Safe and Practical Energy-Efficient Detour Routing in IP Networks

Qi Li, *Member, IEEE*, Mingwei Xu, *Member, IEEE*, Yuan Yang, *Member, IEEE*, Lixin Gao, *Fellow, IEEE*, Yong Cui, *Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

***Abstract*—The Internet is generally not energy-efficient since all network devices are running all the time and only a small fraction of consumed power is actually related to traffic forwarding. Existing studies try to detour around links and nodes during traffic forwarding to save powers for energy-efficient routing. However, energy-efficient routing in traditional IP networks is not well addressed. The most challenges within an energy-efficient routing scheme in IP networks lie in *safety* and *practicality*. The scheme should ensure routing stability and loop- and congestion-free packet forwarding, while not requiring modifications in the traditional IP forwarding diagram and shortest-path routing protocols. In this paper, we propose a novel energy-efficient routing approach called safe and practical energy-efficient detour routing (SPEED) for power savings in IP networks. We provide theoretical insight into energy-efficient routing and prove that determining if energy-efficient routing exists is NP-complete. We develop a heuristic in SPEED to maximize pruned links in computing energy-efficient routings. Extensive experimental results show that SPEED significantly saves power consumptions without incurring network congestions using real network topologies and traffic matrices.**

***Index Terms*—Energy efficiency, intrarouting routing, routing.**

I. Introduction

THE REMARKABLE growth of the Internet entailed an exceptional increase of energy consumption of the whole network infrastructure. However, the Internet is generally not energy-efficient, and only a small fraction of consumed power is actually related to traffic forwarding [14]. One observation is that the Internet has time-varying link utilizations [29]. Even when the traffic load is low, network devices remain in active mode and consume a significant amount of energy. Thus, energy saving becomes an important part of networking research, with the most prominent topics pertaining to the energy consumption of network devices, such as putting these devices into sleep mode when appropriate [10], [11], [32].

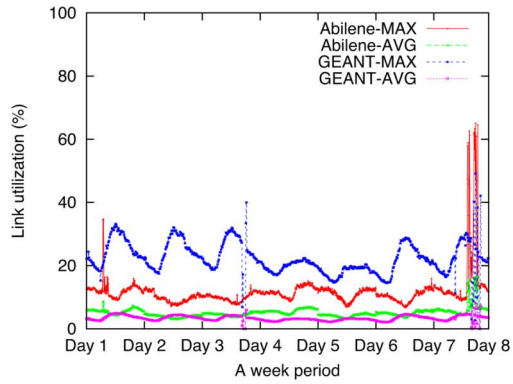


Fig. 1. Maximum and average link utilization in the Abilene and GEANT networks during one-week period. The Abilene traffic matrices are provided by Zhang *et al.* [33], and the traffic is measured every 5 min. GEANT traffic matrices are provided by Uhlig *et al.* [29], and the traffic is measured every 15 min.

To motivate the potential of reducing energy consumption of network devices, we observe that the capacity of a backbone network is generally overprovisioned in order to accommodate traffic shifts and allow rerouting during link failures. The average link utilization of large-scale backbone networks is estimated to be around 30%–40% [11]. Fig. 1 shows the link utilizations of the Abilene and GEANT networks during a week period. We observe that the average link utilizations of both networks during off-peak periods decrease by nearly 100%, and average link utilizations are much lower than the maximum link utilizations, which allow traffic aggregation in few links while still leaving enough spare bandwidth for unexpected traffic bursts. Thus, the energy consumption of the Internet can be greatly reduced if the routers or line cards can be powered off during periods of low utilization [24]. On the other hand, if these devices are carelessly turned off, then a network may have the routing instability issue. That is, routing blackholes and loops are introduced due to the shutdown of network devices. This leads to heavy packet losses, which further trigger packet retransmissions that consume additional energy. Clearly, this violates our original energy-saving goal.

Although energy saving in the Internet is of paramount importance, there are only a few studies that address this issue. Zhang *et al.* [32] proposed energy saving in traffic engineering by leveraging Multiprotocol Label Switching (MPLS) to shut off parallel routing paths under low utilization. Vasić *et al.* [31]

Manuscript received January 09, 2012; revised November 13, 2012 and August 23, 2013; accepted October 18, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Subramaniam. Date of publication November 21, 2013; date of current version December 15, 2014. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2012CB315803; the National Natural Science Foundation of China under Grants 61073166, 61133015, and 61120106008; and the National High-Tech Research and Development Program of China (863 Program) under Grant 2011AA01A101.

Q. Li, M. Xu, Y. Yang, Y. Cui, and J. Wu are with the Department of Computer Science, Tsinghua University, Beijing 100084, China (e-mail: liqi@csnet1.cs.tsinghua.edu.cn; xmw@csnet1.cs.tsinghua.edu.cn; yyang@csnet1.cs.tsinghua.edu.cn; cy@csnet1.cs.tsinghua.edu.cn; jianping@csnet1.cs.tsinghua.edu.cn).

L. Gao is with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003 USA (e-mail: lgao@ecs.umass.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2013.2288790

identified energy-critical paths offline to achieve energy saving in traffic engineering. These approaches advocated centralized computations. However, achieving energy saving in *traditional intradomain routing* [e.g., Open Shortest Path First (OSPF) and IS-IS] in IP networks remains a challenging problem. In particular, we need to address two key issues: 1) *Safety*, i.e., frequent and time-consuming routing computations, i.e., shortest path tree (SPT) computations, for the power saving purpose, may introduce churns in routings and incur more consumptions in routing control planes. More importantly, different next-hops obtained by routing computations may not be loop-free, which will raise routing blackholes and loops. In this sense, it may not be good if routing protocols always try to compute the best routings. 2) *Practicality*, i.e., to simplify deployment, an energy-efficient routing scheme should not require modifications in the standard IP forwarding paradigm and traditional IP routing protocols, e.g., OSPF. In the ideal case, we do not need to implement packet marking or encapsulation in packet forwarding as in [32].

In this paper, we propose a novel energy-efficient routing approach called *Safe and Practical Energy-Efficient Detour Routing* (SPEED) for achieving safe energy saving. SPEED tries to maximize power saving with guaranteed connectivity by maximizing link that can be detoured around in networks. There are two key ideas with SPEED: 1) SPEED precomputes loop-free next-hops for each primary next-hop to effectively detour around links with low traffic load according to the existing shortest-path routings. Each node independently computes different valid loop-free next-hops to detour around links attached to the primary next-hops. 2) SPEED selects loop-free alternate next-hops from precomputed loop-free next-hops as energy-saving next-hops according to real-time traffic load, and then aggregates the traffic to them without extra routing computations. Thus, SPEED can effectively achieve energy saving by putting idle links to sleep after traffic aggregation.

SPEED requires precomputing a limited number of loop-free next-hops only once, and it does not need to recompute routings after traffic loads change but directly select precomputed alternate loop-free next-hops. Loop-free next-hop selections naturally eliminate routing loops between next-hop switching. SPEED does not need to generate and advertise route changes to the networks, which guarantees the *safety* of power saving while ensuring successful packet forwarding. SPEED does not require modifications in the traditional IP forwarding paradigm and shortest-path routing protocols, i.e., ensuring *practicality*. Thus, SPEED offers a new energy-efficient routing approach for traditional IP networks and effectively ensures *safety* and *practicality* of energy-efficient routing. In particular, the paper makes the following contributions.

- We provide a new perspective of energy-efficient routing that does not require changing the existing router architectures and routing protocols.
- We formally define the problem of energy-efficient routing and prove that determining if energy-efficient routing exists is an NP-complete problem.
- We propose a heuristic for computing routings that safely prunes underutilized links and realizing safe traffic aggregation, while avoiding traffic congestion.
- We demonstrate the performance of SPEED by extensive experimental studies with real network topologies and real traffic matrices.

The rest of the paper is organized as follows. Section II presents an overview of SPEED. Section III defines energy-efficient routing formally and presents an analytical study of energy-efficient routing. Section IV develops a heuristic for computing energy-efficient routings with SPEED. The heuristic achieves energy-efficient routing while trying to minimize its impact on packet forwarding performance. Section V evaluates the performance of SPEED. Section VI reviews related work, and Section VII concludes the paper.

II. Overview of SPEED

We propose SPEED, an intradomain routing protocol that aims to detour the traffic of underutilized links and put these links into "sleep" mode for power saving. SPEED has two major design properties. First, SPEED is *safe*, such that it preserves loop-free packet forwarding after detouring traffic. Second, SPEED is *practical*, in the sense that it is built on the standard IP forwarding paradigm and traditional IP routing protocols, without relying on specialized functions such as packet marking or encapsulation as in [32].

By default, a network obtains the routing decision via a traditional shortest-path routing protocol, such as OSPF or IS-IS. The routing protocol determines the shortest path between every pair of source and destination nodes and specifies the next hop on the shortest path to which a source node should forward packets. We call the next hop the *primary next-hop* (PNH). In the case of SPEED, in addition to the PNH, each source node also considers all other alternate next-hops that provide loop-free forwarding paths to the same destination. If the link connected to the PNH is underutilized, then SPEED will have the source node substituting the PNH with one of such alternate next-hops to forward traffic. Since these alternate next-hops are selected from a power-saving perspective, we call them *energy-saving next-hops* (ENHs). We elaborate how we select an ENH in Section III. As a result, each node in SPEED will dynamically select either the PNH or an ENH to forward traffic, according to the predetermined routing results and the current traffic load. That is, if a node finds that the traffic load to the PNH is low while shifting traffic to an ENH does not aggravate congestion, then it will redirect the forwarding traffic to the ENH.

In SPEED, before finding an ENH, a node needs to first determine the set of *loop-free next-hops* for each destination, i.e., the next-hops that provide loop-free packet forwarding to the destination. Let s be the source node that needs to find a loop-free next-hop for destination node d, u be a neighbor node of s, and $D(i, j)$ be the shortest-path distance from node i to node j. Then, u is a loop-free next-hop of s if and only if the following condition holds [6]:

$$D(u, d) < D(u, s) + D(s, d). \tag{1}$$

Intuitively, (1) means that any traffic forwarded from s to u will go to d directly, rather than go back to s again. Thus, u provides loop-free forwarding. Clearly, the PNH of s is a loop-free next-hop.

Fig. 2 illustrates how SPEED achieves power saving. Suppose that the link costs are denoted by the hop counts. Fig. 2(a) shows one possible traffic load distribution that can be obtained from traditional shortest-path routing. Suppose that r_2 is the PNH of r_1 for some destination d. Then, we can see that r_3

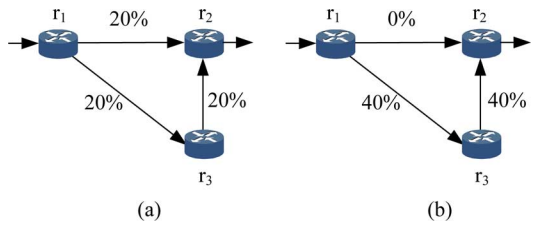


Fig. 2. Illustration of how SPEED achieves energy-efficient routing in a sample topology. We indicate the utilization of each link, i.e., the ratio of link traffic to the link capacity. SPEED basically reduces the number of used links in traditional routing. (a) Traditional (b) Energy-efficent.

is an alternate loop-free next-hop of r_1 for destination d since $D(r_3,d)=D(r_2,d)+1 < D(r_3,r_1)+D(r_1,d) = D(r_3,r_1)+D(r_1,r_2)+D(r_2,d) = D(r_2,d)+2$. Thus, r_3 is a potential candidate of an ENH of r_1.

We then check if detouring r_1's traffic from r_2 to r_3 can actually achieve power saving, subject to the constraint that the link load does not exceed any link capacity. From Fig. 2(a), we observe that if we detour the traffic of link r_1-r_2 to the path r_1-r_3-r_2, then the new utilizations of links r_1-r_3 and r_3-r_2 are both 40%, which is less than 1, implying that the capacity constraint is still satisfied. Thus, with SPEED, we can *safely* free link r_1-r_2 by setting the next-hop of r_1 to r_3, which now becomes an ENH and shifts the traffic to link r_1-r_3 as shown in Fig. 2(b). Then, link r_1-r_2 can be put in sleep mode for power saving [24].

The main goal of SPEED is to select ENHs such that it can detour around links attached to the PNHs and put them into sleep mode so as to achieve energy-efficient routing. However, this remains a challenging issue due to the following reasons. First, while an ENH provides loop-free forwarding guarantees for a single source node, it remains possible to have forwarding loops when multiple source nodes select ENHs independently. For example, in Fig. 2, if r_1 and r_3 independently select each other (i.e., r_3 and r_1, respectively) as their ENHs, then there will be a forwarding loop between r_1 and r_3. Second, the computational complexity for identifying potential ENHs must be low, or routers cannot afford to deploy SPEED. Third, we must choose ENHs in such a way that network congestion is avoided, or energy saving is in vain. The key motivation of this work is to address the above issues in the design of SPEED.

III. Energy-Efficient Routing Model

In this section, we formally define energy-efficient routing that aims to detour the traffic of underutilized links for power saving, and then present an analytical study of energy-efficient routing. The notations and semantics in the energy-efficient model are extended from [20] and [26].

A. Problem Formalization

In general, a network with routing enforced can be modeled as a directed connected graph $G=(V,E)$, where V denotes the node set and E denotes the link set. A directed link in G from node i to node j is denoted by i-j, and $N(i)=\{j \in V \mid i\text{-}j \in E\}$ is the set of node i's neighbors in G. For a destination $d \in V$, $R_d=(V,E_d)$ denotes a routing indicating the paths destined to d from each node, $i \in V\backslash\{d\}$, where $E_d \subseteq E$ [20]. Thus, R_d is a directed acyclic graph (DAG) rooted at d and defines a destination-based routing. Actually, R_d is a subgraph of G, but is dynamically computed by routing protocols. In R_d, each node $i \in V\backslash\{\mathrm{d}\}$ has at least one valid next-hop, $j \in V$, to destination d, i.e., i-$j \in E_d$.

Definition 1: For each destination $d \in V, i \vdash_d j$ defines that j is in the shortest path from i to d, $i \vdash_d ij$ defines that link i-j is in the shortest path from i to d, and $i \nvdash_d ij$ denotes that link i-j is not in the shortest path from i to d.

Let us follow the example shown in Fig. 2. The shortest path from r_1 to r_3 goes through r_3, i.e., $r_1 \vdash_{r_3} r_3$, and link r_1-r_3, i.e., $r_1 \vdash_{r_3} r_1r_3$. It is obvious that link r_1-r_2 is not in the shortest path from r_3 to r_1, i.e., $r_3 \nvdash_{r_2} r_1r_2$.

Definition 2: In routing R_d, link i-j is said to be pruned (where $j \in N(i)$) if there exists a node $k \in N(i)$ $(k \neq j)$ such that $k \nvdash_d ij$ in R_d.

The condition specified in Definition 2 is inspired by [6]. Here, we define the condition from the point of the view of link pruning. The condition means link i-j can be pruned if node i has a neighbor, node k, which has a routing path that does not go through the link. Node k is called an ENH of node i. It implies that after link i-j is pruned, packets can still be rerouted to d by node k in R_d and routing loops never form. For example, Fig. 2 illustrates how routing chooses to prune links for destination d. In this example, link r_1-r_2 can be pruned since $r_3 \nvdash_{r_2} r_1r_2$, where $r_3 \in N(r_1)$.

When computing energy-efficient routings, our goal is to use the minimum number of links in the network to forward traffic and detour around links, e.g., underutilized links specified by the link utilization matrices, in traditional shortest-path routing, while ensuring uninterrupted packet forwarding.

Definition 3: A routing obtained by traditional shortest-path computation is denoted by R_d^{norm}, and a routing under link utilization matrix M is denoted by R_d^M, which is constructed by pruning links in R_d^{norm} according to M.

Definition 4: R_d^M is an energy-efficient routing if every $i \in V$ has a valid path to destination d after detouring around at least one pruned link in R_d^{norm} such that $\bigcup_{d\in V} R_d^M \subset \bigcup_{d\in V} R_d^{\mathrm{norm}}$ under link utilization matrix M.

Definition 5: A weighted graph $G=(V,E)$ is z-aggregatable if the number of pruned links in energy-efficient routing is z, where the overall link weight of each routing path satisfies $w \leq \epsilon$, for a given $\epsilon \in \mathbb{Z}^+$. Note that $w \leq \epsilon$ guarantees that each path in the graph is valid, e.g., ensuring loop-free paths to destinations.

According to Definition 4, in energy-efficient routing, packets to d can be forwarded without interruption after pruning links in R_d^{norm}. The main challenges in energy-efficient routing lie in how to identify which energy-efficient routings are feasible and to maximize the number of pruned links during routing computations (see Section IV). Now we will analyze the topological properties of a weighted graph that are sufficient to ensure graph aggregatability and characterize the complexity of aggregating the graph.

B. Analysis of Energy-Efficient Routing

We first characterize the sufficient conditions of the existence of graph aggregation. All proofs of the theorems can be found in [21].

Theorem 1: For a weighted graph $G=(V,E)$, the graph is 1-aggregatable if there exists at least one link i-j subject to the

following conditions: (i) $i \not\vdash_j ik$; (ii) $i \not\vdash_k ij$; (iii) $k \not\vdash_j ki$, where $k \in N(i)$.

Theorem 1 states that the sufficient condition is required to guarantee 1-aggregatability of a weighted graph. 1-aggregatability means that the network connectivity is still ensured after pruning one link in the network.

Proposition 1: For a weighted graph $G = (V, E)$, if the graph achieves maximum aggregatability, i.e., it is ($|E|$-$|V| + 1$)-aggregatable, at least $|E|$-$|V| + 1$ links in the graph are subject to conditions (i)–(iii).

Proposition 1 implies that the maximum aggregatability requires $|E|$-$|V| + 1$ links must be subject to conditions (i)–(iii). It specifies the necessary condition to ensure that a graph is ($|E|$-$|V|+1$)-aggregatable, where ($|E|$-$|V|+1$)-aggregatability means that the network connectivity is ensured after pruning $|E|$-$|V| + 1$ links in the network. However, even if all links in a graph satisfy conditions (i)–(iii), we cannot conclude if the graph is ($|E|$-$|V| + 1$)-aggregatable.

Theorem 2: For a weighted graph $G = (V, E)$, the graph is not guaranteed to be ($|E|$-$|V|+1$)-aggregatable, if every link in the graph is subject to conditions (i)–(iii).

Theorem 2 establishes that even though each link in a graph is subject to conditions (i)–(iii), the graph is not guaranteed to achieve maximum aggregatability.

The results above show that an arbitrary weighted graph may not be guaranteed to be z-aggregatable where $z \leq |E|$-$|V| + 1$. The problem to determine if a weighted graph is z-aggregatable, for any given $z \in \mathbb{Z}^+$, is defined as the *graph aggregatability problem*. We have the following theorem by a reduction from the clique problem.

Theorem 3: The graph aggregatability problem is NP-complete.

Theorem 3 states that we cannot solve the *graph aggregatability problem* in an arbitrary weighted graph with a polynomial-time algorithm. It means that determining if energy-efficient routings for all destinations achieve z-aggregatability of the graph under a traffic matrix is NP-complete, where the traffic matrix specifies different communicating node pairs. Naturally, we should compute energy-efficient routings with heuristics.

IV. Heuristic Design

In this section, we try to compute $|V|$ routing paths to maximize aggregatability while ensuring network performance (e.g., congestion-free packet forwarding) in SPEED. Section III shows that to determine if energy-efficient routing can achieve maximized graph aggregation in a network is NP-complete. Moreover, for SPEED, we need to consider all routings for $|V|$ destinations, and these $|V|$ routings will contribute to link loads. Thus, considering the network performance (with respect to link load) requires a new dimension of the problem, which makes the problem harder. To address these issues, we design a practical solution with heuristics.

A. Outline

To achieve energy-efficient routing, the heuristic aims to compute routings $R_d, \forall d \in V$, that use the minimal set of links so as to achieve maximized graph aggregatability while avoiding network congestions. By computing $R_d, \forall d \in V$, the number of pruned links for each d, i.e., Ω_d, will be maximized.

Algorithm 1: SPEED Routing

Input: $G = (V, E)$ with link utilization matrix M
Output: R_d^{**} for each $d \in V$
Phase 1:
1: **for** destination $d \in V$ **do**
2: $(R_d^*) \leftarrow$ Link Pruning on d
3: **end for**
Phase 2:
4: Traffic Aggregation by $(R_d^M) \leftarrow$ ENH Assignment on R_d^* under link utilization matrix $M, R_d^{**} \leftarrow R_d^M$
5: **while** Receive LSA announcing link load changes **do**
6: Incremental Traffic Aggregation by $(R_d^M) \leftarrow$ Incremental ENH Assignment on R_d^* under link utilization matrix $M', R_d^{**} \leftarrow R_d^M$
7: **end while**

In the meanwhile, the heuristic will obtain the maximum number of the pruned links according to all computed R_d.

Network congestion is measured through a cost function $\Phi = \sum_{l \in E} \Phi_l$, where Φ_l denotes the congestion cost of l as a function of a link load is computed by the link utilization ratio [12]. The heuristic maximizes Ω_d for $\forall d \in V$, which conforms to that network congestion cost Φ is less than congestion threshold α, where α defines the estimated cost of the network [12] and $\alpha_1 > \alpha_2$ means that the network with α_2 has a lighter traffic load

$$\max_{R_d, \forall d \in V} \Omega_d \tag{2}$$

subject to

$$\Phi < \alpha \tag{3}$$

where $0 < \alpha \leq 5000 \times n$ [12], in which $n = |E|$, controls how much traffic can be aggregated such that SPEED can achieve maximized graph aggregatability under different traffic situations. A smaller Φ value means a lower average link utilization ratio.

Intuitively, a "greedy-search" heuristic can be directly applied to maximize (2). However, since Ω_d is largely restricted by the link weights of networks, maximizing the number of pruned links in each routing is heavily restricted by link weights. To reduce the computation space and keep computational complexity low while preserving the ability to independently compute routings that maximize Ω_d for $\forall d \in V$, we design a two-phase heuristic. Phase 1 independently computes the lowest weight routings after pruning links, which ensures the computed routings have a small deviation from the optimal routings, and Phase 2 aggregates traffic without incurring network congestions by safely assigning ENH to each node according to the computed routings in Phase 1 and real-time link utilization ratio, e.g., learned via TE-LSA [19]. Note that what is computed in Phase 2 is only ENH assignment but not the traditional concept of routing computations.

The two-phase heuristic is shown in Algorithm 1. In Phase 1, the main focus is to compute a routing R_d^* by pruning links in R_d. The pruned links in Phase 1 can be potentially detoured off during traffic forwarding. In Phase 2, the main focus is to achieve traffic aggregation by assigning ENH according to the

computed R_d^* in Phase 1 and realize safe traffic aggregation to detour around pruned links, while ensuring that the network is absence of congestions (from step 4 to 7). Note that we only require the entire computation of ENH assignment for traffic aggregation to maximize Ω_d (step 4) once. The main loop of Phase 2 is to adapt to traffic changes by changing ENH assignments incrementally after receiving an update announcing link utilization change (from step 5 to 7). Therefore, Phase 2 can safely aggregating traffic by assigning ENHs according to routings produced by Phase 1.

B. Phase 1—Link Pruning

In Phase 1, we compute routings to explore links that can be pruned from $R_d, \forall d \in V$. To achieve this, we will precompute an optimal energy-efficient routing, R_d^{opt}, with the minimum link weights. Here, the optimal routing is achieved by computing a minimum spanning tree (MST) [18] according to the link weights[1]. Note that Phase 1 cannot directly use routing paths appearing in MST as the energy-efficient routing paths to forward traffic. Normally, routing paths in MST do not mean that they are always the lowest-cost routing paths for all nodes. Thus, it cannot be directly used as routings in SPEED. Different nodes can compute their own routings according to the precomputed optimal routings. To ensure that all nodes can compute the same optimal routings, if there exist any links having the equal link weights, minimum spanning tree algorithms [18] prefer choosing links attached to the nodes with low node IDs.

Phase 1 uses a greedy search that aims to compute routings with a minimum link weight deviation $\Gamma_d, d \in V$, between a proposed energy-efficient routing R_d and the optimal routing R_d^{opt}. The deviation is measured by using $\Gamma_d = \sum_{i \in V} \Gamma_{i,d}$, where $\Gamma_{i,d}$ indicates the distance from node i to destination d under R_d, with the distance that is computed using the link weights of R_d^{opt} [20]. The smaller Γ_d is, the closer R_d is to R_d^{opt}. The optimization problem can be formalized as follows:

$$\min_{R_d, \forall d \in V} \Gamma_d. \tag{4}$$

Note that the minimized Γ_d is obtained under routing R_d, which ensures connectivity between all node pairs.

In Phase 1, computing routings to minimize Γ_d does not need to consider congestions, and thus computations for different destinations are decoupled because Γ_d is computed based on fixed link weights. This avoids evaluating the cost function Φ for each candidate routing, which is an operation with a significant computation cost. Here, a heuristic can be used to minimize (4). The heuristic needs to start with an initial link set of optimal routings R_d^{opt} $(d \in V)$. Algorithm 2 shows the pseudocode of Phase 1 that examines each node i in increasing order of the node IDs and checks if the node has links under pruning. The heuristic starts with the initial routing $R_d = \text{Set}(R_d^{\text{opt}})$ obtained by extracting a link set from R_d^{opt} (step 1). Note that $\text{Set}(R_d^{\text{opt}})$ includes the directed links in R_d^{opt} and the corresponding directed links in the reverse direction. The main loop locally explores if each node has attached links that can be pruned according to the condition defined in Definition 2 (from step 4 to 12). Its function tries to find effective alternative loop-free next-hops for the PNH of each node by checking if the condition

[1]An optimal routing is a directed graph and different from MST that is an undirected graph.

Algorithm 2: Link Pruning

Input: $G = (V, E)$, R_d^{opt}, and link weights $\mathcal{C}_d$ in R_d^{opt}
Output: R_d^* and Γ_d^*

1: $R_d \leftarrow Set(R_d^{\text{opt}})$
2: $\Gamma_d \leftarrow 0$, $\Gamma_d^{\text{temp}} \leftarrow \Gamma_d$
3: **for** node $i \in V \setminus \{d\}$ **do**
4: **for** $j \in N(i)$ and link i-j is not in R_d **do**
5: Update Γ_d^{temp} using $\mathcal{C}_{i,k}$ and $\mathcal{C}_{k,d}$
6: **if** $k \nvdash_d ij$ where $i \vdash_d k$ and $\Gamma_d^{\text{temp}} > \Gamma_d$ **then**
7: $\Gamma_d^{\text{temp}} \leftarrow \Gamma_d$
8: **else**
9: Add link i-j to R_d
10: **end if**
11: **end for**
12: **end for**
13: $\Gamma_d^* \leftarrow \Gamma_d^{\text{temp}}, R_d^* \leftarrow R_d$

holds and comparing the weight deviations (from step 6 to 10). If the condition holds and the weight deviation is not increased, which means that the current loop-free alternate next-hop can be a candidate ENH and the link attached to the PNH can be pruned, we update weight deviation Γ_d (step 7). Otherwise, the link attached to the PNH needs to be included in R_d (step 9). After all the nodes are visited, computing R_d^* and Γ_d^* is finished, and then Phase 1 stops.

Let us follow the example shown in Fig. 2. We assume that we constructed optimal routings R_d^{opt}. According to the optimal routings to different destinations, we can obtain the set of the links used in the optimal routings. For instance, if we compute R_{r_2}, we can obtain that the optimal routing to $r_2 R_{r_2}^{\text{opt}}$ is $[r_1\text{-}r_3, r_3\text{-}r_2]$ and $\text{Set}(R_{r_2}^{\text{opt}}) = \{r_1\text{-}r_3, r_3\text{-}r_1, r_3\text{-}r_2\}$. Note that r_2-r_3 is excluded from $\text{Set}(R_{r_2}^{\text{opt}})$ because the link is starting from the destination r_2 and will be never used in R_{r_2}. In this example, to check if links attached to r_1 can be pruned, we visit r_1's alternate loop-free next-hops and examine if the attached links are not in R_{r_2}. Since $r_3 \nvdash_{r_2} r_1 r_2$ and $r_1\text{-}r_3 \notin R_{r_2}, r_1\text{-}r_3$ will be included in R_{r_2} and $\Gamma_{r1,r2} = 1$, which means that r_1 can use link r_1-r_3 to reach r_2 and link r_1-r_2 in R_{r_2} can be pruned. We cannot find more loop-free next-hops for r_1, and hence the current $\Gamma_{r1,r2}$ is the minimum value. Next, we can examine the other node, i.e., r_3. Similarly, r_3-r_1 will be included in R_{r_2} and $\Gamma_{r3,r2} = 1$, which means link r_3-r_2 in R_{r_2} can be pruned by using link r_3-r_1. Thus, we can obtain $R_{r_2}^* = [r_1\text{-}r_3, r_3\text{-}r_1, r_3\text{-}r_2]$. The overall $\Gamma_{r2} = \Gamma_{r1,r2} + \Gamma_{r3,r2} = 2$, and we cannot find routings with $\Gamma_{r2} < 2$.

By examining each node in increasing order of its IDs, SPEED ensures that each node achieves the consistent routing tree by independently computing it, and the obtained R_d^* by the node is close to the global optimal solution according to all available ENHs. In the next phase, we can use candidate ENHs to forward traffic and prune links attached to PNHs dynamically under real traffic situations, which guarantees successful packet forwarding.

C. Phase 2—Traffic Aggregation

The routings, R_d^*, achieved by Phase 1 are used as the inputs to Phase 2, and Phase 2 tries to aggregate traffic according to the routing trees and finally achieve link pruning during packet forwarding in runtime. To realize traffic aggregation, Phase 2

only needs to assign ENHs to different nodes and seeks to assign ENH for safe traffic aggregation according to the computed routings in Phase 1, subject to the constraint that Φ is less than congestion threshold α that controls how much aggregation can be achieved. The ENH assignment procedure prefers solutions with the largest possible number of pruned links. It can be formulated using the following optimization:

$$\max_{R_d^*, \forall d \in V} \Omega_d \tag{5}$$

subject to

$$\Phi < \alpha. \tag{6}$$

Normally, to aggregate traffic, nodes may have more than one potential ENH they can use to forward packets to the destination. As discussed earlier, nodes simultaneously pruning their links may incur forwarding loops (see Section II). The algorithm to assign ENH and aggregate traffic in Phase 2 should avoid forwarding such loops as well as ensuring congestion-free routing.

Let $R_d = (V, E_d)$ be the routing for d achieved by Phase 1, where $E_d \subseteq E$ is the set of links attached to PNHs. $R_d^M = (V^M, E_d^M)$ denotes the routing for destination d after pruning some specific links, i.e., the underutilized links, according to M. Let $B_d^M : V \rightarrow V \cup \{\emptyset\}$ be the space of ENH assignments under the current link utilization matrix M. For each node i, $B_d^M(i) \in V \cup \{\emptyset\}$ denotes the set of ENHs of node i that can be used to prune the link attached to its PNH in R_d. Here, if an ENH assignment after Phase 2 is empty, i.e., $B_d^M(i) = \emptyset$, it implies that we do not need to assign an ENH to node i to prune the attached links because assigning ENH may incur forwarding loops or incur congestions. Phase 2 aims to choose a candidate ENH for each node to prune links during traffic forwarding. Note that a node will be pruned if all links attached the node are pruned, which means that the node does not generate and forward any traffic.

Assume that node k is a candidate ENH for node i. *Node k can be selected if E_d^M remains a DAG after the addition of link i-k [condition (C)]*, which guarantees that routing loops are avoided. Since candidate ENHs produced by Phase 1 ensure that node k has at least one next-hop in E_d^M, condition (C) can also ensure that packets will be delivered to d. With the condition, ENHs can be assigned to ensure packet delivery after traffic aggregation.

The algorithm to assign ENH for different nodes is shown in Algorithm 3. The algorithm assigns ENHs for nodes according to routings, $R_d^*, \forall d \in V$, and the current link utilization matrix M. In the pseudocode, Φ and φ_{ij}, where $\forall i, j \in V$, denote the network congestion value and the congestion value of link i-j, respectively. Here, φ_{ij} is computed according to link utilization ratio with the current $R_d^*, \forall d \in V$. First, a loop is used to initialize the set of ENHs $B_d^M(i)$ for each d (from step 1 to 3) and compute and sort $\varphi_{i,j}$ for each link i-j in ascending order (from step 5 to 8). If link i-j is in R_d^{norm} but not in R_d^*, it means that link i-j can possibly be pruned. The main loop is from step 9 to 23 and tries to safely assign ENH to each node so as to avoid loops and congestions. If assigning k satisfies condition (C) and traffic shifting from link i-j to link i-k will not cause Φ exceeding α, k will be assigned to ENH set $B_d^M(i)$. Meanwhile, Φ will be updated. The algorithm stops if all φ_{ij} are visited, and then the traffic to d in node i can be safely shifted to the link that is attached to the node specified in $B_d^M(i)$.

Algorithm 3: ENH Assignment

Input: $G = (V, E)$, α, link utilization matrix M, $(R_d^*, R_d^{\text{norm}})_{d \in V}$
Output: R_d^M for each $d \in V$ and ENH assignment B_d^M

1: **for** node i **do**
2: $B_d^M(i) \leftarrow \emptyset$
3: **end for**
4: $E_d^M \leftarrow R_d^{\text{opt}}$
5: **for** $i = 1$ to $|V|$ **do**
6: Compute $\varphi_{ij} = \Phi_{(i,j)}$ where $j \in N(i)$ and link i-j is in R_d^{norm} but not in R_d^*
7: Add φ_{ij} into φ and sort φ in ascending order
8: **end for**
9: **for** φ_{ij} for node i **do**
10: **for** $k \in N(i)$ where i-k is in R_d^* **do**
11: **if** link i-j is in R_d^{norm} but not in R_d^* **then**
12: **if** link i-k not in R_d^M and $R_d^M \cup i$-k satisfies condition (C) **then**
13: Add i-k into R_d^M
14: **end if**
15: $\Phi' \leftarrow \Phi$, Compute Φ with the updated utilization ratio of the links
16: **if** $\Phi < \alpha$ **then**
17: $B_d^S(i) = k$
18: Break
19: **else**
20: $\Phi \leftarrow \Phi'$
21: Add link i-j to R_d^M
22: **end if**
23: **end if**
24: **end for**
25: **end for**

Note that visiting φ_{ij} in ascending order with the subscript of φ ensures that each node will achieve the consistent view of ENH assignments to avoid forwarding loops. Since it is easy to accurately differentiate link utilization ratio in two directions, e.g., via TE-LSA, we can aggregate computations of ENH assignment for different destinations. That is, if a node shares the same PNH to different destinations, we can select ENHs for these PNHs together. For easy illustration, the algorithm here only considers assigning ENHs for one destination.

Let us follow the example illustrated in Fig. 2. We obtain that $R_{r_2}^* = [r_1\text{-}r_3, r_3\text{-}r_2, r_3\text{-}r_1]$ in Phase 1 and compute $\{\varphi_{r1r2} = 1, \varphi_{r1r3} = 1, \varphi_{r2r3} = 1\}$. Then, $\{r_2, r_3\} \in N(r_1)$, and $r_1\text{-}r_3 \in R_{r_2}^*$. For link r_1-r_2 in $R_{r_2}^{\text{norm}}$, the algorithm checks if: 1) $R_{r_2}^M$ has valid routings to r_2; and 2) the traffic shifting will cause the network congestion cost Φ exceeding congestion threshold α, where $\alpha = 3 \times 3 = 9$. In this example, there does not exist any traffic from r_2 to r_1 via link r_1-r_2. If the traffic on link r_1-r_2 shifts to link r_1-r_3, $\varphi_{r1r2}{}'$ (where φ'_{r1r2} denotes the congest value of link r_1-r_2 before traffic shifting) and φ_{r1r2} (φ_{r1r2} denotes the congestion value of link r_1-r_2 after traffic shifting) are 1 and 0, respectively, and $\varphi_{r1r3}{}'$ and φ_{r1r3} are 1 and 3, respectively. Here, the updated Phi after traffic aggregation is equal to $\Phi - \varphi_{r1r2}{}' + \varphi_{r1r2} + \varphi_{r1r3} = 4 < \alpha = 9$. At the same time, $R_{r_2}^M = [r_1\text{-}r_3, r_3\text{-}r_2]$ satisfies condition (C). Therefore, r_3 can be set into the r_1's ENH set, $B_{r_2}^M(r_1)$. Now we cannot assign

Algorithm 4: Incremental ENH Assignment

Input: $G = (V, E)$, α, updated utilization ratio of link i-j, $(R_d^*, R_d^{\text{norm}}, R_d^{M\prime})_{d \in V}$
Output: R_d^M for each $d \in V$ and ENH assignment B_d^M

1: $\varphi'_{ij} \leftarrow \varphi_{ij}$
2: Compute $\varphi_{ij} = \Phi_{(i,j)}, \Phi = \Phi - {\varphi_{ij}}' + \varphi_{ij}$
3: **if** $\Phi > \alpha$ **then**
4: **while** node $k \in N(i)$ and link i-k is in R_d^{norm} **do**
5: $B_d^S(i) \leftarrow \emptyset$
6: $\varphi'_{ij} \leftarrow \varphi_{ij}$, Compute $\varphi_{ik} = \Phi_{(i,k)}, \varphi_{ij} = \Phi_{(i,j)}$
7: Compute $\Phi = \Phi - {\varphi_{ij}}' + \varphi_{ij} + \varphi_{ik}$
8: **if** $\Phi < \alpha$ **then**
9: Break
10: **end if**
11: **end while**
12: **else if** link i-j is in $R_d^{M\prime}$ but not in R_d^* **then**
13: Remove link i-j from $R_d^{M\prime}$
14: **if** $R_d^{M\prime}$ satisfies condition (C) **then**
15: $B_d^S(i) = k$ where i-k in R_d^*
16: $\varphi'_{ij} \leftarrow \varphi_{ij}$, Compute $\varphi_{ik} = \Phi_{(i,k)}, \varphi_{ij} = \Phi_{(i,j)}$
17: **if** $\Phi + \varphi_{ij} - \varphi_{ik} - {\varphi_{ij}}' > \alpha$ **then**
18: Add link i-j to $R_d^{M\prime}$
19: **else**
20: $\Phi \leftarrow \Phi - \varphi'_{ij} + \varphi_{ik} + \varphi_{ij}$
21: **end if**
22: **else**
23: Add link i-j to $R_d^{M\prime}$
24: **end if**
25: **end if**
26: $R_d^M \leftarrow R_d^{M\prime}$ for each $d \in V$

an ENH for the other node. Although the traffic on link r_2-r_3 shifting to link r_1-r_3 will not cause Φ exceeding α, r_1 cannot be assigned to r_3 as its ENH because r_1's ENH is already set to r_3. If r_1 is assigned to r_3 as its ENH, E_d^M will not be a DAG then, which will violate condition (C).

After assigning ENHs to nodes under an existing traffic situation, SPEED will safely aggregate traffic over different links. However, traffic in a network may always change. We need to dynamically change ENH assignments if the link utilization ratio changes significantly, e.g., network congestion cost Φ exceeds congestion threshold α. We use Algorithm 4 to incrementally change ENH assignment according to the received routing updates. After receiving a routing update announcing link utilization change, we need to recompute φ of the corresponding link (from step 1 to 2). If the overall Φ exceeds α, ENH assignments of different nodes should be adjusted until Φ falls less than α (from step 3 to 12). If Φ is less than α, the algorithm will try to find if there is any traffic can be aggregated (from step 13 to 26), which is similar to that in Algorithm 3. Note that a node that first handles the routing update and change ENH assignments needs to update its link utilization information in the reannounced routing updates to avoid synchronization between different nodes.

D. Discussion

Safety: In SPEED, routings rooted at different nodes are directly aggregated by loop-free alternate next-hops. Thus, SPEED does not require generating routing updates announcing route changes after ENH assignment changes. Therefore, *SPEED can easily ensure routing stability in control plane*. Moreover, SPEED will not induce routing loops with different energy-efficient routings [see condition (C)], which also guarantees each node has valid routing to destinations. Furthermore, according to the previous studies conducted by Fortz *et al.* [12], if network congestion cost Φ in a network is less than the network congestion threshold α where $\alpha \leq 5000 * n$ (n is the number of links in the network), the network will not incur congestions. Thus, SPEED can ensure that a network is congestion-free by setting $\alpha \leq 5000 * n$. In Section V, we will validate this by simulations with real traffic matrices.

SPEED differs from traditional traffic-aware routing approaches. 1) The routes are chosen for energy-efficient routing according to the overall network congestion cost, but not individual link utilizations. It will not be highly sensitive to single link load change, which will significantly reduce the probability of routing oscillations. 2) SPEED directly uses loop-free next-hops to forward traffic so as to achieve energy-efficient routing. It does not require synchronizing routings because all routing paths for energy saving are loop-free routes and routers do not announce link change events to the rest of the networks, which is similar to what is achieved in loop-free alternate (LFA) [6]. Therefore, *traffic shifting and aggregation in SPEED is loop-free and congestion-free while achieving power savings*.

Practicality: The SPEED approach can be directly implemented in the framework of IP Fast Rerouting (IP-FRR), LFA [6]. LFA is used to detour around failures and the standardized IP-FRR technology implemented by major router vendors [2], [4]. Especially, similar to LFA, SPEED does not require updating firmware, but modifies the software implementations of LFA that are already implemented in the mainstream routers. From an implementation point of view, there are two differences between SPEED and LFA in computing and choosing loop-free next-hops: 1) the LFA framework uses any alternative loop-free next-hop to detour around the assumed component (link or node) failure, however SPEED adds a constraint in computing loop-free next-hop so that the computed loop-free alternate next-hops use the minimized number of links; 2) LFA directly uses the computed loop-free next-hop to forward traffic to detour around a failure after the failure occurs, but SPEED dynamically determines if the computed loop-free next-hop should be chosen as the best next-hop to forward traffic according to the current traffic situation, i.e., the current link utilization matrix. We can easily realize SPEED by making some minor modifications in LFA implementations. Thus, it is easy for us to implement SPEED with the LFA framework in routers.

Failure Recovery: SPEED well addresses the failure recovery issue. In presence of network failures, i.e., the adopted next-hops (PNHs or ENHs) fail, the other available next-hops, e.g., PNHs or ENHs, can be used to detour around failures. In other words, PNHs or computed ENHs obtained in Phase 1 are valid next-hops for failure detour. The approach is a bit similar to the existing LFA proposal [6]. Alternately, we can update routings after failures occur. Routers will generate routing update announcements, such as OSPF Link-State Advertisements (LSA),

to announce the failures, and routers will compute new routings upon receiving the announcements. In this case, routers need to recompute their energy-efficient routings and update their routing paths according to the received announcements. Since most network failures are short-term [23], [28], we suggest taking the former approach, i.e., directly changing next-hop assignments to detour around failures so as to realize fast failure recovery [6].

In a nutshell, we do not need to recompute routings under network failures since the routings computed by Phase 1 in SPEED provide valid candidate next-hops to detour around them. Any routers adjacent to failed links or nodes use precomputed alternative loop-free next-hops to detour around these failed components. The routing paths chosen by loop-free next-hops have the smaller link costs than the routing paths that include the routers detecting the failures [see (1)], which ensures that the packets can be successfully delivered to the destinations but not be sent back to form forwarding loops. The approach for failure detour is similar in spirit to the LFA approach [2], [4], . Therefore, SPEED does not impact normal failure recovery in networks. Instead, it can provide efficient failure recovery, while not requiring extra routing computations.

Computation Complexity: The computational complexity to solve the optimization problem in Phase 1 is low since our proposed solution is directly based on an optimal routing that is used to examine the effectiveness of different routings after link aggregations. The overall computation complexity is $O(|E| + |E|\log|E| + |V|\log|V|)$. Phase 2 leverages the results by the previous heuristic and the additional memory to store the routings examined during Phase 1. Thus, Phase 2 needs to compute Φ by evaluating the load of aggregated links, which takes $O(m(m+l))$ time where m is the number of the candidate aggregated links and l is the number of links in the paths between ENHs and the respective destinations. Note that the computational complexity is only for a single prefix. After a complete computation of traffic aggregation, we only need to incrementally compute the network congestion cost. During incremental ENH assignments in Phase 2, we only need to recompute Φ by evaluating the load of the link incurring traffic change, and the worst case of running time is $O(l)$, where l is the number of links in the path between the ENH and the corresponding destination. Thereby, the runtime computation complexity is determined by the number of routers announcing traffic change.

V. Evaluation

This section evaluates the performance of energy-efficient routing and explores the benefits in SPEED.

A. Methodology

We use different real network topologies in the evaluation, including Abilene and GEANT and the two measured topologies, i.e., AT&T and Sprint, from Rocketfuel [27]. Table I shows the detailed information of these four networks. The topologies of Abilene and GEANT can be found in [1] and [3], respectively. The Abilene router-level topology and the measured traffic matrices are available in [33]. The GEANT topology and traffic matrices are provided by the authors of [29]. The traffic matrices are measured every 5 min for Abilene and every 15 min for GEANT. We study and evaluate the performance with different energy-efficient routing schemes by choosing the traffic matrices in two different days. One day is without traffic burst, and the other day is with traffic burst. Abilene-1 and GEANT-1 indicate the Abilene and GEANT networks without traffic burst on August 10, 2004, and August 2, 2005, respectively; Abilene-2 and GEANT-2 indicate the Abilene and GEANT networks with traffic burst on April 8, 2004, and May 31, 2005, respectively. We use the gravity model [20], [32] to generate traffic matrices for the AT&T and Sprint topologies and assign link capacities using the method described in [17].

TABLE I
Network Topologies

Network	The number of nodes	The number of links
Abilene	12	15
GEANT	23	37
AT&T	631	2078
Sprint	315	972

Since the effectiveness of link pruning may be restricted by link weights in IP networks [13], we can optimize link weight assignments in networks to ensure that each underloaded link can be pruned with SPEED. Basically, we can obtain optimal link weight assignments by leveraging integer linear programming (ILP) to achieve the maximum power saving. We can add constraints to the ILP problem to ensure that the routing paths after link weight optimization have the minimum deviation from the default routing paths. Rétvári *et al.* [25] optimize IGP link weights to maximize the number of links under protection by LFA. The methodology can be applied to SPEED to optimize link weights for the purpose of energy-efficient routing. For simplicity, we directly change the link weights to enable maximum graph aggregation to explore the potential power savings with SPEED in our experiments. A sample configuration of the Abilene network can be found in [21].

Moreover, we evaluate the performance of two optimal schemes (OPT) in which traffic will be aggregated in a minimum spanning tree built according to link utilization ratio. All traffic will be aggregated as much as possible, which conforms to the constraint $\Phi < \alpha$. Note that since these two optimal schemes are not restricted by link weights, they can prune the maximum number of links. However, if any traffic aggregation violates the constraint, the traffic cannot be aggregated and will be forwarded by default routings.

In summary, we will evaluate and compare the performance of following schemes:

- ORIG: routings computed by a traditional intradomain routing protocol, e.g., OSPF, without SPEED enforced;
- SPEED: routings computed by a traditional intradomain routing protocol with SPEED according to the existing default network configurations;
- SPEED-C: routings computed by a traditional intradomain routing protocol with SPEED according to modified network configurations where the link weights are modified to achieve maximum graph aggregatability;
- OPT-MAX: routings computed by OPT where traffic over links with heavier loads will be pruned first;
- OPT-MIN: routings computed by OPT where traffic over links with lighter loads will be pruned first.

TABLE II
RESULTS OF Z-AGGREGABILITY

Networks	z-aggregability
Abilene with default configurations	2-aggregability
Abilene with modified configurations	4-aggregability
GEANT with default configurations	5-aggregability
GEANT with modified configurations	15-aggregability
AT&T with default configurations	922-aggregability
Sprint with default configurations	355-aggregability

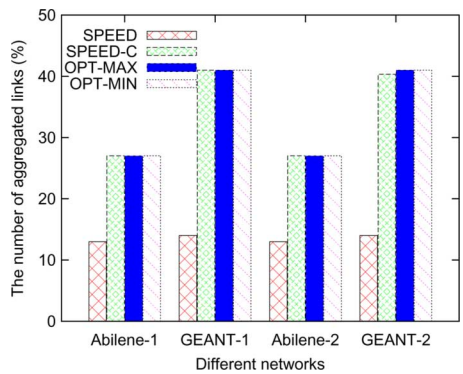


Fig. 3. Number of pruned links in different networks.

Note that the optimization objective of SPEED is to maximize the number of links that can be put into the sleep mode. In this context, OPT-MAX and OPT-MIN are optimal solutions because they both achieve $(|E|\text{-}|V|+1)$-aggregatability if we do not consider network congestion situations.

In the evaluations, for simplicity, we assume that each linecard is only connected to a single link, and then a linecard can be put to "sleep" when there is no traffic over the link. We make this assumption because the information of physical connections among linecards is not available in the data set [32]. However, our methodology can be adapted to the situation when linecards have multiple ports.

B. Z-Aggregability and Link Pruning Ratio

We explore z-aggregability of different networks, and then compare the link pruning performance with different energy-efficient routing schemes[2] using different network topologies and traffic matrices.

As Table II shows, SPEED can achieve only 2-aggregability and 5-aggregability, in Abilene and GEANT networks, respectively, with the default configurations. The networks with modified link weights achieve much better graph aggregability. The modified link weight assignment achieves 5-aggregability and 15-aggregability, respectively, which achieves the maximum graph aggregation rate. SPEED can achieve 922-aggregability and 355-aggregability, in AT&T and Sprint networks, respectively. The aggregation ratio is 44.37% and 36.52%, respectively.

We measure the *link pruning* ratio, defined as $(n_0 - n_1)/n_0$, where n_1 and n_0 are the average number of links used by SPTs before and after traffic aggregations, respectively. Fig. 3 shows the link pruning ratio in different networks. We observe that 14% links can be pruned with SPEED in the Abilene and GEANT network with the default network configurations. Some underutilized links cannot be pruned since the nodes connecting these links do not have valid ENHs to detour around them because of the link weight restrictions. Similarly, SPEED-C achieves more link pruning ratio than SPEED in the Abilene and GEANT networks and achieves the maximum link pruning ratio, i.e., 27% and 41% link pruning ratio, respectively. The main observation is that the link pruning ratio achieved by SPEED-C is very close to that with OPT-MAX and OPT-MIN. These two schemes mostly obtain the maximum link pruning ratio when SPTs rooted at different nodes are successfully aggregated into a minimum spanning tree. Surprisingly, OPT-MAX fails to prune links to use the minimum cost tree to forward traffic during four periods in GEANT-2 because traffic aggregation is restricted by Φ to avoid congestions (see Section IV). Moreover, we conduct the experiments with the AT&T and Sprint topologies and make similar observations to the Abilene network. In the following experiments, we only present the results of Abilene-1 and GEANT-1. The experiment results of the AT&T and Sprint networks can be found in [21].

[2]For simplicity, all routing schemes aim to detour traffic for power savings are called energy-efficient routing schemes for short in the experiments.

TABLE III
POWER CONSUMPTION OF LINECARDS [5]

Linecard	Speed (Mbps)	Power (Watts)
1-Port OC3	155.52	60
8-Port OC3	1244.16	100
1-Port OC48	2488.32	140
1-Port OC192	9953.28	174

C. Power Saving Ratio

We compare the potential power savings by link pruning. The power consumption data of linecards are provided by Cisco [5] (see Table III). Since linecards altogether account for more than 40% of a router power budget [32], it is meaningful to measure the saved powers of linecards. Here, we study the potential power saving by measuring the saved powers of sleeping linecards. According to previous studies [24], if we consider the power consumption for delivering control-plane packets in the sleep model, power saving will be reduced by around 10%. Actually, we can have different approaches to handle the control-plane packets in the sleep model, e.g., explicitly handling the links in the sleep mode and adjusting protocol parameters [32], which is implementation-specific. Some of these approaches do not require additional power consumption. For simplicity, we do not consider the power consumption for delivering routing control packets when the linecards are in the sleep mode in this experiment.

Fig. 4 shows consumed powers in different networks within the span of one day, and Fig. 5 shows the corresponding power saving ratio. The power saving ratio is defined as $(p_0 - p_1)/p_0$, where p_0 and p_1 are the power consumption before and after applying different schemes, respectively. The saved powers of Abilene-1 with SPEED, SPEED-C, OPT-MAX, and OPT-MIN are 8, 17, 17, and 17 kWh. The power saving ratio is 14%, 27%, 27%, and 27%, respectively. The power saving ratio is similar to the link pruning ratio (see Fig. 3), except that the effectiveness of power saving with SPEED is restricted by link weights. We observe that the saved powers of GEANT-1 are 17, 51, 51, and 53 kWh. Note that since OPT-MIN prunes links with

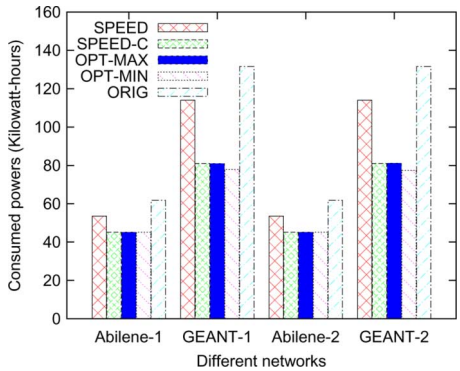


Fig. 4. Power consumption in different networks within the span of one day.

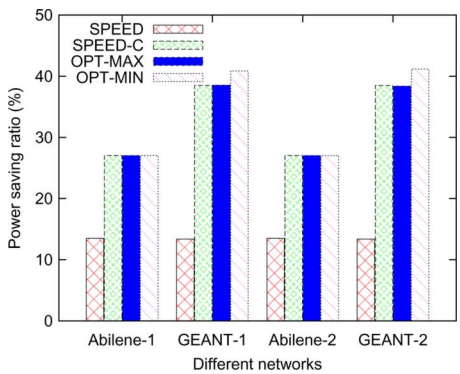


Fig. 5. Potential power savings in different networks within the span of one day.

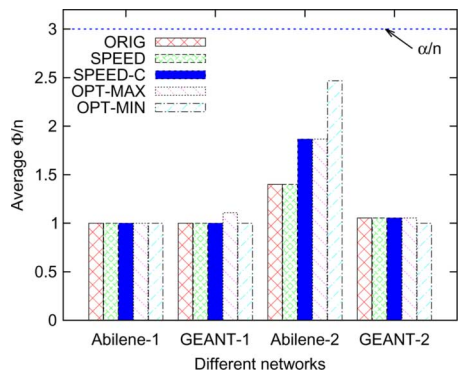


Fig. 6. Different Φ after applying power saving in different networks.

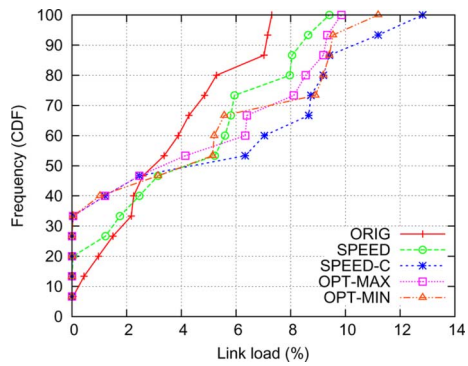


Fig. 7. Link load distribution in Abilene network on August 10, 2004 (Abilene-1).

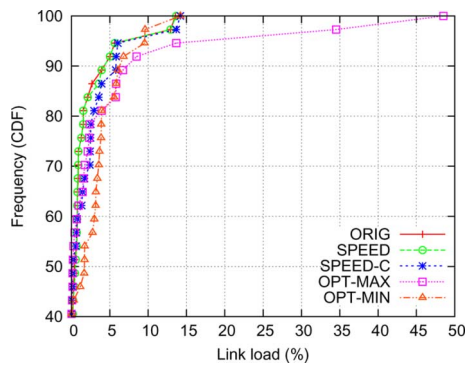


Fig. 8. Link load distribution in GEANT network on August 2, 2005 (GEANT-1).

larger capacity (i.e., with more potential power consumptions), it achieves a bit more power savings. The overall power saving ratios of SPEED-C, OPT-MAX, and OPT-MIN are around 40%. The power saving ratio in Abilene-2 and GEANT-2 are similar to that in Abilene-1 and GEANT-1, respectively. The results demonstrate that SPEED and SPEED-C are effective for power savings in the traditional IP networks.

D. Network Congestion Ratio

Energy-efficient routing schemes will impact the link utilization ratio since some links are detoured off and the other links will carry more traffic after link pruning. We evaluate the impact of different energy-efficient routing schemes on link utilization. Following the discussion of Section IV, we investigate if the schemes achieve congestion-free routing by complying with the congestion constraint, i.e., the network congestion value Φ is less the threshold α. In our evaluations, we set α [see (5)] to $3 \times n$, where n denotes the number of links in the networks, which means that the setting of α tries to ensure that the average link utilization ratio should be smaller than 67% [12].

Fig. 6 illustrates the average Φ/n achieved by each link with different schemes. Average Φ/n with ORIG in Abilene is 1, which means that the link utilization ratio is under 33% [12]. Since the traffic load of each link is light and traffic aggregation does not cause the traffic to shift to the links with heavier loads, average Φ/n achieved by four schemes is similar to ORIG in the Abilene-1, GEANT-1, and GEANT-2 networks. We also observe that the values of Φ in GEANT-1 and GEANT-2 are slightly different after traffic shifting with OPT-MAX and OPT-MIN (we will explain this in a later experiment). In Abilene-2, the traffic burst significantly increases the link loads, and we observe an increase in Φ. Since SPEED shifts a smaller percentage of traffic and balances the traffic over different links, the obtained Φ/n is still similar to ORIG. SPEED-C, OPT-MAX, and OPT-MIN increase the average Φ/n more than 30%. However, Φ/n values in these four networks are still less than what we estimated, i.e., less than $\alpha/n = 3$.

The results show that Φ can effectively control traffic aggregation with respect to network congestions. Now we study if Φ can take effect in controlling individual link utilization ratio. To investigate this issue, we plot the cumulative distribution function (CDF) of link utilization in Abilene-1 shown in Fig. 7. The link utilization ratio with ORIG is less than 8%, and SPEED and SPEED-C increase the maximum link utilization ratio to 9% and

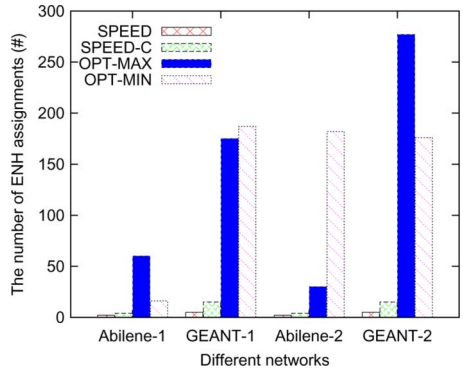


Fig. 9. ENH assignment changes in different networks.

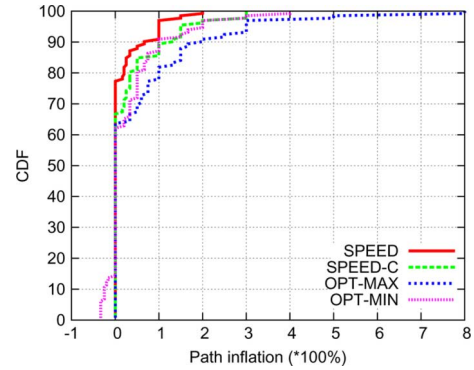


Fig. 10. Routing path inflation in Abilene network on August 10, 2004 (Abilene-1).

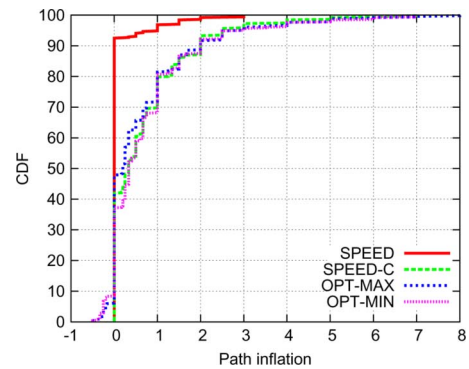


Fig. 11. Routing path inflation in GEANT network on August 2, 2005 (GEANT-1).

13%. In OPT-MAX and OPT-MIN, the link utilization ratio is similar to SPEED-C. We observe a similar pattern of link utilization changes in GEANT-1 (see Fig. 8). Most link utilization ratio in GEANT-1 is less than 15%, except that the link utilization ratio of two links reaches more than 35%. This is the exact reason that we observe that OPT-MAX increases the average value of Φ/n in Fig. 6. Similarly, we do not observe a significant increase of link loads in Abilene-2 and GEANT-2 under traffic burst, except that some nodes in these two networks change their next-hops back to their PNHs during the traffic burst period.

E. Data Plane Stability

To achieve the power saving purpose, different energy-efficient routing schemes need to change ENH assignment so as to change packet forwarding paths to realize efficient link pruning, e.g., to avoid congestions. Control-plane stability of routings with SPEED is guaranteed (see Section IV), and it is more stable than OPT-MAX and OPT-MIN since the latter two always recompute different shortest-path routings once traffic matrices change. Here, we measure the changes of ENH assignments incurred by different schemes and investigate data plane stability.

Fig. 9 shows the changes of packet forwarding paths in different networks based on the snapshot of the traffic matrix. Since packet forwarding paths after traffic aggregation can carry all network traffic no matter if traffic burst occurs, the ENH assignments with SPEED and SPEED-C do not change over time. We observe that SPEED and SPEED-C only change ENH assignments twice and four times, respectively, to prune the underutilized links in Abilene-1 and Abilene-2. Similarly, SPEED and SPEED-C incur ENH assignment changes five times and 15 times, respectively, in GEANT-1 and GEANT-2. Hence, SPEED and SPEED-C do not lead to large traffic shifts in different networks. However, OPT-MAX and OPT-MIN are sensitive to the traffic load changes since they always need to modify ENHs to use links with different capacities under different traffic loads. Thus, they introduce much more ENH assignment changes than SPEED and SPEED-C. Since frequent ENH assignment changes may form many forwarding loops, these two schemes may have a *safety* problem of packet forwarding.

F. Path Inflation Ratio

Intuitively, energy-efficient routing schemes may increase the length of packet forwarding paths since part of the packet forwarding does not follow the shortest routing paths (in terms of link weights) after applying these schemes. In this experiment, we will investigate how many routing paths are inflated and how long the paths are inflated. In this experiment, we measure the path inflations of all source-to-destination pairs.

Figs. 10 and 11 show CDFs of path inflation for Abilene-1 and GEANT-1, respectively. For Abilene-1, over 67% packet forwarding paths in SPEED and SPEED-C remain unchanged, and less than 4% of packet forwarding paths are increased by a factor of 2. Averagely, SPEED and SPEED-C increase 19% and 31% of the length of packet forwarding paths, respectively. In OPT-MAX and OPT-MIN, only 63% of packet forwarding paths remain unchanged, and about 4% of packet forwarding paths are increased by more than twice. Interestingly, compared to ORIG, 10% of the path length in OPT-MIN is reduced. The reason is that OPT-MIN enforces that the forwarding paths of these source–destination pairs follow shorter-length paths constructed by link utilization ratio. The average path inflation ratio of OPT-MAX and OPT-MIN is 64% and 34%, respectively, and much larger than SPEED and SPEED-C. For GEANT-1, more than 90% packet forwarding paths remain unchanged in SPEED. Only 40% of the length of packet forwarding paths is not changed in SPEED-C, OPT-MAX, and OPT-MIN. The average path inflation ratio of these three schemes is around 70%. The reason is that the GEANT topology is constructed by cycles at different scale and the packets on the pruned links will be redirected from a short path in a cycle, e.g., 1 hop, to the rest of the cycle, which incurs relatively longer forwarding paths.

To investigate the impact of path inflation in the GEANT network, we measure traffic delay in the real GEANT network.

We randomly measure the traffic delay of three different routing paths. We observe that the traffic delay is only about 20% when the path inflation reaches 60%. We believe that 20% traffic delay will not significantly impact the performance of TCP connections. Therefore, 60% path inflation may still be acceptable with respect to the achieved power saving in the GEANT network. The detailed measurement results can be found in [21].

The main observation from the path inflation results is that SPEED and SPEED-C realize different tradeoffs between power saving performance and the packet forwarding performance. In particular, SPEED-C achieves more power savings than SPEED by optimizing link weight assignments while inducing more path inflations. As a result, it provides network operators with a tunable solution to configure and optimize link weights such that it can provide a balance between power saving and performance. Network operators can configure congestion threshold α to obtain different power saving. For instance, if they decrease the value of α, power saving will be reduced, and then the path inflation will be also reduced. Also, they can set optimal link weights to achieve more power saving under the same link utilization matrix. SPEED-C achieves much better power saving than SPEED by optimizing link weights. However, it may increase more path inflations. Therefore, network operators can set link weights between that in the default and optimized assignments, and then the power saving can be achieved between SPEED and SPEED-C.

VI. Related Work

Gupta *et al.* [14] addressed the power saving issues in Internet systems and saved powers by putting network devices into sleep modes or energy saving modes, which was realized by changes to Internet protocols. In the follow-up, the authors explored applying device sleeping in a wired LAN network [15]. Nedevschi *et al.* [24] proposed shaping the traffic into small bursts at edge routers to facilitate sleeping and rate adaptation to save powers. All these works focused on link-level power savings. However, SPEED achieves power saving and adapts to different traffic matrices by dynamic traffic aggregations in a network-wide view.

Chabarek *et al.* [7] investigated the different power demands used by Cisco routers and applied mixed integer programming to determine the optimal configuration at each router in a wide area network for a given traffic matrix. They considered energy saving from the point of the view of power awareness network design, but did not aim to propose a design of power awareness routing. SPEED realizes network-wide power savings by changing packet forwarding path according to the routing information learned from the existing routing protocols.

Zhang *et al.* [32] proposed a green traffic engineering scheme by leveraging MPLS in IP networks to shut off parallel underutilized links. Vasić *et al.* identified energy-critical paths offline to achieve energy-efficient traffic engineering [31]. Cianfrani *et al.* [9] addressed the greening issue of OSPF by changing normal OSPF. The scheme identifies the nodes with large out-degrees and aggregates traffic in these nodes. To achieve this, it requires modifying computations of SPTs and requiring announcing SPTs. Moreover, frequent routing computations incur network-wide routing convergence, which may cause routing blackholes and loops. Fisher *et al.* [11] proposed an offline algorithm to shut off cables in bundled links between two routers for the power saving purpose. Heller *et al.* [16] optimized the power consumption by turning off links for tree-based topologies using OpenFlow. In this paper, SPEED achieves energy savings by online computations. It does not require changes to routing computations in the traditional routing protocols and the IP forwarding paradigm, while ensuring safe traffic delivery to destinations.

Many studies proposed several power savings approaches for Internet services and data center networks [8], [16], [22], [30]. Chen *et al.* [8] considered connection-intensive Internet services and proposed several load-dispatching algorithms to reduce energy consumption in servers. Mahadevan *et al.* [22] addressed power saving issues by efficiently allocating jobs in data center networks.

LFA [6] explores and computes different loop-free alternates to detour around failures in networks [6], [26]. However, the effectiveness of failure detour is not guaranteed because of the restrictions of link weights. To solve the problem, Kwong *et al.* [20] proposed a centralized computation approach to compute routing without considering link weights for guaranteed network connectivity. They explore all available links in a graph to precompute routings for each destination. Moreover, two next-hops adopted by a node may not be loop-free, and thus routing loops may be formed. SPEED fully considers the restrictions of link weights in IP routing and only precomputes a limited number of loop-free next-hops to detour around pruned links. After receiving each traffic change announcement, SPEED only incrementally changes ENH assignments between different loop-free next-hops.

VII. Conclusion

This paper offers a new perspective of energy-efficient routing that does not require changes to the traditional IP packets forwarding diagram and routing protocols. The paper identifies different topological properties for graph aggregation for energy savings and proves that computing energy-efficient routings is NP-complete. To solve the problem, we propose *Safe and Practical Energy Efficient Detour Routing* (SPEED) to maximize the number of aggregated links and then aggregate traffic for power savings while avoiding network congestions. The experimental results demonstrate that SPEED can significantly save powers while not exacerbating the routing performance using real network topologies and real traffic matrices.

References

[1] Internet2, "The Abilene Topology," [Online]. Available: http://noc.net.internet2.edu/i2network/maps-documentation.html

[2] Juniper Networks, Sunnyvale, CA, USA, "Ensuring rapid restoration in Junos OS-based networks," 2010 [Online]. Available: http://www.ictnetworks.com.au/pdf/2000345-en.pdf

[3] DANTE, Cambridge, U.K., "The GEANT Topology," [Online]. Available: http://www.geant.net/Network/The_Network/Pages/Network-Topology.aspx

[4] Cisco, San Jose, CA, USA, "OSPFv2 loop-free alternate fast reroute," [Online]. Available: http://www.cisco.com/en/US/docs/ios-xml/ios/iproute_ospf/configuration/15-1s/iro-lfa-frr.html

[5] Cisco, San Jose, CA, USA, "Power management for the Cisco 12000 series router," 2009 [Online]. Available: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12spower.pdf

[6] A. Atlas, A. Zinin, R. Torvi, G. Choudhury, C. Martin, B. Imhoff, and D. Fedyk, "Basic specification for IP fast-reroute: Loop-free alternates," RFC 5286, Sep. 2008.

[7] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. IEEE INFOCOM*, 2008, pp. 457–465.

[8] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. Netw. Syst. Design Implement.*, 2008, pp. 337–350.

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

[10] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin, "LiteGreen: Saving energy in networked desktops using virtualization," in *Proc. USENIX ATC*, 2010, p. 3.

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

[12] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000, pp. 519–528.

[13] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *Proc. ACM CoNEXT*, 2005, pp. 244–245.

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

[15] M. Gupta and S. Singh, "Using low-power modes for energy conservation in ethernet LANs," in *Proc. IEEE INFOCOM*, 2007, pp. 2451–2455.

[16] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *Proc. Netw. Syst. Design Implement.*, 2010, pp. 249–264.

[17] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proc. ACM SIGCOMM*, 2005, pp. 253–264.

[18] D. R. Karger, P. N. Klein, and R. E. Tarjan, "A randomized linear-time algorithm to find minimum spanning trees," *J. ACM*, vol. 42, pp. 321–328, 1995.

[19] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (TE) extensions to OSPF version 2," RFC 3630, Sep. 2003.

[20] K.-W. Kwong, L. Gao, R. Guérin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1235–1243.

[21] Q. Li, M. Xu, Y. Yang, L. Gao, Y. Cui, and J. Wu, "Towards a safe and practical energy-efficient detour routing in IP networks CS Department, Tsinghua University, Beijing, China, Tech. rep., 2012.

[22] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "Energy aware network operations," in *Proc. Global Internet Symp.*, 2009, pp. 1–6.

[23] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, Aug. 2008.

[24] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proc. NSDI*, 2008, pp. 323–336.

[25] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," in *Proc. DRCN*, 2011, pp. 62–69.

[26] G. Retvari, J. Tapolcai, G. Enyedi, and A. Csaszar, "IP fast ReRoute: Loop free alternates revisited," in *Proc. IEEE INFOCOM*, 2011, pp. 321–328.

[27] 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.

[28] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: Understanding the causes and impact of network failures," in *Proc. ACM SIGCOMM*, 2010, pp. 315–326.

[29] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *Comput. Commun. Rev.*, vol. 36, pp. 83–86, 2006.

[30] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proc. CoNEXT*, 2009, pp. 37–48.

[31] N. Vasić, P. Bhurat, D. Novaković, M. Canini, S. Shekhar, and D. Kostić, "Identifying and using energy-critical paths," in *Proc. ACM CoNEXT*, 2011, Art. no. 18.

[32] M. Zhang, C. Yi, B. Liu, and B. Zhang, "GreenTE: Power-aware traffic engineering," in *Proc. IEEE ICNP*, 2010, pp. 1–10.

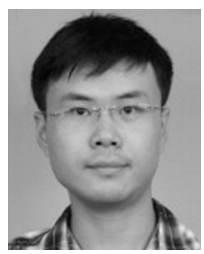
[33] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *Proc. SIGMETRICS*, 2003, pp. 206–217.

Qi Li (M'12) received the B.Sc. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2003 and 2012, respectively.

His research interests include network architecture, protocol design, and system and network security.

Mingwei Xu (M'03) received the B.Sc. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 1994 and 1998, respectively.

He is a Full Professor with the Department of Computer Science, Tsinghua University. His research interests include computer network architecture, high-speed router architecture, and network security.



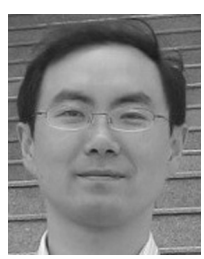
Yuan Yang (M'11) received the bachelor and master degrees in computer science from Tsinghua University, Beijing, China, in 2002 and 2006, respectively, and is currently pursuing the Ph.D. degree in computer science at Tsinghua University.

His major research interests include distributed routing protocol, computer network architecture, and the next-generation Internet.

Lixin Gao (M'98–SM'07–F'10) received the Ph.D. degree in computer science from the University of Massachusetts Amherst, Amherst, MA, USA, in 1996.

She is a Professor of electrical and computer engineering with the University of Massachusetts Amherst. Her research interests include social networks, Internet routing, network virtualization, and cloud computing.

Prof. Gao won the Best Paper Award from IEEE INFOCOM 2010 and the Test-of-Time Award from ACM SIGMETRICS 2010.



Yong Cui (M'10) received the B.E. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 1999 and 2004, respectively.

He is currently an Associate Professor with Tsinghua University. His major research interests include mobile wireless Internet and computer network architecture.

Jianping Wu (SM'05–F'12) received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 1997.

He is now a Full Professor with the Department of Computer Science, Tsinghua University. He has published more than 200 technical papers in academic journals and proceedings of international conferences in the research areas of the network architecture, high-performance routing and switching, protocol testing, and formal methods.