HDEER: A Distributed Routing Scheme for Energy-Efficient Networking

Biyu Zhou, Fa Zhang, Lin Wang, Chenying Hou, Antonio Fernández Anta, *Senior Member, IEEE*, Athanasios V. Vasilakos, *Senior Member, IEEE*, Youshi Wang, Jie Wu, *Fellow, IEEE*, and Zhiyong Liu

***Abstract*—The proliferation of new online Internet services has substantially increased the energy consumption in wired networks, which has become a critical issue for Internet service providers. In this paper, we target the network-wide energy-saving problem by leveraging speed scaling as the energy-saving strategy. We propose a distributed routing scheme–HDEER–to improve network energy efficiency in a distributed manner without significantly compromising traffic delay. HDEER is a two-stage routing scheme where a simple distributed multipath finding algorithm is firstly performed to guarantee loop-free routing, and then a distributed routing algorithm is executed for energy-efficient routing in each node among the multiple loop-free paths. We conduct extensive experiments on the NS3 simulator and simulations with real network topologies in different scales under different traffic scenarios. Experiment results show that HDEER can reduce network energy consumption with a fair tradeoff between network energy consumption and traffic delay.**

***Index Terms*—Distributed algorithms, energy efficiency, green computing, routing, pareto optimization.**

I. Introduction

THE substantial power consumed by a network has become a critical issue for Internet Service Providers (ISPs). It has been reported that the total energy used by the Information and Communication Technology is responsible for a significant fraction of the world total electricity consumption, ranging between 2% and 10% [1]. With the proliferation of new Internet services, such as social networking and cloud computing, this proportion has increased rapidly in recent years [1]. The tremendous energy consumption of large-scale networks will become a stumbling block to their further developments unless energy efficiencies can be significantly improved [2].

A great deal of research has been carried out for network energy efficiency based on two mechanisms: *speed scaling* and *power-down*. These mechanisms are considered to be basic device-level energy saving approaches and have already been applied in industry [3], [4]. However, without coordinating with other devices, an energy-efficient device in a network can only make local decisions, which may be far from the global optimum, leading to limited network-wide energy savings. Therefore, researchers have started to propose network-level energy saving methods based on device-level mechanisms [5]–[9]. These network-wide approaches usually assume some given global traffic matrices in a target network and design traffic-shaping strategies to route the traffic flows. However, global traffic matrices are not easy to obtain, even in a small-scale network [10]. Moreover, most of the approaches are centralized approaches, whereby global information needs to be gathered, decisions are made by a central controller, and then disseminated to network nodes. This centralized model produces many scalability- and vulnerability-related issues when being applied to production networks [11]. As a result, a decentralized approach is a more scalable and flexible choice to implement and apply to real networks.

When designing a new network protocol or applying a new routing scheme, it is very important for an ISP to consider traffic delays [11]. Unfortunately, many existing proposals for energy savings have failed to take reduced traffic delay as a design goal. As a result, although considerable energy savings can be achieved, the traffic delays may be dramatically increased, which consequently leads to very poor user experience. A good energy-efficient design should be able to achieve an optimal balance between energy efficiency and traffic delay.

Motivation. In this paper, our objective is to obtain increased network-wide energy efficiency while considering traffic delays. A decentralized network routing scheme is developed that simultaneously possesses the following properties:

- *High scalability:* With the scale of networks continuously growing, compared with centralized strategies, a decentralized scheme is a more scalable and flexible choice. Specifically, the design needs no centralized controllers,

Manuscript received July 31, 2015; revised January 5, 2016; accepted February 23, 2016. Date of publication March 23, 2016; date of current version May 19, 2016. This work was supported in part by the National Natural Science Foundation of China (NSFC) Major International Collaboration Project 61520106005, in part by NSFC Project for Innovation Groups 61521092, in part by Ministerio de Economia y Competitividad under Grant TEC2014- 55713-R, in part by Regional Government of Madrid (CM) grant Cloud4BigData (S2013/ICE-2894), in part by European Commission H2020 grants ReCred and NOTRE, and in part by NSF grants ECCS 1231461, ECCS 1128209, CNS 1138963, CNS 1065444, CNS 1461932, CNS 1449860, and IIP 1439672. *(Corresponding author: Zhiyong Liu.)*

B.-Y. Zhou, C.-Y. Hou, and Y.-S. Wang are with Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China.

F. Zhang is with Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

L. Wang is with SnT, University of Luxembourg, Luxembourg.

A. F. Anta is with IMDEA Networks Institute, Madrid, Spain.

A. V. Vasilakos is with the Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, SE-931 87 Skellefteå, Sweden.

J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA.

Z.-Y. Liu is with the State Key Laboratory for Computer Architecture, Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. (e-mail: zyliu@ict.ac.cn)

Digital Object Identifier 10.1109/JSAC.2016.2545498

thus avoids a single point of failure and brings high scalability in its implementation.

- *Requires no real-time global traffic matrices:* The route computation process in the proposed scheme should not require global traffic matrix from the network. Each node only needs to continuously monitor and react according to the real-time traffic loads of the links attached to its own, which is very easy to implement.
- *Traffic delay:* The proposed scheme takes both energy efficiency and traffic delay as design considerations.

To this end, we propose HDEER, a Hop-by-hop Distributed Energy-Efficient Routing Scheme. With HDEER, the traffic destined to each node is routed through multiple paths in a Directed Acyclic Graph (DAG) generated for the node. The traffic injected in each node will be distributed among these loop-free paths, and optimality in terms of both energy consumption and traffic delay can be achieved. HDEER is a fully distributed scheme. It does not need to know the traffic matrix beforehand. Each node only needs to know the periodic updating traffic of its own, which can be easily obtained by monitoring the traffic of its own adjacent links.

Main Contributions. Our main contributions can be summarized as follows:

(i) We provide a bi-objective optimization formulation to address the problem of optimizing both energy consumption and traffic delay. We show that good trade-offs between the two objectives can be achieved by adopting Pareto optimal solutions.

(ii) While targeting the network energy-efficient routing problem from a distributed perspective, we observe that it will make the issue easier to settle if we separately consider loop-free routing and optimal traffic allocation. Thus we divide the distributed energy-efficient routing problem into two subproblems and then propose a two-stage distributed routing scheme (HDEER) to solve them accordingly.

(iii) We propose two Distributed, Loop-free, Multi-path Finding Algorithms (D_LoopFree and D_LoopFree-TA) for each node to guarantee distributed loop-free routing. We perform a theoretical analysis of the loop-freeness and connectivity of the generated DAGs.

(iv) We develop two Distributed Routing Algorithms (D_Routing-S and D_Routing-D) for each node to distribute the traffic to the next-hop nodes according to the loads on current node's egress links in these loop-free paths. Using the proposed algorithms, global optimal solutions for the bi-objective optimization can be obtained.

(v) We conduct comprehensive experiments with NS3 simulators to evaluate the performance of HDEER in terms of energy savings, traffic delay, and convergence properties using real network topologies at different scales under different traffic scenarios (generated traffic and real traffic traces).

The remainder of this paper is organized as follows: Section II states the problem and provides the bi-objective optimization model. Section III describes the distributed routing scheme whereby algorithms for both DAG generation and traffic allocation are developed. Section IV discusses the implementation details of the proposed scheme. Section V presents the experimental results for the performance evaluation. Section VI summarizes related work, and Section VII concludes the paper.

II. Problem Statement

In this section, we first state the network model. Then, we define the network optimization problem of simultaneously optimizing the total energy consumption by the network elements and the total delay experienced by the traffic.

A. Network Model

We model a communication network topology by a connected directed graph $G = (V, E)$, where V represents the set of nodes (routers or switches) in the network and E represents the set of bi-directional links (connections) between these nodes in V. Each node $v_i \in V$ is assigned an index i. Each link (i, j) in the set E has a capacity CA_{ij}. Note that the link (i, j), which represents a directed connection from node v_i to node v_j, is not identical to link (j, i).

We adopt the network model provided by Gallager in [11], and restate it for the sake of completeness. For every flow, traffic flow is injected into the network at the source node v_i with the expected rate r_j^i. This traffic has to be routed to the sink node v_j. We assume that fractional routing is permitted, which translates into the packets belonging to the same flow may be split among multiple paths [12] (note that the packet reordering problem has been widely studied [13] and will not be discussed here). The routing variable ϕ_{jk}^i is the fraction of the total traffic in node v_i that is destined for node v_j via link (i, k). This value is restricted to be nonnegative. Intuitively, if node v_k is not a neighbor of node v_i or if i equals j, ϕ_{jk}^i should be zero. Additionally, we have the following constraint: $\sum_{k \in V} \phi_{jk}^i = 1$.

We denote the total expected traffic in node v_i that is destined for node v_j by t_j^i. Evidently, t_j^i consists of the total traffic injected at node v_i and the total traffic passing through node v_i and being injected at other nodes:

$$t_j^i = r_j^i + \sum_{(k,i) \in E} t_j^k \times \phi_{ji}^k \tag{1}$$

The above equation implicitly restricts the flow conservation at each node. Now, let x_{ik} denote the total traffic on link (i, k), it satisfies $x_{ik} = \sum_{j \in V} t_j^i \times \phi_{jk}^i$. We assume that the traffic on each link can never exceed the corresponding link capacity; therefore, $0 \leq x_{ik} \leq CA_{ik}$.

Clearly, a certain configuration of the ϕ-value will result in a unique distribution of traffic t in the network. We study how to choose a proper ϕ_{jk}^i in each node with the objective of energy efficiency without a significant performance sacrifice.

B. Bi-Objective Optimization

We evaluate the network performance using two metrics: energy consumption and traffic delay. We first state these two metrics, and then, a bi-objective optimization problem is proposed to simultaneously minimize these two metrics.

1) Energy Consumption and Traffic Delay: We consider using speed scaling as the architectural support to obtain energy savings. Let $f_{ik}(x_{ik})$ denote the energy consumption function of network links, which means the energy consumed by link (i, k) when transmitting x_{ik} units of traffic. Normally, network devices can be designed so that slower operation speeds use lower power supply [3], [9], [14]. The power consumed by such a variable-speed device is a convex function of its execution speed, with the exact form dependent on the details of the technology. Thus $f_{ik}(x_{ik})$ is assumed to be a convex function of link transmission speed. Note that the function includes the energy cost of the link and other related network components for transmitting packets. Let F_T denote the total energy consumption of the entire network, which satisfies $F_T = \sum_{(i,k)\in E} f_{ik}(x_{ik})$.

Assume that delay is a function of the traffic load of the link. We denote the delay from transmitting x_{ik} units of traffic on link (i, k) by $d_{ik}(x_{ik})$. Similarly, we denote the total traffic delay of the entire network by D_T, $D_T = \sum_{(i,k)\in E} d_{ik}(x_{ik})$. We assume that $d_{ik}(x_{ik})$ is increasing and convex in x_{ik}, which is consistent with most wired data networks [11], [15]. Notice that the delay includes the time for processing, queuing, propagation, and transmission. In some applications where the content of the packets (the payload part of the packets) needs to be processed (e.g. encryption/decryption, and some content security related applications), this causes another part of delay (as well as energy consumption) by the processing elements and memories. This part of delay (and energy consumption) depends on the speeds and other architectural features of the processing cores and memories. This part of delay (and energy consumption) is not considered in this model.

2) Bi-Objective: In this work, we optimize F_T and D_T by adopting an economy-theoretical model. In general, bargaining among multiple objectives is a game played by a decision maker, who makes decisions among multiple objectives in the feasible design space X based on his/her preferences for these objectives [16]. Consider a multi-objective optimization problem $P(F, X)$ with k objectives, where $F(x) = [F_1(x), F_2(x), \ldots, F_k(x)]^T, \forall x \in X,$ and $F_i : X \to \mathbb{R}, \forall i \in [1, k]$. The problem $P(F, X)$ can be formulated as follows:

$$\min_{x\in X} \quad F(x) \tag{2}$$

Intuitively, the optimization goal of the problem $P(F, X)$ is to obtain the corresponding Pareto optimal point when the preferences for these k objectives are predetermined.

Using the weighted sum method [17], we can combine these two objectives using scalarization. We assign a weight to each objective to represent the preference of the objective. By doing this, we obtain a single objective C_T to define the performance of the network.

$$C_T = w_1 \times F_T + w_2 \times D_T \tag{3}$$

When w_1 and w_2 are the corresponding preferences, they satisfy $w_1 + w_2 = 1, w_1, w_2 \in [0, 1]$. A pair (w_1, w_2) corresponds to a Pareto optimal point. The value of (w_1, w_2) serves as a guideline to the optimal trade-off between energy savings and traffic delays.

Finally, we propose the following Bi-objective Optimization Formulation (BOF) to address the routing problem. We will find a set of ϕ-values to minimize the objective function. Note that we seek the Pareto optimal points for minimizing F_T and D_T by minimizing the single objective C_T.

$$\begin{aligned}
&\text{(BOF)} \quad \min \ C_T \\
&\text{s.t.} \quad t_j^i = r_j^i + \sum_{(k,i)\in E} t_j^k \times \phi_{ji}^k \ \ \forall i, j \ \ \text{(flow conservation)} \\
&\sum_{j\in V} t_j^i \times \phi_{jk}^i \le C A_{ik} \ \ \forall i, k \ \ \text{(QoS restriction)} \\
&\sum_{(i,k)\in E} \phi_{jk}^i = 1, \quad \forall i, j \ \ \text{(variable restriction)} \\
&\phi_{jk}^i \in [0, 1], \quad \forall i, j \ \ \text{(variable restriction)}
\end{aligned}$$

The above optimization problem falls into a classical quadratic objective problem. It can be solved using an optimization solver in a centralized manner. Nevertheless, these centralized optimization solvers need the topology information of the entire network and a traffic matrix as the input for computation. When the size of the network scales up, the time spent on the computation will not be tolerable in realistic network operation. In addition, centralized methods rely highly on a central controller, which can be used with SDN, to manage and schedule the network devices. This puts pressure on the central controller and makes the central controller a sensible vulnerability that can affect network stability. Thus, it is of great significance to address such a bi-objective routing problem in a distributed manner. To this end, we propose a completely distributed routing scheme to address the problem in the following section.

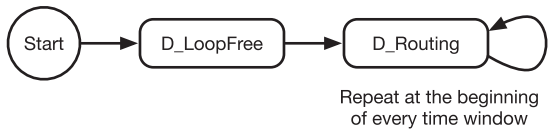


Fig. 1. An overview of HDEER.

III. Distributed Loop-Free Routing Scheme

In this section, we develop a energy efficient routing scheme by deriving optimal routing conditions for minimum C_T. Based on these optimal routing conditions, we propose a Hop-by-hop Distributed Energy-Efficient Routing Scheme (HDEER). HDEER consists of two stages. We first provide a Distributed Loop-free Multi-path Finding Algorithm (D_LoopFree) to guarantee loop-free routing. Then we propose an optimal Distributed Routing Algorithm (D_Routing) for each node to guide the traffic distribution on these multiple loop-free paths built by D_LoopFree.

A. Overview of HDEER

The proposed scheme is illustrated in Figure 1. At the beginning, the D_LoopFree algorithm is called to compute a DAG for each sink node. Each node in the network only needs to send traffic to its neighbors according to the DAG. Consequently, loop-freeness can be guaranteed. When a new time window starts, the D_Routing process will be called to distribute traffic among its neighbors at each node. Then, each node adjusts

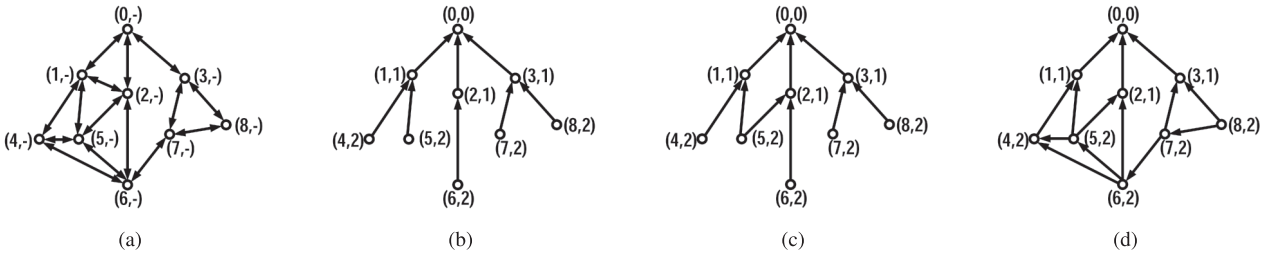


Fig. 2. Generating *GEN_G(0)*. Note that the root node is v_0. Each node v_i is associated with a two-tuple (i, h_i^0). (a) The original graph in which all nodes are assigned an index, while the height of each node is initialized to '-'; (b) generating the shortest path tree; (c) executing the first stage of link addition; (d) executing the second stage of link addition.

its processing speed according to its real-time traffic load. Both D_LoopFree and D_Routing operate in a totally distributed manner.

B. Distributed Loop-Free Multipath Finding (D_LoopFree)

As mentioned in Section II, we perform the distributed routing by letting each node in the network choose its own ϕ-value independently. Once the routing variable has been selected, the corresponding amount of traffic will be routed through the corresponding output link. This process happens at every node in the network simultaneously, thus unless done carefully, loops are likely to occur. Paths with loops will bring about the following consequences: (1) The distributed routing algorithm fails to converge, (2) Additional energy and delay costs are produced due to the cumulative feature of the total cost function. All these situations should be avoided. Therefore, we must carefully choose the next hops for each node to provide a guarantee of loop-free routing.

It is not difficult to see that for a particular destination v_t, all multiple loop-free routing paths originating from every node in the network to the destination v_t form a *DAG*. Leveraging this feature, we find the next hop at each node for each destination, which maintains the loop-free property by generating a *DAG* for each destination, including all nodes in the network topology.

In this work, we first devise a "height-based" algorithm to generate a connected graph $GEN\_G(t)$ from $G(V, E)$ for each destination node v_t. Then, we prove that each $GEN\_G(t)$ is one of the largest *DAGs* of G. The main principle behind our algorithm is to distinguish all the nodes with regard to each destination v_t by labeling them with different heights. If network traffic always flows from higher nodes to lower nodes based on their heights, loops will never be generated. The algorithm takes the graph of the network and the destination node v_t as the input. Then, it returns a *DAG*, denoted as $GEN\_G(t)$, corresponding to v_t after performing the three steps listed below. Now, we describe these three steps in detail.

Step 1: Generating a shortest path tree. We first build a *DAG* that can cover all the nodes in the network. We construct a shortest path tree based on the original graph G with root v_t as the destination to form a *DAG*. Note that every directed graph with no directed cycles is a *DAG*. We select the shortest path tree because it has the following advantages: (1) A shortest path tree based on the original graph G covers all the nodes in G, (2) A shortest path tree can be constructed in a distributed manner using the distributed Dijkstra algorithm.

Step 2: Labeling all nodes with different heights. For each destination node v_t, we label all the nodes in the corresponding *DAG* with integer values. For the sake of brevity, we denote the height of node v_i by h_i^t. First, we label the destination node v_t with 0. Then, we label each node with its node height (distance to v_t) in the shortest path tree. This process terminates when all nodes in the *DAG* have been labeled.

Nevertheless, only considering the shortest path routing will not produce an energy-efficient scheme because some paths that exhibit substantial energy savings may be neglected [12]. We extend the *DAG* with the following:

Step 3: Adding alternative links. For each destination node v_t, we extend the corresponding *DAG* by performing the following processes: (1) We find all the links (i, j), such that, $h_i^t > h_j^t$, then we add the link (i, j) to the DAG, (2) We find all the links (i, j), such that, $h_i^t = h_j^t \wedge i > j$, then we add the link (i, j) to the DAG.

Each node generates DAGs in a distributed way. The above process is only dependent on the network topology, and in particular, it is independent on any traffic load distribution in the network. Thus, every node can execute the process in a distributed manner. Also, it does not need to be executed often but only needs to be executed only when the scheme starts. Therefore, the entire process of our loop-free multi-path finding algorithm can be realized in a distributed manner. The entire process is illustrated in Figure 2. Assume that the original graph is connected and that it does not include parallel links. We now introduce the following theorem on the features of the *DAG* generated by our algorithm.

Theorem 1: For every destination node v_t, $GEN\_G(t)$ is the largest *DAG* of links destined for node v_t, with $|E|/2$ links.

Proof: Consider two arbitrary nodes v_i and v_j in $GEN\_G(t)$.

Case 1: v_i and v_j are adjacent. There will be exactly one link reserved between these two nodes either according to the height order ($h_i^t \neq h_j^t$) or according to the index order ($h_i^t = h_j^t$). Thus, one of the bi-directional links between two arbitrary adjacent nodes will be reserved to build $GEN\_G(t)$.

Case 2: v_i and v_j are not adjacent. In this case, no links will be reserved in $GEN\_G(t)$ according to the algorithm. Thus, this case contributes no links.

In summary, $GEN\_G(t)$ involves $|E|/2$ links in any case.

Suppose that there is a $DAG^*(t)$ that consists of more than $|E|/2$ links. According to the Pigeonhole Principle, there must be more than one link that exists between at least one pair of neighbor nodes, that is, at least one pair of bi-directional links exists, which contradicts the assumption that $DAG^*(t)$ is a DAG. Therefore, such a $DAG^*(t)$ does not exist, thus $GEN\_G(t)$ is the maximal $DAGs$ destined for node v_t. ■

C. Distributed Routing (D_Routing)

In this section, we first discuss our sufficient and necessary conditions for minimizing C_T. Then a distributed routing algorithm (D_Routing) based on the conditions is proposed. With the network model provided in this paper, a given input traffic set r and a certain configuration of the ϕ-value will result in a unique distribution of traffic t in the network. Thus before going into detail, we first calculate the partial derivatives of the objective C_T with respect to the input traffic flow r_j^i and the routing variable ϕ_{jk}^i. For convenience, we denote the total cost relevant to link (i,k) by $c_{ik}(x_{ik})$; thus, $C_T = \sum_{(i,k)\in E} c_{ik}(x_{ik})$. Accordingly, we denote the marginal cost of link (i,k) by $c'_{ik}(x_{ik})$. Note that $c'_{ik}(x_{ik})$ is the partial derivative of C_T with respect to x_{ik}, i.e., $c'_{ik}(x_{ik}) = \frac{\partial C_T}{\partial x_{ik}}$. Assume a small increment ε_r in the input r_j^i. For each adjacent node k, it will cause an increment cost on link (i,k) of $\varepsilon_r \cdot \phi_{jk}^i \cdot c'_{ik}(x_{ik})$. If node k is not the destination node, the increment traffic ε_r will cause new input traffic in node k. This corresponding increment cost will be $\varepsilon_r \cdot \phi_{jk}^i \cdot \frac{\partial C_T}{\partial r_j^k}$. Summing over all adjacent nodes k, we have

$$\frac{\partial C_T}{\partial r_j^i} = \sum_{(i,k)\in E} \phi_{jk}^i \times \left[c'_{ik}(ik) + \frac{\partial C_T}{\partial r_j^k} \right] \tag{4}$$

An increment ε_ϕ in ϕ_{jk}^i causes an increment $\varepsilon_\phi \cdot t_j^i$ in the traffic on link (i,k). If k is not the destination node, this increment will cause an additional $\varepsilon_\phi \cdot t_j^i$ to the traffic at k destined for j, thus we have

$$\frac{\partial C_T}{\partial \phi_{jk}^i} = t_j^i \times \left[c'_{ik}(ik) + \frac{\partial C_T}{\partial r_j^k} \right] \tag{5}$$

To minimize the objective, a widely used method is to find a stationary point for C_T with respect to variations in ϕ. For the constraints $\sum_{(i,k)\in E} \phi_{jk}^i = 1$ and $\phi_{jk}^i \geq 0$, we have $\frac{\partial(\sum_{(i,k)\in E} \phi_{jk}^i - 1)}{\partial \phi_{jk}^i} = 1$ and $\frac{\partial \phi_{jk}^i}{\partial \phi_{jk}^i} = 1$. With positive Kuhn-Karush-Tucker (KKT) multipliers $\lambda_{ij}(i,j \in V)$ and $\lambda'_{ij}(i,j \in V)$ introduced, to minimize C_T, the expressions $\frac{\partial C_T}{\partial \phi_{jk}^i} - \lambda_{ij} - \lambda'_{ij} = 0$ and $\lambda'_{ij}\phi_{jk}^i = 0$ should be satisfied. It is easy to verify that if $\phi_{jk}^i = 0$, we have $\lambda'_{ij} \geq 0$; If $\phi_{jk}^i > 0$, we have $\lambda'_{ij} = 0$. Thus, we have

Theorem 2 ([11]). The necessary condition for a minimum of C_T with respect to ϕ for all $i \neq j$ and link $(i,j) \in E$ is

$$\frac{\partial C_T}{\partial \phi_{jk}^i} \begin{cases} = \lambda_{ij} & \phi_{jk}^i > 0 \\ \geq \lambda_{ij} & \phi_{jk}^i = 0. \end{cases} \tag{6}$$

Theorem 2 is not sufficient because C_T can have inflection points as a function of ϕ. Considering Equation (5), the value of $\frac{\partial C_T}{\partial \phi_{jk}^i}$ is greatly affected by the injected traffic t_j^i. It can be verified that Equation (6) would be sufficient to minimize C_T if t_j^i is removed. Please refer to [11] for detailed proof.

Theorem 3 ([11]). The sufficient condition for a minimum of C_T with respect to ϕ for all $i \neq j$ and link $(i,j) \in E$ is

$$c'_{ik}(x_{ik}) + \frac{\partial C_T}{\partial r_j^k} \geq \frac{\partial C_T}{\partial r_j^i} \tag{7}$$

In this paper, we extend Theorem 2 and Theorem 3 to fit our energy efficient routing problem and propose a new routing scheme. Notice that Gallager also derived a minimum-delay routing algorithm based on Theorem 2 and Theorem 3 in [11]. The main differences are as follows: (1) While Gallager's work only focuses on minimizing the overall traffic delay, our work simultaneously optimizes both energy consumption and traffic delay; (2) Gallager's work establishes routing paths from sources to destinations every instant to guarantee loop-free routing, we develop a different way to guarantee that all the packets go though loop-free paths. That is we develop a path finding algorithm (D_LoopFree) such that loop-free paths are computed at the beginning of the scheme. Then, an algorithm is provided to guide traffic distribution among these loop-free paths to reduce energy consumption. In this way, our method establishes routing paths only once, which reduces the path establishment time; Most importantly, (3) while Gallager's work only applies to quasi-static traffic scenario [11], our work can be easily extended for dynamic situation.

We now seek a routing algorithm to minimize the objective C_T by exploiting Theorem 2 and Theorem 3. Assume that S_j^i is the the neighbor set of node v_i in the $GEN\_G(j)$ built in our loop-free path-finding algorithm. According to Theorem 2 and Theorem 3, to accomplish the routing goal, each node v_i must incrementally decrease the value ϕ_{jk}^i of links for which the sum of $c'_{ik}(x_{ik}) + \frac{\partial C_T}{\partial r_j^i}$ is large. As a result, we put forward an algorithm (D_Routing) to modify the ϕ-value iteratively. Note that D_Routing only routes traffic on the DAG between the current node and the destination, thus is more efficient than Gallager's work. For the sake of simplicity, we denote the marginal distance of node v_i to node v_j by $D_j^i = \frac{\partial C_T}{\partial r_j^i}$ and, accordingly, denote the marginal cost of link (i,k) by $w_{ik} = c'_{ik}(x_{ik})$.

In each iteration κ, each node v_i takes the routing variables for all $v_k \in S_j^i$ as input, which are denoted by $(\phi_j^i)^{\kappa-1}$. When this iteration finishes, the node returns the routing variables that have been updated, which are denoted by $(\phi_j^i)^{\kappa}$. For simplicity, we refer to the κ^{th} iteration as D_Routing(κ). For the destination node v_j, the process of D_Routing(κ) in each node v_i is described as follows:

1) Collect D_j^k and w_{ik} from node v_k, $\forall v_k \in S_j^i$, then calculate D_j^i according to Equation (4) and send this value to outgoing links in $GEN\_G(j)$. Note that if $i = j$, then $D_j^i = 0$.

Algorithm 1. D_Routing(κ)

Input: $(\phi_j^i)^{\kappa-1}$
Output: $(\phi_j^i)^{\kappa}$
1: $k_{min} = \arg\min_{v_k \in S_j^i}\{D_j^k + w_{ik}\}$.
2: **for all** $v_k \in S_j^i$ **do**
3: $\quad \delta_{jk}^i \leftarrow (D_j^k + w_{ik}) - (D_j^{k_{min}} + w_{ik_{min}})$.
4: **end for**
5: **for all** $v_k \in S_j^i$ **do**
6: $\quad$ **if** $k == k_{min}$ **then**
7: $\quad\quad (\phi_{jk}^i)^{\kappa} \leftarrow \phi_{jk}^i + \sum_{k \neq k_{min}} \delta_{jk}^i \times \gamma$.
8: $\quad$ **else**
9: $\quad\quad (\phi_{jk}^i)^{\kappa} \leftarrow \phi_{jk}^i - \delta_{jk}^i \times \gamma$
10: $\quad$ **end if**
11: **end for**

2) Mark the node with the minimum value of $(D_j^k + w_{ik})$ as k_{min}, then calculate $\delta_{jk}^i = (D_j^k + w_{ik}) - (D_j^{k_{min}} + w_{ik_{min}}), \forall v_k \in S_j^i \wedge k \neq k_{min}$.
3) Calculate $(\phi_j^i)^{\kappa}$ for every node $v_k \in S_j^i$. If $k = k_{min}$, $(\phi_{jk}^i)^{\kappa} = (\phi_{jk}^i)^{\kappa-1} + \sum_{k \neq k_{min}} \delta_{jk}^i \times \gamma$. Else, $(\phi_{jk}^i)^{\kappa} = (\phi_{jk}^i)^{\kappa-1} - \delta_{jk}^i \times \gamma$.

Note that γ is the iteration step size. D_Routing(κ) can also be seen in Algorithm 1. After an iteration, each node v_i reallocates the traffic among its output links. When this iterative process terminates, the final set of ϕ-value is the solution of D_Routing, which is denoted by $(\phi_j^i)^*$.

Next, we discuss the convergence property of the distributed iterative algorithm. We denote the cost of the entire network in the current time by C_T, and we denote the cost after an iteration κ by $(C_T)^{\kappa}$. Assume that the value of γ is sufficiently small. The following theorem demonstrates the convergence property of this algorithm:

Theorem 4. After each iteration κ, for all $\epsilon > 0$, if $0 < \gamma < \epsilon$ is small enough, then $(C_T)^{\kappa} - (C_T)^{\kappa-1} \leq 0$.

Theorem 4 was firstly proved by Gallager [11]. Note that Theorem 4 is based on the assumption that γ is sufficiently small. The value of γ plays an important role in the convergence. A small γ guarantees the convergence of the algorithm but leads to slow convergence. When we increase the value of γ, the convergence speed increases as well as the risk that the algorithm will not converge. We will discuss the selection of γ in Section IV by considering various traffic patterns, and performing simulations to address this issue in Section V. Because the marginal distance D_j^i is computed recursively in Equation (4), the time complexity for one iteration is $O(D)$. Note that D is the diameter of the network topology.

D. Discussion

1) Traffic-Aware: As stated above, the construction of the *DAG* in D_LoopFree only requires information about the network topology. Because no traffic information is considered, D_LoopFree is not traffic-aware. Consider a topology with three nodes in Figure 3(a), where each node pair has a bi-directional link. For the destination node v_t, D_LoopFree will first generate a shortest path tree to cover two other nodes v_i and v_j by reserving link (i, t) and link (j, t). Let us assume that v_i has a larger index than node v_j. Noticing that node v_i and v_j have the same heights in the shortest path tree, D_LoopFree will reserve the link from node v_i to node v_j because node v_i has a larger index. After running D_LoopFree, node v_i will have two paths to the destination node v_t: $i \rightarrow t$ and $i \rightarrow j \rightarrow t$. Node v_j has only one path: $j \rightarrow t$. Consider the case whereby node v_j has M units of demand to send to node v_j, while node v_i has no demands to send. Then, node v_j will route all its demands to the destination node via path $j \rightarrow t$ (in Figure 3(b)). The total cost is $C_T = c(M)$. However, if we modify the *DAG* generated by D_LoopFree by replacing link (i, j) with link (j, i), then node v_j will have two paths to destination node v_j: $j \rightarrow t$ and $j \rightarrow i \rightarrow t$ (in Figure 3(c)). Assuming that ψ is the proportion of the total demands routed via $j \rightarrow t$, then the total cost is $(C_T)' = c(\psi M) + 2c((1-\psi)M)$. In some cases, for instance, if $c(x) = \alpha(x)^{\beta}$ and $\alpha > 0, \beta > 1$, we have that $C_T > (C_T)'$. This means that if we modify the *DAG*, we can obtain a better solution. Specifically, the *DAG* generated by D_LoopFree is not always optimal because it is not traffic-aware. To address this issue, we also provide an enhanced version of D_LoopFree: D_LoopFree-TA. This version takes the traffic in different nodes into consideration by reserving links according to the traffic status in each node and not the node index. One possible alternative solution is to replace the criteria of reversing links from the node index by the value of marginal distances.

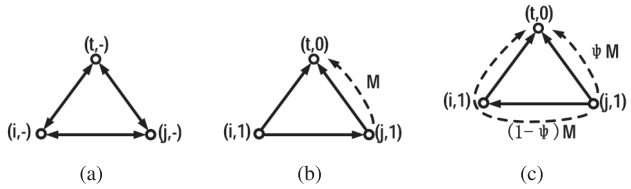


Fig. 3. (a) a topology; (b) $i > j$, generate GEN_G(t); (c) modify GEN_G(t) by replacing link (i,j) with link (j,i) .

2) Stability: The traffic allocation algorithm D_Routing adjusts the traffic distribution iteratively to minimize the objective. Specifically, at each iteration, a router changes its own routing variables. Until the algorithm terminates, it updates the routing variables iteratively after a short time interval, denoted by δt. Now we discuss the network stability of this algorithm. We focus on a single flow (e.g., a TCP connection between two hosts). Consider a situation where the flow experiences more than one iteration of HDEER during its lifetime. Since in every iteration the route for the flow may change, packets from the same flow may follow different paths, leading to routing instability issue. To address the path change issue of flows, we provide two auxiliary policies as follows. Firstly, as a large proportion (approximately 90%) of flows on Internet are typically short flows with the flow lifetime (time used for completing the transmission of all the packets from this flow), denoted by md, which is less than 10 ms [18], [19] in general. Thus setting δt

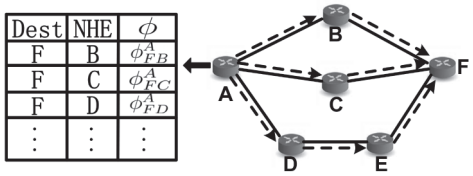


Fig. 4. The RIB in router A, where *Dest* means the Network Id of the destination subnet, NHE means the Next Hop Efficient and ϕ means the corresponding Routing Variable.

several times larger than md (e.g., $\delta t = 5\ md$) would help overcome the above issue. The principle is to guarantee that most of the flows will experience no more than one iteration. Secondly, we keep the paths of all flows remain unchanged during their lifetimes. As the time interval affected by this policy is at most md in most cases, which is only a fraction of δt, the policy is feasible.

IV. Implementation Details

In this section, we discuss the details of the implementation of our distributed routing scheme in realistic networks. Fractional routing is assumed in the scheme, where the flow may be split among multiple paths [12]. The scheme can be conveniently implemented in an Autonomous System of network (AS). Speed-scaling mechanism is provided as an architectural support, thus we assume that the routers in the network are equipped with the Adaptive Link Rate (ALR) [3]. To enable an efficient hardware implementation, we should first address the following issues: (1) How do we build the Routing Information Base (RIB) in each router? (2) How do we implement multipath routing in the network? and (3) How do we adapt our algorithm to both static and dynamic traffic scenarios? We answer these questions in the following.

A. RIB

A RIB with hop-by-hop routing involves at least three information fields: (1) the **Network Id** of the destination subnet; (2) the **Next Hops** to which the packet is to be sent on the way to its final destination; and (3) the **Cost** of the path through which the packet is to be sent. Note that we leave the first field the same as usual RIB. We replace the **Next Hops** with **Next Hops Efficient** in our scheme, which is determined by the D_LoopFree algorithm. Besides, we replace the last one with the **Routing Variable**, which is defined in Section II. Specifically, assume that RIB_i is the routing table of node v_i and that routing entry rib^i_{jk} is a record in RIB_i. Note that $v_k \in S^i_j$. Then the third field of rib^i_{jk} is ϕ^i_{jk}, and

$$rib^i_{jk} \sim (j, k, \phi^i_{jk})$$

means that the proportion of traffic routed from node v_i to destination node v_j via neighboring node v_k is ϕ^i_{jk}. An example of RIB is illustrated in Figure 4.

Algorithm 2. Routing in a Static Traffic Scenario

1: **for all** $v_i \in V$
2: Run D_LoopFree to find the next-hop set S^i_j for each sink node v_j.
3: $\psi \leftarrow Inf, \kappa \leftarrow 0$.
4: **while** $\psi > \theta$ **do**
5: $\kappa \leftarrow \kappa + 1$.
6: Run D_Routing(κ) to modify ϕ^i_j.
7: $\psi \leftarrow \sum_{v_k \in S^i_j} \frac{|(\phi^i_{jk})^{\kappa-1} - (\phi^i_{jk})^{\kappa}|}{|S^i_j|}$.
8: **end while**
9: $(\phi^i_j)^* \leftarrow (\phi^i_{jk})^{\kappa}$.
10: Route traffic with the fixed $(\phi^i_j)^*$.
11: **end for**

B. Traffic Scenarios

The traffic scenarios considered in our paper involves static traffic scenario and dynamic traffic scenario. The static traffic scenario refers to the situation where traffic demands in the nodes are considered to be constant. It happens when a day is partitioned into several time intervals according to the traffic volume. During each time interval, routing variables are computed according to constant peak traffic demands in the nodes. The dynamic traffic scenario refers to the situation where traffic demand in each node changes over time. The routing variables should be adjusted accordingly thus to react to the traffic fluctuation. Link traffic of each interface is measured periodically using the widely applied Simple Network Manager Protocol (SNMP) or other similar protocols.

1) Static Traffic Scenario: In this scenario, each node will first run D_LoopFree to find the next-hop set S^i_j for each sink node v_j, then run D_Routing iteratively to modify the routing variables. Because the updating stage is iterative, we set a threshold θ to control the iteration. When the average distance between vector $(\phi^i_j)^{\kappa}$ and $(\phi^i_j)^{\kappa-1}$ becomes no more than θ, node v_i will stop updating its routing variables. Then, it will notify its neighbors with its latest state and stop sending update information to its neighbors. When all nodes in the network stop updating, the algorithm terminates. This process is depicted in Algorithm 2.

Identifying the next-hop sets for each destination node in our algorithm is similar to the next-hop search process in the Routing Information Protocol (RIP). The main difference is that we use the D_LoopFree proposed above rather than the distributed Bellman-Ford algorithm of the RIP to compute the next hops. Multi-path routing is allowed in our scheme. Because $RFC2991$ discusses multi-path routing in general, we will not describe it here.

To perform the first task of generating the DAG for each destination node, a shortest path tree should first be constructed. There are many link-state routing protocols that can achieve this goal in a distributed manner, such as the most widely used OSPF routing protocol. Because link-state advertisements (LSAs) adopted by many link-state routing algorithms already contain the topology information, each node can easily obtain

the topology of the network by exchanging LSAs without modifications. With sufficient link-state information being released in the network, every node can easily construct the entire network topology. Then, each node computes a shortest path tree rooted in each destination node and adds alternative links into the shortest path tree according to our algorithm to finish the DAG generation process. After D_LoopFree, each node v_i determines the corresponding next hop set S_j^i for each destination node v_j.

We now discuss the implementation of the second stage, i.e. to distribute the appropriate proportion of traffic among the links. To better facilitate our description, we first provide definitions of the *upstream* node and *downstream* node. Given a certain destination node v_t, we call node v_i the *upstream* node of node v_j if $j \in S_t^i$. Correspondingly, we call node v_j the *downstream* node of node v_i. In each iteration, when each node v_i has received the marginal distances from all its *downstream* nodes, the node computes its own marginal distance and transmits this value to all its *upstream* nodes. Due to the LSAs, the values of marginal distances can be incorporated into an LSA. In particular, $RFC3630$ has defined the Traffic Engineering Link State Advertisement (TE-LSA), which records link load information. An iteration finishes when all the nodes in the network have computed their marginal distance. When this algorithm terminates, it will output the optimal configuration of the ϕ-value.

When the optimal routing variable set ϕ is computed, it remains unchanged in the static traffic scenario. Now we discuss the implementation of the data forwarding process under our distributed routing scheme. The packet under HDEER routing scheme is forwarded hop-by-hop from the source node to the destination node. The paths connecting each source node and each destination node are constructed from the corresponding neighbor sets derived from D_LoopFree in Section III-A. When the traffic splitting ratios over multiple paths are computed (i.e. ϕ), packets can be forwarded among these multiple paths accordingly.

2) Dynamic Traffic Scenario: In this scenario, as traffic demands fluctuate over time, we should periodically modify the optimal $(\phi_j^i)^*$ accordingly. However, D_Routing is unable to react to dynamic traffic because the speed of convergence to the optimal routes depends on a global constant. As stated in Section III-C, a small γ guarantees the convergence of D_Routing but leads to slow convergence, while a large γ may cause the algorithm not to converge.

To make traffic allocation process suitable for dynamic traffic scenario, we present a technique for modifying the routing variables in each node. With this technique, traffic is incrementally moved from the links with large values of $D_j^k + w_{ik}$ to links with the least value as D_Routing does. Thus, a new algorithm, Algorithm 3, is developed to using adaptive stepsize to modify the routing variables so that a global constant is no linger necessary. The adaptive stepsize ensures that the amount of traffic moved away from a link is proportional to how large the value of $D_j^k + w_{ik}$ of the link is compared to the average value of $D_j^k + w_{ik}$ among all the adjacent nodes. For coherence, we refer the D_Routing adopted in static traffic scenarios as D_Routing-S and refer this heuristic in dynamic traffic scenarios as D_Routing-D. Correspondingly, we use D_Routing-D(κ) to represent an iteration in D_Routing-D. D_Routing-D(κ) is depicted in Algorithm 3, lines 12–23. This heuristic can quickly get a near optimal solution. We use a constant value F to control the iteration. Specifically, D_Routing-D terminates after only F iterations. It is different from the termination condition in D_Routing-S, where the number of iterations cannot be accurately estimated before D_Routing-S terminates. Note that the value of F has effect to the performance of D_Routing-D, we will show this relationship with experiments in Section V.

Algorithm 3. Routing in a Dynamic Traffic Scenario

1: Each node in the network continues monitoring and computing r_j^i during each $\triangle t$.
2: **for all** $l \in L$ **do**
3: **for all** $v_j \in V$ **do**
4: Notify each node along $GEN\_G(j)$ to start a new epoch l.
5: **end for**
6: **for all** $v_i \in V$ **do**
7: Update the value of $r_j^i(l)$ with the new traffic record r_j^i.
8: Run D_LoopFree to find the next-hop set S_j^i for each sink node v_j.
9: $\kappa \leftarrow 0$.
10: **while** $\kappa < F$ **do**
11: $\kappa \leftarrow \kappa + 1$.
12: **for all** $v_j \in V$ **do**
13: $k_{min} = \arg\min_{v_k \in S_j^i}\{D_j^k + w_{ik}\}$
14: $\tau \leftarrow \frac{1}{|S_j^i|} \times \sum_{v_k \in S_j^i}(D_j^k + w_{ik})$.
15: **for all** $v_k \in S_j^i \wedge k \neq k_{min}$ **do**
16: **if** $\tau < (D_j^k + w_{ik})$ **then**
17: $(\phi_{jk}^i)^\kappa \leftarrow (\phi_{jk}^i)^{\kappa-1} \times \frac{\tau}{D_j^k + w_{ik}}$.
18: **else**
19: $(\phi_{jk}^i)^\kappa \leftarrow (\phi_{jk}^i)^{\kappa-1}$.
20: **end if**
21: **end for**
22: $(\phi_{jk_{min}}^i)^\kappa \leftarrow 1 - \sum_{v_k \in S_j^i \wedge k \neq k_{min}} (\phi_{jk}^i)^\kappa$.
23: **end for**
24: **end while**
25: $(\phi_j^i)^* \leftarrow (\phi_{jk}^i)^\kappa$.
26: Route traffic with the fixed $(\phi_j^i)^*$ in this epoch until a new epoch is provided.
27: **end for**
28: **end for**

Now we address the issue of adapting our fast convergent algorithm $D\_Routing - D$ to the dynamic traffic scenarios. We divide a certain length of time $(L \times \triangle t)$ into L time windows with a fixed length of $\triangle t$. During each time windows $l \in L$, the corresponding routing variables of each node will remain constant. Here, $\triangle t$ is used as the time interval for reconfiguration. Because frequent reconfiguration in realistic networks is not

TABLE I
THE TEST NETWORK TOPOLOGIES

Network	GÉANT	AS1755	Sprint	Random	AT&T
Site	Europe	US	US	-	US
Nodes	23	27	43	50	111
Links	74	92	174	188	290

practical, as it may produce overhead and network failures, the configuration time length $\triangle t$ should be set such that the reconfiguration will not be executed frequently. Each node in the network monitors traffic of its own, computing a new traffic record r_j^i after every fixed time interval $\triangle t$. When a new window $l+1 \in L$ starts, each node drops the current traffic record $r_j^i(l)$ and modifies its routing variable ϕ_j^i with the new traffic record r_j^i. This process is also shown in Algorithm 3, lines 1–28.

V. EVALUATION

This section provides the evaluation results of our distributed routing scheme on the widely used network simulator: $\mathtt{NS3-Simulator}$[1]. We focus this evaluation on four aspects: (1) the tradeoff between energy consumption and traffic delay in our bi-objective optimization, (2) the performance of HDEER compared with the optimal solution of the bi-objective problem and the solution of shortest path routing, (3) the convergence property of HDEER with D_Routing-S and D_Routing-D, and (4) the performance of HDEER under dynamic real traffic traces.

A. Evaluation Settings

1) Environment: We perform our simulation on a laptop with an Intel Core 2 Quad Core CPU $Q8200@2.33GHz \times 4$ with $8.0GiB$ of memory. The simulations are performed using a discrete-event network simulator for Internet systems, namely, NS3, which can capture the system's behavior at the packet level. The optimal solution of the bi-objective problem used as a benchmark is obtained by LINGO.

2) Networks and Traffic: We test various types of topologies in this evaluation, including the topology of GÉANT [20] in 2004 and three other topologies from an ISP topology mapping engine, namely, Rocketfuel [21]. The three networks from Rocketfuel are AS1755, SPRINT and AT&T. The above four topologies all originate from real networks but vary in size. In addition to these real networks, we also evaluate our algorithm on a random network whose degree distribution follows a power law. Detailed information about these test network topologies is listed in Table I. Note that we assume that the network topology is fixed, with no link failure in our evaluations.

We perform the evaluation under two different types of traffic scenarios: static traffic scenario and dynamic traffic scenario. In the static traffic scenario, the traffic demand between each node and sink pair does not change. The traffic model proposed by Nucci *et al.* [22] is assumed in the static traffic scenario, where the 40% is high bit-rate traffic, between 1 $Mbps$ and 80 $Mbps$, and the remaining 60 % is low bit-rate traffic, up tp 1 $Mbps$. Both high and low bit rate traffic follow a Lognormal distribution.} In the dynamic traffic scenario, the traffic demand between each node and sink pair varies during the day. Bursty traffic in real networks always fluctuates rapidly. To be more convincing, we test our algorithm with real network traffic traces in GÉANT [20], which cover the data for one week picked at random. The traffic matrices are sampled on the network every 15 mins.

3) Energy and Delay: $f_{ik}(x_{ik}) = \frac{x_{ik}^2}{CA_{ik}}$ is used as our energy function as in [23]. Similar to most other routing algorithms, $d_{ik}(x_{ik}) = \frac{x_{ik}}{CA_{ik}-x_{ik}}$ is used as our delay function as in [24]. Both energy and traffic delay functions are convex.

To evaluate the performance of our bi-objective optimal routing scheme, we compare the evaluation results with the optimal results, denoted by OPT. Using these network topologies, link capacities and the corresponding traffic matrices according to the traffic generated in the $NS3$ simulator as input, LINGO will output the corresponding optimal solution to our bi-objective model within polynomial time. Due to the absence of the traffic matrix information in real networks and the tremendous computational time and memory costs needed for computing, determining the optimal solution from a centralized optimization solver under a real traffic scenario is not available. Although the optimal solution is often impossible to obtain under real operating conditions in real networks, it can serve as the lower bound for evaluating the performance of HDEER.

In addition to the benchmarking of the optimal solution, we also compare our distributed routing algorithm to a shortest path routing algorithm, denoted by SPT, which is one of the most widely applied routing algorithms in real-world networks. In our evaluation, the "shortest path" refers to the path with the minimum number of hops. We implement the shortest path routing algorithm using the Dijkstra algorithm, which can also be performed in a distributed manner. Because this routing algorithm picks routes for each flow with the minimum number of hops, it can achieve a good traffic delay performance.

B. Simulation Results

1) Pareto Optimal Points: We start by finding the Pareto optimal points in this subsection. The test topology used in this evaluation is from AS1755. As previously stated, a pair (w_1,w_2) corresponds to a Pareto optimal point. We can get the partial curve of the Pareto frontier, which consists of all the Pareto optimal points, by gradually varying the values of w_1 and w_2. We use a static traffic scenario in this simulation. The parameter θ is set to 5×10^{-5}. We refer to the total cost of the bi-objective optimization formulation (BOF) as T, refer to the total energy consumption and total traffic delay as E and D correspondingly. Table II illustrates the values of the total cost savings, denoted by TS ($TS = (1 - \frac{T_{HDEER}}{T_{SPT}}) \times 100\%$), energy savings, denoted by ES ($ES = (1 - \frac{E_{HDEER}}{E_{SPT}}) \times 100\%$) and traffic delay ratio, denoted by DR ($DR = \frac{D_{HDEER}}{D_{SPT}} \times 1$). Noting that each result is averaged among 10 independent tests, where the traffic matrices are generated with different seeds. We set the repeat number as 10 because it is found that the results are similar when the

[1]https://www.nsnam.org/.

TABLE II
PERFORMANCE COMPARISON WITH DIFFERENT (w_1, w_2) PAIRS, EACH VALUE IS AVERAGED AMONG 10 INDEPENDENT TESTS, FOLLOWED BY THE CORRESPONDING STANDARD DEVIATIONS

w_1	w_2	TS: Avg(SD)	ES: Avg(SD)	DR: Avg(SD)
1.00	0.00	32.2%(1.87%)	32.2%(1.87%)	1.022(0.0039)
0.75	0.25	22.8%(1.30%)	31.0%(1.54%)	1.020(0.0084)
0.50	0.50	15.1%(0.97%)	30.8%(1.23%)	1.006(0.0089)
0.25	0.75	7.3%(1.68%)	28.1%(3.21%)	0.996(0.0124)
0.00	1.00	0.7%(0.49%)	9.8%(2.20%)	0.993(0.0049)

TABLE III
PERFORMANCE COMPARISON WITH DIFFERENT TOPOLOGIES, EACH VALUE IS AVERAGED AMONG 10 INDEPENDENT TESTS, FOLLOWED BY THE CORRESPONDING STANDARD DEVIATIONS ($w_1 = 0.5$, $w_2 = 0.5$)

	ES: Avg(SD)		DR: Avg(SD)	
	OPT(%)	HDEER(%)	OPT(1)	HDEER(1)
GEÁNT	19.5(4.05)	12.1(2.16)	0.96(0.036)	1.00(0.004)
AS1755	38.4(5.01)	37.9(1.56)	0.95(0.039)	1.01(0.007)
SPRINT	46.8(9.28)	32.1(5.72)	0.85(0.047)	1.04(0.012)
RANDOM	42.6(7.40)	30.8(4.72)	0.89(0.059)	1.02(0.009)
AT&T	-	16.9(0.94)	-	1.01(0.009)

number is larger than 5. The corresponding standard derivations are also provided. It can be observed in the following:

i) A setting of (w_1,w_2) that increases the energy savings will undermine traffic delay performance and vice versa. In other words, there is a trade-off between energy savings and traffic delay. When $w_1 = 0.75$ and $w_2 = 0.25$, HDEER can reduce 31.0% energy consumption compared with SPT, which is more than the energy savings achieved with $w_1 = 0.25$ and $w_2 = 0.75$ (28.1%). However, the former incurs more delays (1.020) than the latter (0.996) does. To apply our algorithm to real networks, the pareto frontier can serve as a guideline for choosing an optimal setting of (w_1,w_2).

ii) If only the single objective of reducing energy consumption is considered, HDEER can achieve 32.2% energy savings over SPT. This result demonstrates that HDEER performs well for energy consumption reduction.

2) Performances in Static Traffic Scenarios: In this section, we evaluate our distributed routing algorithm under the static traffic scenario. We set both w_1 and w_2 to 0.5 and keep them fixed. We compare the performance of HDEER with OPT and SPT using five topologies listed in Table I in the following (Note that these topologies represent small networks, medium-sized networks and large networks.) The simulation results are shown in Table III, which depicts the normalized performances of the three algorithms in terms of total cost, energy consumption and traffic delay, respectively. Each result is averaged among 10 experiments along with the corresponding standard deviation. From the results, note that:

i) HDEER can save a substantial amount of energy. It can be seen from Table III that the energy consumed by HDEER is close to that of the optimal solution solved by LINGO and represents a significant reduction compared to SPT. Specifically, HDEER can obtain a 26.0% energy savings on average compared with SPT using all the five test topologies.

ii) HDEER produces traffic delays that are within a small percentage of the SPT. It can be seen that HDEER will only produce slightly more traffic delay overhead in the network compared to SPT, 1.6% on average.

iii) HDEER exhibits good scalability. It is worth noting that when using a large topology, e.g., the topology of AT&T, which contains 111 nodes and 290 links, as the input, even LINGO fails to output the optimal solution due to a lack of memory. Nevertheless, our distributed algorithm can still run normally and converge after several iterations. This conclusion is in line with the motivations of our distributed scheme.

We also compare the performance of HDEER with a centralized energy saving algorithm, namely the Iterative Greedy Least-Power Routing algorithm (IGLPR), proposed by Antonakopoulos [23]. w_1 is set to 1 in this simulation, thus the optimization goal of HDEER is only energy conservation in accordance with IGLPR. Simulation results showing in Table IV indicate that HDEER performs better in energy conservation than IGLPR on average (IGLPR: 17.1%, HDEER: 22.5%). Noting that when using the AS1755 topology as the test topology, the IGLPR achieves similar energy savings with HDEER (IGLPR: 31.4%, HDEER: 31.6%), but it brings longer delay than HDEER does (IGLPR: 1.11, HDEER: 1.04).

Finally, we measured the average computation time of one iteration of our algorithm. As we can observe from Table V, the computation time of one iteration of HDEER is at the level of microseconds. As a result, HDEER is fast to react to real-time traffic variations.

3) Convergence Properties: In this section, we investigate the convergence behavior of HDEER with D_Routing-S and D_Routing-D. We first study the convergence property of both D_Routing-S and D_Routing-D by drawing four curves to track the convergence processes. Then we compare the convergence results of these two algorithms. The traffic scenario used here is static black, which has been defined in Section V-A(2). Here we use three traffic, namely Traffic-I, Traffic-II and Traffic-III, generated with three different seeds to test the performance. With the step size γ in D_Routing-S set to 50, 500 and 5000, Figure 5(a), Figure 5(b), Figure 5(c) and Figure 5(d) demonstrate the results. Note the following:

i) When $\gamma = 50$, the energy saving curve in Figure 5(a) decreases smoothly and slowly over time (approximately 60 iterations). Now we increase the value of γ. When $\gamma = 500$, the curves in Figure 5(b) exhibit a faster convergence speed for Traffic-I, but exhibit instability for Traffic-II and Traffic-III. When $\gamma = 5000$, the curves in Figure 5(c) fail to converge.

ii) The energy saving curves in Figure 5(d) decrease fast over time. It needs less than 7 iterations for each curve to reach a similar result.

iii) D_Routing-S outperforms D_Routing-D in energy saving and traffic delay, but the performance gap between the two algorithms is small (energy saving gap is 2.7% and delay gap is 0.01%). In other words, D_Routing-D is almost as efficient as D_Routing-S.

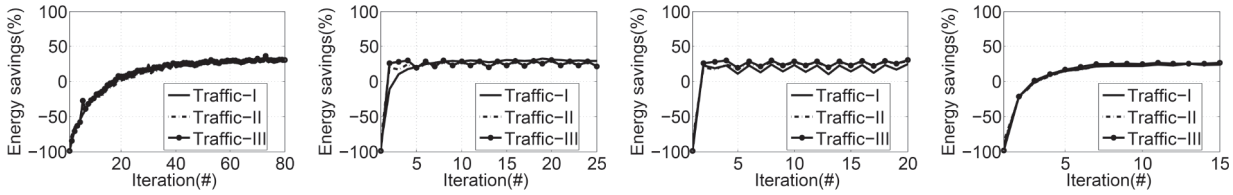
To ensure a smooth convergence process, γ should be chosen carefully. Through a number of experimentations, we found

TABLE IV
PERFORMANCE COMPARISON BETWEEN IGLPR AND HDEER, EACH VALUE IS AVERAGED AMONG 10 INDEPENDENT TESTS, FOLLOWED BY THE CORRESPONDING STANDARD DEVIATIONS ($w_1 = 1, w_2 = 0$)

	ES: Avg(SD)				DR: Avg(SD)			
	GEÁNT	AS1755	SPRINT	Mean	GEÁNT	AS1755	SPRINT	Mean
IGLPR	5.8%(2.60%)	31.4%(2.19%)	14.1%(4.71%)	17.1%	1.08(0.022)	1.10(0.015)	1.14(0.070)	1.11
HDEER	12.2%(1.83%)	31.6%(1.90%)	23.7%(5.95%)	22.5%	1.01(0.005)	1.02(0.017)	1.09(0.019)	1.04

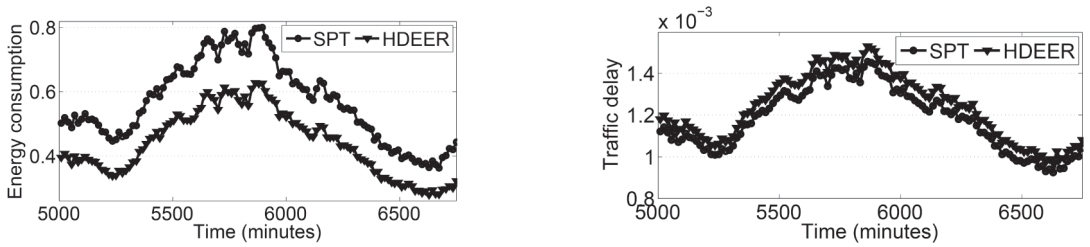
TABLE V
COMPUTATION TIME OF ONE ITERATION OF HDEER, EACH VALUE IS AVERAGED AMONG 10 INDEPENDENT TESTS, FOLLOWED BY THE CORRESPONDING STANDARD DEVIATIONS

Topologies	AS1755	GÉANT	SPRINT	RANDOM	AT&T
Time: Avg(SD)	$37.0\mu s(0\mu s)$	$43.5\mu s(0\mu s)$	$77.5\mu s(10.94\mu s)$	$146.7\mu s(9.43\mu s)$	$1021.2\mu s(18.71\mu s)$



(a) D_Routing-S (γ=50) (b) D_Routing-S (γ=500) (c) D_Routing-S (γ=5000) (d) D_Routing-D

Fig. 5. The comparison among the convergence curves of D_Routing-S with different values of γ and D_Routing-D.



(a) (b)

Fig. 6. Performance comparison during time interval [5000,6750] with real traffic traces ($F = 14$). (a) Energy consumption. (b) Delay.

that a proper value of γ can be 50, whereby the corresponding convergence time is long (see Figure 5(c)). According to our simulation, when γ is set larger than 500, the algorithm fails to converge for some traffic. A simple and effective way of determining such a γ is choosing values on the same order of magnitude as $(D_j^k + w_{ik})$ by experiments. However, it is hard to apply this in dynamic traffic scenarios during real-world operation. D_Routing-D naturally overcomes this issue and converges fast.

4) Performance in Dynamic Traffic Scenarios: In this section, we evaluate our distributed routing algorithm in real dynamic traffic scenario. We conduct our simulations using the topology of GÉANT and a set of real traffic traces [20]. Since it is impossible to get optimal solutions via a centralized solver in real traffic scenario, we no longer use OPT as a benchmark in this section. Instead, we set F to 6, 10 and 14 to study the effect of different values of F. The simulation results are shown in Figure 6(a), Figure 6(b), Figure 7 and Table VI. Note that:

i) Figure 6(a) illustrates that HDEER performs well in energy saving. Figure 6(b) shows that the traffic delay overheads introduced by HDEER is limited. A longer time (a seven-day sample interval) performance results are depicted in Figure 7.

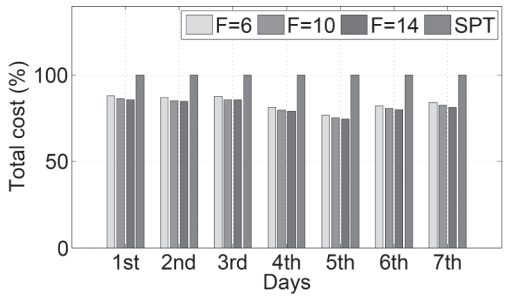


Fig. 7. Performance comparison ($F = 6$, $F = 10$, $F = 14$ and SPT) during a randomly chosen week with real traffic traces.

ii) HDEER always reduces sufficient energy consumption while it introduces delays that are within a small percentage of the shortest path routing algorithm. During these sampled 7 days, the average energy saving ratio of HDEER is 17.47%, and the average traffic delay ratio is 1.063.

iii) Table VI demonstrates that the value of F will affect the performance of HDEER. A larger value of F performs better than a smaller value of F under the same network setting (the average performance in 7 days:

TABLE VI
PERFORMANCE COMPARISON WITH DIFFERENT VALUE OF F DURING A RANDOMLY CHOSEN WEEK WITH REAL TRAFFIC TRACES

	Day-1	Day-2	Day-3	Day-4	Day-5	Day-6	Day-7	Mean	Standard Deviation
ES-6 (%)	12.26	12.92	12.28	18.80	23.36	17.81	15.84	16.18	4.13
ES-10 (%)	13.82	14.81	14.20	20.48	24.88	19.06	17.64	17.84	4.01
ES-14 (%)	14.43	15.24	14.35	21.04	25.38	20.13	18.21	18.40	4.10
DR-6 (1)	1.077	1.066	1.063	1.059	1.062	1.074	1.091	1.070	0.0112
DR-10 (1)	1.070	1.058	1.053	1.051	1.052	1.063	1.081	1.061	0.0111
DR-14 (1)	1.066	1.054	1.048	1.048	1.049	1.060	1.078	1.058	0.0113

ES-6(16.18%), ES-10(17.84%), ES-14(18.40%) and DR-6(1.703), DR-10(1.061), DR-14(1.058).

VI. RELATED WORK

In this section, we summarize some related works on the energy saving problem at the network-wide level in general wired networks. There is a substantial amount of literature that focuses on this issue, which can be generally classified into two categories: centralized solutions and distributed solutions. Recently, some papers have provided thorough reviews on this topic [2], [25]–[27]. Here, we only discuss the methods that are closely related to our scheme.

A. Centralized Approaches

The pioneering work performed by Gupta *et al.* [5] indicated the importance of saving energy from a network protocol point of view. They modified the current Internet protocols to enable some network devices to operate in sleep mode, thus saving energy. In their follow-up studies [28], [29], they proposed various approaches to detect the idle periods of Ethernet interfaces in networks and developed algorithms to decide the interface state transformation between idle and active states. Gunaratne *et al.* [3] also noted the low utilization of network resources and suggested the reduction of the data transmission rate of Ethernet links when possible, thus saving energy over entire networks. Both of these ideas have been adopted by the Ethernet industry [3], [30], [31], which results in two basic energy saving mechanisms: speed scaling and power down. Nedevschi *et al.* [32] explored both speed scaling and power down as techniques for saving energy at the network level and conducted intensive simulations to evaluate the performance of these two techniques to prove their effectiveness. Andrews *et al.* [33], [12] addressed the energy efficiency problem by proposing routing and scheduling algorithms based on these two mechanisms. ESIR, proposed by Cianfrani *et al.* [34], reduces the active links in IP networks by allowing neighboring nodes to share shortest path trees, thus reducing the energy consumption of the network. Fisher *et al.* [35] exploited the fact that many links in core networks are actually "bundles" of multiple physical cables and line cards that can be shut down independently to put as many links as possible to sleep, thus saving energy. [36] proposed by Chiaraviglio *et al.* focuses on shutting down entire routers. There is a set of other works [37]–[42] with various considerations. Different with the above works that only consider power conservation, some works [43], [44] take both energy conservation and network performance into account. Sansò *et al.* [43] were the first to raise awareness on the relationship between Internet power consumption, network performance and reliability planning by exploring the trade-off between energy consumption and network performance based on data collected from Internet sources. Zhao *el al.* in [44] studied the tradeoff between load balance and energy efficiency.

B. Distributed Routing Schemes

To avoid the limitations of the centralized schemes, researchers have proposed distributed schemes. GDRP-PS, proposed by Ho *et al.* [45], is a new architecture that can coordinate various possible routers to enter sleep mode, thus saving energy over the entire network. Vasic *et al.* [46] addressed the energy efficiency problem from a traffic engineering perspective. They proposed EATe, a traffic engineering technique in which flows are assumed to be splittable, and energy can be saved by properly splitting the flows to the links. In their follow-up study [47], they proposed another traffic engineering technique: REsPoNse. In the [47], they first computed a series of paths that are sensitive to the energy consumption according to historic traffic information; then, they used a smart online traffic engineering algorithm for the flow path selection. Shen *et al.* [48] also used a traffic engineering technique. An important difference from the above mentioned distributed schemes is that they take the network performance (maximum link utilization) into consideration in addition to energy consumption. Other researchers have focused on network protocols and routing algorithms. Kim *et al.* [49] formulated the energy consumption minimized network problem as an integer linear programming and exploited the ant colony optimization (ACO) method to solve the problem. Bianzino *et al.* [50] proposed a reinforcement learning technology for nodes to make local energy saving decisions according to the historic and current loads of links. In their later study [51], they used heuristics to turn off links in a distributed manner instead of adopting a learning technique that needs complicated parameter settings. Si *et al.* [52] proposed an energy saving scheme for data centers using power-down mechanism. Coiro *et al.* [53] focused on leveraging the characteristics of MPLS networks to reduce the energy consumption of entire networks using power-down technique. They proposed DAISIES to adjust the capacity reserved by each Label Switch Path (LSP) according to variations in traffic. These works addressed the problems faced by centralized solutions and performed well in terms of energy conservation. However, there is not a distributed scheme that takes into consideration both energy conservation and delay while using speed-scaling energy saving mechanism. In this article, HDEER is developed based on our previous work [54], which uses speed scaling mechanism to reduce network energy

consumption from a routing perspective. However, [54] has limitations of long convergence time and routing instability. HDEER in this paper saves energy while considering the message delay at the same time by using speed-scaling mechanism, and is of shorter convergence time and routing stability.

VII. Conclusions

In this paper, we address the problem of achieving energy efficiency in wired networks from a routing perspective. Unlike most existing green networking solutions, we aim at reducing energy consumption over entire networks with traffic delay considered. We model this problem and provide a bi-objective optimization formulation for it. Through theoretical analysis, we identify the necessary and sufficient conditions for achieving a global optimal solution. Based on the observed conditions, we propose a fully distributed routing scheme that can provide near global optimal solutions. The proposed scheme consists of two stages. First, we generate DAGs for each destination node to ensure loop-freeness of the routing paths. Then, we develop algorithms to allocate traffic in the DAGs under static and dynamic scenarios to save energy. Extensive simulations based on both real network topologies and a generated topology with power law show that the proposed routing scheme can obtain significant energy saving while bringing negligible traffic delay overheads compared to shortest path routing.

References

[1] M. A. Marsan, L. Chiaraviglio, D. Ciullo, and M. Meo, "Optimal energy savings in cellular access networks," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2009, pp. 1–5.

[2] A. P. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A survey of green networking research," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 3–20, Feb. 2012.

[3] C. Gunaratne, K. J. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of Ethernet with adaptive link rate (ALR)," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 448–461, Apr. 2008.

[4] P. Patel-Predd, "Update: Energy-efficient Ethernet," *IEEE Spectr.*, vol. 45, no. 5, pp. 13–13, May 2008.

[5] M. Gupta and S. Singh, "Greening of the Internet," in *Proc. SIGCOMM*, 2003, pp. 19–26.

[6] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 1–39, Mar. 2007.

[7] H. Chan, W. Chan, T. W. Lam, L. Lee, K. Mak, and P. W. H. Wong, "Energy efficient online deadline scheduling," in *Proc. 18th Annu. ACM-SIAM Symp. Discr. Algorithms*, New Orleans, LA, USA, Jan. 7–9, 2007, pp. 795–804.

[8] M. Garrett, "Powering down," *Commun. ACM*, vol. 51, no. 9, pp. 42–46, 2008.

[9] F. F. Yao, A. J. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. 36th Annu. Symp. Found. Comput. Sci.*, Milwaukee, WI, USA, Oct. 23–25, 1995, pp. 374–382.

[10] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *Proc. SIGCOMM*, 2002, pp. 161–174.

[11] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 73–85, Jan. 1977.

[12] M. Andrews, A. Fernández Anta, L. Zhang, and W. Zhao, "Routing for power minimization in the speed scaling model," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 285–294, Feb. 2012.

[13] Y. Wang, G. Lu, and X. Li, "A study of Internet packet reordering," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2004, pp. 350–359.

[14] J. A. Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso, "A measurement-based analysis of the energy consumption of data center servers," in *Proc. 5th Int. Conf. Future Energy Syst. e-Energy*, 2014, pp. 63–74.

[15] T. Bingmann and D. Yordanov, "Robert Gallager's minimum delay routing algorithm using distributed computation," in *Proc. Seminar (WS'07)*, 2007, pp. 15–31.

[16] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidiscip. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[17] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Trans. Autom. Control*, vol. AC-8, no. 1, pp. 59–60, Jan. 1963.

[18] D. Lee and N. Brownlee, "Passive measurement of one-way and two-way flow lifetimes," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 17–28, 2007.

[19] J. Charzinski, "HTTP/TCP connection and flow characteristics," *Perform. Eval.*, vol. 42, nos. 2–3, pp. 149–162, 2000.

[20] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.

[21] N. T. Spring, R. Mahajan, D. Wetherall, and T. E. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[22] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: Initial recommendations," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 19–32, 2005.

[23] S. Antonakopoulos, S. Fortune, and L. Zhang, "Power-aware routing with rate-adaptive network elements," in *Proc. GLOBECOM Workshops*, 2010, pp. 1428–1432.

[24] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987, vol. 2.

[25] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 223–244, May 2011.

[26] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in data center networks," *Comput. Commun.*, vol. 40, pp. 1–21, 2014.

[27] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, no. 5, pp. 86–96, 2010.

[28] M. Gupta and S. Singh, "Dynamic Ethernet link shutdown for energy conservation on Ethernet links," in *Proc. Int. Conf. Commun. (ICC)*, 2007, pp. 6156–6161.

[29] M. Gupta and S. Singh, "Using low-power modes for energy conservation in Ethernet lans," in *Proc. INFOCOM*, 2007, pp. 2451–2455.

[30] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: The road to energy efficient Ethernet," *IEEE Commun. Mag.*, vol. 48, no. 11, pp. 50–56, Nov. 2010.

[31] M. A. Marsan, A. Fernández Anta, V. Mancuso, B. Rengarajan, P. R. Vasallo, and G. Rizzo, "A simple analytical model for energy efficient Ethernet," *IEEE Commun. Lett.*, vol. 15, no. 7, pp. 773–775, Jul. 2011.

[32] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proc. Symp. Netw. Syst. Design Implement. (NSDI)*, 2008, pp. 323–336.

[33] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing and scheduling for energy and delay minimization in the powerdown model," *Networks*, vol. 61, no. 3, pp. 226–237, 2013.

[34] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, and A. V. Vasilakos, "An OSPF-integrated routing strategy for QoS-aware energy saving in IP backbone networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 9, no. 3, pp. 254–267, Sep. 2012.

[35] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: Reducing energy consumption by shutting off cables in bundled links," in *Proc. Green Netw.*, 2010, pp. 29–34.

[36] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *Proc. Int. Conf. Commun. (ICC)*, 2009, pp. 1–6.

[37] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering," in *Proc. Int. Conf. Netw. Protocols (ICNP)*, 2010, pp. 21–30.

[38] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *Proc. Green Netw.*, 2010, pp. 1–8.

[39] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu, "Energy-efficient flow scheduling and routing with hard deadlines in data center networks," in *Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Madrid, Spain, Jun. 30/Jul. 3, 2014, pp. 248–257.

[40] L. Wang *et al.*, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.

[41] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 107–112, 2013.
[42] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 33:1–33:36, 2014.
[43] B. Sansò and H. Mellah, "On reliability, performance and Internet power consumption," in *Proc. 7th Int. Workshop Design Rel. Commun. Netw. (DRCN)*, Oct. 2009, pp. 259–264.
[44] Y. Zhao, S. Wang, S. Xu, X. Wang, X. Gao, and C. Qiao, "Load balance vs energy efficiency in traffic engineering: A game theoretical perspective," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 14–19, 2013, pp. 530–534.
[45] K.-H. Ho and C.-C. Cheung, "Green distributed routing protocol for sleep coordination in wired core networks," in *Proc. 6th Int. Conf. Netw. Comput. (INC)*, May 2010, pp. 1–6.
[46] N. Vasic and D. Kostic, "Energy-aware traffic engineering," in *Proc. e-Energy*, 2010, pp. 169–178.
[47] N. Vasic, P. Bhurat, D. M. Novakovic, M. Canini, S. Shekhar, and D. Kostic, "Identifying and using energy-critical paths," in *Proc. 7th Conf. Emerging Netw. Exp. Technol. (CoNEXT)*, 2011, p. 18.
[48] M. Shen, H. Liu, K. Xu, N. Wang, and Y. Zhong, "Routing on demand: Toward the energy-aware traffic engineering with OSPF," in *Proc. Netw.*, 2012, pp. 232–246.
[49] Y. Kim, E. Lee, H. Park, J. Choi, and H. Park, "Ant colony based self-adaptive energy saving routing for energy efficient Internet," *Comput. Netw.*, vol. 56, no. 10, pp. 2343–2354, 2012.
[50] A. P. Bianzino, L. Chiaraviglio, M. Mellia, and J.-L. Rougier, "GRiDA: Green distributed algorithm for energy-efficient IP backbone networks," *Comput. Netw.*, vol. 56, no. 14, pp. 3219–3232, 2012.
[51] A. P. Bianzino, L. Chiaraviglio, and M. Mellia, "Distributed algorithms for green IP networks," in *Proc. INFOCOM Workshops*, 2012, pp. 121–126.
[52] W. Si, J. Taheri, and A. Y. Zomaya, "A distributed energy saving approach for Ethernet switches in data centers," in *Proc. 37th Conf. Local Comput. Netw. (LCN)*, 2012, pp. 505–512.
[53] A. Coiro, F. Iervini, and M. Listanti, "Distributed and adaptive interface switch off for Internet energy saving," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2011, pp. 1–8.
[54] C. Hou, F. Zhang, A. F. Anta, L. Wang, and Z. Liu, "A hop-by-hop energy efficient distributed routing scheme," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 101–106, 2013.

Biyu Zhou received the B.S. degree from Xidian University, Xi'an, China, in 2012. She is currently pursuing the Ph.D. degree at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. Her research interests include energy-efficient computing, distributed routing, data center networking, and network optimization.

Fa Zhang received the B.S. degree from Hebei University of Science and Technology, Shijiazhuang, China, in 1997, the M.S. degree from Yanshan University, Qinhuangdao, China, in 2000, and the Ph.D. degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences, Beijing, China, in 2005, all in computer science. From 2002 to 2005, he worked as a Visiting Scholar at Ohio State University, Columbus, OH, USA. Then, he worked as an Assistant Professor with the National Research Center for Intelligent Computing Systems (NCIC), ICT from 2005 to 2007. From 2008 to present, he has been an Associate Professor with the Advanced Computing Research Laboratory, ICT. Also, he worked as a Visiting Scientist at the Universidad Rey Juan Carlos, Madrid, Spain, from 2009 to 2010. His research interests include computer algorithms, high performance computing, biomedical image processing, and bioinformatics.

Lin Wang received the B.S. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2010, and the M.S. and Ph.D. degrees in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2012 and 2015, respectively. He is a Research Associate with SnT, University of Luxembourg, Luxembourg. From September 2012 to January 2014, he was with IMDEA Networks Institute, Madrid, Spain, as a Research Intern. His research interests include networked systems, distributed cloud computing, big data analytics, and energy-efficient computing.

Chenying Hou received B.S. degree from Central South University, Changsha, China, in 2010, and the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2013. Her research interests include energy-efficient routing and network optimization.

Antonio Fernández Anta (SM'02) received the M.Sc. and Ph.D. degrees in computer science from the University of Louisiana, Lafayette, LA, USA, in 1992 and 1994, respectively. He is a Research Professor with IMDEA Networks Institute, Madrid, Spain. Previously, he was a Faculty Member with the Universidad Rey Juan Carlos, Madrid, Spain, and the Universidad Politécnica de Madrid, Madrid, Spain, where he received an award for his research productivity. He has been a senior member of the Association for Computing Machinery (ACM) since 2007.

Athanasios V. Vasilakos (M'00–SM'11) is a Professor with Luleå University of Technology, Luleå, Sweden. He served or is serving as an Editor for many journals, such as the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON NANOBIOSCIENCE, IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE, *ACM Transactions on Autonomous and Adaptive Systems*, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He is also the General Chair of the European Alliances for Innovation.

Youshi Wang received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2013. He is currently pursuing the Ph.D. degree at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include energy-efficient computing, data center networking, and discrete optimization.

Jie Wu (M'90–SM'93–F'09) is the Chair and a Laura H. Carnell Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. Before joining Temple University, he was a Program Director with the U.S. National Science Foundation and a Distinguished Professor with Florida Atlantic University, Boca Raton, FL, USA. His research interests include wireless networks, mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He is currently an ACM Distinguished Speaker. He has served on several editorial boards, including the IEEE TRANSACTIONS ON COMPUTERS and *Journal of Parallel and Distributed Computing*. He was the General Co-Chair for IEEE MASS 2006, IEEE IPDPS2008, and IEEE DCOSS 2009 and was the Program Co-Chair for IEEE INFOCOMM 2011. He served as the General Chair for IEEE ICDCS 2013. He was an IEEE Computer Society Distinguished Visitor and the Chair for the IEEE Technical Committee on Distributed Processing. He was the recipient of the China Computer Federation Overseas Outstanding Achievement Award in 2011.

Zhiyong Liu received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1987. He worked as a Professor with the Institute of Computing Technology starting in 1987. He was the Executive Director General of the Directorate for Information Sciences, National Natural Science Foundation of China, from 1995 to 2006. Currently, he is a Chair Professor with the Advanced Research Laboratory, Institute of Computing Technology, Chinese Academy of Sciences. He has worked on networks, high performance architectures and algorithms, parallel and distributed processing, and bioinformatics. His research interests include computer architectures, algorithms, networks, and parallel and distributed processing. He has served as a Board Member, a Referee, an Editor, and an Invited Editor for academic journals, a Program/Organization Committee Member/Chairman, an Advisory Committee Member/Chairman for national and international conferences, a Steering Expert Committee Member for research initiatives, and an Investigator/Principal Investigator of research projects on computer control systems, computer architectures, and algorithms. He was the recipient of the China National Science Congress Prize, a National Prize for Science and Technology in China.