each case is observed.

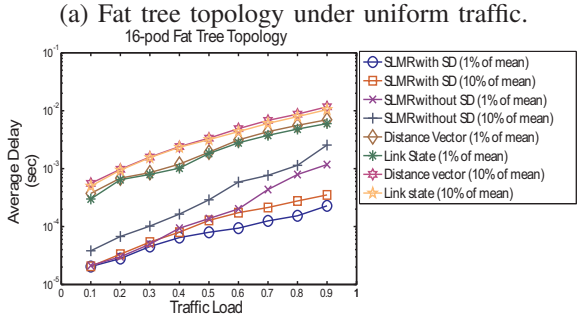
B. Average Delay

Now we present the average end-to-end delay results. We assume that the per hop nodal delay consists of queuing delay, transmission delay, and propagation delay.

Fig. 3(a) and 3(b) show the average end to end delay in a 16-pod fat tree with uniform and non-uniform distribution of the destinations, respectively. We executed SLMR under both the configurations with and without considering the stochastic traffic behavior. When stochastic behavior is considered, SLMR uses σ_k values to calculate the congestion probabilities of the links (refer to Section II-A for details), represented as SLMR with SD in all the figures. When stochastic behavior is not considered, SLMR uses only the μ_k information and ignores the stochastic traffic behavior. In this case, the data are represented as SLMR without SD in the figures. It is evident from the figures that when stochastic behavior is considered, SLMR gives the best result. At high traffic loads, i.e. traffic load values greater than 0.6, we see increase in end to end delay when stochastic behavior is not considered.



(a) Fat tree topology under uniform traffic.



(b) Fat tree topology under non-uniform traffic.

Fig. 3. Average end-to-end delay of a 16 pod Fat tree topology under uniform and non-uniform traffic.

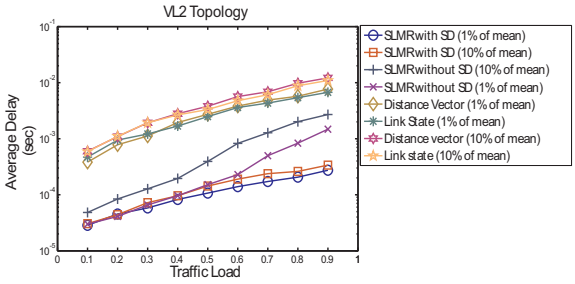
This is because of the fact that at high traffic loads, there is a drastic increase in network traffic congestion. Also, SLMR faces more backtracks which adds up in the delay as it takes more time to find an optimal path. When stochastic behavior is considered, the delay is almost stable even at very high traffic loads. For non-uniform traffic, the average delay in each case decreases, as compared to uniform traffic. Because in this case, most of the flow destinations are situated in the same pod or aggregation layer, and hence need fewer number of hops. Distance Vector and Link State algorithms show large delay under both uniform and non-uniform traffic as they are inefficient in utilizing multiple paths under high traffic loads.

Fig. 4(a) shows the average end-to-end delay under the VL2 topology. In this case, there is a slight improvement in the average delay as compared to the fat tree, due to a greater number of available paths between the source and destination [13].

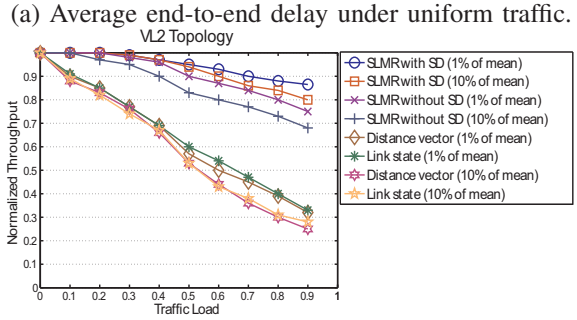
C. Average Normalized Throughput

We now present the average normalized throughput per host under the fat tree topology. Throughput values were taken at each host for all the simulation runs and their average was calculated, as shown in Fig. 5(a) and 5(b) for the fat tree topology. It is evident from the results that SLMR provides the highest throughput when stochastic behavior is considered. As seen in the figure, throughput stays above 90% even at higher traffic load values. This is mainly because of the fact that the probability calculations are more resilient to the traffic behavior, hence give close to optimal results.

Fig. 4(b) shows average normalized throughput under the VL2 topology. Persistently, SLMR shows improved results when we consider the stochastic behavior of traffic, as compared to when the stochastic behavior is not considered.

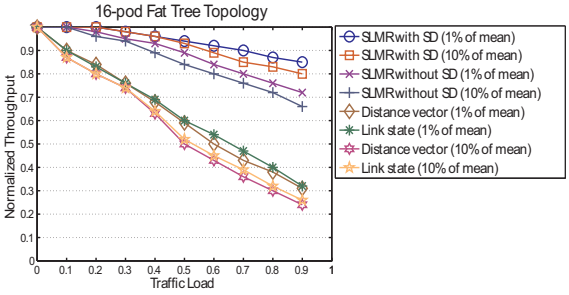


(a) Average end-to-end delay under uniform traffic.

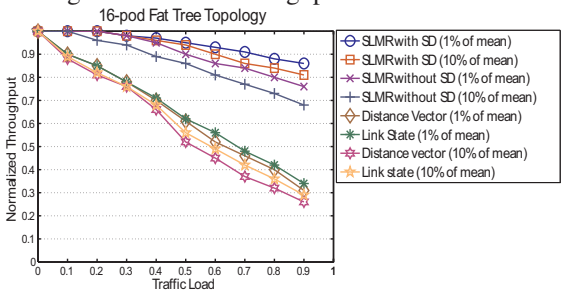


(b) Average normalized throughput under uniform traffic.

Fig. 4. Average end-to-end delay and normalized throughput under VL2 topology.



(a) Average normalized throughput under uniform traffic.



(b) Average normalized throughput under non-uniform traffic.

Fig. 5. Average normalized throughput of a 16 pod Fat tree topology.

V. Conclusion

Our goal in this work is to achieve optimal load balancing and efficiently utilize the multipath properties of muti-rooted data center networks, while considering the stochastic behavior of traffic inside the data centers. We first prove that the considered problem is NP hard by reduction from the integer partition problem. We then propose our stochastic load-balanced multipath routing (SLMR) algorithm that takes a distributed approach, and achieves load balancing by considering the stochastic nature of traffic in the network. By calculating the oversubscription probability of each link, it selects the optimal path for each flow in the network. We implement our proposed algorithm in the NS2 simulator, under both the fat tree and VL2 topologies, and conduct extensive simulations to evaluate the average end to end delay and normalized throughput of SLMR. Our simulations reflect the importance of considering traffic behavior inside the data centers, as SLMR gives the best results when we consider the stochastic behavior of traffic during its execution.

References

[1] Amazon ec2 sla. [Online]. Available: http://aws.amazon.com/ec2-sla/

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *ACM SIGCOMM*, Aug. 2010.

[3] M. Al-Fares, A. L., and A. V., "A scalable, commodity data center network architecture," in *ACM SIGCOMM*, 2008.

[4] T. Benson, A. Akelia, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM IMC*, Nov. 2010.

[5] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *ACM SIGCOMM*, Jan. 2010.

[6] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: Insights and future directions," *IEEE network*, vol. 25, July/August 2011.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. stein, *Introduction to Algorithms*, 3rd ed. The MIT press, 2009.

[8] K. Fall and K. Varadhan, "The ns manual (formerly ns notes and documentation)," UC Berkeley, LBL, USC/ISI, and Xerox PARC., Tech. Rep., Nov. 2011.

[9] L. F. Fenton, "The sum of log-normal probability distributions in scatter transmission systems," in *IRE Transactions on Communications Systems*, 1960.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint ip backbone," *IEEE network*, vol. 17, no. 6, pp. 6–16, Nov. 2003.

[11] I. Gojmerac, P. Reichl, and L. Jansen, "The adaptive multi-path algorithm," in *Elsevier Computer Networks 52 (2894 - 2907)*, 2008.

[12] A. Greenberg, M. Alizadeh, and D. Maltz, "Data center tcp (dctcp)," in *ACM SIGCOMM*, Aug. 2010.

[13] A. Greenberg, J. Hamilton, and N. Jain, "Vl2: A scalable and flexible data center network," in *ACM SIGCOMM*, Oct. 2009.

[14] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in *ACM SIGCOMM*, 2009.

[15] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM*, 2008.

[16] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banarjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *USENIX NSDI*, Apr. 2010.

[17] *Analysis of an Equal-Cost Multi-Path Algorithm*, IETF, RFC 2992 - Nov. 2000 Std.

[18] N. McKeown, T. Anderson, H. Balakrishnan, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," in *ACM SIGCOMM*, Apr. 2008.

[19] X. G. Meng, S. H. Wong, Y. Yuan, and S. Lu, "Characterizing flows in large wireless data networks," in *MOBICOMM*, 2004.

[20] D. S. J. Michael R. Garey, *Computers and Intractability, A guide to the theory of NP Completeness*.

[21] R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM*, Aug. 2009.

[22] Y. Ohara, S. Imahori, and R. Meter, "Mara: Maximum alternative routing algorithm," in *IEEE INFOCOM*, 2009.

[23] J. J. Prevost, K. Nagothu, and B. Kelley, "Prediction of cloud data center networks loads using stochastic and neural models," in *System of Systems Engineering (SoSE)*, Jun. 2011.

[24] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and Statistics for Engineers and Scientists, 9th Edition*. Prentice Hall.