5-10-2012

On Exploiting Flow Allocation with Rate Adaptation for Green Networking

Jian Tang
Syracuse University, jtang02@syr.edu

Brendan Mumey
Montana State University

Yun Xing
Syracuse University

Andy Johnson
Montana State University

On Exploiting Flow Allocation with Rate Adaptation for Green Networking

Jian Tang, Syracuse University
Brendan Mumey, Montana State University
Yun Xing, Syracuse University
Andy Johnson, Montana State University

Index Terms: Green networking, power efficiency, flow allocation, rate adaptation.

Abstract

Network power consumption can be reduced considerably by adapting link data rates to their offered traffic loads. In this paper, we exploit how to leverage rate adaptation for green networking by studying the following flow allocation problem in wired networks: Given a set of candidate paths for each end-to-end communication session, determine how to allocate flow (data traffic) along these paths such that power consumption is minimized, subject to the constraint that the traffic demand of each session is satisfied. According to recent measurement studies, we consider a discrete step increasing function for link power consumption. We address both the single and multiple communication session cases and formulate them as two optimization problems, namely, the Single-session Flow allocation with Rate Adaptation Problem (SF-RAP), and the Multisession Flow Allocation with Rate Adaptation Problem (MFRAP). We first show that both problems are NP-hard and present a Mixed Integer Linear Programming (MILP) formulation for the MF-RAP to provide optimal solutions. Then we present a 2-approximation algorithm for the SF-RAP, and a general flow allocation framework as well as an LP-based heuristic algorithm for the MF-RAP. Simulation results show that the algorithm proposed for the SF-RAP consistently outperforms a shortest path based baseline solution and the algorithms proposed for the MF-RAP provide close-to-optimal solutions.

I. Introduction

Due to fast growth of network users and their communication demands, the Internet has become a major contributor for power consumption. Recent studies have shown that the Internet accounts for up to 10% of the worldwide power consumption [7], and have estimated that the power usage of the US network infrastructure at between 5 and 24 TWh/year, or $0.5-2.4B/year [20]. This has raised public concerns about electricity cost, and green house gas emissions which is known to have a negative impact on global climate. Therefore, green (power efficient) networking has attracted extensive research attention from both industry and academia recently.

Generally speaking, operating a device at a lower frequency can enable a dramatic reduction in energy consumption. It has been shown by recent research [20] that power consumption can be reduced considerably by lowering link data rates. For example, Ethernet links dissipate between 1-4W when operating at between 100Mbps and 1Gbps compared to 10- 20W at 10Gbps [20]. Therefore, if network control software can adapt link data rates to their offered traffic loads (i.e., tune each link to a high rate state when its traffic load is high and tune it down to a low rate state when its traffic load is low), then significant power savings can be achieved. This technique is known as rate adaptation (a.k.a speed scaling) [20], which has been studied previously mostly at a single element level (e.g., a CPU or a multi-CPU server).

In this paper, we exploit how to leverage rate adaptation for power savings from a networking perspective by studying the following flow allocation problem in wired networks: Given a set of candidate paths for each end-to-end communication session, determine how to allocate flow (data traffic) along these paths such that power consumption is minimized, subject to the constraint that the traffic demand of each session is satisfied. Our goal is not to propose any new routing algorithms or protocols but to design standard-compliant flow allocation algorithms that can work together with standard routing protocols. Specifically, for each end-to-end communication session, a standard routing protocol can be used to find a set of candidate paths and then our flow allocation algorithms can be employed to distribute traffic load over these paths. For example, one of the variants of a widely used routing protocol, OSPF, is called OSPF-ECMP (Equal-Cost MultiPath) [21], which can find equal cost paths for each source-destination pair in terms of fixed measures such as line speed or hop count. In addition, according to recent measurement studies [18], [22], the power consumption of a link *l* can be approximated by a discrete step increasing function $W(f_l)$ of its traffic load f_l. Essentially, each link *l* and corresponding switching ports in two ending routers can work in one of J_l states, each of which has a corresponding capacity c_j $(j \in \{1, \cdots, J_l\})$ and a corresponding fixed power consumption w_j such that $w_{j-1} < w_j$ and $c_{j-1} < c_j$ $(j \in \{2, \cdots, J_l\})$. If the traffic load of a link *l* is f_l such that $c_{j-1} < f_l \leq c_j$, then it has to work in state *j*, leading to a power consumption of w_j.

First, we study the single communication session case where a new communication session with a single source and a single destination, and existing traffic load on each link are given. The problem is to find a feasible flow allocation to minimize incremental power consumption. Then we address the multi-session case where multiple communication sessions are given in advance and the objective is to minimize the total power consumption. We formulate them as two optimization problems, namely, the Single-session Flow allocation with Rate Adaptation Problem (SF-RAP), and the Multi-session Flow Allocation with Rate Adaptation Problem (MF-RAP). Note that an algorithm for the SF-RAP can be used in a network with highly dynamic and hard-to-predict traffic demands (i.e., communication sessions arrive and leave the network frequently) such as an enterprise network. However, an algorithm for the MF-RAP can be applied in a network with relatively stable and predictable traffic demands such as a backbone network.

To the best of our knowledge, we are the first to study flow allocation with rate adaptation problems in wired networks with objective of minimizing power consumption, and propose provably good solutions. Our major contributions are summarized as follows:

1) We show that the SF-RAP and MF-RAP problems are NP-hard and present a Mixed Integer Linear Programming (MILP) formulation for the MF-RAP to provide optimal solutions.

2) We present a greedy algorithm for a special case of the SF-RAP where the given candidate paths are link-disjoint. This algorithm has an approximation ratio of 2, i.e., it always finds a solution with an objective value at most *2OPT*, where *OPT* is the corresponding optimal objective value. We then present an approximation scheme for the general case that has the same performance bound.

3) We present a general flow allocation framework as well as an LP-based heuristic algorithm for the MF-RAP problem.

4) We present extensive simulation results to show that the greedy algorithm for the SF-RAP consistently outperforms a shortest path based baseline solution and the algorithms for the MF-RAP provide close-to-optimal solutions.

The rest of this paper is organized as follows. We discuss related work and highlight the differences between this work and these related works in Section II. We describe the system model, and present the NP-hardness proof and the MILP formulation in Section III. The proposed flow allocation algorithms are presented in Section IV. We present simulation results in Section V and conclude the paper in Section VI.

II. Related Work

Recently, green networking has attracted tremendous research attention. From a networking perspective, there are mainly two approaches for power saving: switching off network elements (sleeping) and rate adaptation. In [20], Nedevschi *et al*. showed that even simple schemes for sleeping or rate-adaptation can offer substantial power savings without noticeably increasing loss and with a small increase in latency. In a closely related work [3], Andrews *et al*. studied a routing problem with the objective of provisioning guaranteed bandwidth for a given traffic demand matrix while minimizing power consumption using rate adaptation. They showed that 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 another recent paper [18], 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. In a pioneering work [16], Gupta and Singh discussed the impact of saving power on network protocols by putting network elements to sleep. Using sample packet traces, they first showed that it is indeed reasonable to do this and then they discussed the changes that may need to be made to current Internet protocols to support a more aggressive strategy for sleeping. In [7], the authors considered the problem of switching off network nodes and links while still guaranteeing full connectivity and maximum link utilization for backbone networks. They proposed several heuristic algorithms to solve the problem. Heller *et al*. studied a similar problem for data center networks with tree-like topologies in [11]. They presented a network-wide power manager and several heuristic algorithms to dynamically adjust active network elements to satisfy varying traffic loads. In [22], the authors presented Energy-Aware Traffic engineering (EATe), a technique that takes power consumption into account while achieving the same data rates as the energy-oblivious approaches.

Mathematically, several network flow problems studied in the literature are related to our problems. The mostly related problem is the minimum cost multi-commodity flow problems with an arbitrary discrete step increasing cost function on links. Several exact solutions and heuristic algorithms were introduced in [12] and [19] respectively. Special cases of this problem are the capacitated network design problems which seek a flow routing solution to minimize the total network facility cost subject to the constraint that all end-to-end traffic demands are satisfied. An optimal algorithm was presented by Magnanti *et al*. to solve the single-facility single-flow version of

the problem in [17]. Chopra *et al*. presented an optimal algorithm to solve the two-facility single-flow version of the problem, under the assumption of bounded traffic demand to link capacity ratios, in a later paper [8]. Exact solutions and heuristic algorithms were presented to solve the general cases, i.e., multi-facility multi-commodity cases in [1], [10]. Other related problems include the buy-at-bulk problems which are similar to the capacitated network design problems but address a general subadditive link cost function. Constant factor approximation algorithms were presented for single-flow versions of the problem in [14], [15] and approximation algorithms with logarithmic approximation ratios were presented for multi-commodity versions [2], [4], [6].

We summarize the differences between our work and these related works as follows: 1) We study flow allocation problems which are different from routing problems studied by most related works. 2) Generally, network flow problems are NP-hard. Most related works presented heuristic algorithms [1], [7], [11], [18], [19], [20], which cannot provide any performance guarantees. Our work, however, presents a 2-approximation algorithm (i.e., If the problem is a minimization problem, then the value of a solution given by the algorithm is guaranteed to be no larger than twice the optimal value.) 3) This work considers a practical discrete step increasing function for link cost which is more general than that of the capacitated network design problems [1], [8], [10], [17], and may not be a subadditive function which is assumed for the buy-at-bulk problems [2], [4], [6], [14], [15].

III. Problem Formulation

In this section, we define the problems, show the SF-RAP is NP-hard and present MILP formulations.

We consider a wired computer network where each link is used for communications in both directions. As mentioned before, the power consumption of a link l is given by a discrete step increasing function $\mathcal{W}(f_l)$ of its traffic load f_l:

$$\mathcal{W}(f_l) = \begin{cases} w_1 & \text{if } 0 \le f_l \le c_1; \\ w_2 & \text{if } c_1 < f_l \le c_2; \\ \cdots & ; \\ w_j & \text{if } c_{j-1} < f_l \le c_j; \\ \cdots & ; \\ w_{J_l} & \text{if } c_{J_l-1} < f_l \le c_{J_l}. \end{cases} \quad (1)$$

where J_l is the total number of states a link l can work on, and c_j and w_j are the capacity and the power consumption of link l on state j ($j \in \{1, \cdots, J_l\}$) respectively. Note that we have $w_{j-1} < w_j$ and $c_{j-1} < c_j$ for $j \in \{2, \cdots, J_l\}$.

To clarify notation, we use the vector notation $\hat{f} = \langle f_p \rangle$ to specify the path flow f_p for each given routing path $p \in P$. We also assume that each link l may have some existing traffic load already; this is specified by f_l^{old}. We also define the incremental power consumption of augmenting a flow on a path p by an amount f by

$$\mathcal{W}_p^+(f) = \sum_{l \in p} [\mathcal{W}(f_l^{old} + f) - \mathcal{W}(f_l^{old})]. \quad (2)$$

Definition 1 (SF-RAP): Given the load and state of each link for existing traffic in the network, and a communication session with source node *s*, destination node *t*, traffic demand *d*, and a set *P* of *s*−*t* candidate paths, **the Single-session Flow Allocation with Rate Adaptation Problem (SF-RAP)** seeks a flow allocation $\hat{f}$ that specifies the amount f_p of traffic routed through each given path $p \in P$ such that the incremental power consumption is minimized subject to the constraint that the traffic demand is satisfied.

Definition 2 (MF-RAP): Given *K* communication sessions (each with source node s_k, destination node t_k, traffic demand d_k, and a set P_k of $s_k - t_k$ candidate paths), the **Multi-session Flow Allocation with Rate Adaptation Problem (MF-RAP)** seeks a flow allocation $\langle \hat{f}^k \rangle$ that specifies the amount of traffic f_p^k routed through each given path $p \in P_k$ for every session *k* such that the total network power consumption is minimized subject to the constraint that the traffic demand of each session is satisfied.

A. Computational Complexity

In this section, we show that the SF-RAP is NP-hard via a reduction from the SUBSET-SUM problem ([13]). An instance of this problem is a set of real numbers $X = \{x_i\}_{i=1}^n$ (we may assume $0 \le x_i \le 1$) and a target value *q*. The problem is to determine if there is a subset $Y \subset \{1, \ldots, n\}$ such that $\sum_{i \in Y} x_i = q$.

Theorem 1: The SF-RAP is NP-hard.

Proof: We can reduce an instance of the SUBSET-SUM problem to an instance of the SF-RAP problem as follows: Let $\mathcal{W}(f_l)$ be defined as in equation (1) with

$$c_1 = q + 1, \qquad w_1 = q + 1$$
$$c_2 = c_1 + x_1, \qquad w_2 = w_1 + x_1$$
$$c_3 = c_2 + q/2, \qquad w_3 = w_2 + q$$
$$c_{2i} = c_{2i-1} + x_i, \qquad w_{2i} = w_{2i-1} + x_i$$
$$c_{2i+1} = c_{2i} + q/2, \qquad w_{2i+1} = w_{2i} + q$$

for 1 < i ≤ n. Let $P = \{p_i\}_{i=1}^{n}$ with path $p_i = s \to v_i \to t$. Let $f_{l_i}^{old} = c_{2i-1}$ and $f_{l_i'}^{old} = 1$ be the existing traffic loads on the first link, $l_i = (s, v_i)$, and the second link, $l_i' = (v_i, t)$, on path pi, respectively. Let the required demand $d = q$. We claim that the SUBSET-SUM instance (X, q) is solvable if and only if the minimum power flow allocation solution $\hat{f}$ to the SFRAP instance (P, d) has an incremental power consumption of q. To see this, consider the incremental power consumption, $\mathcal{W}_{p_i}^{+}(x)$, required to augment path p_i by an amount $x \le q$. Observe that $\mathcal{W}_{p_i}^{+}(x) \ge x$ with $\mathcal{W}_{p_i}^{+}(x) = x$ if and only if $x = x_i$. Thus, the only way that a total flow of q with an incremental power consumption of q can be achieved is if there is a set of path indices Y such that $\sum_{i \in Y} x_i = q$.

Since SF-RAP can be regarded as the special case of MFRAP where $K = 1$, it follows that MF-RAP is also NP-hard.

B. MILP Formulation

In this section, we formulate the MF-RAP as an MILP problem to provide optimal solutions, which can serve as the benchmark for performance evaluation. The MILP formulation is not trivial due to the discrete step increasing function for link power consumption. The SF-RAP can also be formulated as an MILP problem. However, as mentioned before, the SF-RAP is defined for networks with dynamic traffic. However, solving an MILP formulation for an instance of SF-RAP sequentially for a sequence of communication sessions randomly arrived at the network may not yield the optimal overall power consumption. So we do not present such an MILP formulation here.

In order to present the MILP formulation for the MF-RAP, 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.

We define the following decision variables for the MF-RAP formulation.

1) $f_p^k \ge 0$: The amount of traffic routed via path p for session k.
2) $x_l^j = \{0, 1\}$: $x_l^j = 1$ if link l works in state j; 0, otherwise.

MILP1: MF-RAP

$$\min \sum_{l} \sum_{j=1}^{J_l} (w_j x_l^j) \tag{3}$$

Subject to:

$$\sum_{p \in P_k} f_p^k = d_k, \quad k \in \{1, \ldots, K\}; \tag{4}$$

$$f_l = \sum_{k=1}^{K} \Big(\sum_{p \in P_k : e \in p} f_p^k + \sum_{p \in P_k : \bar{e} \in p} f_p^k \Big), \quad \forall l = (e, \bar{e}), \; k \in \{1, \ldots, K\}; \tag{5}$$

$$f_l \le c_{J_l}, \quad \forall l = (e, \bar{e}) \in L; \tag{6}$$

$$\frac{(c_j - f_l)}{c_{J_l}} \le \sum_{i=1}^{j} x_l^i \le 1 + \frac{(c_j - f_l)}{c_{J_l}}, \quad \forall l = (e, \bar{e}) \in L, j \in \{1, \ldots, J_l\}. \tag{7}$$

In this formulation, the objective (3) is to minimize total power consumption of the network. Constraint (4) makes sure that the traffic demand d_k of each communication session k is satisfied. Variable f_l is used to give the total amount of traffic carried by link l, which cannot exceed the maximum link capacity c_{J_l} (the capacity of link l working on its highest state). This is ensured by constraint (6). Constraint (7) establishes the relationship between link load variables f_l and link state variables x_l^j according to the discrete step increasing function given in equation (1).

IV. PROPOSED FLOW ALLOCATION ALGORITHMS

In this section, we present algorithms for both the SF-RAP and MF-RAP.

A. Proposed Algorithms for the SF-RAP

For the SF-RAP, we begin with a simple greedy algorithm for the case when the paths in *P* are link-disjoint. This algorithm will then be used as a subroutine for the general case when paths can share links. Let *p* be a candidate path in *P*. Each link l in *P* carries some existing traffic f_l^{old}. As the algorithm proceeds, it will augment flow along paths in *P*. Let $f \geq 0$ be an amount of contemplated augmented flow from *s* to *t* on path *p*. The incremental cost of this augmentation, $\mathcal{W}_p^+(f)$, is defined by (2). Observe that $\mathcal{W}_p^+$ is a step-function composed of the sum of shifted copies of the original link power consumption function $W(\cdot)$ given by equation (1). For infeasible values of f (values f such that there is an $l \in P$ with $f_l^{old} + f > c_{J_l}$), we define $\mathcal{W}_p^+(f) = \infty$. Let the step points of this function be $0 < c_1^p < c_2^p < \ldots < c_{K_p}^p$, where K_P is the number of steps, i.e. $\mathcal{W}_p^+(c_i^p) < \mathcal{W}_p^+(c_i^p + \epsilon)$ for any $\epsilon > 0$.

We define the cost-to-benefit ratio of a path augmentation size f on path *p* as

$$r_p(f_p^{cur}, f, d') = \frac{\mathcal{W}_p^+(f) - \mathcal{W}_p^+(f_p^{cur})}{\min(d', f - f_p^{cur})}, \qquad (8)$$

where $0 \leq d' \leq d$ is the remaining demand to satisfy between *s* and *t* and $f_P^{cur} < f$ is the current flow allocated along path *p*. The idea of the algorithm is to find a path flow augmentation that has the best (lowest) cost-to-benefit ratio and perform the augmentation. At the beginning of each iteration, we also check to see if performing a single remaining augmentation leads to an improved solution, and if so we keep it. This process is repeated until either the total demand *d* between *s* and *t* is satisfied, or there are no remaining augmentations available. We define the total incremental power consumption cost of a flow $\hat{f}$ as

$$\text{cost}(\hat{f}) = \sum_p \mathcal{W}_p^+(f_p).$$

The complete algorithm for the link-disjoint path case is presented in Algorithm 1 (recall d' is the remaining demand).

Algorithm 1 SF-RAP-Greedy-DP(P, d)

Step 1 **for** $p \in P$ Let $\hat{f}^g = 0$; **endfor**
Let $d' = d$;
Step 2 **while** ($d' > 0$ **and**
$\exists (p, i)$ s.t. $c_i^p > f_p^g$ and $r_p(f_p^g, c_i^p, d') < \infty$)
Let $\hat{f}^b = \hat{f}^g$; $f_{p_1}^b = f_{p_1}^g + d'$;
for $p \in P \setminus \{p_1\}$
Let $\hat{f}^z = \hat{f}^g$; $f_p^z = f_p^g + d'$;
if $\text{cost}(\hat{f}^z) < \text{cost}(\hat{f}^b)$
$\hat{f}^b = \hat{f}^z$;
endif
endfor
Select $(p', i') = \text{argmin}_{\{(p,i): c_i^p > f_p^g\}} r_p(f_p^g, c_i^p, d')$;
Let $f = \min(d', c_{i'}^{p'})$;
$f_p^g = f_p^g + f$;
$d' = d' - f$;
endwhile
Step 3 **if** ($d' = 0$)
if $\text{cost}(\hat{f}^g) < \text{cost}(\hat{f}^b)$
$\hat{f}^b = \hat{f}^g$;
endif
Augment path flows by $\hat{f}^b$;
return SUCCESS;
else return FAILURE;
endif

Lemma 1: The SF-RAP-Greedy-DP algorithm is a 2- approximation algorithm for the special case where the candidate paths in *P* are link-disjoint, running in $O(dm_P)$ time, where m_P is the total number of links in *P*.

Proof: Suppose that SF-RAP-Greedy-DP algorithm returns a flow solution $\hat{f}^b$ and let $\hat{f}^*$ be an optimal (least incremental power consumption cost) solution. The algorithms proceeds by choosing a path p and flow augmentation (to a step point $c_i^{p'}$) that minimizes the cost-to-benefit ratio $r_p(f_p^g, c_i^{p'}, d')$. Let the path augmentations

performed be $(p_1, i_1), (p_2, i_2), \ldots, (p_n, i_n)$ and let the residual demand just before augmentation (p_j, i_j) be d'_j and flow assigned to path *p* after this augmentation be $f_p^{g,J}$. Additionally, the algorithm tests whether adding all the remaining demand d' to some path *p* to the current solution yields an improved solution to the best found so far (this check is performed by the inner for loop in Step 2). We consider the point at which there first exists a path p such that $d'_{j+1} < (f_p^* - f_p^{g,J})$ (if this does not occur then $\hat{f}^g = \hat{f}^*$, since d′ → 0). Suppose this occurs at *j* = *k*. It follows that for all $p \in P, 1 \le j \le k$,

$$r_p(f_p^{g,J-1}, f_p^*, d'_j) = \frac{\mathcal{W}_p^+(f_p^*) - \mathcal{W}_p^+(f^{g,J-1})}{f_p^* - f^{g,J-1}}.$$

Let $A = \{p \in P : f_p^{g,k} \le f_p^*\}$ and let *B* = *P* \ *A*. For $p \in A$, $f_p^{g,k} \le f_p^*$ and so the augmentation (p, f_p^*) is an option available to the greedy algorithm that was not chosen to this point. This implies that $\exists M \ge 0$ such that for $p' \in A, 1 \le j \le k$,

$$r_{p_j}(f_{p_i}^{g,J-1}, c_{p_j}^{i_j}, d'_j) \le M \le r_{p'}(f_{p'}^{g,J-1}, f_{p'}^*, d'_j).$$

It follows that for all $p \in P, p' \in A$,

$$\frac{\mathcal{W}_p^+(f_p^{g,k}) - \mathcal{W}_p^+(f_p^*)}{f_p^{g,k} - f_p^*} \le M \le \frac{\mathcal{W}_{p'}^+(f_{p'}^*) - \mathcal{W}_{p'}^+(f_{p'}^{g,k})}{f_{p'}^* - f_{p'}^{g,k}}. \qquad (9)$$

Observe that $\sum_B (f_p^{g,k} - f_p^*) \le \sum_A (f_p^* - f_p^{g,k})$ since $d'_k \ge 0$. Next, we establish an important inequality,

$$\begin{aligned}\sum_{p\in B}[\mathcal{W}_p^+(f_p^{g,k}) - \mathcal{W}_p^+(f_p^*)] &\le M \sum_{p\in B}(f_p^{g,k} - f_p^*) \\ &\le M \sum_{p\in A}(f_p^* - f_p^{g,k}) \\ &\le \sum_{p\in A}[\mathcal{W}_p^+(f_p^*) - \mathcal{W}_p^+(f_p^{g,k})],\end{aligned}$$

where the first and last inequalities follow from (9). Thus,

$$\begin{aligned}\text{cost}(\hat{f}^{g,k}) &= \sum_{p\in A}\mathcal{W}_p^+(f_p^{g,k}) + \sum_{p\in B}\mathcal{W}_p^+(f_p^{g,k}) \\ &= \text{cost}(\hat{f}^*) + \sum_{p\in B}[\mathcal{W}_p^+(f_p^{g,k}) - \mathcal{W}_p^+(f_p^*)] \\ &\quad - \sum_{p\in A}[\mathcal{W}_p^+(f_p^*) - \mathcal{W}_p^+(f_p^{g,k})] \\ &\le \text{cost}(\hat{f}^*).\end{aligned}$$

After the first k greedy augmentations have been made, there is some *p* such that $d'_{k+1} \le f_p^* - f_p^{g,k}$. Thus the remaining demand can be met by augmenting the current flow $f_p^{g,k}$ by d'_{k+1} to achieve a new flow $\hat{f}^b$ at the power cost of

$$\begin{aligned}\text{cost}(\hat{f}^b) &= \text{cost}(\hat{f}^{g,k}) + \mathcal{W}_p^+(f_p^{g,k} + d'_{k+1}) - \mathcal{W}_p^+(f_p^{g,k}) \\ &\le \text{cost}(\hat{f}^{g,k}) + \mathcal{W}_p^+(f_p^*) \\ &\le 2\text{cost}(\hat{f}^{g,k}).\end{aligned}$$

Since the algorithm does not know when the condition $d'_{k+1} \le f_p^* - f_p^{g,k}$ occurs, it checks each possible single-path remaining-demand fulfilling augmentation before each greedy augmentation and so the algorithm is guaranteed to find the above solution $\hat{f}^b$ whose power consumption cost is at most twice optimal.

The running time of the SF-RAP-Greedy-DP algorithm is $O(dm_P)$ since we can compute the path step points $c_1^p, c_2^p, \ldots$ for each path $p \in P$ within $O(m_P)$ time and the number of greedy augmentations is $O(d)$. After each greedy augmentation we need to update the cost-to-benefit ratios for available step points c_i^p on the augmented path and check each path for a remaining demand augmentation. This can all be done in $O(m_P)$ time so the overall time complexity is $O(dm_P)$.

We next discuss an approach to solve the more general case where *P* may contain paths that share links. The idea of the approach is to first group paths that share links together and then calculate an approximate aggregate power consumption cost function for each group. We then treat each group as if it were a single path with this power consumption cost function and apply the SF-RAP-Greedy-DP algorithm. Let *P* = *N*∪*D*, where any path in *D* is link-disjoint from any other path in *P* and any path in *N* share one or more links with another path in *P*. We consider the

graph $G_N = (N, E_N)$, where the vertices are the paths in *N* and $(p_i, p_j) \in E_N$ if paths p_i and p_j share a link. Let the connected components of G_N be $N_1, \ldots, N_c$. We will refer to these connected components as *bundles*. The approach is to consider the power consumption cost of delivering varying amounts of flow in each bundle. In order to control the number of combinations examined, we will allocate flow in discrete units of size *h*. Suppose that we are contemplating how best to allocate *r h*-units of flow among the paths in bundle N_i. Let $|N_i| = n_i$ and let x_i be the number of *h*-units carried by the *i*-th path in N_i. Then the number of ways to allocate the flow is just the number of distinct nonnegative integer solutions to the equation $x_{i,r,1} + \ldots + x_{i,r,n_i} = r$ which is $\binom{r+n_i+1}{r}$. We will enumerate these solutions and calculate the joint power consumption cost of all flows in N_i in each case. Our idea is to replace all the paths in N_i with a single path p_{N_i} with associated incremental power consumption cost function defined as,

$$\mathcal{W}^+_{p_{N_i}}(x) = \min_{x_{i,r,1}+\ldots+x_{i,r,n_i}=r} \sum_{l \in N_i} \mathcal{W}^+_{\{l\}}(h \sum_{j: l \in p_j \in N_i} x_{i,r,j}),$$

for $(r-1)h < x \leq rh$ and $r \geq 0$. When computing this function, we will also record a particular $(x^*_{i,r,1}, \ldots x^*_{i,r,n_i})$ that achieves the minimum above for each *r* examined. Let ϵ be given and let $h = \epsilon/|N|$. At this point, we will run the SF-RAP-Greedy-DP algorithm on the new instance (P', d') where $P' = D \cup \{p_{N_1}, \ldots, p_{N_c}\}$ and $d' = d - \epsilon$. The complete algorithm for the general case is presented in Algorithm 2. For clarity, we assume that the flow requests can be satisfied and omit failure-handling when this is not the case.

Theorem 2: On input (*P*, *d*), SF-RAP-Greedy finds a flow solution whose total flow from *s* to *t* is at least $d - \epsilon$, with power consumption cost at most twice that of the optimal flow

Algorithm 2 Alg:SF-RAP-Greedy(P, d, ϵ)

Step 1 Compute $P = N \cup D$ as described above and find the bundles $\{N_i\}_{i=1}^c$ of G_N;
Step 2 Let $h = \epsilon/|N|$;
Step 3 **for** $i = 1, \ldots, c$
  for $r = 0, \ldots, \lceil \frac{d-\epsilon}{h} \rceil$
    Compute $\mathcal{W}^+_{p_{N_i}}(rh)$;
  endfor
endfor
Step 4 Run SF-RAP-Greedy-DP$(D \cup \{p_{N_i}\}, d - \epsilon)$ and let $\hat{f}'$ be the resulting flow;
Step 5 **for** $i = 1, \ldots, c$
  for $p_j \in N_i$
    Let $f_{p_j} = h x^*_{i, \lceil f'_{p_{N_i}}/h \rceil, j}$;
  endfor
endfor
for $p \in D$
  Let $f_p = f'_p$;
endfor
Step 6 **return** $\hat{f}$;

of size *d* from *s* to *t*, in time $O((\frac{dq}{c} + b + 2)^{b+2} q m_P)$, where m_P is the number of links in *P*, *q* is the number of link-sharing paths and *b* is the maximum bundle size.

Proof: In Step 5, the total flow $\hat{f}$ is set to be at least that of $\hat{f}'$ found in Step 4 which is $d - \epsilon$. Also note that $\text{cost}(\hat{f}) = \text{cost}(\hat{f}')$ since the rounding of $f'_{p_{N_i}}$ up to an integral number of *h*-units in Step 5 does increase the cost of the flow. Let $\hat{f}^*$ be the optimal (least cost) flow solution for the original input (*P*, *d*). We define a new flow $\hat{f}^r$ for the instance $(D \cup \{p_{N_i}\}, d-\epsilon)$ as follows: for each N_i, let $f^r_{p_{N_i}} = \sum_{p \in N_i} \lfloor \frac{f^*_p}{h} \rfloor h$ and for each $p \in D$, let $f^r_p = f^*_p$. Observe,

$$\begin{aligned}\sum_{p \in D} f^r_p + \sum_{i=1}^{c} f^r_{p_{N_i}} &\geq \sum_{p \in D} f^*_p + \sum_{p \in N} (f^*_p - h) \\ &= \sum_{p \in P} f^*_p - |N|h \\ &= d - \epsilon.\end{aligned}$$

and $\text{cost}(\hat{f}^r) \leq \text{cost}(\hat{f}^*)$. Let $\hat{f}'^*$ be an optimal solution to the instance $(D \cup \{p_{N_i}\}, d-\epsilon)$. Since SF-RAP-Greedy-DP is 2-optimal by Lemma 1, we must have

$$\text{cost}(\hat{f}) = \text{cost}(\hat{f}') \le 2\text{cost}(\hat{f}'^{\bullet}) \le 2\text{cost}(\hat{f}^{r}) \le 2\text{cost}(\hat{f}^{*}).$$

Step 3 dominates the running time of the algorithm since Step 1 can be performed in $O(m_P)$ time and Step 4 in $O(dm_P)$ time. Since $|N_i| \le b$ and $r \le \lceil \frac{d-\epsilon}{h} \rceil < \frac{dq}{\epsilon} + 1$, the number of flow combinations examined when computing $\mathcal{W}^{+}_{p_{N_i}}$ is $\binom{\frac{dq}{\epsilon}+b+2}{\frac{dq}{\epsilon}+1} = O((\frac{dq}{\epsilon}+b+2)^{b+1})$. Each combination can be evaluated in $O(m_P)$ time so to compute each $\mathcal{W}^{+}_{p_{N_i}}$ requires $O((\frac{dq}{\epsilon}+b+2)^{b+2}m_P)$ time. There are at most $q/2$ bundles so Step 3 is completed in $O((\frac{dq}{\epsilon}+b+2)^{b+2}qm_P)$ time. We remark that for a constant maximum bundle size b, the running time is polynomial in d, q, m_P and $1/\epsilon$.

B. Proposed Algorithms for the MF-RAP

In this section, we first present a general framework for the MF-RAP based on algorithms for the SF-RAP. Then we present an LP-based algorithm.

Any algorithm for the SF-RAP can be extended to solve the MF-RAP. The basic idea is to use such an algorithm iteratively to find a flow allocation solution for each communication session. A general framework for the MF-RAP is formally presented as Algorithm 3.

Algorithm 3 MF-RAP-Framework

Step 1 Sort the given K communication sessions in the ascending order of their traffic demands and store them in the set S_C;
Step 2 **while** ($|S_C| \neq 0$)
Let (s, t, d) be the first element in S_C;
Find a flow allocation $\hat{f}$ for (s, t, d) using an algorithm for the SF-RAP;
Update the load and state of each affected link according to $\hat{f}$;
$S_C := S_C \setminus (s, t, d)$;
endwhile

In this algorithm, we sort the given set of communication sessions in ascending order of their traffic demands because we try to tackle the easy cases (those communication sessions with low traffic demand) first, which hopefully leads to better performance compared to a random ordering (this was verified by our simulations). Step 1 takes *O*(*K log K*) time. The running time of Step 2 depends on the time complexity of the algorithm for the SF-RAP. If the SF-RAP-Greedy algorithm presented in the last section to solve the SF-RAP with link disjoint candidate paths, then the running time of Step 2 is $O(Kdm)$since the algorithm for the SF-RAP takes $O(dm)$ and the link status updating takes $O(m)$, where *m* is the maximum number of links among all candidate paths sets.
This gives an overall time complexity of $O(K(\log K + dm))$. We call this algorithm the MF-RAP-*Greedy* algorithm.

Next, we present an LP-based algorithm in the following.

Algorithm 4 MF-RAP-SeriesLP

Step 1 Solve LP1($\hat{c}$), where $\hat{c} = [\cdots, c_{J_l}, \cdots]$;
Step 2 **while** (There is a feasible solution for LP1)
Set the state of each link l to j_l s.t. $c_{j_l-1} < f_l \le c_{j_l}$;
Let $\hat{c} = [\cdots, c_{j_l}, \cdots]$;
Let $l^{*} = \text{argmin}_{l \in L} \frac{f_l - c_{j_l-1}}{w_{j_l} - w_{j_l-1}}$;
$j_{l^{*}} := j_{l^{*}} - 1$ and update $\hat{c}$ accordingly;
Solve LP1($\hat{c}$);
endwhile

The basic idea of this heuristic algorithm is to solve a series of linear programs specified by LP1 below. In the beginning, the algorithm obtains an initial feasible flow allocation solution by solving LP1 and setting each link to its highest state such that every link has its maximum capacity. After obtaining a feasible initial flow allocation, we set the state of each link accordingly such that each link has just enough capacity to accommodate its traffic given by the flow allocation. Then in every iteration, the algorithm tries to improve the solution by tuning the state of a link from the current one to a lower state. The link *l* with minimum weight $z(l) = \frac{f_l - c_{j_l-1}}{w_{j_l} - w_{j_l-1}}$ is selected for tune-down. The denominator of the weight function gives the power reduction achieved by tuning down the link state and the numerator gives the traffic amount that needs to be re-routed. We want to have as much reduction as possible

and as little re-routed traffic amount as possible. This metric is somehow similar to the price-to-quality ratio (or cost-to-benefit ratio), which is usually used for customers to select products. Then we solve LP1 again with an updated link capacity vector, i.e., for entry corresponding to l^*, its capacity value is updated from $c_{j_{l^*}}$ to $c_{j_{l^*}-1}$. Note that since we just want to find a feasible flow allocation solution in each iteration, it does not matter which objective function is used. However, in LP1, the objective is set to minimize the maximum link load, which hopefully leads to balanced traffic load and therefore low power consumption.

LP1$(\bar{c})$

$$\min \beta \qquad (10)$$

Subject to:

$$\sum_{p \in P_k} f_p^k = d_k, \quad k \in \{1, \ldots, K\};$$

$$f_l = \sum_{k=1}^{K} \Big(\sum_{p \in P_k : e \in p} f_p^k + \sum_{p \in P_k : \bar{e} \in p} f_p^k \Big), \quad \forall l = (e, \bar{e}), \; k \in \{1, \ldots, K\};$$

$$f_l \le c_{j_l}, \quad \forall l = (e, \bar{e}) \in L; \qquad (11)$$

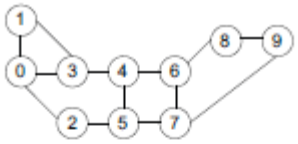
$$f_l \le \beta, \quad \forall l = (e, \bar{e}) \in L. \qquad (12)$$

The time complexity of this algorithm is $m_L J O(\text{LP1})$, where J is the maximum number of states, m_L is the number of links in the network and O(LP1) is the running time for solving LP1. It is known that an LP with a polynomial number of constraints and variables can be solved in polynomial time. Therefore, the MF-RAP-SeriesLP algorithm is a polynomial time algorithm. On average cases, even the simple simplex algorithm [5] can solve the LP efficiently.

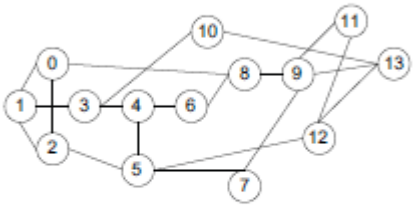
V. Simulation Results

In this section, we present simulation results to show the performance of the proposed algorithms. The software ILOG CPLEX 10.1 [9] was used to solve all the LP and MILP problems. Similar to [3], 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 communication session had randomly selected source and destination nodes, and a randomly generated traffic demand. The link states, the corresponding capacity thresholds and power consumption values are given by Table I [18], [20]. Note the values given here do not include power consumed on line cards and chassises. In addition, for each communication session, we used a simple heuristic algorithm to find link-disjoint paths between its source and destination nodes, which keeps finding a shortest path (in terms of hop-count) and removing links included in the existing set of paths until no more paths can be found.



(a) The Abilene Network



(b) The NSF Network

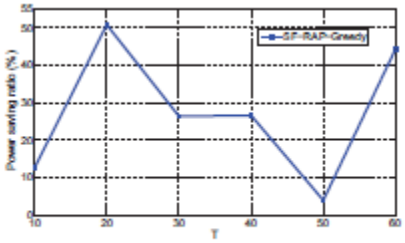
Fig. 1. The network topologies

TABLE I
LINK STATES AND POWER CONSUMPTION

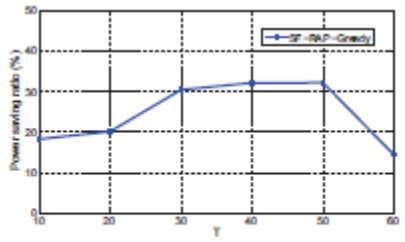
State	Link Capacity	power Consumption
1	10Mbps	0.84W
2	100Mbps	0.96W
3	1Gbps	1.8W
4	10Gps	10W

In the first two scenarios, we tested our SF-RAP-Greedy algorithm with dynamically generated traffic. In each simulation run, communication sessions were added to the network sequentially (one every time unit) and each of them had a lifetime of *T* time units. Each simulation run lasted for 100 time units. A widely used *Shortest Path (SP)* based solution served as the baseline for comparison, which finds a shortest path (in terms of hop-count) for each communication session and routes all its traffic through the shortest path. The power *saving ratio* of an algorithm A is defined as $\frac{\mathcal{W}_{SP}-\mathcal{W}_A}{\mathcal{W}_{SP}}$, where $\mathcal{W}_{SP}$ is the average incremental power consumption given by the baseline solution (i.e., the SP-based solution), and $\mathcal{W}_A$ is the average incremental power consumption given by Algorithm A. We compared the proposed SF-RAP-Greedy algorithm against the baseline solution in terms of the power saving ratio on those two networks by increasing *T* from 10 to 60 with a step size of 10. In scenario 1, each communication session had a randomly generated traffic demand that was uniformly distributed in [1, 100]Mbps. In scenario 2, we tested our algorithm in heavy traffic cases by setting the demand
range to [50, 100]Mbps.

In the other two scenarios, we evaluated our algorithms for the MF-RAP, i.e., the MF-RAP-Greedy algorithm and the MF-RAP-SeriesLP algorithm. In each simulation run, *K* communication sessions and their traffic demands were randomly generated in advance. We solved the MILP formulation for the MF-RAP to provide optimal solutions. We used the *solution quality ratio* as the performance metric, which is defined as $\frac{\mathcal{W}_A}{\mathcal{W}_{OPT}}$, where $\mathcal{W}_{OPT}$ is the optimal total power consumption and $\mathcal{W}_A$ is the total power consumption given by Algorithm A. The closer it is to 1, the better. We compared the two proposed algorithms against the optimal solutions given by solving MILP1 by changing *K* from 10 to 60 with a step size of 10. Similarly, in scenario 3, the demand range for each communication session was set to [1, 100]Mbps, while in scenario 4, it was set to [50, 100]Mbps to generate heavier traffic. The simulation results are presented in Figs. 4–5.



(a) The Abilene network

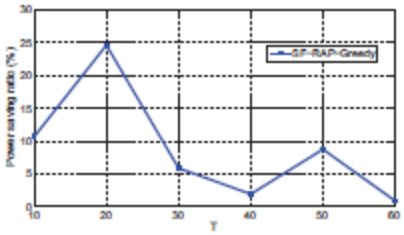


(b) The NSF network

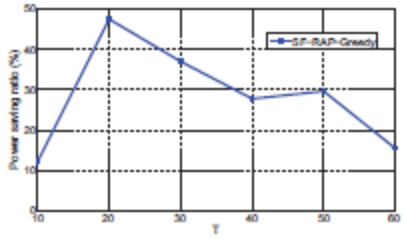
Fig. 2. Scenario 1 (regular traffic): Power saving ratio VS. T

We make the following observations from the simulation results.

1) From Figs. 2–3, we can see that the SF-RAP-Greedy algorithm consistently outperforms the SP-based baseline solution with regards to the average incremental power consumption in networks with dynamic traffic. On average, it achieves 22.2% power savings. An interesting observation is that the SF-RAP-Greedy algorithm performs better on the NSF network than on the Abilene network. Specifically, it achieves average power savings of 26.4% on the NSF network, compared to 18.1% on the Abilene network. This is mainly because compared to the Abilene network, the NSF network

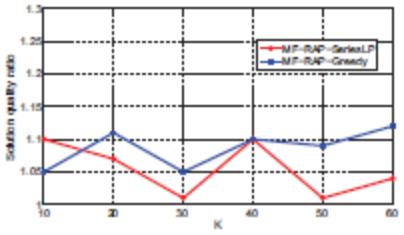


(a) The Abilene network

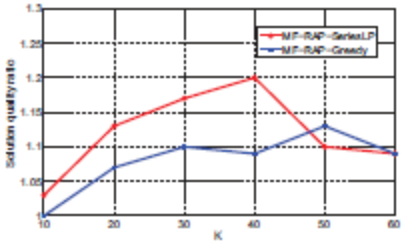


(b) The NSF network

Fig. 3. Scenario 2 (heavy traffic): Power saving ratio VS. T



(a) The Abilene network

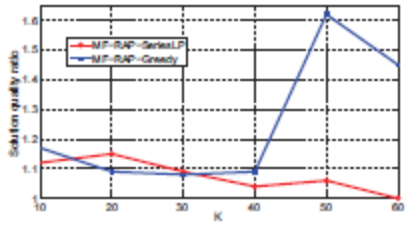


(b) The NSF network

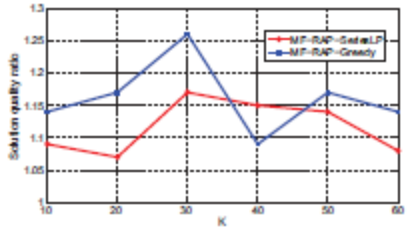
Fig. 4. Scenario 3 (regular traffic): Power saving ratio VS. K

is denser, and there are usually more candidate paths between a pair nodes, which is favorable for flow allocation.

2) From Figs. 4–5, we observe that the MF-RAP-Greedy algorithm and the MF-RAP-SeriesLP algorithms produce close-to-optimal solutions in both regular and heavy traffic cases. Specifically, they achieve average solution quality ratios of 1.14 and 1.09 respectively. Particularly, the solution quality ratio given by the MF-RAP-SeriesLP algorithm is always no



(a) The Abilene network



(b) The NSF network

Fig. 5. Scenario 4 (heavy traffic): Power saving ratio VS. K

more than 1.20. In most cases, the MF-RAP-Greedy gives a solution quality ratio no more than 1.26, however, there are some exceptions in the heavy traffic scenario (scenario 4). The MF-RAP-SeriesLP offers a better performance because it always jointly computes flow allocation for all communication sessions; however, the MF-RAP-Greedy algorithm finds flow allocation for communication sessions sequentially and a bad decision made for one communication session may lead to poor overall performance.

VI. Conclusions

In this paper, we studied how to leverage rate adaptation for green networking by studying two flow allocation problems, namely, the SF-RAP and the MF-RAP. We showed that both problems are NP-hard and presented an MILP formulation for the MF-RAP to provide optimal solutions. Then we presented a 2-approximation algorithm for the SF-RAP, and a general flow allocation framework as well as an LP-based heuristic algorithm to solve the MF-RAP. We performed simulation runs on the Abilene research network and the NSF network. Our simulation results showed that the greedy algorithm for the SFRAP consistently outperforms a shortest path based baseline solution and the algorithms proposed for the MF-RAP produce close-to-optimal solutions.

References

[1] Y. K. Agarwal, Design of capacitated multi-commodity networks with multiple facilities, *Operations Research*, Vol. 50, No. 2, 2002, pp. 333– 344.

[2] M. Andrews, Hardness of buy-at-bulk network design, *Proceedings of IEEE FOCS'2004*, pp. 115–124.

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

[4] B. Awerbuch and Y. Azar, Buy-at-bulk network design, *Proceedings of IEEE FOCS'1997*, pp. 542–547.

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

[6] C. Chekuri, M.T. Hajiaghayi, G. Kortsarz and M. R. Salavatipour, Approximation algorithms for non-uniform buy-at-bulk network design, *Proceedings of IEEE FOCS'2006*, pp. 677–686.

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

[8] S. Chopra, I. Gilboa, S. T. Sastryb, Source sink flows with capacity installation in batches, *Discrete Applied Mathematics*, Vol. 85, 1998, pp. 165–192.

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

[10] A. Frangioni and B. Gendron, 0–1 reformulations of the multicommodity capacitated network design problem, *Discrete Applied Mathematics*, Vol. 157, 2009, pp 1229–1241.

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

[12] V. Gabrel, A. Knippel and M. Minoux, A comparison of heuristics for the discrete cost multicommodity network optimization problem, *Journal of Heuristics*, Vol. 9, 2003, pp. 429–445.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1990.
[14] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problem. In Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, pages 383 C 388, 2001.
[15] A. Gupta, A. Kumar, and T. Roughgarden, Simpler and better approximations for network design, *Proceedings of ACM Symposium on Theory of Computing* 2003, pp. 365–372.
[16] M. Gupta and S. Singh, Greening of the Internet, *Proceedings of ACM SIGCOMM'2003*, pp. 19–26.
[17] T. L. Magnanti and P. Mirchandani, Shortest paths, network design, and associated polyhedra, *Networks*, Vol. 23, No. 2, 2006, pp. 103–121.
[18] P. Mahadevan, P. Sharma, S. Banerjee and P. Ranganathan, power aware network operations, *Proceedings of IEEE Infocom'2009*, pp. 25–30
[19] M. Minoux, Discrete cost multicommodity network optimization problems and exact solution methods, *Annals of Operations Research*, Vol 106, 2001, pp. 19–46.
[20] 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*.
[21] G. M. Schneider and T. Nemeth, A simulation study of the OSPF-OMP routing algorithm, *Computer Networks*, Vol. 39, 2002, pp. 457–468.
[22] Nedeljko Vasic and Dejan Kostic, power-aware traffic engineering, *EPFL Technical Report NSL-REPORT-2008-004.*