Identifying and Using Energy-Critical Paths

Nedeljko Vasić
EPFL, Switzerland
nedeljko.vasic@epfl.ch

Dejan Novaković
EPFL, Switzerland
dejan.novakovic@epfl.ch

Satyam Shekhar*
IIT Guwahati, India
s.satyam@iitg.ernet.in

Prateek Bhurat*
IIT Kanpur, India
bhurat@iitk.ac.in

Marco Canini
EPFL, Switzerland
marco.canini@epfl.ch

Dejan Kostić
EPFL, Switzerland
dejan.kostic@epfl.ch

ABSTRACT

The power consumption of the Internet and datacenter networks is already significant, and threatens to shortly hit the power delivery limits while the hardware is trying to sustain ever-increasing traffic requirements. Existing energy-reduction approaches in this domain advocate recomputing network configuration with each substantial change in demand. Unfortunately, computing the minimum network subset is computationally hard and does not scale. Thus, the network is forced to operate with diminished performance during the recomputation periods. In this paper, we propose REsPoNse, a framework which overcomes the optimality-scalability trade-off. The insight in REsPoNse is to identify a few energy-critical paths off-line, install them into network elements, and use a simple online element to redirect the traffic in a way that enables large parts of the network to enter a low-power state. We evaluate REsPoNse with real network data and demonstrate that it achieves the same energy savings as the existing approaches, with marginal impact on network scalability and application performance.

1. INTRODUCTION

The IT industry is plagued by a large and constantly rising energy consumption. This has become a source of growing environmental, social, and economical concern. The energy consumed by computer networks plays a significant role in this, and a growing body of work within the research community documents the importance of improving their energy efficiency. For example, the US network infrastructure alone requires between 5 and 24 TWh/year [29], which translates into a cost of $0.5-2.4 B/year. Similar trends are also observed in Europe where, for instance, Telecom Italia reports a total power demand of 2 TWh/year [10]. Datacenters are facing the same problem as their networks are accounting for up to 20% of their total power consumption [25]. As an example, the overall power consumed by networking devices in US datacenters alone reached about 3 TWh in 2006, and is still rising [16, 25].

The reason for a network's large energy consumption lies in several factors. Networks are designed with two strategic core principles in mind, namely redundancy and bandwidth overprovisioning, which enable them to tolerate traffic variations and faults, and work around Internet's lack of QoS. While these design features do allow network operators to achieve the service level objectives, networks end up being underutilized most of the time, with network equipment typically running at its least energy-efficient operating point. As network devices draw nearly constant power regardless of the workload [14], a significant fraction of the consumed energy is unjustified waste. As a matter of fact, *networking is hitting the "power wall"* that other computing disciplines are already facing [18], and a number of factors are going to exacerbate this issue. First, network traffic continues to increase exponentially [4]. Second, the CMOS technology that is predominantly used in routers is reaching its energy efficiency limits [14]. Third, multimedia content already dominates other types of traffic and its share will further increase [4] (e.g., due to a shift to high-definition video). Fourth, the amount of traffic datacenter networks have to carry is also increasing as data analytics tools process ever-increasing datasets. In short, the rate of increase in traffic (2.5 every 18 months [13]) is faster than the rate in which silicon technologies improve their energy efficiency (1.65 per 18 months according to Dennard's law).

While a network operator might afford an ever-increasing power and cooling bill for what is perceived to be critical infrastructure, a more insidious problem lurks in the near future. Further increasing the power consumption poses *numerous challenges in delivering sufficient power to the power-hungry network elements.* For instance, typical datacenters already consume up

*work done during an internship at EPFL

ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.

to 60 Amps per rack which is close to the power delivery limits for most of them [31]. Moreover, the dominant trend is to move toward little or no cooling in datacenters [5]. This is in stark contrast with the ever-increasing number of non-energy proportional networking devices for topologies with increased levels of bandwidth [7]. The net effect of all these factors is that the Internet and datacenter networks will not be able to continue their exponential growth without having to substantially increase the investment used for their power and cooling.

While helpful, making the individual networking elements more energy-proportional faces several challenges that are hard to overcome, in terms of CMOS energy-efficiency limits, performance penalties (e.g., increased packet latency and packet drops) due to changes in internal power levels [29], and the existence of always-on components (e.g., chassis).

We therefore argue that the networks themselves should become energy-proportional, i.e., they should be capable of handling the traffic demand by using the minimum possible amount of energy. The energy saving potential comes from the very same design principles that make networks energy inefficient: physical diversity and bandwidth overprovisioning.

Recently there have been a few research efforts [15, 25, 41] toward this end. In these approaches, one needs to: 1) collect information about the current traffic conditions and 2) compute the minimal set of network elements that can accommodate the current traffic, while letting remaining equipment enter a power saving mode. In practice, it is however highly undesirable to often deploy new routing tables – this is likely to require a considerable amount of human effort (e.g. to examine if congestion has occurred with new routing tables, etc.). In contrast, network operators would like to base their designs on longer time scales, measured in weeks or most likely months [42].

In this paper, we analyze the *trade-off between optimal energy savings and scalability* by quantifying the high *recomputation rate* of the existing approaches on real traces of traffic demand. Scalability here refers to the ability of the routing re-computation to quickly reconfigure routing in the face of traffic demand changes for large networks. Further, it has been shown that computing energy-aware routing tables is an NP-hard problem and that computing the optimal solution might take a few hours even for medium-sized networks [14]. During this time, the network can experience congestion due to insufficient resources, or it is using more energy than is needed. All the heuristics for solving this problem can finish sooner [15, 19, 25, 41], but doing so decreases the energy-saving potential.

Given such problems, we investigate whether it is possible to precompute a few **energy-critical** paths that, when used in an energy-aware fashion, can continuously produce optimal or close-to-optimal energy savings over long periods of time. To answer our question, we propose REsPoNse, a framework that: 1) identifies energy-critical paths by analyzing the traffic matrices, 2) installs them into a small number of routing tables (called *always-on*, *on-demand*, and *failover*), and 3) uses a simple, scalable *online* traffic engineering (TE) mechanism to deactivate and activate network elements on demand. Our conjecture is that the network operators can use REsPoNse to overcome power delivery limits by provisioning power and cooling of their network equipment for the typical, low to medium level of traffic. Next, we highlight our contributions:

1. We describe REsPoNse—a framework that allows network operators to automatically identify energy-critical paths.

2. Experimenting with REsPoNse and real network data, we demonstrate that a small number of precomputed paths (e.g., only 3) is required to produce up to 40% energy-savings across various traffic matrices, without necessitating network operators and engineers to frequently recompute and deploy new routing tables.

3. Using replay of traffic demands, ns2 simulations, and running live code in a Click testbed and a network emulator, we demonstrate REsPoNse's responsiveness to changes in traffic demand.

4. We run a media streaming application and a Web workload live over REsPoNse-chosen paths to demonstrate that the power savings have marginal impact on the application-level throughput and latency. To the best of our knowledge, this is the first such evaluation.

2. BACKGROUND

2.1 Networking Hardware is Not Energy-Proportional

Networking hardware is typically built around a chassis that allows a number of line cards to be plugged into it. The chassis alone requires significant energy to run, usually between 25 and 50% of total router power in typical configurations [14]. Moreover, current network elements are not anywhere near being energy-proportional. As a matter of fact, their idle consumption (i.e., when not forwarding any packets) is 90% of their maximum power consumption [14].

2.1.1 Sleeping for Energy Saving

Just like modern CPUs have a number of "sleep" states (e.g., C-states in Intel processors) that enable the CPU and the machine using it to enter a low power state, an energy-saving feature [22, 29] proposed for networking hardware is the ability to put network elements in one of a series of sleep states. Such sleep states typically include a variety of options, each of which con-

sumes progressively less energy but requires more time to return to the normal operating state. In this paper, we assume that the networking elements (chassis, line cards, etc.) can quickly (e.g., a few tens of ms [23]) enter and return from a sleep state when they are not receiving or sending any traffic.

Gupta *et al.* [22] quantify some of the savings that are possible due to inter-packet gaps (i.e., packets not continuously arriving at full speed). Such opportunistic sleeping intervals might however be too short. Thus, Nedevschi *et al.* [29] have quantified the energy savings that are possible when the packets are briefly queued in "upstream" routers, to give "downstream" network elements a chance to sleep longer. However, changing a network element's state might cause packet loss. Further, frequent state switching consumes a significant amount of energy as well.

2.2 Network-Wide Energy-Proportionality

Ideally, it would be possible to build perfectly energy-proportional network elements, and by doing so entire networks would be energy-proportional. Given the technological and deployment difficulties in realizing this goal, the complementary choice is to make the entire ensemble of network elements appear to be energy-proportional by carefully shifting traffic in a way that lets large parts of the network enter a low power state when the demand is low. We point out that doing so enhances the energy-saving efforts created for individual network elements, e.g., they can stay in low-power energy states longer. Next, we describe the model of the problem that needs to be solved in this case.

2.2.1 The Model

The commonly used model for describing the problem in the literature is based on the standard multi-commodity flow formulation, plus a number of additional binary variables and constraints that determine the power state of network elements. We do not claim any novelty about the model, and we summarize it here for the completeness of the exposé.

The inputs to the model are a network topology annotated with link capacities, the power characteristics of its elements, and measurements or estimates of the traffic matrix. The objective minimizes the total network power consumption by solving a mixed-integer linear program whose outcome contains a routing for each origin-destination (O,D) pair that satisfies the constraints. Our formulation uses the following notation:

Network. The network is composed of a set of routers or switches (routers for the sake of simplicity in our exposé) $\mathcal{N}$, for which we use indices i and j, and an arc set $\mathcal{A}$. A link between a pair of routers (i, j) is represented as a directed arc $i \to j$ whose bandwidth capacity is denoted as $C_{i\to j}$. Links are not necessarily symmetric, thus $C_{i\to j} = C_{j\to i}$ does not need to hold. $\mathcal{A}_i$ is the set of arcs originating from i. The binary decision variables are X_i which indicates whether the router i is powered on, and $Y_{i\to j}$ that determines if the link represented by $i \to j$ is active. We impose $Y_{i\to j} = Y_{j\to i}$ as a link cannot be half-powered.

Power consumption. In the interest of clarity, we assume a simple model for router power consumption. We consider it is relatively straightforward to extend our formulation to the more generalized model in [14]. For each router i, $P_c(i)$ is the cost in Watts for operating the chassis. The power cost for a line card is linearly proportional to the number of used ports. Consequently, we denote by $P_l(i \to j)$ the cost in Watts for using a port on router i connected to j. Finally, the power cost of the optical link amplifier(s) is $P_a(i \to j)$ and depends solely on the link's length.

Traffic. For each (O,D) pair, there is a flow $d_{(O,D)}$ that enters the network at the router O, and exits at the router D. We also assume that all flows are fluid, i.e., no losses occur in the network. We denote as $f_{i\to j}(O, D) \in \{0, 1\}$ whether the flow from O to D is routed via the arc $i \to j$. Note that these decision variables are binary; this ensures that each flow is routed on a single link.

The objective function that minimizes the network power consumption is:

$$\sum_{i\in\mathcal{N}} X_i \left[P_c(i) + \sum_{i\to j\in\mathcal{A}_i} Y_{i\to j}(P_l(i \to j) + P_a(i \to j)) \right]$$

We seek a solution that satisfies the multi-commodity flow constraints: capacity, flow conservation and demand satisfaction (omitted for lack of space, see [17]), and these additional constraints:

1. All links connected to a router that is switched off are inactive.

$$\forall i \in \mathcal{N}, \forall i \to j \in \mathcal{A}_i : Y_{i\to j} \le X_i \quad (1)$$

2. A flow cannot be assigned to an inactive link.

$$\forall i \to j \in \mathcal{A} : \sum_{(O,D)} f_{i\to j}(O, D) \quad d_{(O,D)} \le C_{i\to j} Y_{i\to j} \quad (2)$$

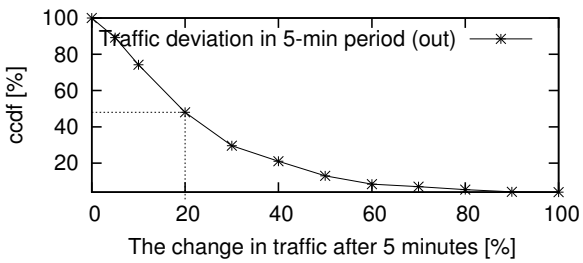
3. A router is powered off when all its links are inactive.

$$\forall i \in \mathcal{N} : X_i \le \sum_{i\to j\in\mathcal{A}_i} Y_{i\to j} \quad (3)$$

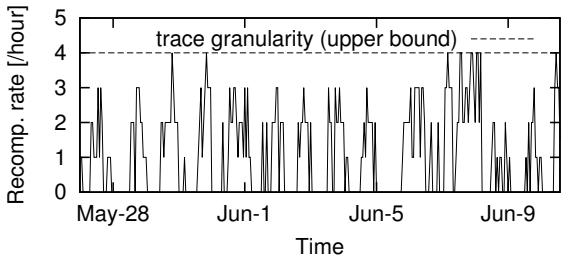
2.2.2 Computational Complexity

The well-known problem of routing flows across finite-capacity links in a network maps to a multi-commodity flow problem that has a polynomial solution. Incorporating the reduction in energy consumption into the model results in an NP-hard problem [17].[1] Thus, it does not lend itself to a distributed, elegant traffic engineering solution such as the one proposed in [24] based on optimization decomposition.

[1]Intuitively, this happens because the network elements can now be only in one of a few energy states.



(a) Traffic in the datacenter changes on a short time scale (Google data from [25]).



(b) State-of-the-art approaches need to run a few times every hour (replay of GÉANT traffic demands).

Figure 1: Both ISP and datacenter traffic exhibits rather high variation.

Tools such as CPLEX [2] (considered to be the state-of-the art for solving mixed-integer programming) are capable of solving the problem over several hours for small- and medium-size ISP topologies. In addition, even the good approximation algorithms (such as greedy bin-packing) still take about 1000 seconds for medium-sized networks [25].

2.3 Related Work

Comprehensive surveys of a number activities for minimizing energy consumption in networks are recently published in [11] and [35]. We classify the most relevant related works as: i) those that seek an optimal solution, and ii) those that use a heuristic to overcome the computational complexity.

Seeking an optimal solution. Chabarek *et al.* [14] present an approach for *provisioning* the network for the peak load and do not consider scaling the power consumption with the offered load.

Heuristics. Chiaraviglio *et al.* [15] try to determine the minimum set of routers and links that can accommodate a given traffic demand. The authors propose a heuristic which sorts the devices according to their power consumption and then tries to power off the devices that are most power hungry. Yet, the paper does not discuss: 1) how this heuristic would be deployed, and 2) how the set of active elements would transition from one configuration to another as traffic changes.

ElasticTree [25] focuses on power savings in data center networks. This approach leverages the tree-based nature of these networks (e.g., a Fat-Tree) and computes at runtime a minimal set of network elements to carry the traffic. Although the authors claim that their solution can be applied to general networks, doing so would incur long path-computation time that is unacceptable for online techniques (the proposed heuristic that scales linearly is only applicable to the fat-tree).

Zhang *et al.* propose GreenTE [41], a power-aware traffic engineering algorithm that periodically collects link utilization and computes a subset of network elements that can carry the traffic. They try to reduce computation time by allowing a solver to explore only the k shortest paths for each (O, D) pair.

Andrews *et al.* [8] formally prove that the problem of energy minimization does not have a bounded approximation in general, but on the positive side, they present a constant approximation under the assumption that the energy curve is a polynomial.

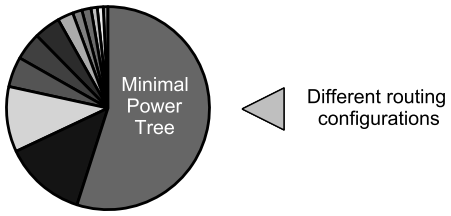
A set of heuristics for a similar problem of identifying the minimal set of cables to carry the traffic is presented by Fisher *et al.* [19]. They start by linearizing the initial problem and solving it as a simple LP problem. They then pass this initial solution to a set of heuristics (which differ in their complexity and running time) to obtain an approximation of the optimal solution after rounding. Similar to other efforts, they report running times of about several minutes.

Distributed heuristics. In Energy-Aware Traffic Engineering (EATe) [36], the edge routers run a complex TE component that tries to aggregate traffic over *predetermined* routing paths.

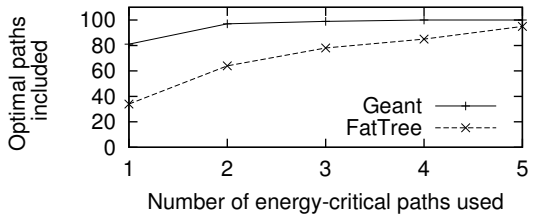
Other efforts. In the context of network design and analysis, Zang *et al.* [42] introduced the criticality concept for traffic matrices. They demonstrate that a small number of traffic matrices are sufficient to perform network analysis. Orthogonal to our work, Abts *et al.* [6] focus on the power characteristics of datacenter networks - they argue that a flattened butterfly topology is more power efficient than the other proposed topology for datacenters. Note that our framework can identify energy-critical paths in an arbitrary topology, including the butterfly topology.

3. NETWORK TRAFFIC ANALYSIS

Although promising in terms of the potential power savings, the deployment of the aforementioned approaches is hampered by the long time needed to compute the minimal network subset after each significant traffic change. Heuristics (described in Section 2.3) try to overcome the so-called scalability-optimality trade-off [25]. Unfortunately, sacrificing optimality does not ensure network performance under changing traffic demands. To explore the trade-off, we analyze traces from: 1) a production Google datacenter, and 2) GÉANT, the European educational network.



(a) The minimal power tree successfully handles the GÉANT traffic 60% of the time.



(b) A large majority of traffic sources route their packets through very few, reoccurring paths.

Figure 2: Only few energy-critical paths for each node pair offer a near-optimal energy saving.

3.1 Traffic Changes Frequently on a Link

We start by analyzing the network traffic measured at 5-min intervals at a production Google datacenter. The analyzed traces are available here [25]. Such a fine granularity of measured traffic (5-min interval) enables us to explore live traffic dynamics, and to evaluate whether the existing approaches can timely react to changes in traffic demands while minimizing power consumption.

Figure 1a reveals that the traffic demand changes on a shorter time scale than that of the energy-aware routing table recomputation. For instance, in almost 50% cases the traffic changes at least by 20% percent over a 5-min interval. We therefore conclude that the computed minimal network subset might no longer be relevant/sufficient for up-to-date demands.

3.2 High Network-Wide Recomputation Rate

Although link-based findings demonstrate that traffic might change too often for existing schemes to adapt, we still do not have a formal metric to quantify our findings. Thus, in this section we quantify the number of times an approach has to change the routing tables when the traffic changes across an entire network are considered. We refer to this new metric as *recomputation rate*, and argue that it can be used to judge the applicability of an approach.

To quantify the impact of on-the-fly network reconfiguration, we inspect the GÉANT traffic demands in a 15-day period (details are in Section 5). Using the optimization problem from Section 2.2.1, we recompute the routing tables after each interval in the trace and only count the intervals for which the set of network elements changes from one interval to the next. Due to the somewhat coarse-grained link utilization information of the trace (15-minute interval), the values we compute are an underestimate. Figure 1b shows that the recomputation rate for existing approaches goes up to four per hour (the maximum possible for our trace), even for the 15-minute interval granularity in the GÉANT trace. Given that the computation takes 5-15 minutes on this network topology even with the fastest heuristics [41], the network is running with inadequate resources to handle the load for a large fraction of time. During the recomputation, the network can experience congestion due to insufficient resources, or it is using more energy than is needed.

3.3 Energy-Critical Paths Exist

We further consider whether it would be possible to simply store and reuse the routing tables as computed above for the GÉANT traces. Figure 2a measures the routing configuration dominance by plotting the fraction of time over which the network was operating under each routing configuration. The routing configuration here refers to the set of routing tables in use at a given interval. Although we observe that a single routing configuration[2] is active almost 60% of times, this approach is not practical – the total number of different routing configurations (13 slices in Figure 2a) is still large, beyond the capabilities of today's network elements.

We then examine if looking at individual (O, D) paths can help reduce the number of routing tables needed. Note that while we observe 13 different routing configurations, several routing configurations typically share the same (O, D) paths for some node pairs. We therefore rank each (O, D) path by the amount of traffic it would have carried over the trace duration. Figure 2b plots the CDF of the percentage of traffic cumulatively accounted to the top X (O, D) paths for each node pair as we increase X from 1 to 5. The results are promising – a large majority of node pairs *route their packets through very few, reoccurring paths* – we refer to these as **energy-critical** paths. In the particular case of GÉANT, only 2 precomputed paths per node pair are enough to cover almost 98% of the traffic, while 3 cover all traffic. We believe that the reason for such a predictable routing behavior lies in the limited, built-in network redundancy, and the typical diurnal traffic patterns [25]. We conclude that, to reduce the number of precomputed routing tables, we simply need the ability to independently choose how to route traffic for any given node pair.

It is also prudent to discuss the scenario where a network has a large degree of redundancy (e.g., datacenter network such as FatTree [7]). One would expect that

[2]Which corresponds to the minimal (w.r.t. power) subset of network elements that provides bare connectivity.

a large number of paths is required to avoid a high recomputation rate in this case. We consider a FatTree network with 36 switches at the core layer (for at least 36 disjoint paths), and drive the traffic volume by reusing the 8-day Google traces from Figure 1a. Figure 2b clearly shows a need for more energy-critical paths than in case of GÉANT, but again, the critical number is still reasonable – 5 precomputed paths are enough to carry the traffic matrices over an 8-day period in a Google production datacenter. Finally, if the traffic matrix analysis reveals that the number of energy-critical paths is large (e.g. more than 10), network operators can deploy only the most important paths, while keeping the remaining ones ready for later use.

4. THE REsPoNse FRAMEWORK

To verify if we can leverage the previous findings to produce solid savings without re-computing routing tables, we implement a prototype called Responsive Energy-Proportional Networks (REsPoNse). Its key insight is to: 1) precompute a few energy-critical routing tables off-line in a way that can approximate the optimal routing from the energy standpoint, 2) install the routing tables into network elements, and 3) use a simple, scalable online traffic engineering component to quickly adapt to changes in the traffic demand.

In this section, we describe the various algorithms we can use to precompute energy-critical routing tables offline. Our discussion is centered on ISP networks (comprised of routers), but is equally applicable to data center networks built with switches.

Precomputing the routing tables. REsPoNse uses the network topology, a power model of the network devices, and traffic matrix estimation (if available) to precompute offline three sets of paths. First, the *always-on* paths comprise the links and the routers (network elements, in general) that are expected to be active most of the time. These elements are selected so as to minimize the energy consumption in the always-on paths. Second, REsPoNse computes the *on-demand* paths, characterized by the fact that most of their network elements can be inactivated to save energy and brought back to the active state to balance traffic demand with additional network capacity. Finally, it computes the set of *failover* paths that can carry traffic in case of network element failure in the always-on or on-demand paths. Later in this section we discuss how these sets of paths can be deployed in practice.

We expect that the computed paths can be maintained unchanged unless the underlying topology changes or the traffic demand experiences a significant and unanticipated deviation from the long term traffic demands averaged over multiple days (potentially months) typically used for traffic estimation [26].

Dynamically (in)activating the network elements. The network uses always-on paths alone as long as they are capable of accommodating the current traffic, which is typical when the demand is low to medium. When the traffic demand increases and these paths exceed ISP's desired maximum utilization, the TE component selectively starts activating the additional resources on the on-demand paths and uses them to accommodate the additional traffic so as to keep energy proportionality under increasing load.

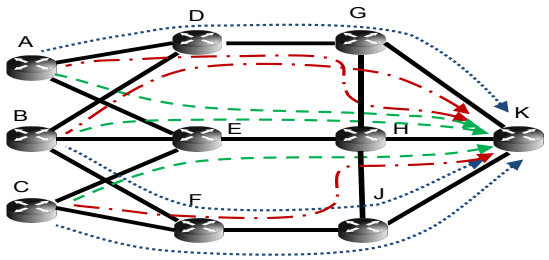


Figure 3: The paths chosen by REsPoNse in the example topology (always-on (dashed), on-demand (dash dotted), and failover (dotted)).

Example. Figure 3 shows the set of paths that could be chosen by REsPoNse on the topology used throughout our examples. Here, A, B, and C use the common *always-on* path $E - H - K$ to route traffic toward K. This arrangement allows the network elements on the $D - G - K$ and $F - J - K$ paths to be inactive when the load is low. The *on-demand* paths are selectively activated to sustain changes in traffic load. Whenever the underlying network allows it, the *failover* paths are chosen to guard against the failure of network elements in the always-on and on-demand paths.

4.1 Computing the Always-On paths

The goal of the always-on paths is to provide a routing that can carry low to medium amounts of traffic at the lowest power consumption. These paths are obtained by solving the optimization problem using the off-peak traffic matrix estimation as input: $d_{(O,D)} = d^{low}_{(O,D)}$. Alternatively, assuming no knowledge of the traffic matrix (as we do for our evaluation), one can set all flows $d_{(O,D)}$ equal to a small value ϵ (e.g., 1 bit/s) to obtain a minimal-power routing with full connectivity between any (O,D) pair. Our experiments show that this approximation is sufficient for a number of settings and that the always-on paths alone can accommodate about 50% of the traffic volume that can be carried by the Cisco-recommended OSPF paths, which are described later.

We use an off-the-shelf solver [2] to compute the always-on paths. This is a reasonable decision given that always-on paths are precomputed only once (for the alternative which does not assume any traffic knowledge) and they do not add run-time overhead.

Given that any routing which minimizes the power consumption can increase the propagation delay, we ex-

plore a version of the problem that uses a constraint to bound the maximum delay:

$$\forall(O,D) : delay(O,D) \leq (1+\beta)\cdot delay^{OSPF}(O,D) \quad (4)$$

Specifically, we ensure that the propagation delay is within a certain percentage β (e.g., 25%) of the delay obtained by the shortest path algorithm computed with the standard, Cisco-recommended setting for OSPF (link weights are the inverse of capacity; we term this baseline *OSPF-InvCap* or simply *InvCap* in our evaluation). This constraint can be used to enforce Service-Level Objectives for latency-sensitive traffic. We call this version *REsPoNse-lat*.

4.2 Computing the On-Demand paths

The always-on paths will not be able to accommodate the full volume of traffic because they are potentially designed in a way which minimizes power consumption without taking into account the maximum amount of traffic that can be carried. The role of the on-demand paths is to start carrying traffic when the load is beyond the capacity offered by the always-on paths. The on-demand table computation is repeated $N-2$ times, where N is the number of energy-critical paths (remaining 2 paths are reserved for always-on and failover sets).

Using the solver for an optimization problem. To compute the on-demand paths with knowledge of the peak-hour traffic matrix, we solve the same optimization problem but with the following additional settings. First, we carry on the X_is and $Y_{i\rightarrow j}$s equal to 1 from the routing solution for always-on paths and maintain them fixed, i.e., a network element already in use stays switched on. Second, we assign each flow as: $d_{(O,D)} = d^{peak}_{(O,D)}$.

Alternatively, without knowledge about the traffic matrix, we propose a probabilistic approach for determining potential bottleneck links that allows us to discover fairly effective on-demand paths. We define the *stress factor* $sf_{i\rightarrow j}$ of a link as the ratio between the number of flows routed via that link in the always-on assignments and the link capacity: $sf_{i\rightarrow j} = \frac{\sum_{(O,D)} Y_{i\rightarrow j}}{C_{i\rightarrow j}}$. Intuitively, this metric captures how likely it is that a link might be a bottleneck. Next, we compute the on-demand paths by solving the optimization problem with the additional constraint of avoiding a certain fraction of links with the highest stress factor. Our sensitivity analysis shows that excluding 20% of the links with the highest stress is sufficient to produce a set of paths that together with the always-on paths can accommodate peak-hour traffic demands.

Using the existing heuristics. There exist several alternatives to using a solver to find a solution to a computationally hard problem of optimal on-demand paths for large networks. We describe the most prominent ones in Section 2.3. For example, GreenTE [41] proposes one such heuristic for ISP networks, and we use it in our evaluation as *REsPoNse-heuristic*. ElasticTree [25] has one for fat-tree datacenter networks.

Using the existing routing tables. In the case of unavailability of such approaches, one can simply substitute the default routing table for the set of on-demand paths. Doing so will ensure that the network can carry the same amount of traffic as before. One of the most widely-used techniques for intradomain routing is OSPF, in which the traffic is routed through the shortest path according to the link weights. We use the version of the protocol advocated by Cisco, where the link weights are set to the inverse of link capacity. We call the resulting set of on-demand paths *REsPoNse-ospf* and demonstrate that REsPoNse can effectively use the existing OSPF paths in Section 5.2.

4.3 Computing the Failover Paths

Our goal is to construct the failover paths in a way that all paths combined are not vulnerable to a single link failure, similarly to [28]. In the case where it is not possible to have such three paths, it is still desirable to find the set of paths that are least likely to be all affected by a single failure. We have opted for a single failover path per (O,D) pair because our analysis revealed that even a single path can deal with vast majority of failures, without causing any disconnectivity in the network. We note however that the origin-destination pairs which are affected by the failure are using the potentially less energy-efficient paths.

4.4 REsPoNse's online component

Inspired by online TE efforts [24, 26, 36], we design **REsPoNseTE**, a simple TE algorithm that operates on top of REsPoNse's paths and tries to maximize energy savings. In REsPoNseTE, the intermediate routers periodically report the link utilization, while the edge routers (called agents), based on the reported information, shift the traffic in a way that preserves network performance and simultaneously minimizes energy.

Aggregating traffic. Because REsPoNse solves the problem of pre-computing the routing tables, ResPoNseTE's agents are fairly straightforward; they aggregate the traffic on the always-on paths as long as the target SLO is achieved, and start activating the on-demand paths when that is no longer the case. Further, REsPoNseTE allows the ISPs to set a link utilization threshold, which in turn enables REsPoNseTE to try to prevent the performance penalties and congestion by activating the on-demand paths sooner.

While shifting traffic, one needs to be careful to avoid persistent oscillations and instability, especially as the traffic frequently changes. Fortunately, making an adaptive and fast-converging protocol stable has been successfully addressed elsewhere [26]. Just as in

TeXCP [26], we implement a stable controller to prevent oscillations in all our experiments.

Collecting link utilization. Like any other TE technique, REsPoNseTE requires up-to-date link utilization to maximize its objective. Every T seconds, REsPoNseTE edge routers fire a probe packet in which they ask intermediate routers to report their link utilization. As opposed to state-of-the-art approaches [25, 41], REsPoNseTE agents require the link utilization only from the paths they originate, which makes our approach highly scalable. This information can be collected using some lightweight technique (e.g., [37]).

Parameter T controls the trade-off between the convergence time versus overhead: a small value guarantees faster convergence but also increases the overhead. Following suggestions from others [26,36], in our implementation we set T to the maximum round trip time in the network.

4.5 Deployment discussion

Can REsPoNse help with power delivery limits and cooling energy? Instead of provisioning the power infrastructure for the peak hours, REsPoNse allows network operators to provision their network for the typical, low to medium level of traffic. Our trace analysis reveals that the average peak duration is less than 2 hours long, implying that alternative power sources can supply necessary power during these periods [20]. Moreover, existing thermodynamic models like [38] can estimate how long the peak utilization can be accommodated without extra cooling, while keeping the temperature at desired levels.

Does REsPoNse increase end-to-end packet delay? We can bound the maximum propagation delay by introducing the constraint (4). We further note that it is possible to easily address the potential queuing delay increase by reserving some network capacity using a safety margin sm to assign the link capacities as $C_{i\to j} \leftarrow sm \cdot C_{i\to j}$. The network operator can use these parameters to specify a limit on link utilization and hence control the trade-off between achieved power savings and reserved capacity to account for unanticipated traffic bursts and processing overheads.

Can REsPoNse deal with unforeseeable traffic peaks? Yes – REsPoNse offers no worse performance than the existing approaches under unexpected traffic peaks. Indeed, network operators can reserve a set of on-demand paths for their existing routing protocol (e.g. OSPF) that are promptly activated if the network does not deliver the required performance (due to unexpected traffic patterns, or any other reasons). We also experiment with REsPoNse-ospf version of our prototype in the evaluation section.

Can REsPoNse deal with long wake-up times? It has been shown that one can exploit network virtualization capabilities to transparently maintain the state of the turned-off elements and thus reduce their wake-up times [12, 40]. Further, REsPoNse can reserve extra capacity from always-on paths which is ready to accommodate potential traffic spikes while we wait for additional (on-demand) resources to wake-up. Finally, future networking elements will most likely support various sleep modes [25] with shorter wake-up times. This will enable REsPoNse to better deal with this issue.

How can REsPoNse be deployed in ISP and datacenter networks?

ISP networks. At a high-level, we need an ability to install the computed routing tables into the routers. MPLS [21] allows flows to be placed over precomputed paths. REsPoNse places modest requirements on the number of paths (three) between any given origin and destination. If we assume that the number of egress points in large ISP backbones is about 200-300 and the number of supported tunnels in modern routers is about 600 (as was at least the case in 2005 [26]), we conclude that REsPoNse can be deployed even in large ISP networks. If the routing memory is limited (e.g. Dual Topology Routing [30] allows only two routing tables), we can deploy only the most important routing tables, while keeping the remaining ones ready for later use.

Datacenter networks. Recent datacenter networks have higher redundancy levels than the ISP networks, but they only require REsPoNse to be instructed to compute multiple on-demand routing tables. Datacenter networks are typically built using equipment from a single vendor however, which makes it easier to request and deploy new protocols and extensions than over the ISP networks.

5. EVALUATION

5.1 Experimental Setup

Techniques. We analyze the performance of various approaches to computing the REsPoNse routing tables (Section 4.1 and 4.2), and contrast them with the theoretical scenario that computes the optimal network subset for every particular traffic demand (called *optimal*).

Datacenter networks. We analyze the power savings potential for datacenter networks by considering a FatTree topology [7]. We experiment with the same sine-wave demand as in [25] to have a fair comparison against the ElasticTree. This demand mimics the diurnal traffic variation in a datacenter where each flow takes a value from [0, 1Gbps] range, following the sin-wave. We considered two cases: near (highly localized) traffic matrices, where servers communicate only with other servers in the same pod, and far (non-localized) traffic matrices where servers communicate mostly with servers in other pods, through the network core.

ISP networks. We experiment with the topologies

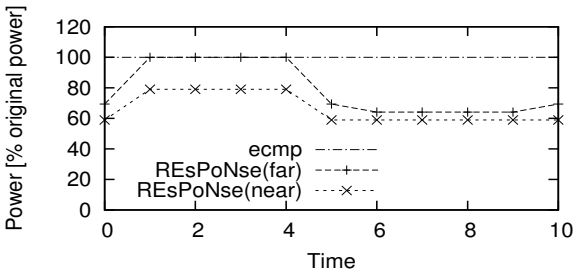


Figure 4: Power consumption for sinusoidal traffic variation in a k=4 fat-tree datacenter topo.

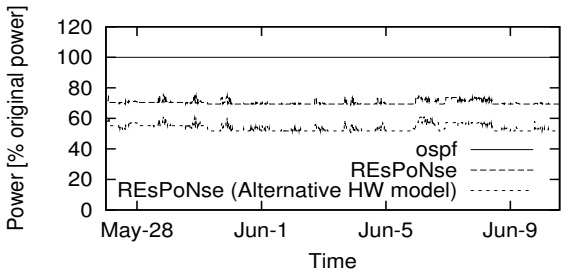


Figure 5: REsPoNse's power consumption for the replay of GÉANT traffic demands.

of: (*i*) large topologies inferred by Rocketfuel [32], (*ii*) GÉANT, the European research and educational network, and (*iii*) an Italian ISP.

We use two Point of Presence (PoP)-level Rocketfuel topologies: Abovenet and Genuity. We leave the link latencies as determined by the Rocketfuel mapping engine. Since the topologies do not originally have link capacities assigned, we keep the values chosen in [26] where links are assigned 100 Mbps if they are connected to an end point with a degree of less than seven, otherwise they are assigned 52 Mbps.

For these networks the traffic demands are not publicly available. Therefore, we infer traffic demands using a capacity-based gravity model (as in [9, 14]), where the incoming/outgoing flow from each PoP is proportional to the combined capacity of adjacent links. To demonstrate energy-proportionality under changing traffic loads (e.g., due to diurnal patterns), we first compute the maximum traffic load as the traffic volume that the optimal routing can accommodate if the gravity-determined proportions are kept. We do this by incrementally increasing the traffic demand by 10% up to a point where CPLEX cannot find a routing that can accommodate the traffic. Then, we mark the largest feasible traffic demand as the 100% load. We select the origins and destinations at random, as in [24].

The topology of GÉANT is described in details in [33] which makes available a dataset of accurate traffic matrices measured over 15-min intervals. In our evaluation, we use a 15-day long trace from 25 May 2005. Also in this case, we select random subsets of origins and destinations as in [24].

The details of an Italian ISP topology are published in [15]. The topology, called PoP-access in the rest of the section, includes backbone and access routers and has a hierarchical design with a significant amount of redundancy at each level. We focus only on the top three levels of the topology: core, backbone and metro (from top to bottom), because the feeder nodes have to be powered on at all times.

We compute the baseline on-demand paths (called simply REsPoNse) using the solver and the demand-oblivious approach, based on the *stress* factor. Namely, we exclude 20% of the links with the highest stress factor for the computation of on-demand paths.

Power consumption model. We evaluated REsPoNse with a set of different power models in mind. First, we used a typical configuration of a Cisco 12000 series router [1] with low to medium interface rates– each line-card (OC3, OC48, OC192) consumes between 60 and 174 W [41], depending on its operating speed, while the chassis consumes about 600 W (around 60% of the router's power budget). Given the power consumption figures we found for optical repeaters (1.2 W as in [3]), we assume that their power consumption is negligible compared to the one of line cards and chassis. Second, we also evaluate REsPoNse with an alternative hardware model in which the power budget for always-on components (chassis) is reduced by factor of 10. This model reflects the potential outcome of ongoing efforts to make individual network elements more energy-proportional. Finally, given that the FatTree [7] approach advocates using commodity hardware, in our experiments with datacenter networks we use a model that captures the energy-unproportionality of off-the-shelf switches, in which the fixed overheads due to fans, switch chips, and transceivers amount to about 90% of the peak power budget even if there is no traffic. We assume that a network element that has its traffic removed can enter a low-power state in which it consumes a negligible amount of power [29].

5.2 Can REsPoNse Match State-of-the-Art Energy Savings?

First, we experiment with a datacenter network where we compare REsPoNse against ElasticTree [25] and Equal-Cost Multi-Path (ECMP) routing for both localized and non-localized traffic demands. Figure 4 plots the network power consumption as we change its traffic demands according to the sine-wave pattern. It confirms that REsPoNse is capable of achieving significant power savings, matching ElasticTree with their formal solution (ElasticTree data points coincide with REsPoNse, and are not shown to improve clarity of the graph). In contrast with ElasticTree, REsPoNse can adapt to different traffic conditions in only a few RTTs, as shown in the next section.

Next, we move to the ISP topologies. Figure 5 shows

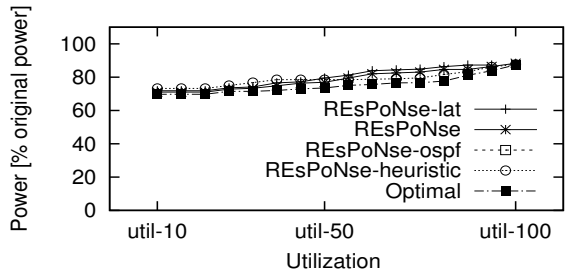


Figure 6: Power consumption for different traffic demands in the Genuity topology.

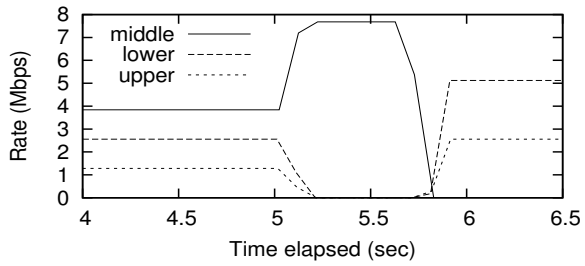


Figure 7: REsPoNseTE quickly enables links to sleep, and restores traffic after link failure.

the power savings that are possible when using REsPoNse on the GÉANT topology. The figure depicts a replayed 15 day-long trace of traffic demands based on the traffic matrices observed over 15-minute intervals. For each traffic demand, we compute the topology, along with its power consumption that is shown in the figure, that will be put into place by running REsPoNseTE. We compare REsPoNse with OSPF-InvCap in this case. The results show that energy savings are around 30% and 42% (for representative hardware today and a future alternative, respectively) for this realistic traffic and topology. We also see that the power consumption varies little with large changes in traffic demand. This happens because the always-on paths are sufficient to absorb the traffic most of the time. Most importantly, there was no need to recompute the on-demand paths - a single computation of always-on and on-demand paths was sufficient for the 15-day period.

Figure 6 shows the corresponding savings for Genuity topology for various gravity-derived traffic demand intensities. We see that REsPoNse makes a significant step towards energy-proportionality. First, the savings are around 30% in lower utilization regions, and as the utilization increases REsPoNse and REsPoNse-lat progressively activate more resources. By enforcing the latency constraint REsPoNse-lat marginally reduces the savings while keeping the latency acceptable.

Even REsPoNse-heuristic and REsPoNse-ospf (on-demand paths computed as the OSPF routing table) exhibit energy-proportionality, which confirms REsPoNse's potential to deal even with large topologies. REsPoNse-heuristic saves more energy at high network loads because it is traffic-aware. On the other hand, REsPoNse and REsPoNse-lat exhibit substantial energy savings while being traffic demand-oblivious. This increases our confidence that REsPoNse can deal with unplanned changes in the traffic volume and pattern.

5.3 Does REsPoNse enable Quick Adaptation to Sudden Traffic Changes?

Given that network traffic can significantly vary over short time intervals (Figures 1), our next step is to explore whether REsPoNse enables timely adaptation to such frequent changes in traffic load. While we implemented REsPoNseTE in both OpenFlow and Click, we show here only results obtained using the Click implementation, which is currently more mature.

Click experiments. We perform our first experiments in a live setup. We implemented REsPoNseTE in the Click [27] modular router and run 10 Click routers in separate virtual machines. The packet processing overhead introduced by REsPoNseTE is modest, between 2 and 3% of the time a packet spends in the router.

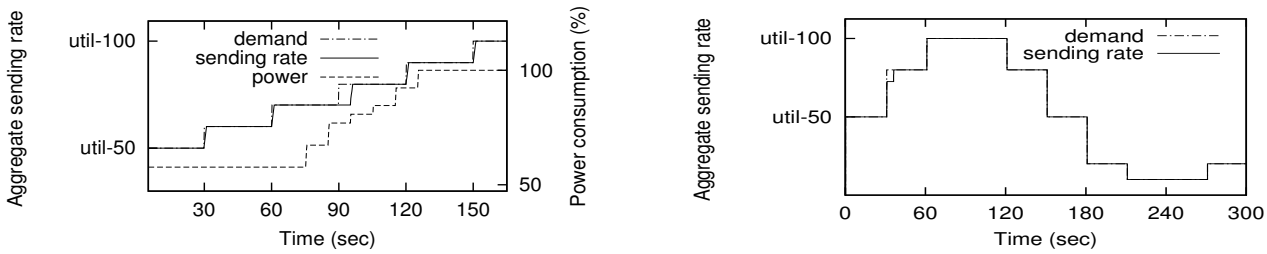
We interconnect the Click instances via virtual links to form the topology shown in Figure 3 (excluding router B), and use `lartc` to enforce 16.67 ms latency and 10 Mbps of bandwidth on each link. We assume that it takes 100 ms for the link failure to be detected (50 ms) and propagated to the sources (50 ms = 3 hops of 16.67 ms), and 10 ms to wake up a sleeping link (the estimated activation times of future hardware [29]).

In this setup, routers A and C each have traffic consisting of 5 flows to router K (10 pps, or about 5 Mbps in total), that can be routed over two different paths. These flows can all traverse the "middle" always-on link E-H. Flows originating from A can also take the "upper" on-demand path, while C flows can also go over the "lower" on-demand path. In this topology, the failover paths are coinciding with the on-demand paths.

Figure 7 shows that after REsPoNseTE starts running at $t = 5$ s, it quickly (200 ms, which is 2 RTTs of 6 hops with 16.67 ms each) shifts traffic away from the links in the "upper" and the "lower" on-demand paths, allowing their links to sleep. We then deliberately fail the "middle" always-on link at $t = 5.7$ s, which causes REsPoNseTE to promptly shift the traffic back to the on-demand/failover paths that are sleeping. Each of these paths now needs to carry more traffic as the always-on path is down.

This experiment demonstrates that it is possible to: 1) quickly and effectively use the on-demand and always-on REsPoNse paths at runtime for energy-saving, 2) quickly tolerate faults using the failover paths, and 3) incur low overhead at the router.

Ns-2 simulations. Next, we run ns-2 simulations with PoP-access ISP and FatTree datacenter topologies. For each topology, we aggressively change the traffic demands every 30 seconds, following the grav-



(a) PoP-access (b) FatTree.

Figure 8: Ns-2 simulations for representative ISP and datacenter networks show that our hybrid approach quickly adapts to traffic changes, while simultaneously saving energy.

ity model and sine-wave pattern, respectively. Further, we set the wake-up time to 5 s, an upper bound on the time reported to power on a network port in existing hardware [25]. Because the routing tables are installed only once, we do not incorporate additional time for that purpose. Finally, for this set of experiments we implemented REsPoNseTE in ns-2 network simulator.

Figure 8a demonstrates that sending rates for each (O,D) pair quickly match the given demands. They are only delayed by a few RTTs which is the time needed by REsPoNseTE's controller to collect the link utilizations and decide which path (always-on or on-demand) to use. Only at time t=90 s, the rates were delayed by 5 s, which corresponds to the time that is needed to wake up additional resources in the on-demand paths. A similar trend is also seen in Figure 8b where REsPoNse's sending rates even more closely match demand because the RTT in a datacenter network is much smaller than that in an ISP network. Here one can see that the on demand resources were waken up at time t=30 s. In summary, REsPoNseTE quickly adapts to changes in network traffic using the REsPoNse routing tables.

5.4 Does Usage of Energy-Critical Paths Affect Application Performance?

Our final step is to examine whether aggregating traffic in pursuit of energy savings impacts application-level performance for representative workloads. For this set of experiments we use ModelNet [34], a network emulator that enables us to run live code on a set of cluster machines. We modified the emulator so that it can use the three routing matrices as needed by REsPoNse. The energy-proportional routing tables we installed are those computed with REsPoNse-lat. Our comparison point is again OSPF-InvCap, in Abovenet.

First, we run BulletMedia [39], a live **streaming application** which is sensitive to both network latency and achievable network throughput. We start 50 participants, with the source streaming a file at 600 kbps. This load corresponds to the lower utilization level that lets REsPoNse aggregate all traffic on the always-on paths. Then, after 300 s, we let 50 additional clients join the system to increase the load so that the on-demand paths have to be activated.

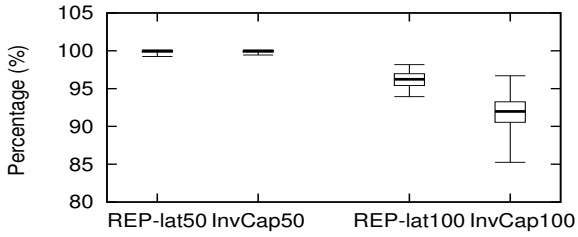


Figure 9: Boxplot with whiskers from min to max of a number of clients that can play the video. REP stands for REsPoNse (for clarity).

Figure 9 depicts the percentage of users that can play the video (i.e., media blocks are arriving before their corresponding play deadlines) under two different utilization levels (50 and 100 clients). As desired, when the network utilization is fairly low or medium (50 clients), carefully consolidating the traffic on the always-on paths has a negligible impact on users' throughput. Turning on on-demand paths similarly does not reduce application-level performance.

We also measure the average block retrieval latency to quantify the overall impact on observable application-level latency. The results show that this latency is increased by about 5% when REsPoNse-lat is used.

We finally quantify the impact of REsPoNse on end-host performance by running a **web workload**. One of the stub nodes is running the Apache Web server, while the remaining four stub nodes are using httperf. The Web workload in our case consists of 100 static files with the file size drawn at random to follow the online banking file distribution from the SPECweb2005 benchmark. The web retrieval latency increases by only 9% when we switch from OSPF-InvCap to REsPoNse.

6. CONCLUSIONS

For networks in which the demands change infrequently, it is possible to minimize energy consumption by solving or approximating an optimization problem. Unfortunately, in most of the today's ISP and datacenter networks the traffic demand is changing on an hourly basis. Thus, a trade-off arises between one's ability to save energy in computer networks, and the ability of the resulting network topologies to quickly adapt to changes in the traffic demand. In this paper, we first quantify

this trade-off by examining the recomputation rate using traces of real link utilization. We proceed to analyze the real traffic matrices and routing tables, and identify *energy-critical* paths that carry the traffic most of the time. We then leverage this finding to propose a responsive, energy-proportional framework (REsPoNse) that overcomes the optimality-scalability trade-off, by avoiding frequent routing table recomputation. One of our most important findings is that the significant power savings can be achieved with only marginal impact on the application-level throughput and latency. As part of our future work, we plan to quantify the level at which topology changes (failures, routing changes, etc.) would warrant recomputing the energy-critical paths.

Acknowledgments This research is funded by the Swiss NSF (grant FNS 200021-130265). Dejan Novaković is supported by the Swiss NSF (grant FNS 200021-125140). We thank our shepherd, Laurent Massoulié, anonymous reviewers, and Nikolaos Laoutaris and Dina Papagiannaki for their constructive feedback.

7. REFERENCES

[1] Cisco xr 12000 and 12000 series. http://www.cisco.com/en/US/prod/collateral/routers/ps6342/prod_qas0900aecd8027c915.html.
[2] ILOG CPLEX. http://www.ilog.com/products/cplex/.
[3] Teleste optical repeater for singlemode fiber. www.teleste.com/ProductDocument/P4P_COR.pdf.
[4] Cisco Visual Networking Index: Forecast and Methodology, 2008-2013., 2009.
[5] Microsoft's Chiller-less Data Center. http://www.datacenterknowledge.com/archives/2009/09/24/microsofts-chiller-less-data-center/, 2009.
[6] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. In *ISCA*, 2010.
[7] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 2008.
[8] M. Andrews, A. Anta, L. Zhang, and W. Zhao. Routing for energy minimization in the speed scaling model. In *INFOCOM*, 2010.
[9] D. Applegate and E. Cohen. Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Trans. Netw.*, 14(6):1193–1206, 2006.
[10] C. Bianco, F. Cucchietti, and G. Griffa. Energy consumption trends in the Next Generation Access Network - a Telco perspective. In *INTELEC*, 2007.
[11] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier. A survey of green networking research. *CoRR*, 2010.
[12] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti. Introducing standby capabilities into next-generation network devices. PRESTO, 2010.
[13] R. Bolla, R. Bruschi, and A. Ranieri. Energy-aware equipment for next-generation networks. In *CFI*, 2009.
[14] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power Awareness in Network Design and Routing. In *INFOCOM*, 2008.
[15] L. Chiaraviglio, M. Mellia, and F. Neri. Energy-aware Backbone Networks: a Case Study. In *GreenComm*, 2009.
[16] U. S. EPA. Report to congress on server and data center energy efficiency. Technical report, 2007.
[17] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science*, 1975.
[18] B. Falsafi. Dark silicon and its implication on server design. http://parsa.epfl.ch/~falsafi/talks/MSR-2010.pdf.
[19] W. Fisher, M. Suchara, and J. Rexford. Greening backbone networks: reducing energy consumption by shutting off cables in bundled links. In *Green Networking*, 2010.
[20] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini. GreenSlot: Scheduling Energy Consumption in Green Datacenters. Technical Report DCS-TR-686, Rutgers University, 2011.
[21] J. Guichard, F. le Faucheur, and J. P. Vasseur. *Definitive MPLS Network Designs*. Cisco Press, 2005.
[22] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOMM*, 2003.
[23] R. Hays. Active/Idle Toggling with Low-Power Idle. http://www.ieee802.org/3/az/public/jan08/hays_01_0108.pdf.
[24] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang. Rethinking Internet Traffic Management: from Multiple Decompositions to a Practical Protocol. In *CoNEXT*, 2007.
[25] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *NSDI*, 2010.
[26] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tightrope: Responsive yet Stable Traffic Engineering. In *SIGCOMM*, 2005.
[27] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3), 2000.
[28] K. W. Kwong, R. Guérin, A. Shaikh, and S. Tao. Balancing performance, robustness and flexibility in routing systems. In *CONEXT*, 2008.
[29] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *NSDI*, 2008.
[30] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. IETF RFC 4915. 2007.
[31] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. In *ISCA*, 2006.
[32] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, 2004.
[33] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon. Providing public intradomain traffic matrices to the research community. *SIGCOMM Comput. Com. Rev.*, 36(1), 2006.
[34] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *OSDI*, 2002.
[35] W. Van Heddeghem, M. De Groote, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester. Energy-efficiency in telecommunications networks: link-by-link versus end-to-end grooming. In *ONDM*, 2010.
[36] N. Vasić and D. Kostić. Energy-Aware Traffic Engineering. In *e-Energy '10*, 2010.
[37] N. Vasić, S. Kuntimaddi, and D. Kostić. One Bit Is Enough: a Framework for Deploying Explicit Feedback Congestion Control Protocols. In *COMSNETS*, 2009.
[38] N. Vasic, T. Scherer, and W. Schott. Thermal-aware workload scheduling for energy efficient data centers. In *ICAC*, 2010.
[39] N. Vratonjić, P. Gupta, N. Knežević, D. Kostić, and A. Rowstron. Enabling DVD-like Features in P2P Video-on-demand Systems. In *P2P-TV*, 2007.
[40] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. SIGCOMM, 2008.
[41] M. Zhang, C. Yi, B. Liu, and B. Zhang. GreenTE: Power-Aware Traffic Engineering. In *ICNP*, 2010.
[42] Y. Zhang and Z. Ge. Finding critical traffic matrices. In *DSN*, 2005.