Hop-by-hop Computing for Green Internet Routing

Yuan Yang*, Dan Wang†, Mingwei Xu*, Suogang Li‡
*Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University
†The Hong Kong Polytechnic University Shenzhen Research Institute, The Hong Kong Polytechnic University
‡CERNET National Network Center
yyang@csnet1.cs.tsinghua.edu.cn, csdwang@comp.polyu.edu.hk, {xmw, lisg}@cernet.edu.cn

***Abstract*—In this paper we study energy conservation in the Internet. We observe that different traffic volumes on a link can result in different energy consumption; this is mainly due to such technologies as trunking (IEEE 802.1AX), adaptive link rates, etc. We design a green Internet routing scheme, where the routing can lead traffic in a way that is green. We differ from previous studies where they switch network components, such as line cards and routers, into sleep mode. We do not prune the Internet topology.**

We first develop a power model, and validate it using real commercial routers. Instead of developing a centralized optimization algorithm, which requires additional protocols such as MPLS to materialize in the Internet, we choose a hop-by-hop approach. It is thus much easier to integrate our scheme into the current Internet. We progressively develop three algorithms, which are loop-free, maximize energy conservation, and jointly consider green and QoS requirements such as path stretch. We comprehensively evaluate our algorithms through simulations on synthetic and real topologies and traffic traces. We show that the power saving in the line cards can be as much as 50%.

I. Introduction

Energy conservation has become a global concern nowadays and energy cost is predicted to increase in the forthcoming future. As a consequence, how to save energy has become an important issue in the design of such areas as data centers [1], building management [2], to name but a few.

In the Internet, routers and switches account for the majority of energy consumption. More and more high performance routers are developed and deployed currently. For example, a Cisco CRS-1 router can draw about one MegaWatt under full configuration, 10,000 times more than a PC. By 2010, 5000 Cisco CRS-1 routers were deployed[1]. Facing such high energy consumption, there are many studies for energy conservation of the Internet [3][4][5][6][7][8].

In general, these studies switch network components, such as line cards and routers, into sleep mode. As such, these studies compute a topology with less nodes and links, which may degrade network resistance against failures. The network components to be turned off are carefully chosen and tradeoffs are investigated to balance network performance and energy conservation. To realize these approaches, MPLS or additional protocols are usually necessary.

In this paper, we study "green" routing where we do not prune the Internet topology. A key observation that makes this possible is that the energy consumption for packet delivery can be different in different traffic volumes [9]. Therefore, we can select paths that consume less power while delivering traffic. Intrinsically, this is caused by technologies including trunking (or bundled links) [10] and adaptive link rates (ALR) [11] [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 40 Gbps link may consist of four 10 Gbps 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 scale with traffic volume. As such, even without changing the topology (i.e., by switching routers into sleeping mode), energy consumption can still vary greatly given different routings that result in different traffic volume on the paths. Intrinsically, our work shows that there can be more refined control than an on-off (0-1) control of the routers in energy conservation. We further illustrate the impact of this through an example.

Example Consider a network in Fig. 1, in which links (a, b) and (b, c) both consist of four parallel OC48 (2.5 Gbps) physical links and the other three links are single OC192 (10 Gbps) physical links. More specifically, for an OC48 link, there is a baseline 125.1 Watt energy consumption and an additional 0.006 Watt for each 1 Mbps traffic; and for an OC192 link, there is a baseline 134.2 Watt energy consumption and an additional 0.004 Watt for each 1 Mbps traffic. In this topology, shortest path routing will generate 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 1 Gbps on each path. Links (a, b) and (b, c) then have to power on two parallel OC48 physical links because the total traffic on these links is 3 Gbps. 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 the shortest path except that path $a \leftrightarrow c = (a, e, d, c)$, then links (a, b) and (b, c) will only carry 2 Gbps and power on only one OC48 physical link each. The total energy consumption is $(125.1+2000\times0.006)\times2+(134.2+4000\times0.004)\times3 = 724.8$ Watt, a 25.7% improvement.

The above example is not special. Yet to systematically study this problem, we first need to quantify an appropriate power model, i.e., the relationship between power consumption

The work is supported by the National Basic Research Program of China (973 Program) under Grant 2012CB315803, the National Natural Science Foundation of China (61133015),and the National High-Tech Research and Development Program of China (863 Program) under Grants 2011AA01A101. Dan Wang's work is supported in part by the National Natural Science Foundation of China (61272464).

[1]http://newsroom.cisco.com/dlls/2010/prod%5F030910.html

and traffic volume following the standards of trunking/ALR. Second, we need designs to maximize energy conservation. There are two possible ways. First, we can formulate the problem into an optimization problem, analyze the problem complexity and design a centralized routing algorithm. The algorithm may find an optimal or near optimal solution; and to establish the routing paths after the computation, we can use MPLS. We plan a future study in this direction.

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 the OSPF protocol. Under this hop-by-hop design, we face the following challenges: 1) to be practical, the computation complexity should be comparable to that of shortest path routing (i.e., Dijkstra) and, more importantly, the routing must be loop-free; 2) hop-by-hop computing should maximize energy conservation; and 3) important QoS performance of the network such as path stretch may be considered concurrently, and can be naturally adjusted.

We present a comprehensive study. We first develop a power model and validate the model using real experiments in commercial routers. We then develop principles and a baseline hop-by-hop green routing algorithm that guarantees loop-free routing. 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 concurrently considers energy conservation and path stretch, and conduct an in-depth study on 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 could 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 properties of routing algebra are discussed in Section IV. Section V is devoted into our design of hop-by-hop green routing algorithms and analysis. Section VI shows the evaluation and Section VII concludes the paper.

II. Related Work

Together with the world-wide objective to build a greener globe, more and more computing systems include energy conservation into their design principles [1] and there are efforts to develop a greener Internet as well [13][14].

First, there are studies on saving energy of the routers. For example, there are studies to develop a better forwarding behavior so as to save energy from the TCAMs [15] of a router.

Second, there are studies on energy conservation of the Internet from upper layers point of view. For example, Energy Efficient TCP [16] is proposed to perform congestion control with dynamic bandwidth adjustment. Note that the energy saving of such upper layer behavior control is realized by translating into better router control in the network layer.

Third, there are studies to save energy from a network routing point of view. GreenTE [3] is proposed to aggregate traffic using MPLS tunnels, so as to switch the under-utilized network components into sleep mode and thus save energy. REsPoNse [4] is proposed to identify energy-critical and on-demand paths offline. The packets are delivered online also with the objective to effectively aggregate traffic and switch more network components into sleep mode. GreenTE and REsPoNse are both centralized schemes. GreenOSPF [5] is proposed to aggregate traffic in a distributed fashion and switch the network components into sleep mode. However, to achieve a good performance, a centralized algorithm [6] is still needed to assign sleeping links. ESACON [7] is proposed to collaboratively select sleeping links with special connectivity properties. Routing paths are then computed after these links are removed. A fully distributed approach is proposed [8] which collects global traffic information and aggregates traffic to switch appropriate network components into sleep mode.

Our approach falls into the third category discussed above, yet it differs from the aforementioned schemes in the aspects as follows. First, all the previous proposals set network devices or links into sleep mode. Our design is based on the observation that different traffic volumes also have different energy consumption. A routing algorithm may take this into consideration. To the best of our knowledge, we are the first to propose such a scheme. Second, though some previous schemes compute the network components to be shut down in a distributed fashion, great changes to the current routing protocols are still needed. Our routing computation is hop-by-hop and Dijkstra-oriented, which we believe is easier to be incorporated into the current routing architecture.

We may consider green as one type of services that the Internet should be provisioned. There are many studies on Internet Quality of Service [17]. There were two different approaches in Internet QoS support beyond shortest path routing. One is centralized computation [18]. The advantage is that since different types of services usually introduce conflicts, a centralized scheme can compute optimal or near optimal solutions. But 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 [19]. A seminal paper [19] develops a routing algebra model and shows that to make hop-by-hop computation loop-free requires the link weights to have certain isotonicity properties.

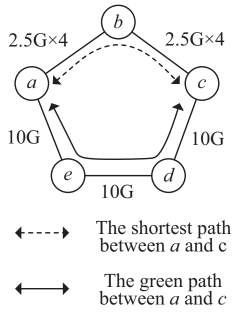
In this paper, we also leverage the algebra model to develop hop-by-hop computing for green Internet routing, which is loop-free. We have a set of algorithms, by which we gradually improve the energy conservation performance.

III. Power Model

Our objective is to model the relationship between link power consumption and traffic volume. We first present the router operation backgrounds and our modeling details. Then we use simulations and experiments to validate our modeling.

A. Router Operation and Power Modeling

A link between two routers is physically connected with two line cards, and the line cards consume the majority power of the routers [9]. We thus use *link power consumption* to abstract the power consumption of the line cards.



2.5G×4
2.5G×4
10G
10G
10G
The shortest path between a and c
The green path between a and c

Fig. 1: An example of green routing with trunk links.

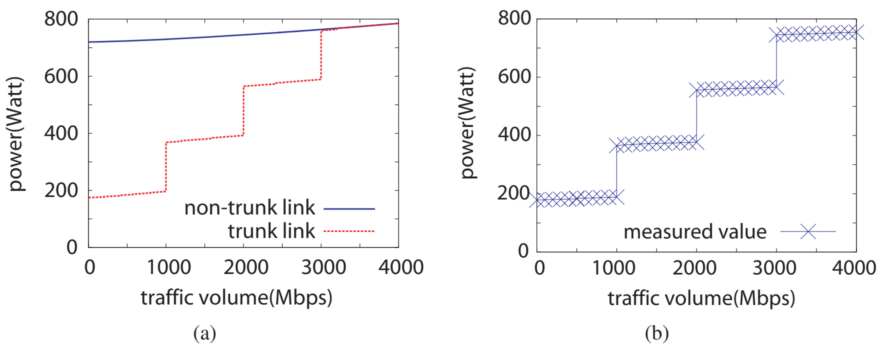


Fig. 2: The link power model. (a) The traffic-power curves defined by Eq. (1) and Eq. (4). (b) The measured traffic-power curve.

We can divide the power consumption into three categories: 1) power consumed by OS and control plane (this is constant to the traffic volume and called the idle power); 2) power consumed by line card CPU processor (this is super-linear to the traffic volume); and 3) power consumed by operations like buffer I/O, packet lookup, etc. (this is usually linear to the traffic volume).

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 below 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[2], all with different power consumption. Similarly, a PCIe bus which connects the chips has the states D0, D1, D2, D3hot and D3cold[3]. Since turning on/off of these components is discrete, we can generally see a discrete stair-like behavior in power consumption (see Fig. 2(a)).

We can classify two types of links: 1) *trunk links* where advanced technologies are adopted and components can be individually turned off, resulting in a stair-like behavior in power consumption and 2) *non-trunk links*. Non-trunk links are still the majority today. Yet, trunk and ALR technologies are witnessing fast development.

We first model the non-trunk links and then the trunk links by including the stair-like behavior.

Note that a logical link in the Internet may consist of several bundled physical links. Let l be a link, and n_l 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 and α_l are constants ($\alpha > 1$) [20]. 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-trunk link, and then we have

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

[2]http://www.intel.com/support/processors/sb/CS-028739.htm

[3]http://www.pcisig.com/specifications/conventional/pcipm1.2.pdf

Here $2n_l$ denotes the fact that the power is consumed by 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 in different traffic volumes; and 2) different components in line cards can be turned-off in different traffic volumes.

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. Then we have:

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

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 i-th state ($0 = \delta_0 < \delta_1 < \cdots < \delta_{n_s-1}$). Then we have

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

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 and we have

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

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 the illustration purpose, we plot numerical examples in Fig. 2(a). We set the link to consist of four 1 Gbps 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 the suggested values in [9] and [20]. 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. We obtain these thresholds from our experiments.

We see that for a non-trunk link, the power consumption is slightly super-linear to the traffic volume. For a trunk link, the power consumption shows a much bigger difference and a discrete stair-like behavior. This means that smaller traffic volume leads to more energy conservation for a trunk link if appropriately managed. We further validate this power model with experiments using a real commercial router.

We set up the experiment by generating packets of 64 bytes with a PC and sending the packets to a commercial BitEngine12000 router[4], through four 1 Gbps ethernet links. The traffic volume varies from 1 Mbps to 4000 Mbps. The router has four 4GE line cards and powers on a proper number of line cards to forward the traffic. We measure the power of the 4GE line cards by connecting an AC ammeter with the AC-input power supply circuit. We can read the electric current value from the ammeter. The results are shown in Fig. 2(b). We see that the curve matches our model closely. As an example, when we increase the traffic from 1000 Mbps to 1100 Mbps, the power consumption shows a sharp increase from 210 Watt to 380 Watt.

In this paper, we will focus on the power consumed by traffic. 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 into the current Internet routing architecture.

Formally, a network is modeled as $G(V, E)$, where V denotes the set of nodes and E the set of 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 $\min \sum_{l \in E} P_l(x_l)$, under 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 the total energy consumption. The traffic of different paths collectively increases the utilization ratio of links, and leads to greater 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 that reflect the volume of traffic flowing 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 [21], and 2) the procedure to estimate a traffic matrix from partial data is of high complexity, since the associated optimization problem is non-convex [21].

[4]http://www.bit-way.com/product-a-6.html

Thus, for a hop-by-hop scheme whose complexity is comparable to that of Dijkstra, we design a path weight similar to the path weight used by Dijkstra, where the weight reflects the 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 the routing is not loop-free. We show 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 is $P_{(a,b)}(x_l) = 0.1x_l$, $P_{(a,c)}(x_l) = 0.3x_l$, and $P_{(b,c)}(x_l) = 0.1x_l$ if $x_l < 10$, or $P_{(b,c)}(x_l) = 0.1x_l + 5$ if otherwise.

Given the traffic demand of a source node, assume a hop-by-hop scheme straightforwardly chooses to compute a path weight as the sum of the power consumption of all the links on the path (i.e., the "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 to c will never reach c.

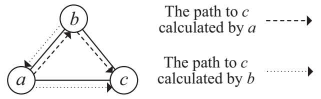


Fig. 3: An example of routing loops.

Intrinsically, to achieve a loop-free routing, there are certain properties that the path weights should follow. A seminal work [19] 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 paths 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 [22]: 1) certain properties need to be satisfied (intrinsically, path concatenation should follow certain properties) and 2) a routing algorithm is designed accordingly. We introduce a few definitions from [22].

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 the 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 the 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 the paths 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 the paths p_1, p_2, q.

We have the following theorems [22][23].

Theorem 4.1: [22] If the weight structure $(S, \oplus, \preceq, w)$ is left-isotonic, for every $s, d \in V$, there exists a 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. If the left-isotonicity is strict, all lightest paths are D-lightest paths.

We can get a consistent (thus loop-free) hop-by-hop routing if every node uses D-lightest paths to forward packets. To compute D-lightest paths, we have:

Theorem 4.2: [23] Dijkstra's algorithm that uses d as the root node is guaranteed to find the D-lightest paths if and only if the path weight structure $(S, \oplus, \preceq, w)$ is strictly left-isotonic.

V. Hop-by-hop Green Routing Algorithms

We now study hop-by-hop green routing (Green-HR). We first propose a path weight and a baseline algorithm Dijkstra-Green-B to achieve loop-free. We then study some intrinsic relationships between link weights and power consumption, and develop an advanced algorithm Dijkstra-Green-Adv that improves energy conservation. We further develop algorithm Dijkstra-Green that concurrently considers energy conservation and path stretch. Finally we present the intrinsic hardness of designing hop-by-hop green routing with QoS requirements.

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.

A preliminary observation is that though we cannot choose the "greenest" paths, for energy conservation from the whole network point of view, we should not choose a path that is too long either, since it accumulatively 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 $p = (s = v_0, v_1, \dots, v_n = d)$, we set a "virtual traffic volume" for each link $l = (v_i, v_{i+1})$, $i \in \{0, 1, \dots, n-1\}$ 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 on 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 $p = (s = v_0, v_1, \dots, v_n = d)$ is the sum of the weight of each link:

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

Here function $\text{Hops}(p)$ returns the hop number of path p. Note that a link's weight is not a static value, and may vary with path p and destination node d. However, we can prove 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 is 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, assume p_1 and p_2 are two paths from node s to node d. Without losing generality, assume that p_1 is lighter than p_2, i.e., $w_b(p_1) \prec w_b(p_2)$. We 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.

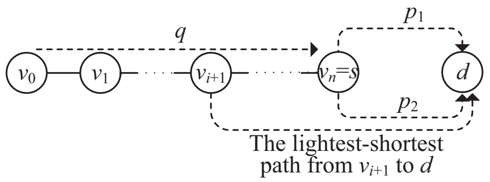


Fig. 4: The topology used to prove the strict left-isotonicity of the path weight structure defined by Eq. (6).

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

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

and

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

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 4.1, 4.2 and 5.1, we can achieve a consistent (thus loop-free) hop-by-hop routing by applying a Dijkstra-like algorithm. 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 that of 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 so the complexity in the best

Algorithm Dijkstra-Green-B()

```
Input: $G(V, E)$, $s$, $d$, $\boldsymbol{P}$, $x_0^v$, $\beta$;
Output: the green path from $s$ to $d$ which is stored in $\varphi[]$;
1: for each node $v \in V$
2:     $w[v] \Leftarrow \infty$; $\varphi[v] \Leftarrow null$; $h[v] \Leftarrow \infty$;
3: $Q \Leftarrow V$; $w[d] \Leftarrow 0$; $h[d] \Leftarrow 0$;
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:         $x \Leftarrow x_0^v \cdot \beta^{h[u]}$;
10:        $\varpi \Leftarrow P_{(v,u)}(x)$;
11:        if $w[u] + \varpi < w[v]$
12:            $\varphi[v] \Leftarrow u$;
13:            $w[v] \Leftarrow w[u] + \varpi$; $h[v] \Leftarrow h[u] + 1$;
14:        else if $w[u] + \varpi = w[v]$ and $h[u] + 1 < h[v]$
15:            $\varphi[v] \Leftarrow u$; $h[v] \Leftarrow h[u] + 1$;
16: return null; //unreachable
```

case is $O(1)$. Thus, we can expect that the average complexity is less than that of Dijkstra.

B. Link Weights vs. Energy Conservation

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

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 rise of the traffic volume. The link weight should reflect this. As a matter of fact, if the traffic volume x_l is proportional 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 that 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 a 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 path stretches is less than the leap of power consumption.

Generally, we take a heuristic by multiplying the weight of a 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 still the starting weight; and r_u and r_d are calculated as follows. Given traffic volume x_l, r_u is the least traffic volume where a leap of power consumption 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 leap of power consumption may occur and $r_d < x_l$, and let $r_d = 0$ if r_d cannot be found. 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 in shortest path routing. The intuition is that if x_0^v is big and/or $r_u - r_d$ is small, a leap of power consumption is likely to happen, and we multiply a bigger penalty k_l in selecting this link l.

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

C. Dijkstra-Green-Adv Algorithm

We design a link weight 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. Second, we scale up the link weight by k_l if link l is a trunk link. 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 an algebra $(S, \oplus, \preceq, w_{adv})$ based on Eq. (8). S is R^+ and $\preceq$ is $\leq$. w_{adv} is given in Eq. (8), and $w_{adv}(p) \oplus w_{adv}(q) = w_{adv}(p \circ q)$, 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 losing 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 prove strict left-isotonicity.

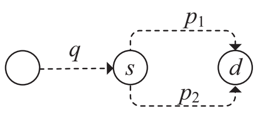


Fig. 5: The topologies used to prove the strict isotonicity of the path weight structure defined by Eq. (8).

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. ■

Based on Theorem 4.1, 4.2 and 5.2, we design an 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

```
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;
```

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 major difference between Dijkstra-Green-Adv and Dijkstra 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 that of the Dijkstra-Green-B algorithm, i.e., $O(|E| + |V| \log |V|)$ in the worst case.

D. Dijkstra-Green Algorithm

We now study the balance between green and normal QoS requirements for the routing paths. In other words, we want to investigate whether the pursuit of green may sacrifice typical routing metrics such as end-to-end delay or bandwidth etc; and how a balance can be made. As an example, we will develop an algorithm that jointly consider green and path length. Note that our pervious algorithms consider path length in the sense to make the paths greener. Here the path length is considered as a separate parameter that reflects end-to-end delay.

Clearly, green paths and shortest paths cannot be simultaneously achieved. A typical metric to evaluate how a computed path differs from shortest path computation is *path stretch*: the ratio of the length of a source-destination path to that of the shortest path between this source-destination pair.

We analyze the path stretch 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 an algorithm which takes additional considerations for the "short" paths. Specifically, let $Len(p)$ be the length of path p. We divide the link length by 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 consumption will dominate for long paths. The weight of path $p = (s = v_0, v_1, \ldots, v_n = d)$ is defined as

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

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. When setting $\kappa = 0$, the weight naturally converge to the weight in Eq. (8) in Section V-C.

We can similarly define an algebra $(S, \oplus, \preceq, w_g)$ according to Eq. (9).

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

Proof: We move the proof to the appendix. ■

Based on Theorem 4.1, 4.2 and 5.3, we develop a loop-free hop-by-hop algorithm named Dijkstra-Green.

Dijkstra-Green is similar to Dijkstra-Green-Adv and we omit the details for the sake of smoothness of presentation. 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 involves path length in the link weight to Eq. (8). 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.

VI. Performance Evaluation

A. Methodology

We evaluate our algorithms using both synthetic and real topologies. For the synthetic topologies, we use BRITE[5] to generate network topologies and we set the parameters following [24]. Each dot in our 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).[6]

The link capacities of the synthetic topologies are determined based on the fact that a node with a big degree is more likely to hold links with a large capacity [25]. We set a link's capacity to 9953.28 Mbps (OC192-1 port) if both end nodes of the link have a degree greater than 5. The capacity is set to 2488.32 Mbps (OC48-1 port) if one end node has a degree larger than 5 and the other has a degree less than 6 but greater than 2. Finally, the other links' capacities are set to 622.08 Mbps (OC12-1 port).

For synthetic topologies and CERNET, we create traffic matrices according to the gravity model [21]. 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 inversely proportional to the square of the hop number of the shortest path from s to d, as shown in Eq. (10)

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

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. We create traffic volumes that result in an average link utilization ratio between 5% and 70%. There are too many links overloaded if we try to create an average link utilization ratio larger than 70%. Such a case rarely happens in the real world even for a heavily-loaded data center network [26]. For Abilene, we use real traffic matrices[7] and one traffic matrix is summarized every five minutes. The traffic volume on an Abilene link is multi-hundred Mbps and the link utilization ratio is around 10%.

[5]http://www.cs.bu.edu/brite/

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

[7]http://www.cs.utexas.edu/%7Eyzhang/research/AbileneTM/

For 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 consists 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 consumption per unit traffic volume (ρ_l) of different operation rates is set as constants, referring to the measurement results given by [9] and [27]. The idle power consumption of different operation rates is calculated using the maximum power[8] and shown in Table I.

TABLE I: Power consumption 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, and β is 1.5. γ is 10 and κ is 0.0004.

We compare our algorithms with shortest path routing. We evaluate the power saving ratio, defined as $(P_s - P_g)/P_s$, where P_s is the total power consumed by line cards under shortest path routing, and P_g is the total power under our algorithms.

There is no similar green routing schemes. We select two recent green routing approaches, i.e., DLF (Distributed Least Flow) [8] and REsPoNse [4]. These schemes put link/node into sleep and our scheme does not need link/node level sleep. In DLF, given that network are connected and capacity are satisfied, the least loaded links are selected to sleep. In REsPoNse, energy-critical paths are computed offline, and used for traffic aggregation. The links that are not in the energy-critical paths are switched into sleep mode. Our scheme is more robust facing failures. To evaluate, we use the total disruption time under single link failures, defined as $\sum_{s,d \in V} t_{sd}$, where t_{sd} denotes the time period during which node s cannot reach node d under the failure.

In addition, we also study the path stretch ratio of our algorithms.

B. Results In Synthetic Topologies

1) Results On Different Traffic Levels: Fig. 6 shows the power saving ratio as against of traditional Dijkstra. We see that the power saving ratio can be as much as 55%, when the average link utilization is low. The power saving ratios decrease when the average link utilization ratio increases. 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 Dijkstra-Green is slightly worse than Dijkstra-Green-Adv, mainly when the network is in high utilization.

[8]http://www.cisco.com/en/US/docs/ios/12%5F0s/feature/guide/12spower.html

Fig. 7 shows the average path stretch of our algorithms. We see that the path stretch is consistent and relatively low under any link utilization ratio. The average path length of Dijkstra-Green-Adv is about 1.22 times to that of the shortest path. We consider such stretch may be a fine value in most cases. The path stretch of Dijkstra-Green is only 1.04. It successfully considers path length when saving energy.

2) Results On Different Trunk Link Ratios: Fig. 8 shows the power saving ratio as a function of trunk link ratio λ. Clearly, the more trunk links are deployed, the more opportunity that the power can be saved. When all the links are trunk links, a 65% power-saving can be achieved.

Fig. 9 shows the path stretch under different trunk link ratios. We can see that Dijkstra-Green-B is not affected by the trunk link ratio. This is because it does not specifically consider trunk links. On the other hand, as we specially consider trunk links in the Dijkstra-Green-Adv design, it is more affected as the number of trunk link increases. The path stretch of Dijkstra-Green also increases when λ increases, but in a much limited scale and is always under 1.05.

3) Results On Different Link Densities: Fig. 10 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. 11 shows the average path stretch increases with the increment of link density. This is because the shortest path has a smaller length when the link density is higher.

C. Results In Real Topologies

Fig. 12 shows the power saving ratio, using the Abilene topology and the traffic matrices collected on March 8, 2004. The results using the data in other time periods are similar. We can see the result changes with time, because the traffic matrix is changing, as shown in Fig. 13. However, the average power saving ratios of the algorithms are always around 57%. The results of the three algorithms are similar. Dijkstra-Green-Adv performs a little worse (within 0.2%) than the other two algorithms. This is because the topology scale is small (only 12 nodes) and less paths can be found. Thus, Dijkstra-Green-Adv may choose a longer path, but no more power is saved.

Fig. 13 shows the link utilization ratio. Dijkstra-Green results in a link utilization ratio very close to that of shortest path routing, while DLF, which chooses the links with the least traffic to sleep, results in a nearly doubled link utilization ratio. This is because our algorithms do not prune links, but successfully save link power at some critical energy waste points, e.g., preventing to leap to a higher "stair". So the traffic will not aggregate excessively.

Our algorithms do not need to turn link/node level component off in the Internet. This is also useful when a failure occurs, as pruning links easily makes the Internet more stressful in connectivity and traffic support. With real traffic,

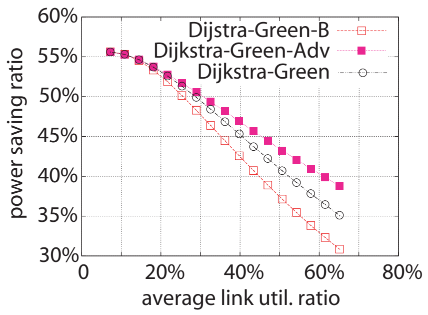


Fig. 6: Power saving ratio as a func. of avg. link util. ratio (synthetic topology).

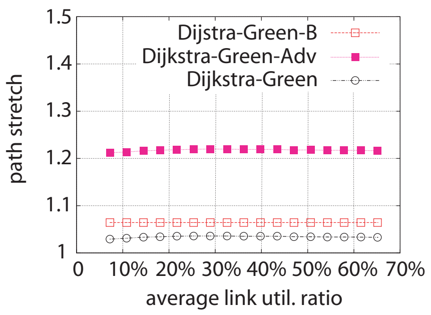


Fig. 7: Avg. path stretch as a function of avg. link util. ratio (synthetic topology).

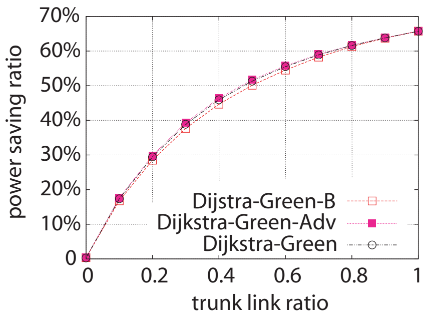


Fig. 8: Power saving ratio as a function of trunk link ratio (synthetic topology).

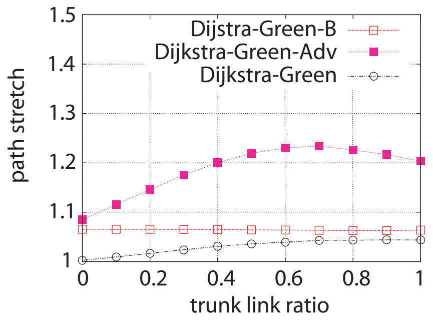


Fig. 9: Avg. path stretch as a function of trunk link ratio (synthetic topology).

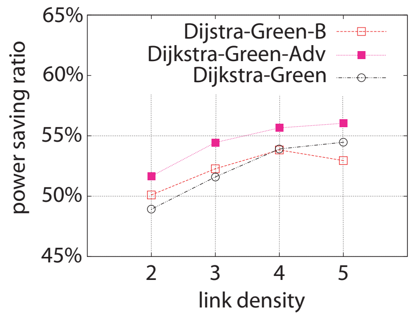


Fig. 10: Power saving ratio as a function of link density (synthetic topology).

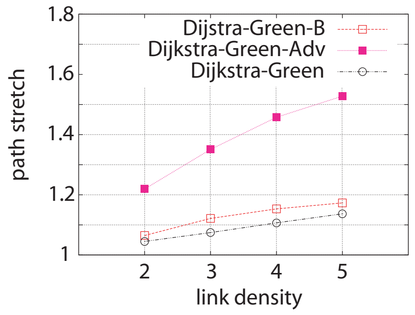


Fig. 11: Avg. path stretch as a function of link density (synthetic topology).

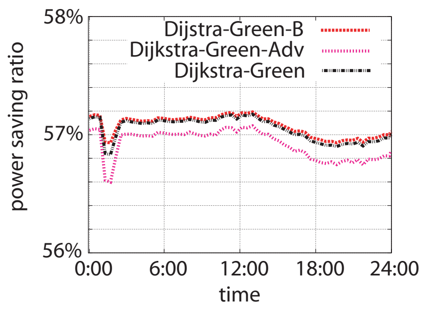


Fig. 12: Power saving ratio as a function of time (Abilene).

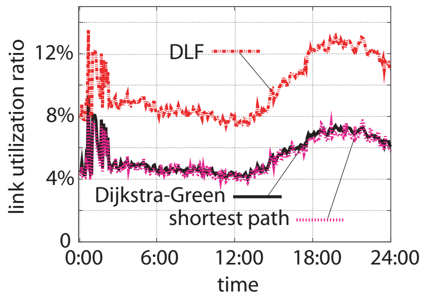


Fig. 13: Avg. link util. ratio as a function of time (Abilene).

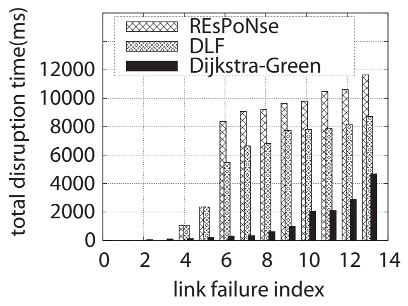


Fig. 14: Total disruption time (Abilene).

we can study the disruption time under single link failures. We compare Dijkstra-Green, DLF and REsPoNse, using the Abilene topology and one traffic matrix on March 8, 2004.

Fig. 14 shows the results. We see that Dijkstra-Green induces much less disruption time than the other two algorithms, and for most link failures the time is less than 2 seconds. This is because we do not prune links from the topology and less source-destination paths are disrupted by a link failure. The disruption is mainly caused by routing inconsistency during convergence. DLF and REsPoNse result in a longer disruption time (3 - 20 times longer) because they both prune links from the topology, and more paths are disrupted by a link failure, until some sleeping links are waked up and the routing is rebuilt. REsPoNse is the worst because a link failure needs to be detected by end nodes. Note that the computation time of DLF and REsPoNse is distributed and reasonable. We conjecture that centralized approaches such as GreenTE [3] may have even longer disruption time.

Fig. 15 shows the power saving ratio as a function of the average link utilization ratio, using the CERNET topology. CERNET is also a small topology with 8 nodes. Therefore, the three algorithms perform similarly. Nevertheless, we still see a 65% of energy saving when the utilization is low and and Dijkstra-Green can save more than 20% of the energy when the utilization is as high as 70%.

VII. Conclusion

In this paper, we studied green Internet routing. We presented a power model that quantifies the relationship between traffic volume and power consumption. We validated our model using real experiments. We proposed a hop-by-hop approach and progressively developed algorithms that guarantee loop-free routing, substantially reduce energy footprint in the Internet, and jointly consider QoS requirements such as path

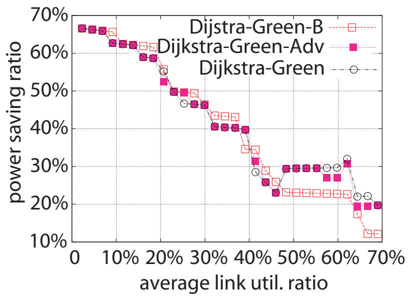


Fig. 15: Power saving ratio (CERNET).

stretch.

As a very first work, we admit that there are many unsolved questions. Especially, we are interested in further investigating a centralized scheme. This is useful when MPLS can be applied, and may provides theoretical bounding for the possible maximum power conservation.

Appendix A Proof of Theorem 5.3

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

Proof: As shown in Fig. 16, suppose p_1 and p_2 are two paths from node s to node d. Without losing generality, we suppose that p_1 is lighter than p_2, i.e., $w_g(p_1) \prec w_g(p_2)$.

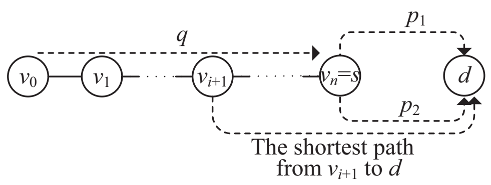


Fig. 16: The topology used to prove the strict left-isotonicity of the path weight structure defined by Eq. (9).

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

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

and

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

Note that the length of the shortest path from v_{i+1} to d is independent of p_1 or p_2. Because $w_g(p_1) \prec w_g(p_2)$ 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. ■

References

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] F. Cuomo, A. Abbagnale, A. Cianfrani, and M. Polverini, “Keeping the Connectivity and Saving the Energy in the Internet,” in *IEEE INFOCOM Workshop on GCN*, April 2011.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] M. Gupta and S. Singh, “Greening of the Internet,” in *ACM SIGCOMM*, 2003.

[14] 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 COMMUNICATIONS SURVEYS & TUTORIALS*, vol. 13, pp. 223–244, 2011.

[15] W. Lu and S. Sahni. Low-Power TCAMs for Very Large Forwarding Tables. *IEEE/ACM ToN*, 18:948–959, 2010.

[16] L. Gan, A. Walid, and S. H. Low. Energy-efficient congestion control. In *Proc. of ACM SIGMETRICS*, pp. 89-100, 2012.

[17] P. Paul and S. V. Raghavan, “Survey of QoS routing,” in *Proc. of the 15th ICCC*, 2002, pp. 50–75.

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

[19] J. L. Sobrinho, “Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet,” *IEEE/ACM TON*, vol. 10, pp. 541–550, 2002.

[20] 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.

[21] M. M. Rahman, S. Saha, U. Chengan, and A. Alfa, “IP Traffic Matrix Estimation Methods: Comparisons and Improvements,” in *Proc. of IEEE ICC*, 2006, pp. 90–96.

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

[23] 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.

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

[25] T. Hirayama, S. Arakawa, S. Hosoki, and M. Murata, “Models of link capacity distribution in ISP's router-level topologies,” *JCNC*, vol. 3, pp. 205–216, 2011.

[26] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in *Proc. of IMC*, pp. 267–280, 2010.

[27] 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.