Hop-by-hop Computing for Green Internet Routing
(Technical report, under revision)

Yuan Yang
Tsinghua Univ.
yyang@csnet1.cs.tsinghua.edu.cn

Dan Wang
Hong Kong Polytechnic Univ.
csdwang@comp.polyu.edu.hk

Mingwei Xu
Tsinghua Univ.
xmw@cernet.edu.cn

Suogang Li
CERNET National Network Center
lisg@cernet.edu.cn

***Abstract*—Energy conservation is a global concern nowadays and it is widely expected that energy cost will continuously increase in the near future. The design of the Internet, unfortunately, has not taken energy issues into consideration at the very beginning. There are emerging studies on reducing Internet energy footprint. These studies aim at turning network components such as line cards or routers into sleeping modes. We make an observation that different traffic volumes on links result in different energy consumption, even without turning-off network devices. As such, we for the first time design a green Internet traffic routing scheme; and it is orthogonal to those turning-off network components schemes.**

There can be many design choices for our green routing schemes. As an example, an optimization problem can be formulated where we centralized compute routing paths to minimize the overall energy consumption. Such approach, however, requires to deploy additional protocols for the Internet. In this paper, we propose a hop-by-hop routing scheme. Such scheme can be easily incorporated into current OSPF protocol. We solve many challenges. First, we present a power model to quantify link rate and power consumption. We validate our model using real world experiments. Second, unlike centralized computing, a key challenge for hop-by-hop computing is to avoid routing loops. We thus design green routing algorithms that are isotonic and guarantee that no loop will be formulated. Third, hop-by-hop routing without loops does not naturally lead to minimized energy consumption. Therefore, we develop a set of enhanced algorithms that substantially improve the performance. Fourth, a "green" path may degrade QoS performance such as end-to-end delay. We study this problem and develop an algorithm that co-considers energy conservation and path stretch.

We comprehensively evaluate our algorithms through simulations on synthetic and real topologies and traffic traces. We show that with green routing, we can save the power that is consumed by line cards for as much as 50%.

I. Introduction

Energy conservation is a global concern nowadays and energy cost is expected to increase for the forthcoming future. As a consequence, energy has become an important issue in the designs of such area as data centers [1][2], building management [3][4], to name but a few.

There are emerging studies for saving energy for the Internet [5][6][7][8][9][10]. In general, these studies turn the network components such as line cards or routers into sleeping modes. The network components to be turned off are carefully chosen and trade-offs are investigated to balance network performance and energy conservation. Internet routing is then conducted in the residual topology or realized by MPLS assistance, etc.

In this paper, we study "green" routing in the sense that we deliver traffic by selecting paths that can consume less power. This fundamentally differs from previous schemes, though our approach is orthogonal and can be jointly applied with them.

At the very beginning of the Internet routing designs, shortest path routing was adopted in the hope to save such critical resources as router computing capacity, bandwidth, etc. Energy conservation was not listed in the design space. A commonly accepted vision nowadays, however, is that computing and information are becoming cheaper and energy is becoming more expensive and we may trade computing capacity for energy saving opportunities. In-line with such vision, we may want to search a path that is "greener", even though longer.

A key observation that makes this possible is that even if we do not turn off a network component, the energy consumption for packet delivery can be different under different traffic volume [11]. Intrinsically, this is due to such technologies as trunking (or bundled links) [14] and adaptive link rates (ALR) [12]. Trunking, standardized in IEEE 802.1AX, refers to the fact that a logical link in the Internet often reflects multiple physical links, e.g., a 40Gbps link may consist of four 10Gbps links; and when traffic volume is less, less physical links can be used and less energy is consumed. ALR is an ethernet technology where link rate and power dynamically scales with traffic volume. Such observation leads us to think to incorporate energy conservation into Internet routing designs. We show an example as follows.

Example Consider a network in Fig. 1 where two links (a, b) and (b, c) are both consisted of four parallel OC48 (2.5Gbps) physical links and the other tree links are single OC192 (10Gbps) physical links. More specifically, for an OC48 link, there is a baseline 125.1Watt energy consumption and an additional 0.006Watt for each 1Mbps traffic; and for an OC192 link, there is a baseline 134.2Watt energy consumption and an additional 0.004 Watt for each 1Mbps traffic. In this topology, shortest path routing will result in three paths on each link. For example, (a, b) will support paths $a \leftrightarrow b$, $a \leftrightarrow c$ and $b \leftrightarrow e$. Assume that the traffic volume is 1Gbps on each path. Link (a, b) and (b, c) then have to power on two parallel OC48 physical links because the total traffic on these links is 3Gbps. The total energy consumption is $(125.1 + 1500 \times 0.006) \times 4 + (134.2 + 3000 \times 0.004) \times 3 = 975.0$Watt. If, however, we use a routing where every path is the same as shortest path except that path $a \leftrightarrow c = (a, e, d, c)$. Then link

(a, b) and (b, c) will only carry 2Gbps and power on only one OC48 physical link for each. The total energy consumption is $(125.1 + 2000 \times 0.006) \times 2 + (134.2 + 4000 \times 0.004) \times 3 = 724.8$Watt, a 25.7% improvement.

One way to generalize the above example and maximize energy conservation is to formulate the problem into an optimization problem; analyze the problem complexity and design a centralized routing algorithm. Such approach requires developing a separate protocol to establish the routing paths. In this paper, we instead choose a hop-by-hop approach. More specifically, each router can separately compute next hops, the same as what they do in Dijkstra today. We can then easily incorporate the routing algorithm into OSPF protocol.

Under this hop-by-hop design decision, we face three main challenges: 1) we need to clarify an appropriate power model; 2) to be practical, the computation complexity should be comparable to shortest path routing (i.e., Dijkstra) and loop-free; 3) hop-by-hop computing should maximize energy conservation; and 4) important QoS performance of the network such as path stretch may be co-considered, and can be naturally adjusted.

In this paper, we present a comprehensive study. We first develop a power model and we validate our power model using real experiments. We then develop principles and a baseline hop-by-hop green routing algorithm that guarantees loop-free. The algorithm follows the widely known algebra with isotonic property. We further develop an advanced algorithm that substantially improves the baseline algorithm in energy conservation. We also develop an algorithm that co-considers energy conservation and path stretch, and do a in-depth study on the problem of maximizing energy conservation with QoS requirements. We evaluate our algorithms using comprehensive simulations on synthetic and real topologies and traffic traces. The results show that our algorithms may save more than 50% energy on line cards.

The rest of this paper is organized as follows. Section II presents related work. The power model is presented in Section III. The design outline and some properties of routing algebra are discussed in Section IV. After that, the design of hop-by-hop green routing is presented in Section V. Section VI presents the evaluation while Section VII concludes the paper.

II. Related Work

Together with the world-wide objective to build a greener globe, more and more computing systems put energy conservation into their design principles [1][2].

There are efforts to develop a greener Internet as well [15][16]. Some studies save energy of specific network devices [17][18]. GreenTE [5] is proposed to use MPLS tunnels to aggregate traffic so as to turn the under-utilized network components into sleeping modes to save energy. REsPoNse [6] is proposed to offline identify energy-critical paths and on-demand paths. The packets are online delivered also with the objective to effectively aggregate traffic to turn more network components into sleeping modes. GreenTE and REsPoNse are both centralized schemes. GreenOSPF [7] is proposed to aggregate traffic in a distributed fashion and turn the network components into sleeping modes. However, to achieve a good performance, a centralized algorithm [8] is still needed to assign sleeping links. ESACON [9] is proposed to collaboratively select sleeping links with special connectivity properties. Routing paths are then computed after removing these links. A fully distributed approach is proposed [10] which collects global traffic information and aggregates traffic to turn appropriate network components into sleeping mode.

Our approach differs from all the aforementioned schemes as follows. First, all previous proposals set network devices or links into sleeping modes. Our design is based on the observation that different traffic volume also has different energy consumption. Internet routing algorithm may take this into consideration. To the best of our knowledge, we are the first to propose such scheme. Second, though some previous schemes compute the network components to be shut down in a distributed fashion, large changes to current routing protocols are still needed. Our routing computation is hop-by-hop and Dijkstra-oriented. We believe this is easier to be incorporated into current routing architecture.

We may consider green as one type of service that the Internet should be provisioned. There are many studies on Internet Quality of Service [19]. There were two different approaches in Internet QoS support beyond shortest path routing. One is centralized computation [20]. The benefit is that since different types of services usually introduce conflictions, a centralized scheme can compute optimal or near optimal solutions. The disadvantage is that centralized computation requires additional protocols, which is a non-trivial overhead. The other is to maintain hop-by-hop computation by managing different types of services into a singular link weight [21]. A seminal paper [21] shows that to make hop-by-hop computation loop-free requires the link weights to have certain isotonicity properties and a routing algebra model is developed.

In this paper, we also leverage the algebra model to develop hop-by-hop computing for green Internet routing that is loop-free. We have a set of algorithms where we gradually improve the energy conservation performance.

III. Power Model

We model the relationship between the power consumption and the traffic volume. We first present the router operation backgrounds and our modeling details. We then use simulations and experiments to validate our modeling.

A. Router Operation Backgrounds and Power Modeling

A link between two routers is physically connected by two line cards; and the line cards consume the majority power of the routers [11]. We thus use *link power consumption* to abstract the power consumption of the line cards.

We can divide the power consumption into two different categories: 1) power consumed by line card CPU processor; this is super-linear to the traffic volume and 2) power consumed by such operations as buffer I/O, packet lookup, etc; this is usually linear to the traffic volume.

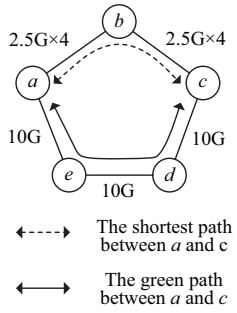
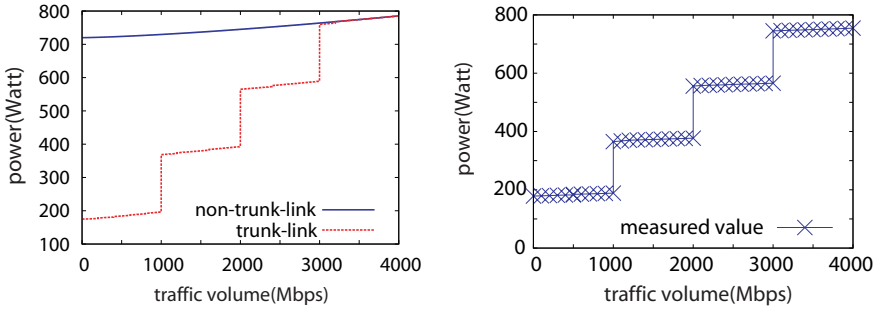


Fig. 1: An example of green routing with adaptive link rates.



(a) (b)

Fig. 2: The link power model. (a) The traffic-power curves defined by Eq. (1) and Eq. (4). (b) The measured traffic-power curve, where two links are sharing the traffic when the traffic is larger than 400Mbps.

There are many components in a line card. With advanced technologies, many components can individually change to low-power states or be turned off after the traffic volume is reduced under different levels of thresholds. For instance, an Intel processor has active state C0, auto halt state C1, stop clock state C2, deep sleep state C3 and deeper sleep state C4, all with different power consumption. Similarly a PCIe bus which connects the chips has the states of D0, D1, D2, D3hot and D3cold. Since turning on-off of these components are discrete, we can generally see a discrete stair-like behavior in power consumption (see Fig. 2(a)).

With these backgrounds, we can classify two types of links: 1) *trunk-links* (i.e., adaptive link rate) where advanced technologies are adopted and components can be individually turned off, resulting a stair-like behavior in power consumption and 2) *non-trunk-links*. Today, non-trunk-links are still the majority, yet trunk and ALR technologies are under fast development.

We first model the non-trunk-links and then we model the trunk-links by revising the non-trunk-link model to include the stair-like behavior. Our objective is to model the relationship between link power consumption and traffic volume.

Note that a logical link in the Internet topology may consist of several bundled physical links. Let l be a link, and n_l be the number of physical links. Let x_l be the traffic volume on this link; then the traffic on each physical link is $\frac{x_l}{n_l}$. Let δ_l be the idle power of a line card. The power consumption for the first category (i.e., CPU, super-linear) on each physical link can be modeled as $\mu_l \left(\frac{x_l}{n_l}\right)^{\alpha_l}$, where μ_l, α_l are constants ($\alpha > 1$) [22]. The power consumption for the second category (buffer I/O, packet lookup, etc, linear) on each physical link is $\rho_l \frac{x_l}{n_l}$, where ρ_l is a constant. Finally, let P_l^{no} be the total power consumption of a non-ALR-link, we have

$$P_l^{no}(x_l) = 2n_l \times \left(\delta_l + \rho_l \frac{x_l}{n_l} + \mu_l \left(\frac{x_l}{n_l}\right)^{\alpha_l}\right). \quad (1)$$

Here $2n_l$ denote the fact that the power is consumed for the line cards on both ends of the link.

For a trunk-link, the difference is the discrete stair-like behavior. We model two intrinsic reasons for the discrete stair-like behavior: 1) physical links can be powered off under different traffic volume; and 2) different components in line cards can be turned-off under different traffic volume.

Let $n_c \in \{0, 1, \ldots, n_l - 1\}$ be the number of physical links being powered off and $r_0, r_1, \ldots, r_{n_l-1} (r_0 < r_1 < \cdots < r_{n_l-1})$ are traffic volume thresholds to power off a physical link, we have:

$$n_c = \begin{cases} n_l - 1, & \text{if} \quad r_0 \le x_l < r_1 \\ n_l - 2, & \text{if} \quad r_1 \le x_l < r_2 \\ \ldots, & \ldots \\ 1, & \text{if} \quad r_{n_l-2} \le x_l < r_{n_l-1} \\ 0, & \text{if} \quad r_{n_l-1} \le x_l \end{cases} \quad (2)$$

Similarly, let the number of line card states be n_s, and $\delta_c = \delta_i$ for $i \in \{0, 1, \ldots, n_s - 1\}$ be the power reduced by switching a line card into the ith state ($0 = \delta_0 < \delta_1 < \cdots < \delta_{n_s-1}$), we have

$$\delta_c = \begin{cases} \delta_{n_s-1}, & \text{if} \quad r'_0 \le \frac{x_l}{n_l - n_c} < r'_1 \\ \delta_{n_s-2}, & \text{if} \quad r'_1 \le \frac{x_l}{n_l - n_c} < r'_2 \\ \ldots, & \ldots \\ \delta_1, & \text{if} \quad r'_{n_s-2} \le \frac{x_l}{n_l - n_c} < r'_{n_s-1} \\ \delta_0, & \text{if} \quad r'_{n_s-1} \le \frac{x_l}{n_l - n_c} \end{cases} \quad (3)$$

where $r'_0, r'_1, \ldots, r'_{n_s-1}$ are traffic volume thresholds.

Finally, let P_l^a be the total power consumption of a trunk-link, we have

$$P_l^a(x_l) = 2(n_l - n_c)\left(\delta_l - \delta_c + \frac{\rho_l x_l}{n_l - n_c} + \mu_l \left(\frac{x_l}{n_l - n_c}\right)^{\alpha_l}\right) \quad (4)$$

Eq. (4) is equivalent to Eq. (1) if n_c and δ_c equal 0.

B. Simulation and Experimental Validation

Eq. (1) and Eq. (4) are abstract. For illustration purpose, we plot them in Fig. 2(a). In Fig. 2(a) we set the link to consist of four 1Gbps physical links (i.e., $n_l = 4$). The idle power δ_l for each physical link is set to 180 Watt. We set $\rho_l = 0.0005$, $\mu_l = 0.001$ and $\alpha_l = 1.4$; these are based on suggested values in [11] and [22]. We set r_0, r_1, r_2, r_3, r_4 to $0, 1000, 2000, 3000, 4000$ Mbps. We assume that there are 5 states for the line card components, which can reduce power by $5, 3.5, 2, 1, 0$ Watt respectively. We set $r'_0, r'_1, r'_2, r'_3, r'_4$ to

$0, 200, 400, 600, 800, 1000$ Mbps. This can be seen as the fact that smaller traffic volume leads to more energy conservation for a trunk-link.

Not surprisingly, we see that for the non-trunk-link, the power consumption is dynamically correlated to the traffic volume; and for the trunk-link, the power consumption shows a discrete stair-link behavior. We validate this power model with experiments using a real commercial router.

We set up the experiment by generating packets of 64 bytes using a PC and sending the packets to a commercial BitEngine12000 router [23], through four 1Gb ethernet links. The traffic volume varies from 1Mbps to 4000Mbps. The router has four 4GE line cards and powers on proper number of line cards to forward the traffic. We measure the power of the 4GE line cards and the results are shown in Fig. 2(b). We see that the curve matches our model closely.

In this paper, we will focus on the power consumed for traffics. Therefore, we subtract the idle power $P_l^a(0)$ or $P_l^{no}(0)$. The final power model $P_l(x_l)$ is

$$P_l(x_l) = \begin{cases} P_l^a(x_l) - P_l^a(0) & l \text{ is a trunk-link} \\ P_l^{no}(x_l) - P_l^{no}(0) & l \text{ is a non-trunk-link} \end{cases} \quad (5)$$

IV. Overview and Preliminaries

The objective of our green Internet routing is to minimize the total energy consumption in the network. We choose a hop-by-hop approach because it can be easily integrated to the current Internet routing architecture.

Formally, a network is modeled as $G(V, E)$, where V denotes the set of vertexes (nodes) and E denotes the set of edges (links). A path from node s to d is a sequence of nodes $(v_0 = s, v_1, v_2, \ldots, v_n = d)$, where $(v_i, v_{i+1}) \in E$ for $0 \leq i < n$. Following Section III, let x_l be the traffic volume and $P_l(x_l)$ follow the power model in Eq. (5). The objective is thus $\min \sum_{l \in E} P_l(x_l)$, with the constraint that the source-destination paths are *loop-free* and can be *polynomially computed* at each node.

For a hop-by-hop approach, simply computing the "greenest" path (i.e., with smallest energy consumption) for each source and destination pair may not minimize total energy consumption. The traffics of different paths collectively increase a link utilization ratio, and lead to a larger energy consumption. This is a standard local vs. global optimal problem. One possible solution is to let each router compute routing based on global traffic matrices which reflect the volume of traffic that flows between all possible source and destination pairs. However, it is not easy to obtain a traffic matrix, because 1) direct measurements to populate a traffic matrix is typically prohibitively expensive [24], and 2) the procedure to estimate a traffic matrix from partial data is of high complexity, as the associated optimization problem is non-convex [24].

Thus, for a hop-by-hop scheme whose complexity can be comparable to Dijkstra, we design a path weight similar to the path weight used by Dijkstra, where the weight should reflect total energy conservation based on partial traffic data.

The path weights must be carefully designed to make sure hop-by-hop computing is loop-free. We show an example where loop-freeness fail. We see a topology in Fig. 3. Assume that $(a, b), (a, c)$ are non-trunk-links and (b, c) is a trunk-link. The power consumption of the links are $P_{(a,b)}(x_l) = 0.1x_l$, $P_{(a,c)}(x_l) = 0.3x_l$, and $P_{(b,c)}(x_l) = 0.1x_l$ if $x_l < 10$, or else $P_{(b,c)}(x_l) = 0.1x_l + 5$.

Given the traffic demand of a source node, assume a hop-by-hop scheme straightforwardly chooses to compute path weight as the sum of the power consumption of all the links in the path (i.e., "greenest" path). Suppose node a has a traffic demand of 5 to send to node c. The path weight of (a, b, c) is $0.1 \times 5 + 0.1 \times 5 = 1$ and the path weight of (a, c) is $0.3 \times 5 = 1.5$. Thus node a will choose path (a, b, c) to deliver packets. Meanwhile, suppose node b has a traffic demand of 10 to send to node c. The path weight of (b, a, c) is $0.1 \times 10 + 0.3 \times 10 = 4$ and the path weight of (b, c) is $0.1 \times 10 + 5 = 6$. Thus node b will choose path (b, a, c) to deliver packets. As a result, a loop is introduced between node a and b, and packets destined for c will never reach c.

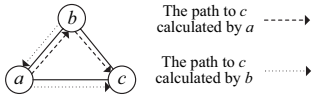


Fig. 3: An example of routing loops.

Intrinsically, to achieve loop-freeness, there are certain properties that the path weights should follow. A seminal work [21] first explained the properties through a routing algebra model. Here we briefly present the background.

A routing algebra (also called a path weight structure) is defined as quadruplet $(S, \oplus, \preceq, w)$. S denotes the set of path weights. $\oplus$ is a binary operation upon the path weights. $\preceq$ is an order relation to compare two path weights. w is a function that maps a path to a weight. Let $p \circ q$ denotes the concatenation of path p and q. Then $w(p \circ q) = w(p) \oplus w(q)$. In particular, the weight of a path $p = (v_0, v_1, \ldots, v_n)$ is $w(p) = w(v_0, v_1) \oplus w(v_1, v_2) \oplus \cdots \oplus w(v_{n-1}, v_n)$. We call the weight of a path which has only one hop *a link weight*. The path with the *lightest* weight is preferred. Formally, path p is *the lightest path* if $w(p) \preceq w(q)$ for any q.

For example, in shortest path routing, the path weight is the sum of the link lengths. Thus, $S = R^+$ and $\oplus$ is $+$. The shortest path is preferred and thus $\preceq$ is $\leq$. In widest routing, where one needs to find a path with the largest bandwidth, the path weight equals the bandwidth of the bottleneck link. Thus $S = R^+$, $w(p) \oplus w(q)$ means $\min(w(p), w(q))$, and $\preceq$ is $\geq$.

There are two steps to avoid loops in hop-by-hop computing [25]: 1) certain properties need to be satisfied (intrinsically, path concatenation should follow certain properties) and 2) routing algorithm is designed accordingly. We introduce a few definitions from [25].

Definition 4.1: $(S, \oplus, \preceq, w)$ is left-isotonic if $w(p_1) \preceq w(p_2)$ implies $w(q \circ p_1) \preceq w(q \circ p_2)$, for all paths p_1, p_2, q. Similarly, $(S, \oplus, \preceq, w)$ is strictly left-isotonic if $w(p_1) \prec w(p_2)$ implies $w(q \circ p_1) \prec w(q \circ p_2)$, for all paths p_1, p_2, q.

Definition 4.2: $(S, \oplus, \preceq, w)$ is right-isotonic if $w(p_1) \preceq w(p_2)$ implies $w(p_1 \circ q) \preceq w(p_2 \circ q)$, for all p_1, p_2, q. Similarly,

$(S, \oplus, \preceq, w)$ is strictly right-isotonic if $w(p_1) \prec w(p_2)$ implies $w(p_1 \circ q) \prec w(p_2 \circ q)$, for all paths p_1, p_2, q.

We have the following theorems [25][26].

Theorem 4.1: Dijkstra's algorithm is guaranteed to find the lightest paths if and only if the path weight structure $(S, \oplus, \preceq, w)$ is right-isotonic [25].

Dijkstra's algorithm mentioned in Theorem 4.1 uses source node s as the root node for computation. If we use the destination node d as the root node for computation, we have:

Theorem 4.2: Dijkstra's algorithm that uses d as the root node is guaranteed to find the lightest paths if and only if the path weight structure $(S, \oplus, \preceq, w)$ is strictly left-isotonic [26].

V. Hop-by-hop Green Routing Algorithms

We now study hop-by-hop green routing (Green-HR). We first study a baseline Green-HR algorithm Dijkstra-Green-B that is loop-free. We then study some intrinsic relationships between link weights and power consumption and we finally develop an advanced algorithm Dijkstra-Green-Adv.

A. Dijkstra-Green-B Algorithm

From Section IV, we see that the key is to develop an appropriate weight for a path so that it incorporates "green" and holds isotonicity. A Dijkstra-oriented algorithm can then be developed to achieve loop-free hop-by-hop routing.

In this algorithm, we emphasize on isotonicity. A preliminary observation is that though we cannot choose the "greenest" paths; for global energy conservation, we also should not choose a path that is too long, which collectively consumes more energy.

We thus set the weight as follows. For each destination node d, we assign x_0^v as a starting weight. This x_0^v is determined by the total bandwidth associated with d and we will specify this later. For a path $p = (s = v_0, v_1, \ldots, v_n = d)$, we set a "virtual traffic volume" for each link $l = (v_i, v_{i+1})$, $i \in \{0, 1, \ldots, n-1\}$ to $x_l^v = x_0^v \cdot \beta^h$. Here β ($\beta > 1$) is a constant and h is the hop number of the *lightest-shortest* path $p_{ls}(v_{i+1}, d)$, i.e., one of the lightest paths from v_{i+1} to d which has the least number of hops. Intuitively, we pose an exponential penalty to each additional hop in a path. The weight of a link is set to $P_l(x_l^v)$, where $P_l(\cdot)$ follows the power function in Section III. The weight of path $p = (s = v_0, v_1, \ldots, v_n = d)$ is the sum of the weight of each link:

$$w_b(p) = \sum_{i=0}^{n-1} P_{(v_i, v_{i+1})} \left(x_0^v \cdot \beta^{\mathrm{Hops}(p_{ls}(v_{i+1}, d))} \right), \quad (6)$$

where function $\mathrm{Hops}(p)$ returns the hop number of path p. Note that a link's weight is not a static value, and may differ with path p and destination node d. However, we can proof the strict left-isotonicity of this path weight structure.

We define an algebra $(S, \oplus, \preceq, w_b)$ based on Eq. (6) where S is R^+, $\preceq$ is $\leq$, w_b are given in Eq. (6), and $w_b(p) \oplus w_b(q)$ is equal to $w_b(p \circ q)$ which can be calculated by Eq. (6).

Theorem 5.1: The algebra $(S, \oplus, \preceq, w_b)$ defined by Eq. (6) is strictly left-isotonic.

Proof: As shown in Fig. 4. Suppose p_1 and p_2 are two paths from node s to node d. Without loss of generality, we suppose that p_1 is lighter than p_2, i.e., $w_b(p_1) \prec w_b(p_2)$. Let us check the order relation between $w_b(q \circ p_1)$ and $w_b(q \circ p_2)$, i.e., the weights after concatenating p_1 and p_2 to path q, respectively.

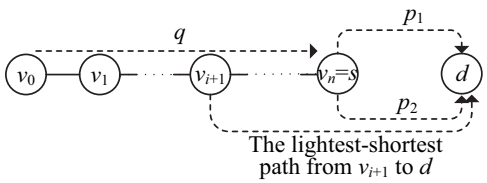


Fig. 4: The topology used to prove the strict left-isotonicity of path weight structure defined by Eq. (6).

Assume $q = (v_0, v_1, v_2, \ldots, v_n = s)$. According to Eq. (6) we have

$$w_b(q \circ p_1) = \sum_{i=0}^{n-1} P_{(v_i, v_{i+1})} \left(x_0^v \cdot \beta^{\mathrm{Hops}(p_{ls}(v_{i+1}, d))} \right) + w_b(p_1)$$

and

$$w_b(q \circ p_2) = \sum_{i=0}^{n-1} P_{(v_i, v_{i+1})} \left(x_0^v \cdot \beta^{\mathrm{Hops}(p_{ls}(v_{i+1}, d))} \right) + w_b(p_2).$$

Note that the length of the lightest-shortest path from v_i to d is independent of p_1 or p_2. Because $w_b(p_1) \prec w_b(p_2)$ means $w_b(p_1) < w_b(p_2)$, we can obtain from the above equations $w_b(q \circ p_1) < w(q \circ p_2)$, which means $w_b(q \circ p_1) \prec w_b(q \circ p_2)$.

This implies that the strict left-isotonicity holds, and completes the proof. ■

Based on Theorem 4.2 and Theorem 5.1, we can achieve a consistent (thus loop-free) hop-by-hop routing by applying Dijkstra-like algorithm and we develop Algorithm Dijkstra-Green-B. $\boldsymbol{P}$ in the inputs denotes the set of the traffic-power functions of all the links in E. In the algorithm, $w[v]$ denotes the weight of the current path from v to d and $\varphi[v]$ denotes the successor (or next hop node) of v. $N(u)$ denotes the set of neighbor nodes of u. $h[u]$ is used to store the hop number of the lightest-shortest path from u to d. There are a few differences between Dijkstra-Green-B and the standard Dijkstra. 1) A sink tree rooted at d is calculated and the algorithm halts once s is extracted (Step 5 to 7). 2) $h[u]$ is used to record the lightest-shortest path. 3) A link weight is calculated according to Eq. (6) in Step 9 and 10.

The computation complexity of Dijkstra-Green-B is the same as the standard Dijkstra in the worst case, i.e., $O(|E| + |V| \log |V|)$. However, the algorithm can stop once the path from s to d is finished and the complexity in the best case is $O(1)$. Thus, we can expect that the average complexity is less than Dijkstra.

B. Link Weights vs. Energy Conversation

In order to achieve greater energy conversation, we take a closer look at two main factors affecting power consumption.

Algorithm Dijkstra-Green-B()

Input: $G(V, E)$, s, d, $\boldsymbol{P}$, x_0^v, β;
Output: the green path from s to d which is stored in $\varphi[]$;

```
1: for each node v ∈ V
2:     w[v] ⇐ ∞; φ[v] ⇐ null; h[v] ⇐ ∞;
3: Q ⇐ V; w[d] ⇐ 0; h[d] ⇐ 0;
4: while Q ≠ φ
5:     u ⇐ Extract_Min(Q);
6:     if u = s
7:         return φ[];
8:     for each node v ∈ N(u)
9:         x ⇐ x_0^v · β^{h[u]};
10:        ϖ ⇐ P_(v,u)(x);
11:        if w[u] + ϖ < w[v]
12:            φ[v] ⇐ u;
13:            w[v] ⇐ w[u] + ϖ; h[v] ⇐ h[u] + 1;
14:        else if w[u] + ϖ = w[v] and h[u] + 1 < w[v]
15:            φ[v] ⇐ u; h[v] ⇐ h[u] + 1;
16: return null; //unreachable
```

1) Link weights vs. Power consumption per unit traffic volume: Recall the traffic-power functions (Eq. (1) and Eq. (4)) in Section III. The power consumption of a link increases with the increasing of the traffic volume. The link weight should reflect this. As a matter of fact, if the traffic volume x_l is proportion to ρ_l, we can achieve an optimal routing.

Lemma 5.1: If $P_l(x_l) = \rho_l x_l$ for any $l \in E$, the minimum power routing can be achieved by setting the weight of link l to ρ_l, and running Dijkstra in each router.

Proof: The total power consumption can be represented by the sum of the power consumed by each path, because $P_l(x_l) = \rho_l x_l = \rho_l \sum_p x_l^p$, where x_l^p is the traffic volume of path p which traverses link l. For any path $p = (v_0, v_1, \ldots, v_n)$ which has a traffic volume x^p, the power consumption is $\sum_{i=0}^{n-1} \rho_{(v_i,v_{i+1})} x^p = x^p \sum_{i=0}^{n-1} \rho_{(v_i,v_{i+1})}$. By setting the weight of each link l to ρ_l and running Dijkstra in each router, $\sum_{i=0}^{n-1} \rho_{(v_i,v_{i+1})}$ can be minimized. Thus, the power of path p is minimized. As a result, the total power consumption is minimized. ∎

In general, a link weight should reflect $\frac{dP_l}{dx_l}$, i.e., the power consumption per unit traffic volume. We can set the link weight to $P_l(x_l + \Delta x) - P_l(x_l)$ where Δx is a small constant.

2) Link weight vs. trunk-link: We know that for a trunk-link, if the traffic volume results in a leap to a higher "stair", there can be great power loss. We tend to assign a higher weight for a trunk-link to reduce its traffic volume. However, this comes with a tradeoff that the end-to-end paths may become longer and the extra hops also consume power. We can reduce the power consumption only if the power increment induced by the path stretches is less than the power leaping.

Generally, we take a heuristic by multiplying the weight of an trunk-link with a factor k_l

$$k_l = \gamma \sqrt{\frac{x_0^v}{r_u - r_d}}, \tag{7}$$

where γ is used to balance the link weights of non-trunk-links and trunk-links; x_0^v is an estimated end-to-end traffic and is the same as in Section V-A; r_u and r_d are calculated as follows. Given traffic volume x_l, r_u is the least traffic volume where a power leaping may occur and $r_u > x_l$ (recall that we have traffic thresholds $r_0, r_1, \ldots, r_{n_l-1}$ in our power model in Section III), and let $r_u = c_l$ if r_u cannot be found, where c_l is the capacity of link l; r_d is the largest traffic volume where a power leaping may occur and $r_d < x_l$, and let $r_d = 0$ if r_d cannot be found. In practical, we use the historical link load $\bar{x}_l$ instead of the realtime value x_l to avoid routing oscillations, because $\bar{x}_l$ has a diurnal pattern under shortest path routing. We can see that if x_0^v is big and/or $r_u - r_d$ is small, a power leaping is likely to happen, and we have a big k_l.

In what follows, we prove that we can develop a path weight that is isotonic based on these two improvements. We however admit that these two improvements are still preliminary and we leave more in depth investigation into future work.

C. Dijkstra-Green-Adv Algorithm

Based on the discussion above, we design an advanced path weight for Green-HR. We also prove the isotonicity to achieve the loop-free hop-by-hop routing, after which we develop the Dijkstra-Green-Adv algorithm.

Specifically, a link weight is assigned in two steps. First, the weight of link l is set to $P_l(\bar{x}_l + x_0^v) - P_l(\bar{x}_l)$, where $\bar{x}_l$ is the historical traffic volume estimation for link l and x_0^v is the same as in Section V-A. Second, we scale up the link weight by k_l if link l is a trunk-link, where k_l is defined as in Eq. (7). For a path p, the weight function $w_{adv}(p)$ is defined as follows.

$$w_{adv}(p) = \sum_{l \in p} \left(P_l(\bar{x}_l + x_0^v) - P_l(\bar{x}_l)\right) \cdot k_l. \tag{8}$$

We define the path weight structure $(S, \oplus, \preceq, w_{adv})$ based on Eq. (8). S is R^+ and $\preceq$ is $\leq$. w_{adv} are given in Eq. (8), and $w_{adv}(p) \oplus w_{adv}(q)$ is equal to $w_{adv}(p \circ q)$ which is equal to $w_{adv}(p) + w_{adv}(q)$.

Theorem 5.2: The path weight structure defined by Eq. (8) is strictly left-isotonic.

Proof: As shown in Fig. 5. Suppose p_1 and p_2 are two paths from node s to node d. Without loss of generality, we suppose that p_1 is lighter than p_2, i.e., $w_{adv}(p_1) \prec w_{adv}(p_2)$. We need to check the order relation between $w_{adv}(q \circ p_1)$ and $w_{adv}(q \circ p_2)$ to proof strictly left-isotonicity.

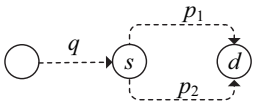


Fig. 5: The topologies used to prove the strict isotonicity of path weight structure defined by Eq. (8).

According to Eq. (8), we have $w_{adv}(q \circ p_1) = w_{adv}(q) + w_{adv}(p_1)$ and $w_{adv}(q \circ p_2) = w_{adv}(q) + w_{adv}(p_2)$. This is because $\bar{x}_l$ and x_0^v do not change when concatenating p_1 and p_2 to q, respectively. Because $w_{adv}(p_1) \prec w_{adv}(p_2)$, i.e. $w_{adv}(p_1) < w_{adv}(p_2)$, we obtain $w_{adv}(q \circ p_1) < w_{adv}(q \circ p_2)$, which means $w_{adv}(q \circ p_1) \prec w_{adv}(q \circ p_2)$. This implies that the path weight structure is strictly left-isotonic. ∎

Algorithm Dijkstra-Green-Adv()

Input: $G(V,E)$, s, d, $\boldsymbol{P}$, x_0^v, $\bar{\boldsymbol{x}}$;
Output: the advanced green path from s to d which is stored in $\varphi[]$;
1: **for** each node $v \in V$
2: $w[v] \Leftarrow \infty; \varphi[v] \Leftarrow null$;
3: $Q \Leftarrow V; w[d] \Leftarrow 0$;
4: **while** $Q \neq \phi$
5: $u \Leftarrow \text{Extract\_Min}(Q)$;
6: **if** $u = s$
7: **return** $\varphi[]$;
8: **for** each node $v \in N(u)$
9: $\varpi \Leftarrow P_{(u,v)}(\bar{x}(u,v) + x_0^v) - P_{(u,v)}(\bar{x}(u,v))$;
10: $\varpi \Leftarrow \varpi \cdot k(u,v)$;
11: **if** $w[u] + \varpi < w[v]$
12: $\varphi[v] \Leftarrow u$;
13: $w[v] \Leftarrow w[u] + \varpi$;
14: **return**;

Based on Theorem 4.1 and Theorem 5.2, we design the advanced algorithm which can run in a hop-by-hop manner, namely the Dijkstra-Green-Adv algorithm.

The algorithm makes only a few modifications to the standard Dijkstra's algorithm. There are some new inputs including the set of traffic-power functions $\boldsymbol{P}$, the set of historical traffic volumes $\bar{\boldsymbol{x}}$, and x_0^v. In the algorithm, $w[v]$ denotes the weight of the current path from v to d and $\varphi[v]$ denotes the successor (or next hop node) node of v. $N(u)$ denotes the set of neighbor nodes of u. The main difference between Dijkstra-Green-Adv and the standard Dijkstra's algorithm is that Dijkstra-Green-Adv calculates the link weight of (u,v) in Step 9 and 10 according to Eq. (8). The computation complexity of Dijkstra-Green-Adv is the same as the Dijkstra-Green-B algorithm, i.e., $O(|E| + |V|\log|V|)$ in the worst case.

D. Dijkstra-Green Algorithm

When designing the Green-HR algorithm for energy conservation, we should also take into account the QoS requirements such as end-to-end delay and bottleneck bandwidth. This is because QoS guarantee is important to users and is a major concern of Internet Service Providers (ISP). However, a greener path may be longer or more congested, which degrades the QoS performance. We will first develop an algorithm that co-considers energy conservation and path length. We will then further discuss the relationship between Green-HR and QoS requirements.

Let $w_{adv}(p)$ be the path weight defined as Eq. (8), which reflects the power consumption. Let $Len(p)$ be the length of path p. We will combine $w_{adv}(p)$ and $Len(p)$ in order to get small path stretch (the ratio of a path of the source-destination against the shortest path of the source-destination). We analysis the path stretches of $w_{adv}(p)$ and find that the path stretch is small for most of the paths; yet there exists some big stretch when the length of the shortest path is small. Thus, we develop the algorithm which takes additional special considerations for the short paths.

Specifically, we take a heuristic by dividing the link length with the root of the shortest path length to node d. In this way, path length will dominate in the path weight for short paths, and power will dominate for long paths. The weight of path $p = (s = v_0, v_1, \ldots, v_n = d)$ is defined as

$$w_g(p) = w_{adv}(p) + \sum_{i=0}^{n-1} \frac{\kappa \cdot Len(v_i, v_{i+1})}{\sqrt{Len(p_s(v_{i+1}, d))}} \tag{9}$$

where $p_s(v_i, v_j)$ denotes the shortest path from node v_i to node v_j, and κ is a constant factor which we can use to adjust the path stretch performance.

We define the path weight structure $(S, \oplus, \preceq, w_g)$ similar to $(S, \oplus, \preceq, w_b)$ and $(S, \oplus, \preceq, w_{adv})$, except that $\oplus$ and w_{adv} is defined by Eq. (9).

Theorem 5.3: The path weight structure defined by Eq. (9) is strictly left-isotonic.

Proof: As shown in Fig. 4. Suppose p_1 and p_2 are two paths from node s to node d. Without loss of generality, we suppose that p_1 is lighter than p_2, i.e., $w_g(p_1) \prec w_g(p_2)$.

Assume $q = (v_0, v_1, v_2, \ldots, v_n = s)$. According to Eq. (9) we have

$$w_g(q \circ p_1) = \sum_{i=0}^{n-1} \frac{\kappa \cdot Len(v_i, v_{i+1})}{\sqrt{Len(p_s(v_{i+1}, d))}} + w_g(p_1)$$

and

$$w_g(q \circ p_2) = \sum_{i=0}^{n-1} \frac{\kappa \cdot Len(v_i, v_{i+1})}{\sqrt{Len(p_s(v_{i+1}, d))}} + w_g(p_2).$$

Note that the length of the shortest path (not the light-shortest path as shown in Fig. 4) from v_{i+1} to d is independent of p_1 or p_2. Because $w(p_1) \prec w(p_2)$ means $w_g(p_1) < w_g(p_2)$, we can then obtain $w_g(q \circ p_1) < w_g(q \circ p_2)$, which means $w_g(q \circ p_1) \prec w_g(q \circ p_2)$. This implies that the strict left-isotonicity holds. ■

Based on Theorem 4.2 and Theorem 5.3, we develop a loop-free hop-by-hop algorithm named Dijkstra-Green.

Algorithm Dijkstra-Green()

Input: $G(V,E)$, s, d, $\boldsymbol{P}$, x_0^v, $\bar{\boldsymbol{x}}$, $\boldsymbol{p}_s$, κ;
Output: the green path with less path stretch from s to d which is stored in $\varphi[]$;
1: **for** each node $v \in V$
2: $w[v] \Leftarrow \infty; \varphi[v] \Leftarrow null$;
3: $Q \Leftarrow V; w[d] \Leftarrow 0$;
4: **while** $Q \neq \phi$
5: $u \Leftarrow \text{Extract\_Min}(Q)$;
6: **if** $u = s$
7: **return** $\varphi[]$;
8: **for** each node $v \in N(u)$
9: $\varpi \Leftarrow P_{(u,v)}(\bar{x}(u,v) + x_0^v) - P_{(u,v)}(\bar{x}(u,v))$;
10: $\varpi \Leftarrow \varpi \cdot k(u,v)$;
11: $\varpi \Leftarrow \varpi + \kappa \cdot Len(u,v)/\sqrt{Len(p_s(u,d))}$;
12: **if** $w[u] + \varpi < w[v]$
13: $\varphi[v] \Leftarrow u$;
14: $w[v] \Leftarrow w[u] + \varpi$;
15: **return**;

The algorithm is based on Dijkstra-Green-Adv. We add new inputs including the set of shortest paths $\boldsymbol{p}_s$ and κ. The main modification made to Dijkstra-Green-Adv is that Dijkstra-Green combines path length in the link weight in Step 11. Since we have to maintain the shortest paths when the topology changes, the computation complexity of Dijkstra-Green is $O(|E||V| + |V|^2 \log|V|)$ in the worst case.

E. Discussion on Green-HR and QoS Requirements

We now study the problem of hop-by-hop green routing with limited degrading on general QoS parameters. Naturally, the goal is to find the paths which minimize the power consumption, subjected to some QoS constraints. We study a simple case when only one QoS parameter is considered. Formally, for a path p from node s to node d, let $w_1(p)$ be the weight of path p which reflects the QoS parameter, and $w_2(p)$ be the path weight which reflects the power consumption. Let p_l be the path with the lightest QoS weight $w_1(p_l)$. Given a threshold value ζ, our objective is to find a path p^* from s to d, such that for any path p from s to d, we have $w_2(p^*) \preceq w_2(p)$ and $w_1(p^*) \preceq w_1(p_l) \oplus \zeta$.

However, we find that it is impossible to design an isotonic path weight structure that meet the above conditions. More generally, if we combine two path weights in order to optimize one while bounding the other, the result path weight structure will never be isotonic. We find that this is due to the intrinsic nature of routing algebra. We summarize the conclusion in the following lemma.

Lemma 5.2: Given two isotonic path weight structures $(S_1, \oplus_1, \preceq_1, w_1)$ and $(S_2, \oplus_2, \preceq_2, w_2)$, path weight structure $(S, \oplus, \preceq, w)$ defined as follows is not isotonic: path p^* from s to d is the lightest for $(S, \oplus, \preceq, w)$ if and only if
(1) $w_1(p^*) \preceq_1 w_1(p_l) \oplus_1 \zeta$ for a given ζ, where p_l is the lightest path from s to d for $(S_1, \oplus_1, \preceq_1, w_1)$;
(2) $w_2(p^*) \preceq_2 w_2(p)$, where p is a path from s to d and $w_1(p) \preceq_1 w_1(p_l) \oplus_1 \zeta$.

Proof:

We prove the lemma by contradiction. Assume that $(S, \oplus, \preceq, w)$ is isotonic. Then, according to [25], there exists at least one lightest path from s to d such that all its subpaths with destination d are also lightest paths. Such a lightest path is called a D-lightest path. Now, we will show a counter example where we can find no lightest path that is a D-lightest path.

The example is shown in Fig. (6), where $w_1(p_1) = w_1(p_2)$ and $w_1(q) = \zeta$ and $w_1(p_3) \preceq_1 w_1(p_2) \oplus_1 \zeta$, and $w_2(q \circ p_3) \prec_2 w_2(q \circ p_2) \prec_2 w_2(p_1)$.

There are three paths from s to d: p_1, $q \circ p_2$, and $q \circ p_3$. $q \circ p_3$ is not a lightest path because

$$w_1(q \circ p_3) = w_1(q) \oplus_1 w_1(p_3) \succ_1 \zeta \oplus_1 w_1(p_2) = w_1(p_1) \oplus_1 \zeta,$$

which does not meet condition (1) of Lemma 5.2. Similarly, p_1 is not a lightest path because

$$w_2(p_1) \succ_2 w_2(q \oplus_2 p_2),$$

which does not meet condition (2) of Lemma 5.2.

Thus, the lightest path from s to d is $q \circ p_2$. However, the lightest path from s' to d is not p_2 but p_3, because $w_1(p_3) \prec_1 w_1(p_2) \oplus_1 \zeta$ and $w_2(p_3) \prec_2 w_2(p_2)$. This means that the lightest path from s to d is not a D-lightest path, and we conclude that $(S, \oplus, \preceq, w)$ is not isotonic.

■

Note that in Lemma 5.2, the QoS parameter is bounded by $w_1(p_l) \oplus_1 \zeta$. Sometimes, we may wish to bound the parameter by a scale factor. For example, let $Len(p)$ be the length of path p, then we want $Len(p) \leq \zeta \cdot Len(p_s)$ where $\zeta > 1$. However, in such a case, the same conclusion as Lemma 5.2 can still be deduced.

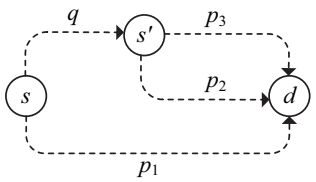


Fig. 6: An example of routing loops.

Lemma 5.2 implies that a QoS parameter can not be bounded if we minimize a path weight for energy conservation by using a hop-by-hop routing that is guaranteed to be loop-free. This is also true for multiple QoS parameters. Thus, we have to study the intrinsic relationship between a particular QoS parameter and a green path weight structure when combining them.

VI. Performance Evaluation

A. Methodology

We use both synthetic and real topologies in the simulations. For the synthetic topologies, we use BRITE [27] to generate network topologies and the parameters follow [28]. Each dot in the figures is an average of 1000 random and independent simulations. We have two real topologies: 1) the Abilene backbone with 12 nodes and 15 two-directional links, and 2) the China Education and Research Network (CERNET) backbone, which has 8 nodes and 12 two-directional links (9 links are trunk-links).[1]

The link capacities of the synthetic topologies are determined based on the fact that a node with a big degree is likely to hold links with large capacity [29]. We set a link's capacity to 9953.28Mbps (OC192-1 port) if both end nodes of the link have a degree larger than 5. The capacity is set to 2488.32Mbps (OC48-1 port) if one end node has a degree larger than 5 and the other has a degree less than 6 but larger than 2. Finally, the other links' capacities are set to 622.08Mbps (OC12-1 port).

For synthetic topologies and CERNET, we create traffic matrices according to the gravity model [24]. The traffic volume from node s to d, namely $f(s,d)$, is proportional to the total output capacity of s and the total input capacity of d, and is inverse proportional to the square of the hop number of the shortest path from s to d, as shown in Eq. (10)

$$f(s,d) = \frac{\eta \cdot \sum_{v \in N(s)} c(s,v) \cdot \sum_{u \in N(d)} c(u,d)}{(\mathrm{Hops}(p_s(s,d)))^2} \qquad (10)$$

where η is a scale factor by which we can create different levels of traffic volume, $c(s,v)$ the capacity of link (s,v), $N(s)$ the set of neighbors of s, and $p_s(s,d)$ is the shortest path from s to d. For Abilene, we use real traffic matrices which

[1]We remove the stub nodes which have only one link to the backbone.

can be found in [30] and one traffic matrix is summarized each five minutes. The traffic volume on an Abilene link is hundreds of Mbps and the link utilization ratio is around 10%.

For the synthetic topologies, a link is designated to be a trunk-link with probability λ. For Abilene, seven links are randomly selected to be trunk-links.

We assume that a trunk-link is consisted of four physical links with a lower rate than this link's original capacity. The traffic volume thresholds for state changes are set to the operation rates of the links, shown in Table I. The power consumptions per unit traffic volume (ρ_l) of different operation rates are set as constants, referring to the measurement results given by [11] and [31]. The idle power consumptions of different operation rates are set according to [32], shown in Table I.

TABLE I: Power consumptions of line cards

line card	operation rate(Mbps)	maximum power(W)	ρ_l (W/Mbps)	calculated idle power(W)
1-Port OC3	155.52	60	0.01	58.4
1-Port OC12	622.08	80	0.008	75.0
1-Port OC48	2488.32	140	0.006	125.1
1-Port OC192	9953.28	174	0.004	134.2

Some default values are set as follows. The node number of a synthetic topology is 100 and the link density is 2 (i.e. total 200 links). Trunk-link ratio λ is 0.5. Synthetic traffic matrices are set to create an average link utilization ratio of 25%. For Dijkstra-Green-B and Dijkstra-Green-Adv, $x_0^v(d)$ is 1/800 of the sum of the input capacity of node d, β is 1.5.

We compare our algorithms with the standard shortest path routing. We will evaluate the power saving ratio, defined as $(P_s - P_g)/P_s$, where P_s is the total power consumed by line card under shortest path routing, and P_g is the total power under our algorithms. We will also evaluate path stretch ratio and average hop number.

B. Results In The Synthetic Topologies

1) Results On Different Traffic Levels: Fig. 7 shows the power saving ratio as against of traditional Dijkstra. We see that the power saving can be as much as 55%, when the average link utilization is low. The power saving ratios decease when the average link utilization ratio increases. This is not surprising as our algorithm is closely related to traffic volume; and the more traffic there is, the less trade-off can be found. Yet we still see a power saving ratio of 38% when the average link utilization ratio is 65%. Dijkstra-Green-Adv is better than Dijkstra-Green-B as its design takes more factors that affect power consumption into consideration. We also see that Green-Dijkstra is only slightly worse than Green-Dijkstra-Adv, mainly when the network is in high utilization.

Fig. 8 shows the average path stretch of our algorithms. We see that the average the path length of Green-Dijkstra-Adv is about 1.22 times to that of the shortest path and the path length of Green-Dijkstra is only 1.04 times. This is not surprising as Dijkstra-Green-Adv focuses on energy conservation and Green-Dijkstra co-considers path length when saving energy. Fig. 9 shows the average hop number. Dijkstra-Green-B has the least hop number (about 3.3), because it saves energy mainly by choosing the paths that traverse less routers. Dijkstra-Green-Adv and Dijkstra-Green have a hop number around 3.8 and 3.7, respectively. We also see that traditional Dijkstra has the largest hop number, which implies that the shortest paths do not always traverse the least number of routers.

2) Results On Different trunk-Link Ratios: Fig. 10 shows the power saving ratio as a function of trunk-link ratio λ. We can see that the more trunk-links are deployed, the more power is saved. When all the links are trunk-links, a 65% power-saving can be achieved. However, the curves increase in a sub-linear manner. This implies that the majority of the power saving comes from avoiding trunk-links, so the gain is intrinsically limited when trunk-links are too many (we cannot find a path avoiding trunk-links). Dijkstra-Green-Adv and Dijkstra-Green save more power than Dijkstra-Green-B when λ is near 0.5, because their designs can avoid trunk-links more easily.

Fig. 11 and 12 show the path stretch and the hop number, respectively. We can see the result of Dijkstra-Green-B changes little, because Dijkstra-Green-B considers trunk-links little and the routing changes little with trunk-link ratio λ. The result of Dijkstra-Green-Adv and Dijkstra-Green increases when $\lambda < 0.7$ and decreases a little when $\lambda > 0.7$, because these two algorithms use the paths of more hops to avoid trunk-links.

3) Results On Different Link Densities: Fig. 13 shows the power saving ratio as a function of link density. We find that the higher the link density is, the more the power is saved for Dijkstra-Green-Adv and Dijkstra-Green. This is not surprising as there are more trunk-links when the link density is higher. The power saving ratio of Dijkstra-Green-B decreases when the link density increases from 4 to 5. This may be because Dijkstra-Green-B can choose a path with less hops easily and cannot avoid trunk-links when there are too many links in the network.

Fig. 14 shows the average path stretch, which increases with the increment of link density. This may be mainly because the shortest path has less length when the link density is higher. Fig. 15 shows the number of hops. We can see the result decreases with the increment of link density.

C. Results In The Real Topologies

Fig. 16 shows the average link utilization, using the Abilene topology and the traffic matrices collected on March 8, 2004. We can see the value is around 5% to 10%. Fig. 17 shows the power saving ratio as a function of time. The results using the data in other time periods are similar. We can see the result changes with time, because the traffic matrix is always changing. However, the average power saving ratios of the algorithms are all around 57%. The results of the three algorithms are similar because the traffic in the network is small (recall the results in Fig. 7). Furthermore, we find Dijkstra-Green-Adv performs worse than the other two algorithms.

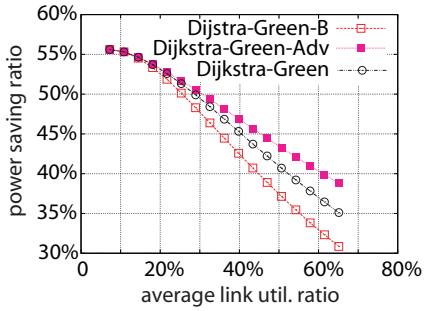


Fig. 7: Power saving ratio as a function of average link util. ratio (synthetic topology).

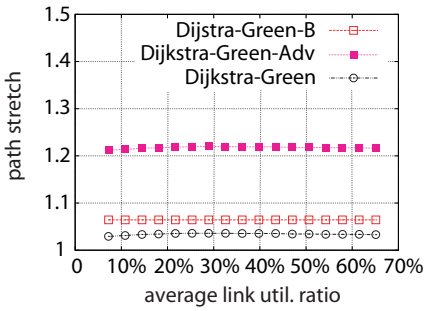


Fig. 8: Average path stretch as a function of average link util. ratio (synthetic topology).

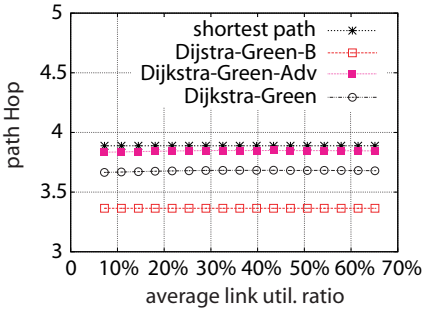


Fig. 9: Average path hop number as a function of average link util. ratio (synthetic topology).

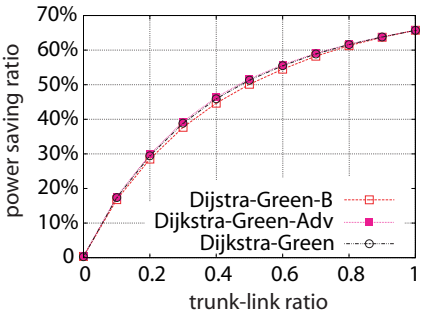


Fig. 10: Power saving ratio as a function of trunk-link ratio (synthetic topology).

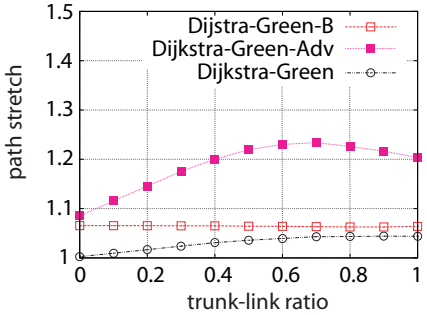


Fig. 11: Average path stretch as a function of trunk-link ratio (synthetic topology).

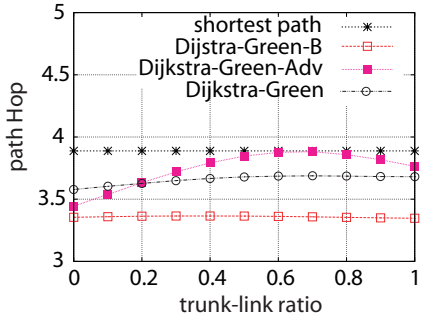


Fig. 12: Average path hop number as a function of trunk-link ratio (synthetic topology).

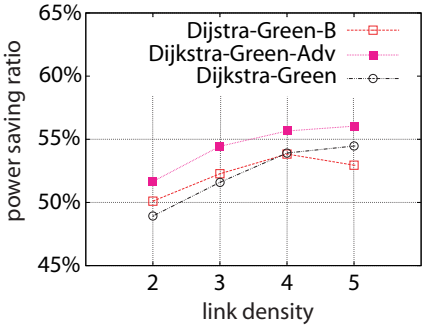


Fig. 13: Power saving ratio as a function of link density (synthetic topology).

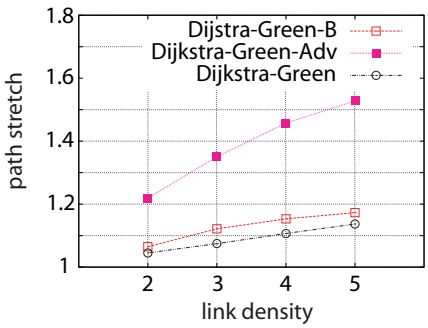


Fig. 14: Average path stretch as a function of link density (synthetic topology).

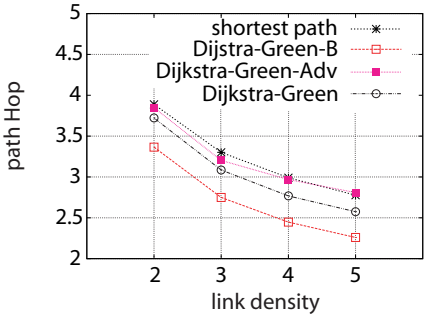


Fig. 15: Average path hop number as a function of link density (synthetic topology).

However, the difference is within 0.2%. This may be because Abilene has less nodes and links than the synthetic topologies and less paths can be found. Dijkstra-Green-Adv may choose a longer path, but no more power can be saved.

Fig. 18 shows the power saving ratio as a function of average link utilization ratio, using the CERNET topology. Like the results in synthetic topologies, the power saving ratio decreases with the increment of traffic volume. The performance of the algorithms are similar. When the link utilization raito is 50%, Dijkstra-Green-Adv and Dijkstra-Green can save 29.5% of the energy, about 7% more than Dijkstra-Green-B. However, sometimes Dijkstra-Green-B performs better. Like in Abilene, this is because CERNET has small numbers of nodes and links.

VII. Conclusion

In this paper, we studied to incorporate energy conservation into Internet routing design. We presented a power model to quantify the relation between traffic volume and power consumption; and validated our model using real experiments. We proposed a hop-by-hop approach and developed algorithms that guarantee loop-freeness and substantially reduce energy footprint in the Internet.

As a very first work, we admit that there are many unanswered questions. First, we believe our algorithms have much room to be improved on energy conservation. Second, we are

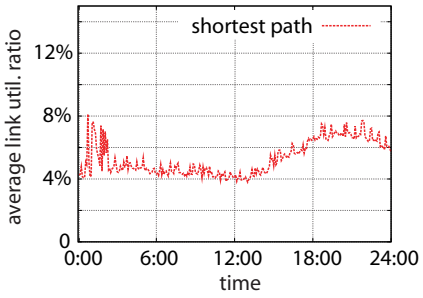


Fig. 16: Average link utilization ratio as a function of time (Abilene).

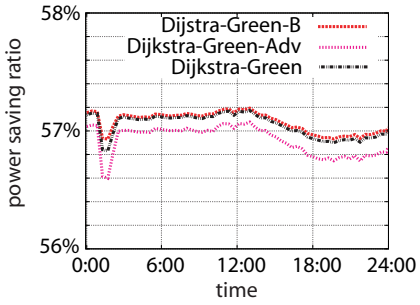


Fig. 17: Power saving ratio as a function of time (Abilene).

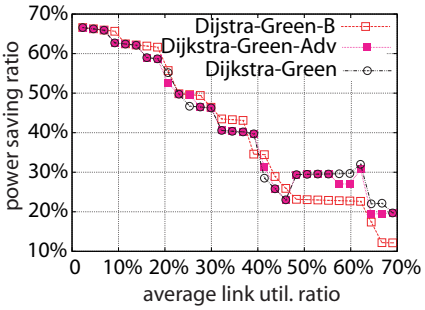


Fig. 18: Power saving ratio as a function of average link util. ratio (CERNET).

interested to further investigate a centralized scheme; this is useful when MPLS can be applied, and also provide theoretical bounding on the maximum possible power conservation.

REFERENCES

[1] J. Carter and K. Rajamani, "Designing Energy-Efficient Servers and Data Centers," *Computer*, vol. 43, pp. 76–78, 2010.

[2] I. Goiri, K. Le, M. Haque, R. Beauchea, T. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling energy consumption in green datacenters," in *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011, pp. 1–11.

[3] Y. Kim, T. Schmid, Z. M. Charbiwala, and M. B. Srivastava, "ViridiScope: Design and Implementation of a Fine Grained Power Monitoring System for Homes," in *Proc. of 11th international conference on Ubiquitous Computing(Ubicomp)*, 2009, pp. 245–254.

[4] V. Siddhartha, P.V. Ramakrishna, T. Geetha and A. Sivasubramaniam, "Automatic generation of energy conservation measures in buildings using genetic algorithms," *Energy and Buildings*, vol. 43, pp. 2718–2726, 2011.

[5] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-Aware Traffic Engineering," in *Proc. of IEEE ICNP*, October 2010, pp. 21–30.

[6] N. Vasic, P. Bhurat, D. Novakovic, M. Canini, S. Shekhar, and D. Kostic, "Identifying and using energy-critical paths," in *Proc. of CoNext'11*, 2011.

[7] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An Energy Saving Routing Algorithm for a Green OSPF Protocol," in *INFOCOM IEEE Conference on Computer Communications Workshops*, March 2010, pp. 1–5.

[8] A. Cianfrani, V. Eramo, M. Listanti, and M. Polverini, "An OSPF enhancement for energy saving in IP networks," in *IEEE INFOCOM Workshop on Green Communications and Networking*, April 2011, pp. 325–330.

[9] F. Cuomo, A. Abbagnale, A. Cianfrani, and M. Polverini, "Keeping the Connectivity and Saving the Energy in the Internet," in *IEEE INFOCOM Workshop on Green Communications and Networking*, April 2011.

[10] A. P. Bianzino, L. Chiaraviglio, and M. Mellia, "Distributed Algorithms for Green IP Networks," in *IEEE INFOCOM Workshop on Green Networking and Smart Grid*, March 2012.

[11] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing," in *Proc. of IEEE INFOCOM*, 2008, pp. 457–465.

[12] R. Kubo, J. Kani, H. Ujikawa, T. Sakamoto, Y. Fujimoto, N. Yoshimoto, and H. Hadama, "Study and Demonstration of Sleep and Adaptive Link Rate Control Mechanisms for Energy Efficient 10G-EPON," *IEEE/OSA JOCN*, vol. 2, pp. 716–729, 2010.

[13] C. Gunaratne and K. Christensen, "Ethernet Adaptive Link Rate: System Design and Performance Evaluation," in *Proc. of the 31st IEEE Conference on Local Computer Networks*, 2006, pp. 28–35.

[14] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links," in *Proc. of the first ACM SIGCOMM workshop on Green networking*, 2010, pp. 29–34.

[15] M. Gupta and S. Singh, "Greening of the Internet," in *ACM SIGCOMM*, 2003.

[16] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy ef..ciency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,*, vol. 13, pp. 223–244, 2011.

[17] K. Kant and N. Udar, "Multi-state power management of communication links," in *Proc. of IEEE Communication Systems and Networks (COMSNETS)*, 2011, pp. 1–10.

[18] N. Dinh and A. Walid, "Power Saving Protocol for 10G-EPON Systems: A Proposal and Performance Evaluations," in *Proc. of IEEE GLOBECOM*, 2012.

[19] P. Paul and S. V. Raghavan, "Survey of QoS routing," in *Proc. of the 15th international conference on Computer communication*, 2002, pp. 50–75.

[20] F. Ergun, R. K. Sinha, and L. Zhang, "QoS Routing with Performance-Dependent Costs," in *Proc. of IEEE INFOCOM*, 2000, pp. 137–146.

[21] J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 10, pp. 541–550, 2002.

[22] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for Energy Minimization in the Speed Scaling Model," in *Proc. of IEEE Infocom*, 2010, pp. 1–9.

[23] "BitEngine12000 router." [Online]. Available: http://www.bitway.com/product-a-6.html

[24] M. M. Rahman, S. Saha, U. Chengan, and A. Alfa, "IP Traffic Matrix Estimation Methods: Comparisons and Improvements," in *Proc. of IEEE International Conference on Communications*, 2006, pp. 90–96.

[25] Y. Yang and J. Wang, "Design Guidelines for Routing Metrics in Multihop Wireless Networks," in *Proc. of IEEE INFOCOM*, 2008, pp. 1615–1623.

[26] J. Wang and K. Nahrstedt, "Hop-by-Hop Routing Algorithms For Premium-class Traffic In Diffserv Networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, pp. 73–88, 2002.

[27] "Boston University Representative Internet Topology Generator, BRITE." [Online]. Available: http://www.cs.bu.edu/brite/

[28] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "Generating realistic ISP-level network topologies," *IEEE Communications Letters*, vol. 7, pp. 335–336, 2003.

[29] T. Hirayama, S. Arakawa, S. Hosoki, and M. Murata, "Models of link capacity distribution in ISP's router-level topologies," *International Journal of Computer Networks & Communications*, vol. 3, pp. 205–216, 2011.

[30] "Yin Zhangs Abilene TM." [Online]. Available: http://www.cs.utexas.edu/ yzhang/research/AbileneTM/

[31] W. Vereecken, W. V. Heddeghem, M. Deruyck, B. Puype, B. Lannoo, W. Joseph, D. Colle, L. Martens, and M. Pickavet, "Power Consumption in Telecommunication Networks: Overview and Reduction Strategies," *IEEE Communications Magazine*, vol. 49, pp. 62–69, 2011.

[32] "Power Management for the Cisco 12000 Series Router." [Online]. Available: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12spower.html