A Hop-by-Hop Routing Mechanism for Green Internet

Yuan Yang, *Student Member, IEEE*, Mingwei Xu, *Member, IEEE*, Dan Wang, *Member, IEEE*, and Suogang Li

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, substantially reduce energy consumption, and jointly consider green and QoS requirements such as path stretch. We further analyze the power saving ratio, the routing dynamics, and the relationship between hop-by-hop green routing and QoS requirements. We comprehensively evaluate our algorithms through simulations on synthetic, measured, and real topologies, with synthetic and real traffic traces. We show that the power saving in the line cards can be as much as 50 percent.

Index Terms—Energy conservation, internet routing, hop-by-hop routing, routing algebra

1 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, 5,000 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], [9], [10], [11], [12], [13], [14].

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. Therefore, we can select paths that consume less power while delivering traffic. Intrinsically, this is caused by technologies including trunking (or bundled links) [18], [19] and adaptive link rates (ALR) [21], [22]. 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 the 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 power consumption and an additional 0.006 Watt for each 1 Mbps traffic; and for an OC192 link, there is a baseline 134.2 Watt power consumption and an additional 0.004 Watt for each 1 Mbps traffic. In this topology, shortest

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

- *Y. Yang and M. Xu are with Tsinghua National Laboratory for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University. E-mail: yyang@csnet1.cs.tsinghua.edu.cn, xmw@cernet.edu.cn.*
- *D. Wang is with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: csdwang@comp.polyu.edu.hk.*
- *S. Li is with CERNET National Network Center, Beijing, China. E-mail: lisg@cernet.edu.cn.*

Manuscript received 6 Aug. 2014; revised 15 Jan. 2015; accepted 18 Jan. 2015. Date of publication 20 Jan. 2015; date of current version 16 Dec. 2015. Recommended for acceptance by M. Steinder. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPDS.2015.2394794

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 $(125.1 + 2000 \times 0.006) \times 2 + (134.2 + 4000 \times 0.004) \times 3 = 724.8$ Watt, a 25.7 percent improvement.[2]

The above example is not special. In a network that includes many links with trunking or ALR, the energy conservation can be greater than that of topology pruning approaches (See Section 6.1 for more details). An approach without topology pruning can also be used in a network after pruning some links or nodes for further energy conservation. Yet to systematically study this problem, we first need to quantify an appropriate power model, i.e., the relationship between power consumption 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. Such an approach is suitable for a network that deployed MPLS-TE already. We plan a future study in this direction.

In this paper, we instead choose a hop-by-hop approach. Such an approach is suitable for the networks without MPLS deployed. 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 routing 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. Then, we discuss and analyze a few issues related to our approach. We discuss the power saving ratio of our approach and topology pruning approaches, analyze the routing dynamics, 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 percent energy on line cards.

The rest of this paper is organized as follows. Section 2 presents related work. The power model is presented in Section 3. The design outline and properties of routing algebra are discussed in Section 4. Section 5 is devoted into our design of hop-by-hop green routing algorithms. Discussion and analysis are presented in Section 6. Section 7 shows the evaluation and Section 8 concludes the paper.

2 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 [23], [24].

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 [25] of a router.

Second, there are studies on energy conservation of the Internet from upper layers point of view. For example, Energy Efficient TCP [26] 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 underutilized 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. An energy efficient routing scheme that jointly considers admission control is proposed in [5]. These approaches use sleep mode to save energy. Also, there are studies to save energy without sleep mode. They leverage bundled links [18], [19] or speed scaling model [31]. These approaches all need centralized computing and use MPLS to construct routing paths.

For distributed or hop-by-hop energy efficient routing approaches, GreenOSPF [6] is proposed to aggregate traffic and switch the network components into sleep mode. However, to achieve good performance, a centralized algorithm [7] is still needed to assign sleeping links. ESACON [8] is proposed to collaboratively select sleeping links with special connectivity properties. Routing paths are then computed after these links are removed. A set of fully distributed approaches is proposed in [9] and [10], which collects global traffic information and aggregates traffic to switch appropriate network components into sleep mode. There is a set of

2. We realize that there are devices in layers other than the network layer in an Internet backbone, such as optical transport systems, cooling systems, etc.. However, saving energy from such systems is different from saving energy from routers, since the latter can be achieved by carefully designed routing protocols, i.e., green internet routing, while the former may not involve traffic. In this paper, we focus on network layer devices-routers.

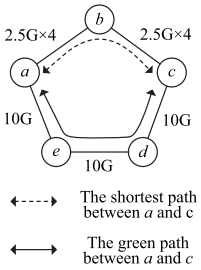


Fig. 1. An example of green routing with trunk links.

other works [11], [12], [13], [14] with various considerations. These works use sleep mode to save energy. A hop-by-hop approach that does not use sleep mode is proposed in [15]. The approach considers a power model similar to the speed scaling model, which does not capture the properties of trunking. Further, the routing protocol needs to be changed largely, e.g., multipath needs to be supported.

Our approach falls into the third category discussed above, yet it differs from the aforementioned schemes in the following aspects. First, most previous proposals switch network devices or links into sleep mode and prune the network topology. Our design is based on the observation that the energy consumption of a link can be dependent on the traffic volume. A routing algorithm may take this into consideration. 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 note that using sleep mode for energy conservation is preferred in certain scenarios, specifically, in networks where the link power changes little with the traffic volume, such as a data center network [16]. However, techniques of trunking and ALR are deployed more and more in the Internet, and there are increasing efforts recently in developing power-proportional routers [20]. Existing approaches for such a situation are centralized [18], [19], [31] or change current routing protocols greatly [15]. Thus, a hop-by-hop and practical approach is needed, and this motivates our work.

We may consider green as one type of services that the Internet should provision. There are many studies on Internet Quality of Service (QoS) [27]. There were two different approaches in Internet QoS support beyond shortest path routing. One is centralized computation [28]. 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 [29]. A seminal paper [29] 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.

3 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.

3.1 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 [17]. We thus use *link power consumption* to abstract the power consumption of the line cards.

We can classify two types of links: 1) *trunk links* where advanced technologies are adopted and line cards in a logical link can be individually turned off, resulting in a stair-like behavior in power consumption and 2) *non-trunk links*. We first model the non-trunk links.

1) Non-trunk links. 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 linear to the traffic volume).

Note that a logical link in the Internet may consist of several bundled physical links. Formally, let l be a logical link, and n_l the number of physical links. Let x_l be the traffic volume on this logical 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 second category (i.e., CPU, super-linear) on each physical link can be modeled as $\mu_l(\frac{x_l}{n_l})^{\alpha_l}$, where μ_l and α_l are constants ($\alpha > 1$) [32]. The power consumption for the third 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

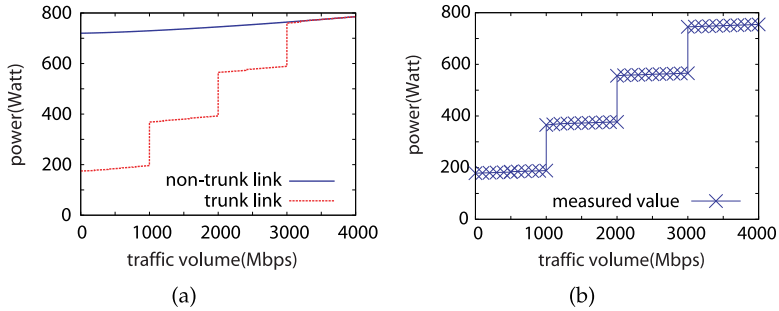


Fig. 2. The link power model. (a) The power-traffic curves defined by Eq. (1) and Eq. (4). (b) The measured power-traffic curve.

consumption of a non-trunk link, and then we have

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

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

2) Trunk links. 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. 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,[3] all with different power consumption. Similarly, a PCIe bus which connects the chips has the states D0, D1, D2, D3hot and D3cold.[4] Turning on/off of these components is discrete in general.

Formally, 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

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

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

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

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 and we have

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

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

3.2 Simulation and Experimental Validation

Eqs. (1) and (4) are abstract. For the illustration purpose, we plot numerical examples in Fig. 2a. 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 [17] and [30]. We set r_0, r_1, r_2, r_3, r_4 to $0, 1{,}000, 2{,}000, 3{,}000, 4{,}000$ Mbps. We assume that there are five 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 $0, 200, 400, 600, 800, 1{,}000$ 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. Another observation is that the power consumption changes little when the line card components change power state; the slope of each step of the trunk link curve is similar to the slope of the non-trunk link curve. This is because the idle power of a line card is a dominant factor in current stage. 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,[5] through four 1 Gbps ethernet links. The traffic volume varies from 1 to 4,000 Mbps. The router has four 4 GE line cards and powers on a proper number of line cards to forward the traffic. We measure the power of the 4 GE 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. 2b. We see that the curve matches our model closely. As an example, when we increase the traffic from 1,000 to 1,100 Mbps, the power consumption shows a sharp increase from 210 to 380 Watt.

The power model we proposed is based on analysis and measurements on real routers. Similar results are reported in a recent independent work [33]. The main difference we made is the stair-like behavior when line cards in a trunk link can be switched off individually. Again, we emphasize that we focus on network layer devices (routers) in this paper. Although routers made by different vendors have different power consumptions, we believe that the stair-like relationship between power consumption and traffic holds for modern routers that operate in a modular fashion.

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

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

4 Overview and Preliminaries

The objective of 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 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 $(v_0 = s, v_1, v_2, \ldots, v_n = d)$,

3. http://www.intel.com/support/processors/sb/CS-028739.htm
4. http://www.pcisig.com/specifications/conventional/pcipm1.2.pdf
5. http://www.bit-way.com/product-a-6.html

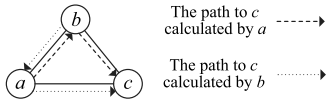


Fig. 3. An example of routing loops.

where $(v_i, v_{i+1}) \in E$ for $0 \leq i < n$. Following Section 3, 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 (s-d) 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 the smallest energy consumption) for each s-d 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 [34], and 2) the procedure to estimate a traffic matrix from partial data is of high complexity, since the associated optimization problem is non-convex [34].

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 the hop-by-hop routing 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 five 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.

Intrinsically, to achieve a loop-free routing, there are certain properties that the path weights should follow. A seminal work [29] first explained the properties through a routing algebra model. Here we briefly present the background.

A routing algebra (or a path weight structure) is defined as quadruplet $(S, \oplus, \preceq, w)$. S denotes the set of path weights, $\oplus$ a binary operation upon the path weights, $\preceq$ an order relation to compare two path weights, and w is a function mapping 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 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 with 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 hop-by-hop routing loops [35]: 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 [35].

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 [35], [36].

Theorem 4.1. *[35] 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. *[36] 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.*

5 Green-HR 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.

5.1 Dijkstra-Green-B Algorithm

From Section 4, 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

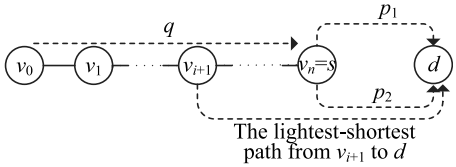


Fig. 4. The topology used to prove the strict left-isotonicity of the path weight structure defined by Eq. (6).

path that is too long either, since it accumulatively consumes more energy.

We thus set the weight as follows. For each link in the path to destination node d, we assign an estimated traffic volume or "virtual traffic volume". We compute the virtual traffic volume by posing an exponential penalty to a start traffic volume for each additional hop. Then, with the virtual traffic volume, the link power is computed following the power function in Section 3. Formally, for each destination node d, we assign x_0^v as the start traffic volume. This x_0^v is determined by the total bandwidth associated with d and we will specify this later. For a path $p = (s = v_0, \ldots, v_n = d)$, the virtual traffic volume for each link $l = (v_i, v_{i+1})$ $i \in \{0, 1, \ldots, n-1\}$ is set 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 $p_{ls}(v_{i+1}, d)$. Here, the lightest-shortest path $p_{ls}(v_{i+1}, d)$ means the path from v_{i+1} to d that has the lightest path weight and the least number of hops. The weight of link l is set to $P_l(x_l^v)$, where $P_l(\cdot)$ follows the power function in Section 3. The weight of path $p = (s = v_0, v_1, \ldots, v_n = d)$ is the sum of the weight of each link:

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

Here function $\mathrm{Hops}(p)$ 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.

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 Theorems 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 power-traffic 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 Steps 9 and 10.

Algorithm 1. 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 $v \in V$
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: $u \Leftarrow \mathrm{Extract\_Min}(Q)$;
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; //d is unreachable

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 case is $O(1)$. Note that Dijkstra-Green-B computes the routing for one destination. However, routings of different destinations are independent from each other, so parallel processing can be used to accelerate the computing. We can expect that the running time is less than that of Dijkstra.

5.2 Link Weights vs. Energy Conservation

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

5.2.1 Link Weights vs. Power per Unit Traffic Volume

Recall the power-traffic functions (Eq. (1) and Eq. (4)) in Section 3. The power consumption of a link increases

with the rise of the traffic volume. The link weight should reflect this. We consider an extreme case that the power consumption is proportional to traffic volume x_l. Such an assumption is a special case of our power model. Though the assumption is ideal, it is consistent with the trend of developing power-proportional routers [20]. We have the following lemma.

Lemma 5.1. *If $P_l(x_l) = \rho_l x_l$ ($\forall l \in E$), 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.

5.2.2 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. Generally, we take a heuristic by multiplying the weight of a trunk link with a factor. However, the factor for different trunk links should not be the same. On one hand, if we put a big traffic volume on a trunk link, the power consumption is likely to leap to a higher stair. In this case, we need a big factor for the link, such that the traffic volume can be reduced. On the other hand, if a small traffic volume can cause the power consumption to leap to a higher stair, we also need a big factor for the link. Formally, we define the factor k_l as

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

where γ is used to balance the link weights of non-trunk links and trunk links; x_0^v is still the starting virtual traffic volume; and $r_u - r_d$ is the traffic volume that can make the link power consumption leap to a higher stair. 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. 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 3), 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.

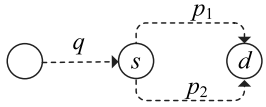


Fig. 5. The topologies used to prove the strict isotonicity of the path weight structure defined by Eq. (8).

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.

5.3 Dijkstra-Green-Adv Algorithm

Now we develop the Dijkstra-Green-Adv algorithm based on the improvements above. Note that Dijkstra-Green-B in Section 5.1 is a baseline algorithm, which focuses on loop-free hop-by-hop routing while considering energy conservation. Dijkstra-Green-Adv focuses on achieving more energy conservation, and follows the principles of Dijkstra-Green-B to guarantee loop-free routing.

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:

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

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.

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 Theorems 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.

Algorithm 2. 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 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**;

The algorithm makes only a few modifications to the standard Dijkstra's algorithm. New inputs include the set of power-traffic 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 major difference between Dijkstra-Green-Adv and Dijkstra is that Dijkstra-Green-Adv calculates the link weight of (u,v) in Steps 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.

5.4 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 previous 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, the green paths and the shortest paths cannot be simultaneously achieved. A typical metric to evaluate how a computed path differs from shortest path is *path stretch*: the ratio of the length of an s-d path to that of the shortest path between this s-d pair.

We analyze the path stretch of $w_{adv}(p)$ and find that the path stretch is small for most 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 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 5.3.

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.*

The proof is very similar to that of Theorem 5.1 and we omit the detail due to page limit. Based on Theorems 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. New inputs include 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.

6 Discussion and Analysis

Now we will discuss and analyze a few issues that are important to hop-by-hop green routing. First, we discuss the bounds on the the optimal power saving without topology pruning, and the power saving ratio that Green-HR can achieve. Also, we compare the power saving ratio of Green-HR with that of topology pruning approaches. Second, we discuss the routing dynamics of Green-HR, and show that routing oscillations and transient micro-loops can be avoided. Third, we study the relationship between Green-HR and QoS requirements, and show that it is impossible to find a strictly left-isotonic path weight structure optimizing one path weight while bounding another, due to the intrinsic nature of routing algebra.

6.1 Power Saving Ratio of Green-HR

The hop-by-hop green routing approach saves energy by leveraging trunk links, without any topology pruning. The problem of maximizing the power saving with trunk links using MPLS-like routing has been shown to be NP-hard [18]. Green-HR uses heuristics to solve the problem, and our main concern is to develop an OSPF-like routing which requires loop-free hop-by-hop forwarding and Dijkstra-oriented computing. Thus, there is no guarantee on the optimality of power saving. However, we can still do some analysis.

Without topology pruning, the power saving that can be achieved is related to the traffic load. In the case when the network is heavy-loaded, all links consume the maximum power and no power can be saved. Clearly, Green-HR will not induce more power consumption in this case. In the case when the traffic is extremely small, each trunk link has only one physical link powered on. Green-HR can achieve such a power saving even when the traffic is larger, because our path weights tends to put less traffic on the trunk links.

These two cases show the lower and the upper bounds of the power saving.

Formally, we consider a simplified model. Without loss of generality, let λ be the ratio of the trunk link number to the total link number. Each trunk link consists of m parallel physical links and each physical link consumes the same power. The upper bound of optimal power saving ratio without topology pruning can be presented by

$$\frac{\lambda|E|(m-1)}{\lambda|E|m+(\lambda-1)|E|}=\frac{\lambda(m-1)}{\lambda m+\lambda-1}. \tag{10}$$

Generally, a larger λ and/or m results in more power saving. In an extreme case when all links are trunk links, i.e., $\lambda=1$, the upper bound is $\frac{m-1}{m}$.

We consider the relation between Green-HR and topology pruning approaches. Even after some links or nodes are pruned, Green-HR can still be used on the remaining topology for further energy conservation. However, it is interesting to ask whether Green-HR saves more energy or less than topology pruning approaches. We do a comparison using the simplified model above. Let σ be the average number of powered on physical links in a trunk link with Green-HR ($\sigma \leq m$), and σ' be the number in the pruned topology ($\sigma < \sigma' \leq m$). We can derive the condition under which Green-HR can save more power

$$\lambda|E| > \frac{\sigma'}{\sigma'-\sigma}(|E|-|V|+1). \tag{11}$$

Generally, in networks that have a big λ and a small σ/σ', Green-HR may outperform topology pruning approaches. In the Internet, trunk links are deployed more and more, which is a common method to upgrade the network capacity. For example, in the China Education and Research Network (CERNET) backbone, 9 out of 12 links are trunk links. Thus, Green-HR is expected to have a high power saving ratio. We will show the simulation results of both types of approaches in Section 7.

6.2 Routing Dynamics

The traffic in a network changes much frequently than the topology does. This may lead to frequent routing computations in Green-HR, which may incur routing oscillations. Furthermore, transient routing micro-loops [37] may be incurred. Such loops may only be induced during the process of routing convergence, and are different from that induced by a link weight structure which is not isotonic. It is natural to discuss such routing dynamics of Green-HR.

Routing oscillations may be incurred when the traffic on a path is affected by a routing computation, and this traffic change in turn affects the path weight and triggers another routing computation. We show that Green-HR does not have such a situation. For Dijkstra-Green-B, the path weight is determined by a virtual traffic volume x_0^v that is only associated to destination node d, so a routing change will not affect x_0^v. For Dijkstra-Green-Adv and Dijkstra-Green, in addition to x_0^v, the historical traffic volume estimation for link l, i,e., $\bar{x}_l$, is also used. However, we can use the average traffic volume of a long period to weaken the affection of current routing change and avoid a routing oscillation. For example, we can use the average traffic volume of the same period (e.g., an hour) in the past seven days, based on the fact that the traffic has a diurnal pattern.

Transient routing micro-loops may be incurred if the routers have inconsistent routing information [37]. Such an inconsistency usually happens during routing convergence, when part of the routers have received link state advertisements (LSAs) that announce topology changes or TE-LSAs [38] that announce traffic changes. However, such a situation can be avoided in Green-HR, by controlling the order of routing re-computations. This method is feasible because the routing computation for energy conservation is not a time-critical mission and can be performed with delay. We develop the control algorithm, namely AvoidTL.

Algorithm 3. Algorithm AvoidTL()

Input: $G(V,E)$, local v, destination d, current sink tree T_d rooted at d computed by Dijkstra-Green(-B/-Adv)
1: $A_d \Leftarrow null$;
2: **for** each node u that is a neighbor of v
3: **if** link $(u,v) \in T$
4: $A_d \Leftarrow A_d \cup \{u\}$;
5: $u \Leftarrow$ the next hop of v in T_d;
6: **wait until** each node in A_d has completed recomputing and notified v;
7: **call** Dijkstra-Green(-B/-Adv);
8: notify node u;
9: **return**;

Specifically, the path to destination node d can be computed by node v (Step 7) if the following condition is met: for each node u that is a neighbor of v and the original path from u to d traverses v (Steps 1 to 4), u has computed the new path to d (Step 6). This can be implemented by a little modification to the routing protocol. We have the following lemma.

Lemma 6.1. *No micro-loop can be incurred by Green-HR with AvoidTL.*

Proof. We prove the lemma by contradiction. Assume that there is a micro-loop between nodes v_i and v_j, i.e., nodes v_i and v_j use each other as the next hop to destination d. Then one of v_i and v_j has computed the new path, and the other has not. This is because the path weight structures in Green-HR are strictly left-isotonic, and if both or none of v_i and v_j have computed the new paths, there will be no loop between them.

Without losing generality, assume v_i is the one that computed the new path, and v_j is using the original path. According to the AvoidTL algorithm, because the next hop of v_j is v_i in v_j's original path, v_j must have computed the new path. This is a contradiction. □

Lemma 6.1 implies that Green-HR can avoid transient routing micro-loops by AvoidTL.

6.3 Green-HR and QoS Requirements

In Section 5.4, we design the Dijkstra-Green algorithm which considers energy conservation and path stretch

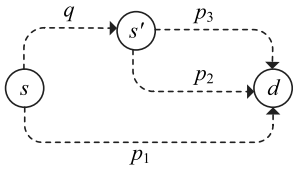


Fig. 6. An example used to prove Lemma 6.2.

concurrently. Naturally, one may hope for a stronger objective, i.e., find the paths that minimize the path weight reflecting the power consumption, and bound the path weights reflecting some QoS parameters. However, we find that for independent path weights, we cannot design a combined isotonic path weight structure, such that one path weight is minimized while others are bounded. This is due to the intrinsic nature of routing algebra. We conjecture that to achieve a min-bound objective, we have to search for the solution with centralized routing control, e.g., using MPLS; and we leave this into future work.

We show this negative result even with one QoS parameter. Formally, consider a path p from s to d. Let $w_1(p)$ be the path weight of p reflecting the QoS parameter, and $w_2(p)$ be the path weight reflecting the power consumption, which is independent of $w_1(p)$. Let p_l be the path with the lightest QoS weight $w_1(p_l)$. Given a threshold ζ, let our objective be finding an s-d path p^*, such that for any s-d path p, we have $w_2(p^*) \preceq w_2(p)$ and $w_1(p^*) \preceq w_1(p_l) \oplus \zeta$.

Lemma 6.2 shows that this is impossible.

Lemma 6.2. *Given two independent isotonic path weight structures $(S_1, \oplus_1, \preceq_1, w_1)$ and $(S_2, \oplus_2, \preceq_2, w_2)$, define path weight structure $(S, \oplus, \preceq, w)$ as follows: s-d path p^* 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 s-d path for $(S_1, \oplus_1, \preceq_1, w_1)$; (2) $w_2(p^*) \preceq_2 w_2(p)$, where p is an s-d path and $w_1(p) \preceq_1 w_1(p_l) \oplus_1 \zeta$. Then, such $(S, \oplus, \preceq, w)$ is not isotonic.*

Proof. We prove the lemma by contradiction. Assume that $(S, \oplus, \preceq, w)$ is isotonic. According to Theorem 4.1, there exists at least one lightest path from s to d such that all its subpaths to destination d are also lightest paths. Such a lightest path is called a D-lightest path. 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 $w_1(p_1) = w_1(p_2)$, $w_1(q) = \zeta$, $w_1(p_3) \preceq_1 w_1(p_2) \oplus_1 \zeta$, 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 6.2. 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 6.2.

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 $w_2(p_3) \prec_2 w_2(p_2)$. Thus, the lightest path from s to d is not a D-lightest path, and we conclude that $(S, \oplus, \preceq, w)$ is not isotonic. □

Lemma 6.2 implies the limit that we have reached by algorithm Dijkstra-Green.

TABLE 1
The Rocketfuel Topologies

As No.	name	No. of nodes	No. of links
1,221	ASN-TELSTRA	104	153
1,239	SprintLink	315	972
1,755	STUPI AB	87	161
3,257	TINET-BACKBONE	161	328
3,967	Nationstar Mortgage	79	147
6,461	Metromedia	138	374

7 Performance Evaluation

7.1 Methodology

We evaluate our algorithms using synthetic, measured, and real topologies. First, we use BRITE[6] to generate synthetic network topologies, and we set the parameters following [39], which captures certain properties of real networks, such as the power law node degree. Each synthetic topology has 100 nodes, and the link density changes from 2 to 5 (i.e., 200 to 500 links). Each dot in our figures is an average on 1,000 random and independent synthetic topologies. Second, we use the six measured intra-domain topologies provided by the Rocketfuel project [40], as shown in Table 1. Third, we have two real topologies: 1) the Abilene backbone with 12 nodes and 15 two-directional links,[7] and 2) the China Education and Research Network backbone, with 8 nodes and 12 links (9 links are trunk links).[8]

The link capacities of the synthetic and measured topologies are determined based on the fact that a node with a big degree is likely to hold links with a large capacity [41]. We set a link's capacity to 9,953.28 Mbps (OC192-1 port) if both end nodes of the link have a degree greater than 5. The capacity is set to 2,488.32 Mbps (OC48-1 port) if one end node has a degree greater 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 the synthetic topologies, the measured topologies, and CERNET, we create traffic matrices according to the gravity model [34]. 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. (12)

$$f(s,d) = \frac{\eta \cdot \sum_{v \in N(s)} c(s,v) \cdot \sum_{u \in N(d)} c(u,d)}{\left(\mathrm{Hops}(p_s(s,d))\right)^2}, \qquad (12)$$

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 percent. There are too many links overloaded if we try to create an average link utilization ratio greater than 70 percent. Such a

6. http://www.cs.bu.edu/brite/

7. We do not use the topology of Internet2 which superseded Abilene, because we use real traffic matrices which were measured in Abilene, and we do not find available data for Internet2.

8. We remove the stub nodes, which have only one link to the backbone.

TABLE 2
Power Consumption of Line Cards

line card (1-port)	operation rate(Mbps)	maximum power(W)	ρ_l (W/Mbps)	calculated idle power(W)
OC3	155.52	60	0.01	58.4
OC12	622.08	80	0.008	75.0
OC48	2,488.32	140	0.006	125.1
OC192	9,953.28	174	0.004	134.2

case rarely happens in the real world even for a heavily-loaded data center network [42]. For Abilene, we use real traffic matrices[9] 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 percent.[10]

For the synthetic and measured 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 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 2. The power consumption per unit traffic volume (ρ_l) of different operation rates is set as constants, referring to the measurement results given by [17] and [43]. The idle power consumption of different operation rates is calculated using the maximum power[11] and shown in Table 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 percent. 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.

For comparison, we select three recent green routing approaches, i.e., Distributed Least Flow (DLF) [9], Distributed Most Power (DMF) [9], and REsPoNse [4]. DLF and DMF are also OSPF-based approaches, while REsPoNse uses MPLS.[12] These schemes put link/node into sleep mode and our scheme does not need link/node level sleep. In DLF, the least loaded links are selected to sleep if the resulting topology is connected and the traffic volume is less than the link capacity. DMP is the same as DLF except that the most power hungry links are selected to sleep. DLF and DMP are simulated because they can switch the maximum number of links into sleep mode when the traffic is small enough. 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 use the total disruption time under traffic bursts to evaluate the transient micro-loops during routing convergence of Green-HR. We do this by set $x_0^v(d)$ to a large value. We also study the path stretch ratio of the algorithms. We compute the power saving ratio and path stretch ratio statically, while we write a discrete event simulator program to measure the total disruption time.

7.2 Results in Synthetic Topologies

7.2.1 Results on Different Traffic Levels

Fig. 7 shows the power saving ratio as against of traditional Dijkstra. We see that the power saving ratio of Green-HR can be as much as 55 percent, 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 percent when the average link utilization ratio is 65 percent. 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. DLF and DMP save more power than Green-HR when the network is underutilized, but they save less power than Green-HR when the average link utilization is larger than 25 percent. This is because in a network with a large traffic, the link utilization will increase largely when switching links into sleep mode. In this case, Green-HR can save more power than topology pruning approaches.

Fig. 8 shows the average path stretch of the algorithms. We see that the path stretch of Green-HR 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 a stretch to 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. The average path stretches of DLF and DMP are much larger than those of Green-HR when the link utilization is low. For example, the path length of DLF is two times to that of the shortest path when the link utilization ratio is 10 percent. This is because packets have to be delivered in a long path to detour around the sleeping links. The path stretches of DLF and DMP decreases with the increment of the link utilization, because less links can be switched into sleep mode when the traffic is larger, since a link may be overloaded easily. This also implies that Green-HR has a more refined control on routing than topology pruning approaches.

7.2.2 Results on Different Trunk Link Ratios

Fig. 9 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 percent power-saving can be achieved by Green-HR. We also see that Green-HR saves more power than DLF and DMP when λ is larger than 0.6, which is consistent with our analysis in Section 6.1, i.e., Ineq. (11).

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

10. Note that the traffic matrices are used as inputs of our simulations, but not inputs of our routing algorithms.

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

12. We do not choose more approaches such as [18] and [19] since most of them need centralized computing and MPLS, whose complexity is incomparable with OSPF-based approaches.

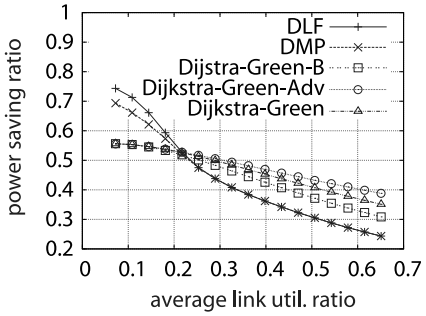


Fig. 7. Power saving ratio as a func. of avg. link util. ratio (synthetic topo.).

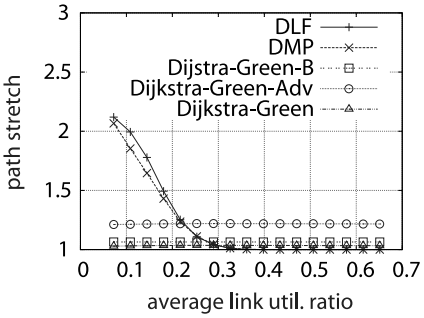


Fig. 8. Path stretch as a function of avg. link util. ratio (synthetic topo.).

Fig. 10 shows the path stretch under different trunk link ratios. We can see that the path stretch of Dijkstra-Green-B is not affected by the trunk link ratio. This is because Dijkstra-Green-B does not specifically consider trunk links. On the other hand, as we specially consider trunk links in the Dijkstra-Green-Adv design, the path stretch 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. The path stretches of DLF and DMP are larger than those of Green-HR. The result of DMP is affected by the trunk link ratio, while the result of DLF is not. This is because when selecting sleeping links, DMP prefers a link with larger power (i.e. a trunk link), while DLF only considers the traffic.

7.2.3 Results on Different Link Densities

Fig. 11 shows the power saving ratio as a function of link density. We find that the larger the link density is, the more the power is saved. This is not surprising as there are more trunk links when the link density is larger. DLF and DMP save less power than Green-HR in networks with a small link density and save more in networks with a large link density. This is because a network with a small link density has less links to be pruned, and Green-HR can outperform topology pruning approaches. Fig. 12 shows that 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.

7.3 Results In Measured Topologies

Fig. 13 shows the average power saving ratios in the Rocketfuel topologies. Dijkstra-Green saves about 55 percent of the power in all the six topologies, which implies that Green-HR has a good scalability in networks with different sizes. DLF saves more power because the synthetic traffic is small. However, the difference is small. We consider the power saving ratio of Green-HR to be fine, since DLF induces larger path stretches and degrades the network resilience against failures, as presented below. DMF has similar results as DLF in the measured and the real topologies, so we do not plot the results.

Fig. 14 shows the average path stretches in the Rocketfuel topologies. We can see that Dijkstra-Green induces a path stretch between 1.05 and 1.35, much less than that of DLF, which can be larger than 2 in AS1755 and AS3257. We also

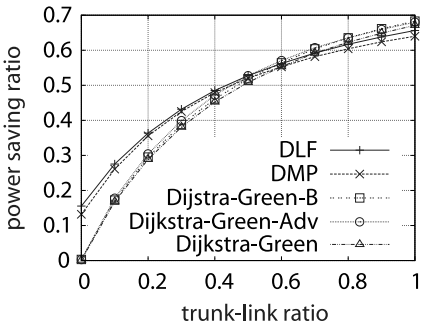


Fig. 9. Power saving ratio as a func. of trunk link ratio (synthetic topo.).

find that the difference between the path stretch of Dijkstra-Green and that of DLF is not the same, which is less in AS1239 and AS6461 than that in AS1221, AS1755, AS3257, and AS3967. This is because the link density of AS1239 and AS6461 is larger, i.e., about 3 (please refer to Table 1), and thus shorter paths can be found when DLF switches links into sleep mode.

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. We compare Dijkstra-Green with DLF, and study the disruption time under single link failures in the AS1239 (Sprint) topology, which is the largest among our topologies.

Fig. 15 shows the results. The x-axis is the index of the failed link, sorted in increasing order according to the corresponding disruption time. We see that a few link failures induce the most disruption time. Dijkstra-Green induces much less disruption time than DLF, and for 99.2 percent of the link failures the time is less than 5 seconds. This is because we do not prune links from the topology and less s-d paths are disrupted by a link failure. DLF results in a longer disruption time (13.5 times longer on average) because it prunes 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. Note that DLF is distributed and the computation time is reasonable. We conjecture that centralized approaches such as GreenTE [3] may have even longer disruption time.

7.4 Results in Real Topologies

Fig. 16 shows the power saving ratio, using the Abilene topology and the traffic matrices collected on March 8, 2004.

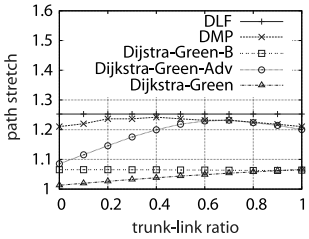


Fig. 10. Path stretch as a function of trunk link ratio (synthetic topo.).

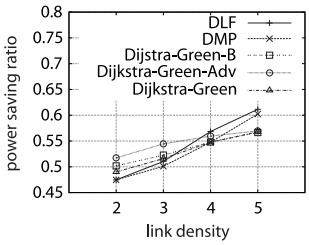


Fig. 11. Power saving ratio as a function of link density (synthetic topo.).

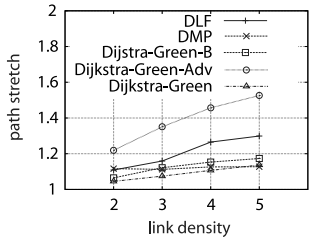


Fig. 12. Avg. path stretch as a function of link density (synthetic topology).

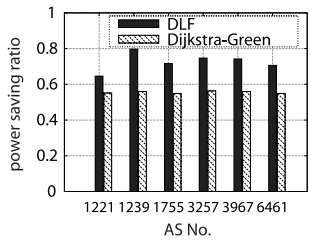


Fig. 13. Power saving ratios in the Rocketfuel topologies.

The results using the data in other time periods are similar. We can see the result changes little with time, i.e. within 1 percent. This is because the traffic matrix is changing, but the change is not big enough to power on/off a physical link, as shown in Fig. 17. The average power saving ratios of the three Green-HR algorithms are around 57 percent. The results of the Green-HR algorithms are similar, because the topology scale is small (12 nodes). DLF saves 10 percent more power than Green-HR.

Fig. 17 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.

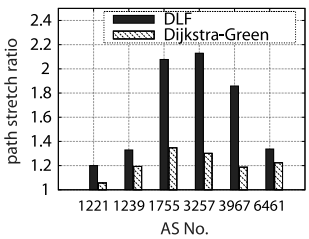


Fig. 14. Path stretches in the Rocketfuel topologies.

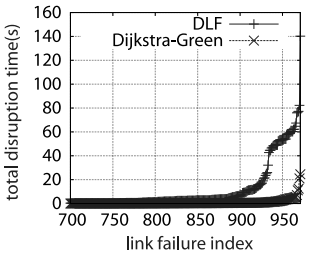


Fig. 15. Total disruption time under single link failures in AS1239.

Fig. 18 shows the average path stretch. All the Green-HR algorithms result in stable path stretches, which implies that Green-HR does not induce routing oscillations, and this confirms our analysis in Section 6.2. Dijkstra-Green has an average path stretch close to 1 as it considers path length when saving energy. The result of DLF is larger, and changes frequently with the traffic, because DLF always tries to switch into sleep mode the links with the least traffic.

With real traffic, we study the total 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. 19 shows the results. Similar to Fig. 15, 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. DLF and REsPoNse result in a longer disruption time (3-20 times longer) because they both prune links. REsPoNse is the

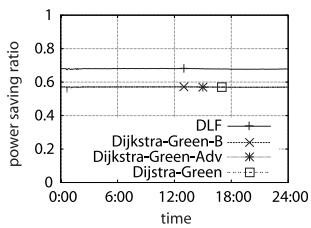


Fig. 16. Power saving ratio as a function of time (Abilene).

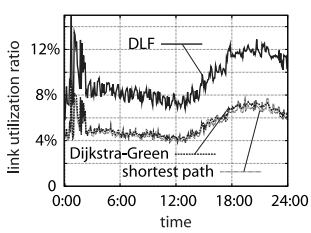


Fig. 17. Avg. link util. ratio as a function of time (Abilene).

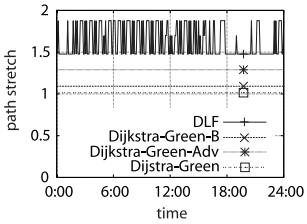


Fig. 18. Path stretch as a function of time (Abilene).

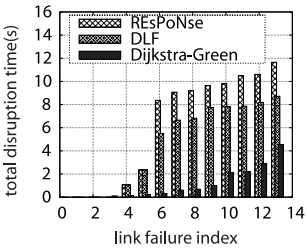


Fig. 19. Total disruption time under single link failures (Abilene).

worst because a link failure needs to be detected by end nodes.

To study the transient micro-loops during routing convergence of Green-HR, we measure the total disruption time of Dijkstra-Green and Dijkstra-Green with AvoidTL, using the Abilene topology and one traffic matrix on March 8, 2004, in which we inject traffic bursts for each node. Fig. 20 shows the results. The disruption time induced by transient micro-loops is less than 2 seconds, which is less than that induced by link failures as shown in Fig. 19. Dijkstra-Green with AvoidTL has a disruption time of 0 because AvoidTL successfully avoids transient micro-loops. We also see that a larger traffic burst induces a longer disruption time. Thus, in networks with few traffic bursts, Green-HR may induce few transient micro-loops even without AvoidTL.

Fig. 21 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 percent of energy saving when the utilization is low and Dijkstra-Green can save more than 20 percent of the energy when the utilization is as high as 70 percent.

8 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 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.

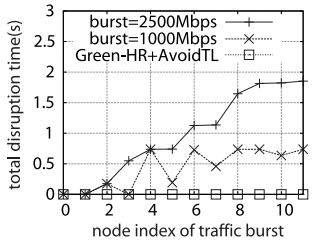


Fig. 20. Total disruption time under traffic bursts (Abilene).

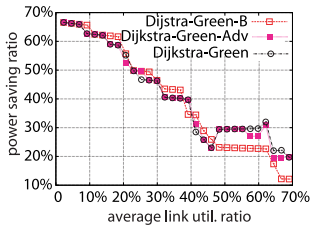


Fig. 21. Power saving ratio (CERNET).

Acknowledgments

The work was supported by the National Basic Research Program of China (973 Program) under Grant 2012CB315803, the National Natural Science Foundation of China under Grant 61133015 and 61161140454, Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20120002110060. Dan Wang's work was supported in part by National Natural Science Foundation of China (No. 61272464), RGC/GRF PolyU 5264/13E, HK PolyU G-YM06, A-PK95, 1-ZVC2. The corresponding author of this paper is Mingwei Xu.

References

[1] J. Carter and K. Rajamani, "Designing energy-efficient servers and data centers," *Computer*, vol. 43, no. 7, pp. 76–78, Jul. 2010.

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

[3] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-Aware Traffic Engineering," in *Proc. IEEE Int. Conf. Netw. Protocol*, Oct. 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. 7th Conf. Emerging Netw. Experiments Technol.*, 2011, p. 18.

[5] S. Avallone and G. Ventre, "Energy efficient online routing of flows with additive constraints," *Comput. Netw.*, vol. 56, no. 10, pp. 2368–2382, 2012.

[6] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An energy saving routing algorithm for a green OSPF protocol," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Mar. 2010, pp. 1–5.

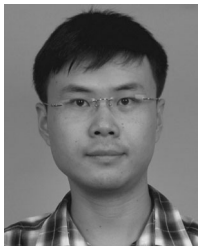
[7] A. Cianfrani, V. Eramo, M. Listanti, and M. Polverini, "An OSPF enhancement for energy saving in IP networks," in *Proc. IEEE Workshop Green Commun. Netw.*, Apr. 2011, pp. 325–330.

[8] F. Cuomo, A. Abbagnale, A. Cianfrani, and M. Polverini, "Keeping the connectivity and saving the energy in the internet," in *Proc. IEEE Conf. Comput. Commun. Workshop GCN*, Apr. 2011, pp. 319–324.

[9] A. P. Bianzino, L. Chiaraviglio, and M. Mellia, "Distributed algorithms for green IP networks," in *Proc. IEEE Conf. Comput. Commun. Workshop Green Netw. Smart Grid*, Mar. 2012, pp. 121–126.

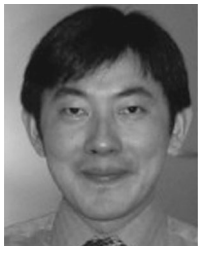
[10] A. P. Bianzino, L. Chiaraviglio, M. Mellia, and J.-L. Rougier, "GRiDA: GReen distributed algorithm for energy-efficient IP backbone networks," *Comput. Netw.*, vol. 56, no. 14, pp. 3219–3232, 2012.

[11] Y. M. Kim, E. J. Lee, H. S. Park, J.-K. Choi, and H.-S. Park, "Ant colony based self-adaptive energy saving routing for energy efficient Internet," *Comput. Netw.*, vol. 56, no. 10, pp. 2343–2354, 2012.
[12] E. Amaldia, A. Caponea, and L. Gianoli, "Energy-aware IP traffic engineering with shortest path routing computer networks," *Comput. Netw.*, vol. 57, no. 6, pp. 1503–1517, 2013.
[13] Y. Yang, M. Xu, and Q. Li. "Towards fast rerouting-based energy efficient routing," *Comput. Netw.*, vol. 70, no. 9, pp. 1–15, 2014.
[14] Q. Li, M. Xu, Y. Yang, L. Gao, Y. Cui, and J. Wu. "Safe and practical energy-efficient detour routing in IP networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 1925–1937, Dec. 2014.
[15] C. Hou, F. Zhang, A. F. Anta, L. Wang, and Z. Liu, "A Hop-by-hop energy efficient distributed routing scheme," *ACM SIGMETRICS Performance Evaluation Review*, vol. 14, no. 3, pp. 101–106, 2013.
[16] M. Xu, Y. Shang, D. Li, and X. Wang, "Greening data center networks with throughput-guaranteed power-aware routing," *Comput. Netw.*, vol. 57, no. 15, pp. 2880–2899, 2013.
[17] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 457–465.
[18] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: Reducing energy consumption by shutting off cables in bundled links," in *Proc. 1st ACM SIGCOMM Workshop Green Netw.*, 2010, pp. 29–34.
[19] G. Lin, S. Soh, K. Chin, and M. Lazarescu, "Efficient heuristics for energy-aware routing in networks with bundled links," *Comput. Netw.*, vol. 57, no. 8, pp. 1774–1788, 2013.
[20] L. Niccolini, G. Iannaccone, S. Ratnasamy, and J. Chandrashekar, "Luigi rizzo, building a power-proportional software router," in *Proc. USENIX Conf Annu. Tech. Conf.*, 2012, p. 8.
[21] 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 J. Opt. Commun. Netw.*, vol. 2, no. 9, pp. 716–729, Sep. 2010.
[22] C. Gunaratne and K. Christensen, "Ethernet adaptive link rate: System design and performance evaluation," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, 2006, pp. 28–35.
[23] M. Gupta and S. Singh, "Greening of the internet," in *Proc. ACM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2003, pp. 19–26.
[24] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Commun. Survey Tuts.*, vol. 13, no. 2, pp. 223–244, Second Quarter 2011.
[25] W. Lu and S. Sahni, "Low-power TCAMs for very large forwarding tables," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 948–959, Jun. 2010.
[26] L. Gan, A. Walid, and S. H. Low, "Energy-efficient congestion control," in *Proc. 12th ACM Joint Int. Conf. Meas. Model. Comput. Syst.*, 2012, pp. 89–100.
[27] P. Paul and S. V. Raghavan, "Survey of QoS routing," in *Proc. 15th Int. Conf. Comput. Commun.*, 2002, pp. 50–75.
[28] F. Ergun, R. K. Sinha, and L. Zhang, "QoS routing with performance-dependent costs," in *Proc. IEEE Conf. Comput. Commun.*, 2000, pp. 137–146.
[29] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 541–550, Aug. 2002.
[30] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *Proc. IEEE Conf. Comput. Commun.*, 2010, pp. 1–9.
[31] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for power minimization in the speed scaling model," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 285–294, Feb. 2012.
[32] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems: Optimality and robustness," *Elsevier Perform. Eval.*, vol. 69, no. 12, pp. 601–622, 2012.
[33] A. Vishwanath, K. Hinton, R. W. A. Ayre, and R. S. Tucker, "Modeling energy consumption in high-capacity routers and switches," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 8, pp. 1524–1532, Aug. 2014.
[34] M. M. Rahman, S. Saha, U. Chengan, and A. Alfa, "IP traffic matrix estimation methods: Comparisons and improvements," in *Proc. IEEE Int. Conf. Commun.*, 2006, pp. 90–96.
[35] Y. Yang and J. Wang, "Design guidelines for routing metrics in multihop wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 1615–1623.
[36] J. Wang and K. Nahrstedt, "Hop-by-hop routing algorithms for premium-class traffic in DiffServ networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 73–88, 2002.
[37] P. Francois and O. Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1280–1292, Dec. 2007.
[38] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (TE) extensions to OSPF Version 2," *IETF RFC3630*, 2003.
[39] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "Generating realistic ISP-level network topologies," *IEEE Commun. Lett.*, vol. 7, no. 6, pp. 335–336, Jul. 2003.
[40] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.
[41] T. Hirayama, S. Arakawa, S. Hosoki, and M. Murata, "Models of link capacity distribution in ISP's router-level topologies," *Int. J. Comput. Netw. Commun.*, vol. 3, pp. 205–216, 2011.
[42] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.
[43] W. Vereecken, W. V. Heddeghem, M. Deruyck, B. Puype, B. Lannoo, W. Joseph, D. Colle, L. Martens, and P. Demeester, "Power consumption in telecommunication networks: Overview and reduction strategies," *IEEE Commun. Mag.*, vol. 49, no. 6, pp. 62–69, Jun. 2011.



Yuan Yang received the BSc and MSc degrees from Tsinghua University. He is currently working toward the PhD degree at the Department of Computer Science & Technology, Tsinghua University. He was a visiting PhD student at the Hong Kong Polytechnic University between 2012 and 2013. His major research interests include distributed routing protocol, computer network architecture and the green Internet. He is a student member of the IEEE.

Mingwei Xu received the BSc and PhD degrees from Tsinghua University. He is a full professor in the Department of Computer Science at Tsinghua University. His research interest includes computer network architecture, high-speed router architecture, and network security. He is a member of the IEEE.



Dan Wang received the BSc degree from Peking University, Beijing, the MSc degree from Case Western Reserve University, Cleveland, OH, and the PhD degree from Simon Fraser University, Vancouver, Canada, all in computer science. He is an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. His research interest includes sensor networks, internet routing and applications. He is a member of the IEEE.

Suogang Li received the BSc and PhD degrees from Tsinghua University. He is a senior engineer in the China Education and Research Network (CERNET) Operation Center. His research interests include computer network architecture and network security.