Optimizing Network Performance using Weighted Multipath Routing

Junjie Zhang*, Kang Xi†, Liren Zhang†, H. Jonathon Chao*

Polytechnic Institute of New York University* UAE University† jzhang10@students.poly.edu*, kxi@poly.edu†, lzhang@uaeu.ac.ae†, chao@poly.edu*

Abstract—Equal-Cost Multipath (ECMP) routing has been widely adopted to perform load balancing. With ECMP, a router can maintain multiple next hops for a destination IP prefix. The most common method used by such routers is to split traffic with per-flow basis evenly among those next hops. This approach, although simple, cannot achieve optimal load balancing. In this paper we study the optimal configuration of weighted ECMP, where traffic splitting among the available paths is based on a set of pre-determined ratios. The contribution of this paper is two-fold. First, we develop a model to obtain the split ratios such that the overall network end-to-end delay is optimized. This is important because better delay performance is a result of better bandwidth allocation and has a direct impact on application, while most existing work tries to minimize the traffic load on the most utilized link. Second, we prove that the problem can be first solved by using a simple flow-based routing model and then converting the results to apply to IP networks, where destination-based forwarding is used. We present a heuristic algorithm to find the near-optimal weight configurations and demonstrate the effectiveness of the algorithm using computer simulations.

Index Terms—Multipath, ECMP, Load balancing.

I. INTRODUCTION

Multipath routing has been widely recognized as more efficient than single path routing. In the former, multiple paths are configured between a pair of nodes, and then packets are split among these paths by using label switching [1] or flow-based hashing [2]. In richly interconnected networks, multipath routing can provide load balancing, improve bandwidth utilization and mitigate congestion [3]. There are two critical issues for the design of multipath routing:

1) How to find multiple paths.

2) How to distribute traffic among them.

In networks using Open Shortest-Path First (OSPF), the nodes compute the paths using the cost of each link, based on Dijkstra’s shortest path algorithm [4]. If there are multiple shortest paths between a pair of source and destination nodes, each router on the paths would apply the Equal-Cost Multipath (ECMP) split rule, then equally split traffic among all available next hops corresponding to the shortest paths [5]. Even splitting is based on a hash function applied to the packet header (e.g., source and destination addresses and port numbers). The hash values are evenly mapped to all available next hops. Thus, packets are supposed to be delivered evenly to the next hops, according to the hash values [2]. However, even traffic distribution cannot achieve optimal load balancing. An example is given in Figure 1. Under the ECMP routing scheme, the link between nodes 3 and 4 becomes the bottleneck link, because the link is shared by the flows of two source-destination pairs (A, C) and (B, D). The situation could be even worse in large-scale networks, where even traffic splitting at each node may overload certain links while leaving some others under-utilized.

In this paper, we focus on the traffic split issue within the framework of ECMP. Instead of even splitting as ECMP, we present a delay-sense weighted ECMP (wECMP-d). The idea is to find a set of ratios to split traffic at each node among the available next hops for given traffic demands, so that the network delay performance is optimized. While most existing works try to minimize the traffic load on the most utilized link, we aim at optimizing the end-to-end delay of each pair of nodes in the network, which has a direct impact on the quality of service in most cases. The major contributions of this paper include the following:

1) We show that the problem can be formulated by using a simple flow-based routing model and then prove that the results using the flow-based routing model can be gracefully converted to a set of ratios that can be applied to destination-based routing, thus maintaining the same forwarding complexity as ECMP.

2) Because of the difficulty in solving the problem for large-scale networks by using the nonlinear programming model we formulated, we present a heuristic algorithm to obtain the near-optimal weight configurations.
We conducted extensive performance evaluation using computer simulations in various networks, including practical and randomly generated topologies. The results show that the proposed scheme can effectively improve the overall network performance and outperforms existing schemes. The implementation of our scheme is very straightforward. It only requires to modify routers’ hash parameters without any hardware or protocol changes. Therefore, weECMP-d can be applied to new and existing networks with minor modifications of the router software/configuration.

The remainder of this paper is organized as follows: Section II discusses related works. Section III presents the model and problem formulation. Section IV describes the heuristic algorithm for the problem. Section V discusses implementation details. In Section VI, we show the simulation results and related discussion. Finally, we conclude our paper in section VII.

II. RELATED WORK

As we mentioned in the previous section, there are two critical issues for multipath routing: how to find multiple paths and how to distribute traffic along those paths. There are several solutions [6][7][8][9][10][11][12][13][14] trying to deal with them. The schemes in [6] and [7] are based on ECMP with even traffic splitting. The idea is to fine tune the link weights to affect the shortest path computation, so that the traffic distribution can be optimized for load balancing. While this approach works for some topologies, it does not work for other topologies. For instance, no matter how we adjust the link weights in Fig.1, it is impossible to achieve load balancing along the three paths. Our solution can be an incremental enhancement of above solutions, since our scheme is based on shortest paths routing but release the constraints of even traffic distribution to next hops. The scheme in [8] is also based on shortest path routing and the ECMP split rule. For the flexibility of traffic allocation, the main idea of [8] is to modify the forwarding table of routers to define a subset of allowable next hops for each destination IP prefix by selecting this subset from the set of available next hops which correspond to the shortest paths. Therefore, the network performance is optimized by evenly distributing the traffic among these subsets of next hops. However, our scheme offers a finer granularity in traffic allocation than [8]. Another approach is to use two-phase routing [9][10][11][12]. In these schemes, traffic is sent from each source to a set of intermediate nodes with certain split ratios first, and then deliver from the intermediate nodes to the destinations finally. Network performance is optimized by carefully tuning the split ratios to intermediate nodes. The ability of handling highly dynamic and changing traffic is the advantage of two-phase approach. However, the approach increases the routing overhead and impairs the performance and reliability of networks, when forcing all traffic to go through a set of intermediate nodes.

There are also dynamic schemes [13][14], as opposed to the static ones described above, to improve the multipath problems. OSPF-OMP [13] allows splitting traffic unevenly, the idea is to dynamically determine the split ratios of traffic distributed along multiple equal-cost paths, based on the special traffic-load control messages exchanged by routers. The scheme in [14] uses minimal congestion feedback signals from routers to dynamically split the flows. By dynamically changing the split ratios, the schemes in [13] and [14] can alleviate link congestion in the network and increase the network throughput. However, these schemes are too complicated and perform optimization locally (per router), which often results in inefficient traffic allocation.

III. PROBLEM DESCRIPTION

A. Notation

The network is represented by a connected graph \( G(V,E) \) with a node set \( V \) and a directed edge set \( E \). Let us denote the notations as follows:

- \( c_e \) the capacity of link \( e \).
- \( y_e \) the traffic load on link \( e \).
- \( h_{ij} \) the traffic load between a source-destination pair \((i,j)\) \((i,j) \in V; i \neq j\).
- \( P_{ij} \) the shortest path set of a source-destination pair \((i,j)\) \((i,j) \in V; i \neq j\).
- \( \alpha_{ijp} \) traffic split ratio for path \( p \) \((p \in P_{ij})\), \( 0 \leq \alpha_{ijp} \leq 1 \).
- \( \delta_{ijp} \) =1 if link \( e \) belongs to path \( p \) \((p \in P_{ij})\); \( 0 \), otherwise.

B. Modeling

The objective of our design is to distribute the traffic so that the queuing delay in the entire network is minimized. For this purpose, we need to first model the delay on each link. Given the load \( y_e \) on link \( e \), we use M/M/1 to derive the average queuing delay \( t_e \) on that link. Therefore, \( t_e \) is given by:

\[
t_e = 1/(c_e - y_e)
\]  

The average queuing delay on a path \( t_p \) is the sum of the average queuing delay on each hop:

\[
t_p = \sum_{e \in E} \delta_{ijp}(c_e - y_e)
\]  

Since optimizing the delay of a path with a higher traffic load would give us higher performance gain, we assign path load \( h_{ij} \alpha_{ijp} \) as a weight to each path. The total weighted end-to-end delay for a source-destination pair \((i,j)\) is given by:

\[
t_{ij} = \sum_{p \in P_{ij}} \sum_{e \in E} h_{ijp} \alpha_{ijp} \delta_{ijp}(c_e - y_e)
\]  

Our objective is to minimize the overall end-to-end delay in the network \( \sum_{i,j\in V} t_{ij} \), where

\[
\sum_{i,j\in V} t_{ij} = \sum_{i,j\in V} \sum_{p \in P_{ij}} \sum_{e \in E} h_{ijp} \alpha_{ijp} \delta_{ijp}(c_e - y_e)
\]
Since the traffic load on a link is the sum of the load share from each pair whose paths include that link.

\[ y_e = \sum_{i,j \in V, p \in P_{ij}} \sum_{i \neq j} h_{ij} \alpha_{ijp} \delta_{esjp} \]  

(5)

After transformation, our objective turns out to be the minimizing of the average packet delay\(^1\).

\[ \sum_{i,j \in V, p \in P_{ij}} \sum_{e \in E} h_{ij} \alpha_{ijp} \delta_{esjp} / (c_e - y_e) = \sum_{e \in E} y_e / (c_e - y_e) \]  

(6)

Let \( F_e(y_e) \) denote \( y_e / (c_e - y_e) \). The problem can be formulated as follows:

**minimize** \( \sum_{e \in E} F_e(y_e) \)  

subject to

\[ \sum_{p \in P_{ij}} \alpha_{ijp} = 1 \quad i, j \in V, i \neq j \]  

(7b)

\[ \sum_{i,j \in V, p \in P_{ij}} \delta_{esjp} \alpha_{ijp} h_{ij} = y_e \quad e \in E \]  

(7c)

\[ 0 \leq y_e \leq c_e \quad e \in E \]  

(7d)

\[ \alpha_{ijp} \geq 0 \quad p \in P_{ij}, i, j \in V, i \neq j \]  

(7e)

Eq.(7b) states that the sum of the split ratio \( \alpha_{ijp} \) over all paths of a source-destination pair \( (i, j) \) is equal to 1. Eq. (7c) indicates the traffic load on link \( e \). Eq. (7d) is constrained by the link capacity. Since the problem is complicated and computational intractable for medium to large sized networks, we design a heuristic algorithm for the problem in Section IV.

**C. Objective Function**

There are two main reasons why we choose minimizing end-to-end delay of any pair of nodes \( \min \left( \sum_{e \in E} F_e(y_e) \right) \) as our objective function. First, as shown in Fig.2(a), since we used minimizing maximum link utilization \( \min(\max_{e \in E}(y_e / c_e)) \) as the objective function, the optimization algorithm would be terminated and no more improvement could be made. However, in Fig.2(b), where we use minimizing overall end-to-end delay in the network as the objective function, the scenario is the same as that in Fig.2(a), yet we can achieve better load balancing and improve network performance. Second, the term \( F_e(y_e) = y_e / (c_e - y_e) \) for link \( e \) corresponds to the average number of packets in the M/M/1 system. Thus, our objective is to try to minimize the number of packets in queues (i.e., the average packet delay), which means minimizing the overall number of packets buffered in the network. This reduction has a direct impact on the quality of service. We compare the performance of two objective functions by simulations in Section VI.

**IV. HEURISTIC ALGORITHM DESCRIPTION**

In this section we develop a heuristic algorithm that computes the near-optimal split ratios for multiple paths of each source-destination pair. The approach we develop is adapted from the Simulated Annealing algorithm [15], which is a general method of searching for a near-optimal solution. Normally the algorithm keeps searching for a better state, but also allows current states to temporarily move to a worse state with a non-zero probability, to avoid being trapped at the local optima. This approach turns out to have higher chances to obtain a better solution. The terms of the algorithm are defined as follows:

- State \( S \): determined by split ratios for the multiple paths of each source-destination pair in the network.
- Energy \( E \): means the overall end-to-end delay in the network. \( E \) is calculated by an energy function \( E() \) which is equivalent to the objective function described in Eq.(6). \( E \) is used for state evaluation.
- Temperature \( T \): the remaining iterations before termination. \( T_0 \) is the initial value of \( T \).
- Acceptance Probability \( A \): the probability of transition from the current state \( S \) with energy \( E = E(S) \) to a neighbor state \( S^* \) with energy \( E^* = E(S^*) \),

\[ A(E, E^*, T) = \begin{cases} 
1, & E^* < E \\
\exp((E-E^*)/T), & E^* \geq E
\end{cases} \]  

(8)
where $c$ is a parameter. We set $c = 100$ in our simulations, based on our experience.

- Neighbor Generator:
  1) Randomly choose a source-destination pair (e.g., $(i, j)$);
  2) Select the path with the highest end-to-end delay from the shortest path set $P_{ij}$ (e.g., $\eta, \eta \in P_{ij}$);
  3) Randomly find an alternative path from the same path set (e.g., $\eta', \eta' \in P_{ij}$);
  4) Adjust split ratios of two paths (e.g., $\alpha_{ij\eta}, \alpha_{ij\eta'}$). Let the split ratio of the path with highest end-to-end delay ($\alpha_{ij\eta}$) decrease by $\Delta$, and let the split ratio of the alternative path ($\alpha_{ij\eta'}$) increase by $\Delta$, where $\Delta$ is randomly chosen between $(0, \alpha_{ij\eta})$;
  5) After adjustment, a neighbor state $S^*$ with a new set of split ratios is generated.

The algorithm is illustrated in Fig.3 and summarized in the following 3 steps. We start with even split ratios $S$, and then repeat the following steps until the algorithm terminates with $T = 0$.

1) Use Neighbor Generator to generate a new set of split ratios $S^*$, and decrease $T$ by 1;
2) Based on the current and new split ratios, $S$ and $S^*$, we calculate the state energy $E$ and $E^*$ respectively, and then do the following according to $E$ and $E^*$:
   a) If $E^*$ is smaller than $E$, accept the neighboring state as the current state, and go to Step 3.
   b) If $E^*$ is larger than or equal to $E$, randomly pick a value $\rho$ between $(0, 1)$, and compare with the acceptance probability $A$. If $\rho \leq A$, accept the neighboring state as the current state, and go to Step 3. Otherwise, go to Step 1.
3) Update the current state as $S \leftarrow S^*$

When the above procedure terminates, the best set of split ratios for our objective in the whole procedure is obtained.

V. LOAD BALANCING FOR DESTINATION-BASED ROUTING

The proposed heuristic algorithm is designed to obtain a set of near-optimal split ratios. However, such ratios are obtained based on a flow-based model, where traffic for each source-destination pair is split among the multiple paths. Note that IP networks perform destination-based routing, and usually do not use source addresses. The results from the proposed heuristic algorithm cannot be directly applied to IP networks. Fortunately, we prove that the results using the flow-based routing model can be gracefully converted to a set of ratios that can be applied to destination-based routing.

In destination-based routing (hop by hop routing) networks, if the paths of $n$ different pairs pass the same node to the same destination, they will overlap with the remaining sub-paths of $n$ pairs after that node. This is because the segments of shortest paths are also shortest paths. Based on this observation, we can convert flow-based split ratios to destination-based split ratios.

Before we present the theoretic results, let us first illustrate with a simple example how the conversion works. Fig.4 shows a simple network topology. Assume that each link has a capacity of 5 units and has the same cost. There are three shortest paths between nodes $A$ and $C$. Source $A$ wants to send 1 unit of traffic to destination $C$ along the three paths. To achieve the best load balancing, assume that the flow-based split ratios obtained by our proposed heuristic algorithm for the three paths are set to $3/12, 1/12, 8/12$, respectively. There are two shortest paths between nodes $B$ and $C$. Source $B$ want to send 1 unit of traffic to destination $C$ along the two paths. Again, to achieve the best performance, assume that the flow-based split ratios obtained by our proposed heuristic algorithm for the two paths are set to $5/12, 7/12$, respectively. Node $A$ forwards the packets of each flow from source $A$ to destination $C$ to the next nodes $B$ and “1” according to the pre-determined ratios, as shown in Fig.4(a). Similarly, node $B$ forwards packets of each flow from source $A$ to destination $C$, and from source $B$ to destination $C$ to nodes “2” and “3” according the split ratios, as shown in Fig.4(a). As we can see, the number of entries of the forwarding table is determined by the number of source-destination pairs passing the router, but not the number of destination nodes as in the traditional IP networks. Fig.4(b) shows how we can converge the flow-based splitting into destination-based splitting. For node $A$, the destination-based split ratio $(1/3, 2/3)$ is equivalent to the flow-based split ratio $(3/12, 1/12, 8/12)$. For node $B$, based on

![Fig. 3. Flow Chart for the proposed heuristic algorithm for weighted ECMP.](image)

---

- Neighbor Generator:
  1) Randomly choose a source-destination pair (e.g., $(i, j)$);
  2) Select the path with the highest end-to-end delay from the shortest path set $P_{ij}$ (e.g., $\eta, \eta \in P_{ij}$);
  3) Randomly find an alternative path from the same path set (e.g., $\eta', \eta' \in P_{ij}$);
  4) Adjust split ratios of two paths (e.g., $\alpha_{ij\eta}, \alpha_{ij\eta'}$). Let the split ratio of the path with highest end-to-end delay ($\alpha_{ij\eta}$) decrease by $\Delta$, and let the split ratio of the alternative path ($\alpha_{ij\eta'}$) increase by $\Delta$, where $\Delta$ is randomly chosen between $(0, \alpha_{ij\eta})$;
  5) After adjustment, a neighbor state $S^*$ with a new set of split ratios is generated.

The algorithm is illustrated in Fig.3 and summarized in the following 3 steps. We start with even split ratios $S$, and then repeat the following steps until the algorithm terminates with $T = 0$.

1) Use Neighbor Generator to generate a new set of split ratios $S^*$, and decrease $T$ by 1;
2) Based on the current and new split ratios, $S$ and $S^*$, we calculate the state energy $E$ and $E^*$ respectively, and then do the following according to $E$ and $E^*$:
   a) If $E^*$ is smaller than $E$, accept the neighboring state as the current state, and go to Step 3.
   b) If $E^*$ is larger than or equal to $E$, randomly pick a value $\rho$ between $(0, 1)$, and compare with the acceptance probability $A$. If $\rho \leq A$, accept the neighboring state as the current state, and go to Step 3. Otherwise, go to Step 1.
3) Update the current state as $S \leftarrow S^*$

When the above procedure terminates, the best set of split ratios for our objective in the whole procedure is obtained.

V. LOAD BALANCING FOR DESTINATION-BASED ROUTING

The proposed heuristic algorithm is designed to obtain a set of near-optimal split ratios. However, such ratios are obtained based on a flow-based model, where traffic for each source-destination pair is split among the multiple paths. Note that IP networks perform destination-based routing, and usually do not use source addresses. The results from the proposed heuristic algorithm cannot be directly applied to IP networks. Fortunately, we prove that the results using the flow-based routing model can be gracefully converted to a set of ratios that can be applied to destination-based routing.

In destination-based routing (hop by hop routing) networks, if the paths of $n$ different pairs pass the same node to the same destination, they will overlap with the remaining sub-paths of $n$ pairs after that node. This is because the segments of shortest paths are also shortest paths. Based on this observation, we can convert flow-based split ratios to destination-based split ratios.

Before we present the theoretic results, let us first illustrate with a simple example how the conversion works. Fig.4 shows a simple network topology. Assume that each link has a capacity of 5 units and has the same cost. There are three shortest paths between nodes $A$ and $C$. Source $A$ wants to send 1 unit of traffic to destination $C$ along the three paths. To achieve the best load balancing, assume that the flow-based split ratios obtained by our proposed heuristic algorithm for the three paths are set to $3/12, 1/12, 8/12$, respectively. There are two shortest paths between nodes $B$ and $C$. Source $B$ want to send 1 unit of traffic to destination $C$ along the two paths. Again, to achieve the best performance, assume that the flow-based split ratios obtained by our proposed heuristic algorithm for the two paths are set to $5/12, 7/12$, respectively. Node $A$ forwards the packets of each flow from source $A$ to destination $C$ to the next nodes $B$ and “1” according to the pre-determined ratios, as shown in Fig.4(a). Similarly, node $B$ forwards packets of each flow from source $A$ to destination $C$, and from source $B$ to destination $C$ to nodes “2” and “3” according the split ratios, as shown in Fig.4(a). As we can see, the number of entries of the forwarding table is determined by the number of source-destination pairs passing the router, but not the number of destination nodes as in the traditional IP networks. Fig.4(b) shows how we can converge the flow-based splitting into destination-based splitting. For node $A$, the destination-based split ratio $(1/3, 2/3)$ is equivalent to the flow-based split ratio $(3/12, 1/12, 8/12)$. For node $B$, based on
the flow-based split ratios, the traffic load on the path \((B, 2, C)\) is 2/3 \((= 5/12 + 1 + 3/12 + 1)\), and the traffic load on the path \((B, 3, C)\) is also 2/3 \((= 7/12 + 1 + 1/12 + 1)\). Therefore, at node \(B\), splitting traffic based on a destination-based split ratio \((1/2, 1/2)\) can achieve the same traffic allocation as splitting traffic based on the flow-based split ratios. Therefore, our weighted ECMP scheme maintains the same forwarding complexity as the traditional ECMP.

We can formulate the above convergence concept into a theorem as below.

**Theorem 1:** In a destination-based routing network, for any node \(X\), if there are \(n\) sources sending traffic to a destination \(Y\) via node \(X\), there are \(m\) shortest paths between node \(X\) and destination \(Y\), and the \(n\) sources distribute traffic among these \(m\) paths, based on the flow-based split ratios \(\alpha_{iYp}\) \((i = 1, 2, \ldots, n; \ p = 1, 2, \ldots, m)\) (see Fig.5), then the destination-based split ratio \(\beta_{yp}\) at node \(X\) for the traffic destined to \(Y\) is given by

\[
\beta_{yp} = \frac{\sum_{i=1}^{n} \alpha_{iYp} h_{iY}}{\sum_{i=1}^{n} h_{iY}} \quad p = 1, 2, \ldots, m \tag{9}
\]

**Proof:** Let \(h_{XY} = \sum_{i=1}^{n} h_{iY}\) denote the total traffic load from \(X\) to \(Y\). When using the destination-based splitting with a ratio \(\beta_{yp}\), the traffic load on each path becomes:

\[
l_{\alpha,Yp}^{\beta} = \beta_{yp} \ast h_{XY} = \sum_{i=1}^{n} \alpha_{iYp} h_{iY}, \quad p = 1, 2, \ldots, m.
\]

When using the flow-based splitting with a ratio \(\alpha_{iYp}\), the traffic load on each path is:

\[
l_{\alpha,Yp} = \sum_{i=1}^{n} \alpha_{iYp} h_{iY}, \quad p = 1, 2, \ldots, m.
\]

As shown above, the traffic load on each path based on destination-based split ratio \(\beta_{yp}\) is the same as that based on flow-based split ratio \(\alpha_{iYp}\).

It implies that traffic allocation of the two kinds of split ratios is the same. Therefore, for every node in the network, the flow-based split ratios can be converted to the destination-based split ratios by Eq.(9).

According to \(\beta_{yp}\), for the packets destined to \(Y\), we can determine different hash ranges in the forwarding table of router \(X\). The range size is proportioned to the ratio allocated to each next hop.

**VI. SIMULATION AND DISCUSSION**

In this section, we evaluate the performance of our proposed delay-sense weighted ECMP (wECMP-d) scheme. Our simulations include both practical networks and a randomly generated network with constraints. We compare the performance of wECMP-d with the traditional ECMP in two kinds of networks and show that wECMP-d outperforms the traditional ECMP. We also compare the performance of two objective functions, minimizing overall end-to-end delay in the network (wECMP-d) and minimizing maximum link utilization (wECMP-u), and show that wECMP-d has a better objective function.

**A. Topologies**

1) **Practical Networks:** Practical topologies used in our evaluation are shown in Fig.6. Each link is bidirectional. Each direction can be assigned a different capacity. The link capacities of each topology are randomly selected between [10, 20]. The selected node sends random units (1 to 5) of traffic to every other selected node. For practical topologies, the heuristic algorithm terminates after 2,000 iterations.

It uses the same heuristic algorithm described in section IV; Energy becomes the maximum link utilization in the network; Step 2 of Neighbor Generator becomes selecting the path with the maximum link utilization.
2) Random Network: We randomly generate a connected topology with 200 nodes and 1,000 directed links. The maximum degree of node is 5. The link capacities are randomly selected between $[25, 125]$. We randomly choose 1,000 pairs from the total source and destination pairs in the network, and assign random units (1 to 5) of traffic to each chosen pair. For the random topology, the heuristic algorithm terminates after 20,000 iterations.

For all networks we simulated, we guarantee that there are no overloads on links under the traditional ECMP routing scheme. Each link cost of the networks is assumed to be 1, and we use Dijkstra’s shortest path algorithm to generate the shortest paths for each pair (Some pairs may have only one shortest path.).

B. ECMP vs wECMP-d

Table I shows the maximum link utilization ($U$) and the overall end-to-end delay in the network ($D$) of ECMP and wECMP-d on the 5 topologies. As shown, wECMP-d outperforms ECMP on all topologies. Fig.7(a) and 8(a) show the average end-to-end delay of each pair of two schemes under topology (d) and the random topology. The results indicate wECMP-d reduces the end-to-end delay of most pairs in the network. This means wECMP-d allocates traffic more efficiently than ECMP and achieves better load balancing. Fig.7(b), 8(b) and Table II show the effectiveness of the heuristic algorithm. Fig.7(b) and 8(b) show that the objective function converges quickly after a certain number of iterations under two topologies, in terms of the total number of iterations. Table II compares wECMP-d with the optimal results derived from Eq.7 (solved by the AMPL/MINOS solver); the results of our proposed heuristic algorithm (wECMP-d) are very close to the optimal results.

C. Two Objective Functions

Table III shows the maximum link utilization ($U$) and the overall end-to-end delay in the network ($D$) optimized by two objective functions (wECMP-d and wECMP-u) on the 5 topologies. On topology (b), wECMP-u obtains a smaller maximum link utilization than wECMP-d does, while wECMP-d obtains a smaller overall end-to-end delay in the network. On topology (c), wECMP-d and wECMP-u obtain the same maximum link utilization, but wECMP-d does not improve much on end-to-end delay. On topologies (a), (d) and random, wECMP-d obtains the same maximum link utilization as wECMP-u and further reduces the overall end-to-end delay in the network. These results show that when the scenario described in Fig.2 occurs, wECMP-d achieves better performance than wECMP-u. In terms of the volume of traffic load affected, wECMP-d obtains different performance gains. If the scenario in Fig.2 does not occur, wECMP-d obtains better delay performance, and the maximum link utilization of wECMP-d is comparable with that of wECMP-u.

VII. CONCLUSION

We propose an efficient scheme, a delay-sense weighted ECMP, to achieve optimized load balancing for given traffic.

![Fig. 6. Topologies used for performance evaluation.](image)

**TABLE I**

<table>
<thead>
<tr>
<th>Topology</th>
<th>ECMP</th>
<th>wECMP-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.96</td>
<td>0.88</td>
</tr>
<tr>
<td>b</td>
<td>0.86</td>
<td>0.76</td>
</tr>
<tr>
<td>c</td>
<td>0.97</td>
<td>0.87</td>
</tr>
<tr>
<td>d</td>
<td>0.98</td>
<td>0.87</td>
</tr>
<tr>
<td>Random</td>
<td>0.99</td>
<td>0.96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topology</th>
<th>$U$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>45.74</td>
<td>46.28</td>
</tr>
<tr>
<td>b</td>
<td>15.38</td>
<td>16.34</td>
</tr>
<tr>
<td>c</td>
<td>35.56</td>
<td>35.70</td>
</tr>
<tr>
<td>d</td>
<td>34.33</td>
<td>37.63</td>
</tr>
</tbody>
</table>

$U$ stands for maximum link utilization

$D$ stands for overall end-to-end delay in the network
demands. The scheme distributes traffic among equal-cost paths based on a set of pre-determined ratios, so that the overall end-to-end delay in the network is minimized. The contributions of this paper include: First, we show that the problem can be formulated by using a simple flow-based routing model, and then prove that the results using the flow-based routing model can be converted to a set of ratios that can be applied to destination-based routing. Second, we present a heuristic algorithm to obtain the near-optimal weight configuration for large-scale networks. Our evaluation shows that the delay-sense weighted ECMP can achieve better load balancing than the traditional ECMP, and effectively reduce the overall end-to-end delay in the network.

REFERENCES


<table>
<thead>
<tr>
<th>Topology</th>
<th>wECMP-u</th>
<th>wECMP-d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U</td>
<td>D</td>
</tr>
<tr>
<td>a</td>
<td>0.88</td>
<td>52.91</td>
</tr>
<tr>
<td>b</td>
<td>0.75</td>
<td>16.40</td>
</tr>
<tr>
<td>c</td>
<td>0.87</td>
<td>35.83</td>
</tr>
<tr>
<td>d</td>
<td>0.87</td>
<td>40.44</td>
</tr>
<tr>
<td>Random</td>
<td>0.96</td>
<td>1473.96</td>
</tr>
</tbody>
</table>

Fig. 7. Weighted ECMP vs traditional ECMP on topology (d).

Fig. 8. Weighted ECMP vs traditional ECMP on a random topology.