Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links

Will Fisher, Martin Suchara, and Jennifer Rexford
Princeton University
Princeton, NJ 08544, U.S.A.
{wafisher, msuchara, jrex}@princeton.edu

ABSTRACT

In backbone networks, the line cards that drive the links between neighboring routers consume a large amount of energy. Since these networks are typically overprovisioned, selectively shutting down links during periods of low demand seems like a good way to reduce energy consumption. However, removing entire links from the topology often reduces capacity and connectivity too much, and leads to transient disruptions in the routing protocol. In this paper, we exploit the fact that many links in core networks are actually "bundles" of multiple physical cables and line cards that can be shut down independently. Since identifying the optimal set of cables to shut down is an NP-complete problem, we propose several heuristics based on linear optimization techniques. We evaluate our heuristics on topology and traffic data from the Abilene backbone as well as on two synthetic topologies. The energy savings are significant, our simplest heuristic reduces energy consumption by 79% on Abilene under realistic traffic loads and bundled links consisting of five cables. Our optimization techniques run efficiently using standard optimization tools, such as the AMPL/CPLEX solver, making them a practical approach for network operators to reduce the energy consumption of their backbones.

1. INTRODUCTION

Reducing energy consumption has been an important part of networking research, with the most prominent topics pertaining to the energy consumption of servers and wireless devices. Unfortunately, the energy consumption of wired networks has been traditionally overlooked. The potential savings in this area are significant—powering wired networks in the United States alone costs an estimated 0.5–2.4 billion dollars per year [1]. Moreover, more energy-efficient network architectures would allow network deployments in less developed parts of the world [2]. In this work, we describe optimization techniques that allow significant energy savings in wide-area networks.

The capacity of backbone networks is overprovisioned in order to accommodate traffic shifts, and to allow rerouting when links fail. The average link utilization in backbone networks of large Internet service providers is estimated to be around 30–40%. While some overprovisioning is necessary, it is possible to reduce the energy consumption by dynamically reducing the available capacity, and bringing links back up as needed. For example, network utilization in off-peak hours may decrease by a factor of three or more, which allows for a significant reduction in capacity while still leaving enough spare bandwidth for unexpected traffic shifts. As the power consumption of backbone routers and their line cards is essentially independent of the link load [3], it is natural to consider powering off routers or line cards during periods of low utilization. Even though today's backbone routers cannot put line cards in "sleep mode," or bring shutdown interfaces back up quickly, we believe that these advances will come in the years ahead, especially if they offer a big energy savings.

In core networks, pairs of routers are typically connected by multiple physical cables that form one logical bundled link [4] that participates in the intradomain routing protocol. Bundled links are also sometimes called aggregate links or composite links, and they are standardized by IEEE 802.1AX [5]. Link bundles are prevalent because when capacity is upgraded, new links are added alongside the existing ones, rather than replacing the existing equipment with a higher-capacity link. For example, a 40 Gbps bundled link may comprise of four OC-192 cables with capacity 10 Gbps each. Bundled links are also necessary when the aggregate capacity of the bundle exceeds the capacity of the fastest available link technology. In today's backbone networks, a vast majority of links would be bundled, with bundles consisting of two to approximately twenty cables, a majority between the two extremes. Our approach is to power down individual cables in the bundle (and the line cards that serve them) during periods of low utilization. This results in significant energy savings because line cards represent a large fraction of the router energy consumption [3].

In our work, we propose that network operators run a network-management system that exploits opportunities to selectively power down individual cables in bundled links, based on the current or projected traffic

matrix. The network-management system must solve an optimization problem that takes the network topology and traffic matrix as input, and identifies the maximum number of cables that can be powered down while still having sufficient headroom for carrying the offered traffic. Since finding the optimal solution is an NP-complete problem, we introduce several heuristics whose performance we evaluate in a realistic setting. To the best of our knowledge, this is the first work that considers the reduction of power consumption by turning off cables in a bundled link. The work that is closest in spirit to ours is [6] which considers powering down entire routers and links. Their approach is similar to ours in that they also formulate the problem as an integer linear program, and solve the problem using heuristics that determine the order of edges to remove.

The rest of the paper is organized as follows. In section 2 we formulate the NP-complete optimization problem and in section 3 we describe several heuristics that solve it. In section 4 we demonstrate the performance of the heuristics on the Abilene network and on two synthetic topologies. The related work and conclusions appear in sections 5 and 6 respectively.

2. MAXIMIZING SHUTDOWN CABLES

To maximize energy savings, the network-management system solves an optimization problem that considers the network topology, a traffic matrix, and the bundle size (i.e., the number of cables per link). An optimal solution shuts down the maximum number of cables, while still routing all traffic demands on paths with sufficient bandwidth. In this section, we formulate the optimization problem and explain why a naive solution can perform quite badly.

2.1 Network Topology and Traffic Flow

The network is represented as a directed graph $G(V, E)$ that consists of a set V of routers and a set E of links, where each link $(u, v) \in E$ between two routers $u, v \in V$ has a capacity $c(u, v)$. Each link consists of B cables that can be shut down independently. For example, when links are indivisible we have $B = 1$; when a 40 Gbps link consists of four 10 Gbps links, we have $B = 4$. The traffic demand between a pair of routers is represented as a triple (s_d, t_d, h_d) where $s_d \in V$ is the ingress router, $t_d \in V$ is the egress router, and h_d is the amount of traffic exchanged between the ingress-egress router pair. Let D denote the collection of all these demands. The network topology $G(V, E)$, bundle size B, and traffic demands D are the inputs to the optimization problem.

Given the three inputs, the network-management system calculates a network configuration that employs the fewest cables such that all the traffic demands are satisfied. In particular, the output of the optimization problem is (i) a set of cables to shut down and (ii) the paths that the traffic should take over the remaining cables in the network. For any given solution, let $f_d(u, v)$ represent the flow of commodity d that is assigned to edge (u, v). The total flow assigned to edge (u, v) is denoted $f(u, v)$ and can be obtained by simply summing the corresponding flows of all demands: $f(u, v) = \sum_D f_d(u, v)$. To avoid overloading the links, the load $f(u, v)$ on link (u, v) should not exceed the corresponding link capacity $c(u, v)$.

Variable	Description
$G(V, E)$	backbone router and link representation
$\|V\|, \|E\|$	cardinality of set V and E, respectively
$c(u, v)$	capacity of edge (u, v)
B	number of cables in a bundle
D	set of demands
s_d	source of demand $d \in D$
t_d	destination of demand $d \in D$
h_d	flow requirement for demand $d \in D$
$f_d(u, v)$	flow on edge (u, v) of demand d
$f(u, v)$	total flow on edge (u, v)
n_{uv}	number of powered cables in link (u, v)

Table 1: Summary of notation.

In practice, network operators do not run their networks at full utilization, to prevent transient congestion and accommodate shifts in traffic over time. Still, in our problem formulations, we simply impose a constraint that $f(u, v)$ is less than or equal to $c(u, v)$. In practice, we envision that the network-management system would "scale up" the traffic demands D to account for variation in traffic, or "scale down" the link capacities $c(u, v)$ to have sufficient spare capacity to tolerate typical fluctuations in traffic. The network-management system may also periodically reoptimize the choice of paths, as well as the cables to power down, to adjust to larger changes in the traffic matrix over the course of the day or week. As such, in the rest of the paper, we assume that the traffic matrix and link capacities are fixed, and that the network-management system need only enforce that link load does not exceed capacity.

2.2 Naive Solution: Maximize Spare Capacity

Given that optimization problems with variables restricted to integers are usually NP-complete, a seemingly natural way to formulate the problem is to maximize the spare capacity by directing traffic over paths that minimize the *sum of loads over all links*. Then, the spare capacity on each link could be powered down, subject to the constraint of removing capacity in discrete units. Formulating the problem in this way is

useful because it leads to a simple linear-programming solution that will be used for:

(i) determining if a given network topology can satisfy all demands (important for determining whether *any* cables could be shut down),

(ii) calculating the maximum amount of spare capacity (serving as an upper bound on the energy savings of any solution constrained by the bundle size), and

(iii) providing an initial distribution of traffic that can serve as a starting point for heuristics that search for better solutions.

The resulting linear-programming problem formulation is as follows:

$$\begin{array}{ll} \min & \sum_{(u,v)\in E} f(u,v) \\ \text{s.t.} & \sum_D f_d(u,v) \le c(u,v) \quad \forall (u,v) \in E, \\ & \sum_{v\in V} f_d(u,v) = \sum_{v\in V} f_d(v,u) \quad \forall d, u \neq s_d, t_d, \\ & \sum_{v\in V} f_d(s_d,v) = \sum_{v\in V} f_d(v,t_d) = h_d \quad \forall d. \end{array} \tag{1}$$

The objective function minimizes the total flow summed over all links. A capacity constraint ensures that no edge carries more traffic flow than its capacity. A flow-conservation constraint ensures that no flow is lost or created except at the source and destination. The final constraint ensures that the sum of the flows leaving the source, or entering the destination, of commodity d sums to h_d. Obviously the optimization problem has a feasible solution exactly when the given network topology can satisfy all of the demands; therefore, we use this optimization in the next section as a building block to test feasibility after a cable is powered down.

In addition, we can use the linear-programming solution to obtain an upper bound on the energy savings of *any* feasible solution as follows. For each edge (u, v) we simply "round down" by using *one fewer* cable in the bundle than is needed to carry the traffic $f(u, v)$ obtained by the solution. For example, if a solution assigns 6.3 Gbps of traffic to an edge (u, v) whose capacity is 10 Gbps, and the edge consists of ten 1 Gbps cables, we remove four cables to have 6 Gbps of capacity. It is easy to verify that no flow assignment that satisfies all the demands can use fewer cables.

A lower bound is obtained similarly. By "rounding up" to the next discrete number of cables (e.g., from 6.3 Gbps to 7 Gbps), we obtain an approximate solution that provably satisfies all the demands, serving as a strawman heuristic for identifying which cables to shut down. Unfortunately, the following subsection shows that this technique often results in extremely suboptimal solutions, leading us to formulate our problem as an integer optimization problem.

2.3 Integer Linear Program Formulation

To see why the naive solution is suboptimal, consider the example in Figure 1. The network has $k + 1$ edges on the direct path from source S to each destination T_i, and all edges have unit capacity. The source S has a small amount of traffic to send to each destination T_i; that is, demands (S, T_i, ϵ) for $i = 1, 2, \ldots, n$, where $\epsilon \ll 1$. Similarly, S sends a small amount of traffic to B_0; that is, a demand (S, B_0, ϵ). And, each node B_i sends a small of traffic to its right neighbor; that is, demands (B_i, B_{i+1}, ϵ) for $i = 0, 1, \ldots, k - 1$ and (B_k, T_i, ϵ) for $i = 1, 2, \ldots, n$. Let us also assume that $B = 1$.

The optimal solution of the linear program (1) uses the shortest path for each demand to maximize the spare capacity. Therefore, the demands connecting S and T_i follow the shortest path with $k + 1$ hops, rather than the longer path through the B_i nodes. Similarly, each of the links in the lower path carry a small amount of traffic due to the demands with a source or destination of $B_0, B_1, \ldots, B_k$. As such, all links need to be used and hence the cost of this solution is $(k + 1)n + (k + 1) + n = k(n + 1) + 2n + 1$ cables. A better solution would place all of the demands connecting S and T_i on the lower paths through nodes B_i, so that the links along the direct paths could be shut down. This solution requires only $k + n + 1$ cables. Looking at the ratio of the two solutions, we see that

$$\lim_{k\to\infty} \frac{k(n+1) + 2n + 1}{k + n + 1} = n + 1.$$

That is, the optimal solution is better by a factor of $n + 1$.

Since the naive solution performs poorly, we consider an integer linear program that takes the bundle size B into account explicitly. This can be achieved by modifying the optimization problem (1) to minimize the number of powered cables, where $n_{uv} \in \mathbb{N}$ is the number of

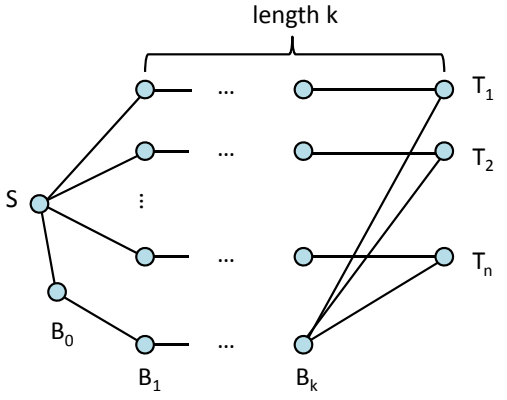

Figure 1: The simple linear program is highly suboptimal in this example.

powered cables in link (u, v). The resulting objective is

$$\text{minimize} \sum_{(u,v)\in E} n_{uv}.$$

We also need to add a constraint guaranteeing that each edge has enough active cables to carry the traffic:

$$f(u, v) \leq \left(\frac{n_{uv}}{B}\right) c(u, v) \quad \forall (u, v) \in E.$$

Note that the constraint $n_{uv} \leq B$ need not be added as it is implied by the other constraints.

This formulation is an NP-complete integer linear program. NP-hardness is obtained by reducing a less general NP-complete problem, the simple two-commodity integral flow in directed graphs (simple D2CIF) problem [7]. We only need to use two demands that correspond to the two commodities in simple D2CIF, all edges have a unit capacity, and we have one cable per link. A feasible solution of our problem correspond to a yes instance of simple D2CIF. The problem is clearly in NP as given a solution, we can verify its cost in polynomial time. To make the problem tractable, the next section introduces heuristics that can be solved almost as easily as the original linear program, and which achieve energy efficiency comparable to the optimum.

3. EFFICIENT HEURISTICS

In this section, we present three heuristics that solve the integer linear programming (ILP) formulation of section 2.3. The heuristics remove cables in certain order until no further cables can be removed. The heuristics differ in the order in which they consider the cables, as well as in the number of cable combinations they consider for removal.

3.1 Fast Greedy Heuristic (FGH)

We start by solving the linear program (1) to obtain the flow $f(u, v)$ assigned to each edge. Then we remove the maximal number of cables so that all the flows are still satisfied. After these cable removals (i.e., "rounding up"), we proceed by identifying the edge with the greatest spare capacity, i.e., we find the (u, v) for which

$$\arg\max_{(u,v)} \left(\frac{n_{uv}c(u, v)}{B} - f(u, v)\right), \tag{2}$$

where $n_{u,v}$ denotes the number of remaining cables after the cable removals. We try to remove one cable from the edge (u, v) because after the removal, the excess traffic that needs to be rerouted is the smallest. With the cable removed, we solve the linear program (1) with the new link capacities to find the new distribution of the traffic. If the problem has a feasible solution, we permanently remove the cable that was identified in 2. Otherwise, we do not remove the cable and mark the corresponding edge as final—no additional cables are removed from final edges. We continue by identifying the edge with the greatest spare capacity, ignoring all final edges, until all edges become final.

This algorithm is attractive due to its simplicity. However, it is clear that if the algorithm makes the "wrong" choice by removing a suboptimal cable, it will never backtrack to correct the mistake. The effects of this design can be seen on the topology in figure 1. When some cable (B_i, B_{i+1}) is removed, the rerouted traffic must take a long path, and the optimal set of cables can no longer be removed due to this rerouting. Therefore, the next heuristic improves on FGH by making more careful choices.

3.2 Exhaustive Greedy Heuristic (EGH)

To improve over FGH, the EGH algorithm calculates a penalty value for each candidate cable, and removes the cable that leads to the smallest increase in the penalty. The penalty considers the path that the excess traffic needs to follow after a cable is removed, and increases as traffic must flow over ever longer paths. The penalty associated with removing a cable is calculated as the difference of the objective value of the linear program (1) after and before the removal. The higher the increase of the objective, the longer the reroute must be. We call the heuristic exhaustive because the penalty allows us to do a "look-ahead" operation on each cable, and then decide how to proceed.

The EGH heuristic is similar to FGH in that the linear program (1) is solved first, and the maximal number of cables is removed. No edge is marked as final yet. Then the cable with the smallest penalty is removed if the removal results in a feasible solution. Otherwise the corresponding edge is marked as final. We continue removing cables with the smallest penalty until all edges are final. The penalty favors solutions that choose shorter paths on the presumption that satisfying demands with fewer links makes it easier to drop cables later on. This heuristic finds the optimal solution on the topology in figure 1. However, further improvements could be achieved if we consider more cable combinations for removal before making each decision.

3.3 Bi-level Greedy Heuristic (BGH)

The BGH heuristic improves on EGH by considering cables for potential removal in pairs. A penalty is associated with the potential removal of each pair of cables, and the cable pair with the smallest penalty is removed first. Should removing the pair of cables make the solution infeasible, we consider the cable pair with the second lowest penalty, and so on, until no pair of cables can be removed. Finally, the one last cable with the lowest penalty is removed, if this is possible without affecting feasibility.

The BGH heuristic is more time consuming than the

other ones because all pairs of cables need to be considered. However, the running time can be improved significantly by adding checks that stop the computation if removing *one* cable causes the optimization problem to become infeasible. If so, we need not consider the pairs of cables that include the concerned cable.

4. EXPERIMENTAL EVALUATION

In this section we evaluate the three heuristics using simulations. First we describe our experimental setup. Then we quantify the energy savings on both synthetic and realistic topologies. We also discuss the running time of the optimizations which would need to be performed by the network operator.

4.1 Experimental Setup

The heuristics were implemented as optimization problems in AMPL, and solved using the CPLEX optimization solver. The heuristics are simple to implement, and the core of the AMPL/CPLEX calculations is a repeated optimization of the linear program (1). An implementation where a separate flow variable $f_d(u, v)$ is only maintained for distinct classes of demands that share a common destination allowed us to significantly reduce the number of variables compared to maintaining a separate variable $f_d(u, v)$ for every demand. Our AMPL/CPLEX optimizations were performed on a Sun V880 server running Solaris 10 (64-bit version) with eight 1.2 GHz SPARC III CPUs.

The performance of the heuristics was evaluated on the Abilene topology and on two synthetic topologies—a two level hierarchical graph, and the Waxman graph, which are frequently used to simulate wide area network topologies. The parameters of the topologies are summarized in table 2. The Abilene topology uses realistic edge capacities, and the demands were determined by measurements. The hierarchical graph was generated by GT-ITM [8]. In the Waxman graph, the probability that two nodes are connected by an edge decays exponentially by the distance between them. This probability is fixed at $P(u, v) = ae^{-d/(bL)}$ where a and b are constants, d is the distance between u and v, and L is the maximum distance between any pair of nodes in the graph. Realistic demands were generated using a classical entropy model for urban traffic that appears in [9].

Name	Topology	Nodes	Edges	Demands
hier50	hierarchical	50	148	2,450
wax50	Waxman	50	169	2,450
Abilene	backbone	39	28	253

Table 2: Summary of network topologies.

4.2 Energy Savings

The performance of the three heuristics on the Abilene topology is depicted in figure 2. The same figure also depicts the upper and lower bound on the energy savings. How these bounds were obtained was described in section 2.2. We note that the upper bound is not tight and it is impossible to achieve it for small values of the bundle size B.

An important observation in figure 2 is that the energy savings increase sharply as B increases from 1 to 2 or 3. That is, while switching entire links on or off as suggested previously [6] would only allow us to reduce the energy consumption of the associated line cards by 54%, if all links consist of three cables, the savings increase significantly to 73%. This shows the importance of considering link bundles in energy optimizations.

We observe that the performance of the three heuristics in figure 2 is almost indistinguishable and differs by at most a couple percent. This was also the case in all the other experiments, and thus our evaluations of the synthetic topologies only depict the simplest Fast Greedy Heuristic (FGH). The power savings in the synthetic topologies are depicted in figure 3. For these topologies we reconfirm that the energy savings increase sharply as the bundle size increases to 2 or 3 cables.

4.3 Running Time

Since a network operator would need to use one of our heuristics to reoptimize the solution when the offered load changes significantly, it is important to understand the running time, which is summarized in table 3. Although the number of cables that can be removed increases proportionally with increasing bundle size B, the running time is essentially independent of the bundle size for all heuristics because the vast majority of

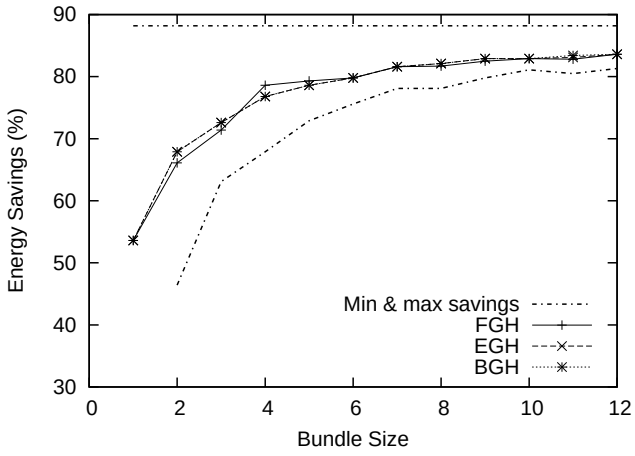

Figure 2: Energy savings on the Abilene topology as a function of the bundle size.

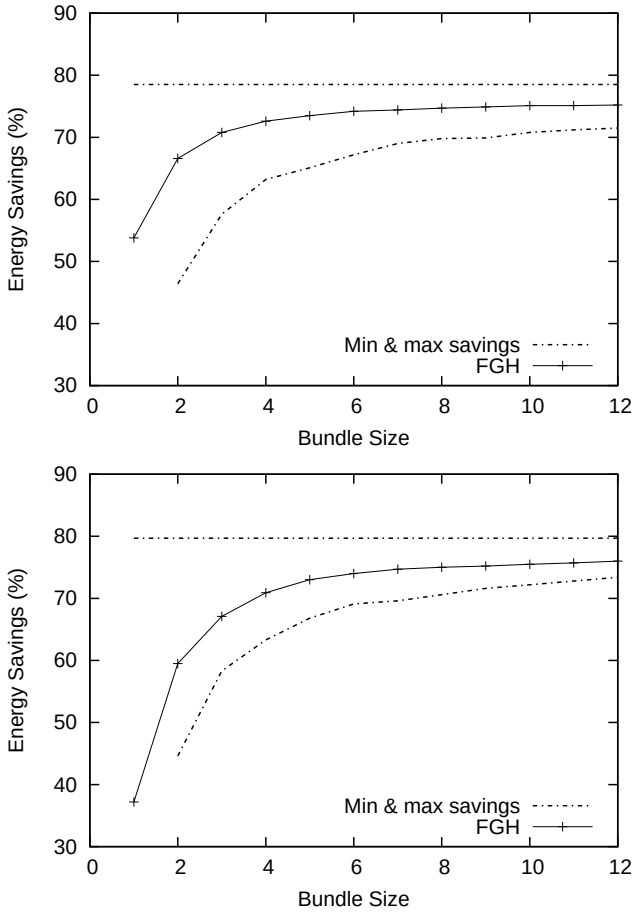

Figure 3: Energy savings of FGH on the Waxman graph (top) and the hierarchical topology (bottom).

the cables are powered down in the initial step.

The running time of the heuristics differs significantly because FGH and EGH need to solve the linear program (1) up to $O(|E|^2)$ times, and BGH up to $O(|E|^3)$ times. Moreover, in the average case FGH identifies an edge for removal after examining a constant number of edges, resulting in $O(|E|)$ executions of the linear program.

Because the energy savings of the three heuristics do not differ significantly, it is clear that the simplest heuristic (FGH) should be preferred. It can be executed rapidly while achieving significant energy savings.

5. RELATED WORK

The authors of [2] acknowledge the energy efficiency problems of wide area networks and outline two solutions that save energy by putting network interfaces and other router and switch components to sleep. In *coordinated sleeping* routers act in concert to aggregate traffic onto as few active paths as possible. The drawback is the need for a dynamic protocol. In *uncoordinated sleeping* routers only use local information at the cost of only being able to save energy during idle packet interarrival periods. Our solution combines the benefits of both solutions by relying on centralized pre-calculation of the best configuration. Another approach is used in [10] where it is proposed that network hardware should support slow-speed mode with low power consumption.

Topology	FGH	EGH	BGH
Abilene	8 ± 2 sec	50 ± 8 sec	5 ± 2 min
Waxman	50 ± 20 min	17 ± 5 hr	*
Hierarchical	14 ± 4 min	*	*

Table 3: Table of running times with mean and standard deviation. Asterisk represents impractical running time.

Cutting edge hardware designs of ethernet interfaces used in hosts and switches in *local area networks* exploit periods of inactivity to operate in low power modes. For example, Broadcom [11] as well as Intel [12] introduced network interface cards that contain programmable sleep timers that allow the link to remain off for a certain time, and algorithms have been developed to optimize the length of the idle periods [13]. Unfortunately, the power usage of current routers in *wide area networks* remains essentially independent on the actual load [3], and depends only on the sum of the power draw of the chassis and the appropriate number of line cards. Our work attempts to address this problem by developing optimization methods to put unneeded line cards to sleep. Similar approaches that allow to power down individual hardware components are [6] and [14]. These works, however, do not model bundled links. Furthermore, the greedy heuristics in [6] also differ in the order in which they consider edges for removal. Whereas our solution recalculates the flow assignment before each edge removal to identify which edge is the best to remove, their solution considers edges in a fixed order determined before the first edge removal. Although [14] solves a more general problem where traffic is rerouted on alternate paths dynamically as the demands change, they assume that the end-to-end paths are fixed and given on the input. [15] optimizes the energy consumption of data centers by powering down unneeded links and switches while still meeting performance and fault tolerance goals.

6. CONCLUSION

The Internet has experienced a tremendous growth and so has its energy consumption. This paper developed and evaluated techniques that save energy in core networks by selectively powering down individual cables of large bundled links. First we showed that

the problem formulation is an integer linear program that is NP complete. Then, we developed several easy-to-implement heuristics. We demonstrated that the AMPL/CPLEX solver can efficiently solve these heuristics, and we presented an analysis of the energy savings on several realistic topologies. As our solution can be implemented using existing hardware technologies, we believe that our heuristics will be of significant interest to the network operators community.

7. REFERENCES

[1] K. W. Roth, F. Goldstein, and J. Kleinman, "Office and telecommunications equipment in commercial buildings - volume I: Energy consumption baseline," Technical Report 72895-00, Arthur D. Little, Inc., 2002.

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

[3] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," *IEEE INFOCOM*, 2008.

[4] R. Doverspike, K. K. Ramakrishnan, and C. Chase, "Structural overview of ISP networks," in *Guide to Reliable Internet Services and Applications* (C. Kalmanek, S. Misra, and R. Yang, eds.), Springer, 2010.

[5] IEEE Computer Society, *IEEE Standard 802.1AX: Link Aggregation*, 2008.

[6] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," *IEEE ICC*, 2009.

[7] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," *IEEE FOCS*, 1975.

[8] E. W. Zegura, *GT-ITM: Georgia Tech internetwork topology models (software)*, 1996.

[9] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE Journal on Selected Areas in Communications*, 2002.

[10] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," *USENIX NSDI*, 2008.

[11] "Broadcom website." http://www.broadcom.com/.

[12] "Intel website." http://www.intel.com/.

[13] M. Gupta and S. Singh, "Energy conservation with low power modes in Ethernet LAN environments," *IEEE INFOCOM*, 2007.

[14] N. Vasić and D. Kostić, "Energy-aware traffic engineering," *EPFL Technical Report*, 2008.

[15] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving energy in data center networks," *USENIX NSDI*, 2010.