6-10-2012

Enabling Green Networking with a Power Down Approach

Brendan Mumey
Montana State University

Jian Tang
Syracuse University, jtang02@syr.edu

Saiichi Hashimoto
Montana State University

Recommended Citation

B. Mumey, J. Tang, and S. Hashimoto, "Enabling green networking with a power down approach," in 2012 IEEE International Conference on Communications, ICC 2012, June 10, 2012 - June 15, 2012, Ottawa, ON, Canada, 2012, pp. 2867-2871.

Enabling Green Networking with a Power Down Approach

Brendan Mumey, Jian Tang and Saiichi Hashimoto

Abstract—**The most straightforward way to reduce network power consumption is to turn off idle links and nodes (switches/routers), which we call the power down approach. In a wired network, especially in a backbone network, many links are actually "bundles" of multiple physical cables and line cards that can be shut down independently. In this paper, we study the following routing problem for green networking in wired networks: Given a set of end-to-end communication sessions, determine how to route data traffic through the network such that total power consumption is minimized by turning off unused cables in bundled links and nodes, subject to the constraint that the traffic demand of each session is satisfied. We present an integer linear programming to provide optimal solutions. We also present two fast and effective heuristic algorithms to solve the problem in polynomial time. It has been shown by simulation results based on the Abilene network and the NSF network that the proposed heuristic algorithms consistently provide close-to-optimal solutions.**

Index Terms—**Green networking, power efficiency, routing, power down.**

I. INTRODUCTION

The Internet has become a major source of power consumption due to fast growth of network users and new applications (such as P2P and video streaming) that bring a large volume of traffic. It has been shown by recent studies that the Internet contributes up to 10% of the worldwide power consumption [3], and have estimated that the energy usage of the US network infrastructure at between 5 and 24 TWh/year, or \$0.5-2.4B/year [9]. This has raised public concerns about electricity cost and green house gas emissions, which are known to have a negative impact on global climate. Therefore, how to make the Internet green (power efficient) has become a very important problem and has attracted extensive research attention from both industry and academia recently.

The most straightforward way to reduce network power consumption is to turn off idle links and nodes, which we call the *power down approach*. In a wired network, especially in a backbone network, many links are actually bundles of multiple physical cables and line cards that can be shut down independently. Turning off whole links might lead to poor network connectivity and unsatisfied traffic demands [5]. Therefore, we consider a new power down approach that only shuts down unused cables in bundled links.

In this paper, we exploit the power down approach for power savings from a networking perspective by studying the following routing problem in wired networks: Given a set of end-to-end communication sessions, determine how to route data traffic through the network such that total power consumption is minimized by turning off unused cables in bundled links and nodes, subject to the constraint that the traffic demand of each session is satisfied. Note that most related works on the power down approach [3], [5], [6] considered special cases of the problem studied in this paper, which will be discussed in greater details in Section II. Our major contributions are summarized as follows:

1) We study a general case of the Routing with Power down Problem (RPP) and present a Mixed Integer Linear Programming (MILP) formulation to provide optimal solutions, which can serve as the benchmark for performance evaluation.

2) We present two fast and effective heuristic algorithms to solve the problem in polynomial time.

3) We present simulation results based on the Abilene network and the NSF network to show that the proposed algorithms consistently provide close-to-optimal solutions.

II. RELATED WORK

From a networking perspective, there are two main approaches for power savings: turning off network elements (power down) and adapting link date rates to their offered traffic loads (rate adaption). In a pioneering work [7], Gupta and Singh discussed the benefits brought by a power down approach and its impact on network protocols. In a closely related work [3], the authors considered the problem of switching off network nodes and links while still ensuring full connectivity and maximum link utilization for backbone networks. Several heuristic algorithms were presented to solve the problem. Heller *et al.* studied a similar problem for data center networks with tree-like topologies in [6]. They presented a network-wide power manager and several heuristic algorithms to dynamically turn on/off network elements to satisfy varying traffic loads. In [5], the authors considered the problem of minimizing total power consumption by shutting off cables in bundled links. They proposed an ILP formulation and several heuristics to solve it and justified their effectiveness via simulations. In [9], Nedevschi *et al.* showed that even simple schemes using power down and/or rate adaptation approaches can offer substantial power savings without noticeable increase in packet loss and with a small increase in latency. In a recent paper [8], the authors conducted an extensive case study of several simple power saving algorithms by simulating a real Web 2.0 workload in a real data center network topology. Their results indicated that 16% power savings can be obtained merely by appropriately adjusting active network elements' data rates. In another recent work [1], Andrews *et al.* studied a routing problem with the objective of provisioning guaranteed bandwidth for end-to-end traffic demands while minimizing power consumption using rate adaption. They showed that

Brendan Mumey and Saiichi Hashimoto are with the Department of Computer Science at Montana State University, Bozeman, MT 59717. Email: mumey@cs.montana.edu. Jian Tang is with the Department of Electrical Engineering and Computer Science at Syracuse University, Syracuse, NY 13244. Email: jtang02@syr.edu

if the link power consumption curve is superadditive, there is no bounded approximation in general for integral routing. However, for common power cost curves such as polynomials, they proposed a constant factor approximation algorithm based on randomized rounding. In [10], the authors presented Energy-Aware Traffic engineering (EATe), a technique that takes power consumption into account while achieving the same data rates as energy-oblivious approaches.

We summarize the differences between this work and related works as follows: 1) Compared to related works on the power down approach [3], [5], [6], this paper address a more general case since a whole link is assumed to be turned either on or off completely in [3], [6] and a node is assumed to be on all the time in [5]. 2) The optimization problem studied in this work is mathematically different from those problems studied in the related works on the rate adaption approach [1], [8]. 3) In this work, we present a mathematical formulation and efficient algorithms to provide optimal and close-to-optimal solutions respectively for an optimization problem, instead of performing a measurement study to justify the benefits of the power down approach as [7], [9], [10].

III. Problem Formulation

We consider a wired computer network, where each link is used for communications in both directions. Following previous studies [5], [8], we adopt a simple power consumption model: each link l is a bundled link containing $N_l \leq N$ cables that can be turned on or off selectively. The power consumption of a cable in a bundled link l is assumed to be a constant P_l. In order for a link to be operational, both of its endpoint nodes (switches/routers) and corresponding line cards must be turned on. The power usage of an on-node v is a fixed constant P_v that includes power consumed for operating its chassis and line cards. A node will be shut down if all of its adjacent cables are turned off. The *effective capacity* of an operational link l is $c_l = \frac{x_l}{N_l} C_l$, where x_l is the number of cables turned on and C_l is the capacity of the bundled link l that is the summation of the capacities of all of its cables. We are interested in finding a routing solution for a given set of end-to-end communication sessions. A routing solution is *feasible* if according to it, the traffic load of each link does not exceed its effective capacity, the flow conservation constraint is satisfied and the traffic demand of each communication session is met. All the unused cables and nodes (i.e., carry no traffic) will be switched off to save power. The corresponding optimization problem is defined as follows.

Definition 1 (RPP): Given K end-to-end communication sessions (each with a source node s_k, a destination node t_k and a traffic demand d_k), the **Routing with Power down Problem (RPP)** seeks a *feasible* routing solution for all the communication sessions such that total network power consumption is minimized.

A. MILP Formulation

We formulate the RPP as an MILP problem to provide optimal solutions. In order to present the MILP formulation, we construct a directed graph $G(V, E)$ to model the network where each vertex $v \in V$ corresponds to a node (router) and each pair of edges $(e, \bar{e})$ sharing the same ending vertices (i.e., $e = (u, v)$ and $\bar{e} = (v, u)$) correspond to a (physical) link l which is used for communications in both directions. We use L to denote the set of such edge pairs (links) in G. Let E_v be the set of links (edge pairs) that are adjacent to node v, let E_v^{in} be the set of incoming edges into v and E_v^{out} be the set of outgoing edges from v.

We define the following decision variables for the MILP formulation.

1) $f_e^k \geq 0$: The amount of traffic on edge e for communication session k.
2) $x_l \in \{0, 1, \ldots, N_l\}$: The number of bundle cables turned on in link l.
3) $y_v \in \{0, 1\}$: $y_v = 1$ if node v is turned on; $y_v = 0$, otherwise.

MILP: RPP

$$\min \sum_{l \in L} P_l x_l + \sum_{v \in V} P_v y_v \tag{1}$$

subject to

$$\sum_{e \in E_{s_k}^{out}} f_e^k - \sum_{e \in E_{s_k}^{in}} f_e^k = d_k, \quad k \in \{1, 2, \cdots, K\}; \tag{2}$$

$$\sum_{e \in E_v^{out}} f_e^k - \sum_{e \in E_v^{in}} f_e^k = 0, \quad \forall v \in V \setminus \{s_k, t_k\},$$
$$k \in \{1, 2, \cdots, K\}; \tag{3}$$

$$\sum_{k=1}^{K} (f_e^k + f_{\bar{e}}^k) \leq \frac{x_l}{N_l} C_l, \quad \forall l = (e, \bar{e}) \in E; \tag{4}$$

$$\sum_{l \in E_v} x_l \leq M y_v, \quad \forall v \in V; \tag{5}$$

In this formulation, the objective (1) is to minimize total power consumption of the network. Constraint (2) makes sure that the traffic demand d_k of each communication session k is satisfied. Constraint (3) ensures that the conservation of flow holds. Constraint (4) ensures that the total amount of traffic carried by a link (including traffic in both directions) does not exceed its effective capacity. Constraint (5) ensures that endpoint nodes of active links are turned on (the constant M is set to the maximum number of cables connected to a node), i.e., a node is shut down only if none of its adjacent cables are turned on.

IV. Proposed Routing Algorithms

Even though solving the MILP can provide optimal solutions for the RPP, it may take exponentially long time [2], especially for large problem instances. In this section, we present two polynomial-time heuristic algorithms for the RPP, which can be used in an on-line manner.

A. Shortest Path Based Algorithm

The first algorithm is based on simple shortest path routing. Since both nodes and links have associated power consumption costs (weights), we cannot directly apply a standard shortest path algorithm, such as Dijkstra's algorithm, which deals with edge weights only. To overcome this, we make a simple graph transformation. Each vertex $v \in V$ is replaced by two new vertices v_{in} and v_{out}. We also modify the edges in the graph so that any incoming edge into v of the form (u, v) is replaced with (u, v_{in}) and any outgoing edge of the form (v, u) is replaced with (v_{out}, u). These new edges have the same power costs and capacities as the ones they replace. We also add a new edge (v_{in}, v_{out}) to E, with weight

$$w(v_{in}, v_{out}) = \begin{cases} P_v & \text{if } v \text{ is currently off;} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Let the resulting auxiliary graph be $G' = (V', E')$, after the above transformations have been done. It is easy to see that a path in G' corresponds to a path in G. The power costs incurred in the path on G' are just edge weights and are equal to the costs incurred in the corresponding path in G which includes both node and edge power costs.

In the beginning, all cables and nodes are assumed to be off. The basic idea of the proposed algorithm is to find a shortest path for communication sessions one by one, turn on nodes and cables accordingly and update edge weights in G' every time according to existing traffic and the statuses of the nodes and links. Suppose that we are trying to find a routing solution for a communication session from s to t with demand d. We define the weight for each edge $e = (u, v)$ in G' as follows: Let f_e be the existing traffic on edge e. For a link $l = (e, \bar{e})$, let $f_l = f_e + f_{\bar{e}}$, be total existing traffic on l, and let x_l be the number of cables in l that are currently turned on. Clearly, $x_l \geq \lceil \frac{N_l f_l}{C_l} \rceil$, in order to support the existing traffic. Let

$$a_e = \frac{x_l}{N_l} C_l - f_l \tag{7}$$

be the available "free" capacity on link e. We define the weight of e as

$$w(e) = \begin{cases} 0 & \text{if } a_e > 0; \\ P_l & \text{if } a_e = 0 \text{ and } x_l < N_l. \end{cases} \tag{8}$$

If $f_l = C_l$, then there is no remaining capacity on link $l = (e, \bar{e})$, and we will mark both e and $\bar{e}$ as unavailable for additional flow routing and remove them from G'. The algorithm finds a shortest path p from s to t in G' using the above edge weight functions and then augments the flow of every edge along p. The amount a of flow to augment is determined as follows: initially, a is set to the (remaining) demand between s and t left to provision. All edges on path p are examined: if $w(e) = 0$, then a is set to the minimum of a and a_e; otherwise, an additional cable must be turned on in the corresponding link l and so a is set to the minimum of a and $\frac{C_l}{N_l}$. After the augmentation, the available capacity and weight of each edge on path p are updated according to (6), (7) and (8). This process is repeated until all traffic demands are met (or no augmenting path can be found). The complete algorithm is given in Algorithm 1.

We note that after each path p augmentation of size $a > 0$ is performed, either the remaining demand r between the current (s_k, t_k) pair is satisfied ($r = a$), or there is an edge $e \in p$ which has just become saturated ($a_e = 0$) after the augmentation. The number of times this can occur for an edge is $N_e \leq N$, so the total number path augmentations done (over all of the sessions) is at most $K + N|E'|$. Each shortest path computation can be done with Dijkstra's algorithm in $O(|E'| + |V'| \log |V'|)$ time, so the time complexity of this algorithm is $O((K + N|E'|)(|E'| + |V'| \log |V'|))$.

Algorithm 1 RPP-ShortestPath

```
Input: $U = \{(s_k, t_k, d_k)\}$
 Step 1  Construct the auxiliary graph $G' = (V', E')$.
 Step 2  Calculate $w(e)$ for $e \in E'$ using (6) and (8);
 Step 3  **forall** $(s_k, t_k, d_k) \in U$
            $r := d_k$; // the remaining demand
            **while** $(r > 0)$
                Compute a shortest path $p \in G'$ from $s$ to $t$;
                **if** $(\nexists\ p \in G')$ **return** FAILURE; **endif**
                $a := r$;
                **forall** $e \in p$
                    **if** $(w(e) = 0)$
                        $a := \min(a, a_e)$;
                    **else**
                        $a := \min(a, \frac{C_e}{N_e})$;
                    **endif**
                **endforall**
                **forall** $e \in p$
                    Augment the flow of $e$ by $a$;
                    **if** $((f_e + f_{\bar{e}}) = C_{(e,\bar{e})})$
                        Remove $e$ and $\bar{e}$ from $G'$;
                    **else**
                        Update $w(e)$;
                    **endif**
                **endforall**
                $r := r - a$;
            **endwhile**
         **endforall**
 Step 4  **return** SUCCESS;
```

B. Tree-Based Algorithm

The second algorithm uses a tree structure to route all the communication sessions. The reason for using a tree for routing is that links in the tree will be used frequently and so the fixed power costs of setting up each link can be amortized over more communication sessions. For ease of computation, we choose to use a *Minimum Spanning Tree (MST)* for routing. We consider the network undirected here since every link is used for communications in both directions. Without abuse of notation, we use an undirected graph $G(V, L)$ to model the given network. Note that there is a unique path p from s_k to t_k on any MST T.

In the beginning, all cables and nodes are assumed to be off. The basic idea of our algorithm is to first weight each link $(u, v) \in L$ according to how much available capacity it has and also whether its endpoint nodes are turned on and whether it is either a source or destination of at least one of the communication sessions. Then we find an MST T and use T to route as many of the input communication sessions as possible. If not all of the sessions can be routed with T (this happens when a link in a routing path on T has no remaining capacity), then we simply update link weights based on the traffic successfully routed through T and find another MST T' to route remaining sessions. This process is repeated until either all traffic demands are satisfied, or a new MST cannot be found. Again, let f_l be the existing traffic on link l. Let $\text{off}(v) \in \{0, 1\}$ indicate whether the node v is currently turned off and let $\text{st}(v) \in \{0, 1\}$ indicate whether v is the source or destination node of any communications session. When computing the MST, we use the following link weight function for each link $l = (u, v) \in L$ (remember we treat $G(V, L)$ as an undirected graph now),

$$w(u, v) = \frac{f_l}{C_l} + \frac{\text{off}(u) - \text{st}(u) + \text{off}(v) - \text{st}(v)}{2}. \tag{9}$$

The first two terms in (9) add weight to the undirected link $l = (u, v)$ according to how much existing traffic is already present in either direction on (u, v). Links with more traffic have a higher weight and will be less likely to be selected in the MST. The remaining terms in (9) modify the weight of $l = (u, v)$ according to whether u or v are currently off (and so must be turned on for the link (u, v) to be used) and whether u or v are endpoint nodes in at least one of the communication sessions. This last contribution to the link weight is a heuristic to influence the MST T to go through source/destination nodes and by doing so, hopefully the length of the routing path on T can be reduced. We will also assume that those links l that become saturated ($f_l = C_l$) are marked unavailable and removed from the graph. The complete algorithm is given in Algorithm 2.

Algorithm 2 RPP-Tree

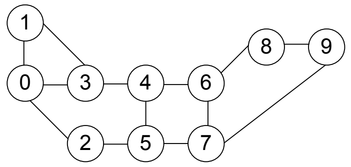
```
Input: $U = \{(s_k, t_k, d_k)\}$
 Step 1  while $U \neq \emptyset$
             Calculate $w(u, v)$ for all $(u, v) \in L$
             according to (9);
             Find a MST $T$;
             if $G$ is disconnected return FAILURE endif
             forall $(s_k, t_k, d_k) \in U$
                 Let $a$ be the available capacity on the path $p$
                 from $s_k$ to $t_k$ on $T$;
                 Augment the flow of each link on $p$ by
                 $\min(a, d_k)$;
                 Remove links $l \in p$ with $f_l = C_l$ from $L$;
                 $U := U - (s_k, t_k, d_k)$;
                 if $(a < d_k)$
                     $U := U + (s_k, t_k, d_k - a)$;
                 endif
             endforall
         endwhile
 Step 2  return SUCCESS
```

We note that in each iteration of the outer **while** loop of the algorithm, either the current (s_k, t_k, d_k) pair is successfully routed (its traffic demand is met), or there is a link $l \in L$ which has just become saturated ($f_l = C_l$). Thus, a simple bound on the number of iterations is $K + |L|$. The running time of each iteration is dominated by the time required to find the current MST T, which can be done within $O(|L| + |V| \log |V|)$ time using Prim's algorithm. So the time complexity of this algorithm is $O((K + |L|)(|L| + |V| \log |V|))$.

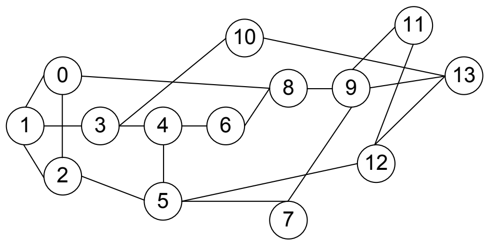
V. Simulation Results

In this section, we present simulation results to show the performance of the proposed algorithms. The software ILOG CPLEX 10.1 [4] was used to solve all the MILP problems.

Similar as in [1], the simulation runs were performed on two well-known network topologies: the Abilene research network with 10 nodes and 13 links, and the NSF network with 14 nodes and 20 links, which are shown in Fig. 1. Every



(a) The Abilene Network



(b) The NSF Network

Fig. 1. The network topologies

communication session had randomly selected source and destination nodes, and a randomly generated traffic demand. The link capacity values, and link and node power consumption values are given in Table I, which were set according to the measurement results presented in [8].

TABLE I
SIMULATION SETTINGS

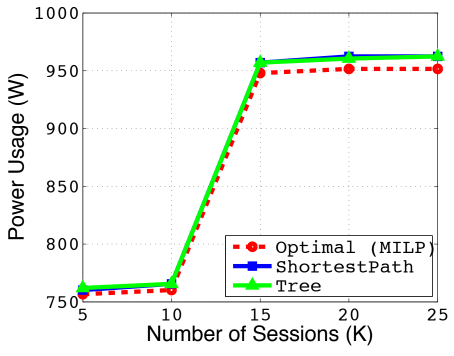
Parameter	Value
Link capacity (C_l)	3Gbps
The number of cables in each link (N_l)	3
Cable power usage (P_l)	1.8W
Node power usage (P_v)	93W

We tested the performance of the proposed algorithms in both regular traffic and heavy traffic cases. Specifically, we performed simulation runs based on two scenarios for each network: In scenario 1, each communication session had a randomly generated traffic demand that was uniformly distributed in $[0, 100]$ Mbps. In scenario 2, we increased traffic demands by setting the traffic demand range to $[0, 200]$ Mbps. In each simulation run, K communication sessions were created; K was increased from 5 sessions to 25 sessions with a step size of 5. The optimal solution was obtained by solving the MILP problem given in section III-A. The simulation results are presented in Fig. 2–3. Note that CPLEX was not able to find the optimal solutions to the MILP problems on the NSF network in the heavy traffic case due to very long running time. So in Fig. 3(b), we only present simulation results corresponding to the two heuristic algorithms.

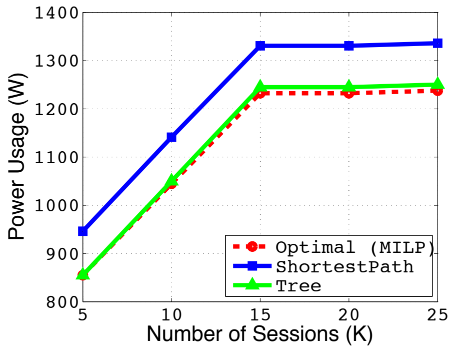
We make the following observations from the simulation results.

1) From the figures, we observe that both the shortest path based algorithm and the tree based algorithms consistently produce close-to-optimal solutions in both regular and heavy traffic cases. Specifically, the average difference between the power usage values given by the shortest path based algorithm and the optimal ones is only 4.4%. For the tree based algorithm, the average difference is 0.7%. An interesting observation is that the tree based algorithm performs slightly better the shortest path algorithm.

2) As expected, the power usage values given by both algorithms increase monotonically with the number of communication sessions in both scenarios since more communication sessions lead to heavier traffic loads therefore higher power usages. However, we notice that the power usage usually levels off after certain points. This is because in the beginning, with more traffic introduced into the network, more nodes and cables usually need to be turned on to support them, leading to significant increase on power usage; however, after certain points, most of newly introduced traffic can be supported by using available capacities on nodes and cables that have already been turned on, which will not cause significant power usage increase.

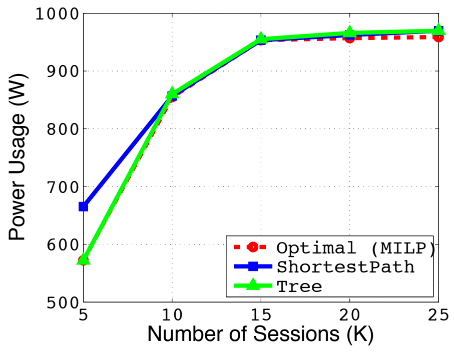


(a) The Abilene network

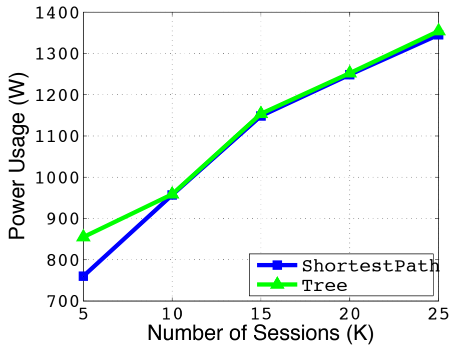


(b) The NSF network

Fig. 2. Scenario 1 (regular traffic): Power Usage VS. K



(a) The Abilene network



(b) The NSF network

Fig. 3. Scenario 2 (heavy traffic): Power Usage VS. K

VI. Conclusions

In this paper, we examined how to leverage the power down approach for green networking by studying a novel routing problem, namely, the RPP. We presented an MILP formulation for the RPP to provide optimal solutions. Then we presented a shortest-path based algorithm and a tree based routing algorithm to solve it in polynomial time. Our simulation results showed both algorithms consistently provide close-to-optimal solutions and the tree based algorithm slightly outperforms the shortest path based algorithm on the Abilene network and the NSF network.

References

[1] M. Andrews, A. F. Anta, L. Zhang and W. Zhao, Routing for power minimization in the speed scaling model, *Proceedings of IEEE Infocom'2010*.

[2] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows (3rd edition)*, John Wiley & Sons, 2005.

[3] L. Chiaraviglio, M. Mellia and F. Neri, Reducing power consumption in backbone networks, *Proceedings of ICC'2009*.

[4] ILOG Software Inc., CPLEX 10.1, http://www.ilog.com

[5] W. Fisher, M. Suchara and J. Rexford, Greening backbone networks: reducing energy consumption by shutting off cables in bundled links, *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, 2010.

[6] B. Heller *et al.* , ElasticTree: saving power in data center networks, *Proceedings of USENIX NSDI'2010*.

[7] M. Gupta and S. Singh, Greening of the Internet, *Proceedings of ACM SIGCOMM'2003*, pp. 19–26.

[8] P. Mahadevan, P. Sharma, S. Banerjee and P. Ranganathan, Power aware network operations, *Proceedings of IEEE Infocom'2009*, pp. 25–30.

[9] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy and D. Wetherall, Reducing network power consumption via sleeping and rate-adaptation, *Proceedings of USENIX NSDI'2008*.

[10] Nedeljko Vasic and Dejan Kostic, Power-aware traffic engineering, *EPFL Technical Report NSL-REPORT-2008-004*.