Segment Routing based Traffic Engineering for Energy Efficient Backbone Networks

Radu CÂRPA, Olivier GLÜCK and Laurent LEFEVRE
Inria Avalon - LIP Laboratory
École Normale Supérieure de Lyon, University of Lyon, France
Email: radu.carpa@ens-lyon.fr, olivier.gluck@ens-lyon.fr, laurent.lefevre@inria.fr

***Abstract*—Energy consumption has become a limiting factor for deploying large-scale distributed infrastructures. This work[1] seeks to improve the energy efficiency of backbone networks by providing an intra-domain Software Defined Network (SDN) approach to selectively turn off a subset of links. We propose the STREETE framework (SegmenT Routing based Energy Efficient Traffic Engineering) that dynamically adapts the number of powered-on links to the traffic load. The core of the solution relies on SPRING, a novel protocol being standardized by IETF. It is also known under the name of Segment Routing. The algorithms have been implemented and evaluated using the OMNET++ simulator. Experimental results show that the consumption of 44% of links can be reduced while preserving good quality of service.**

***Index Terms*—energy efficiency, backbone networks, SDN, traffic engineering, SPRING, segment routing, MPLS.**

I. INTRODUCTION

Networks play an important role in today's highly connected world, and their traffic is expected to increase by a factor of three in the next five years[2]. Although they are responsible for only a fraction of the energy the IT sector consumes, networks can be made more efficient as their components often consume a constant amount of power regardless of their utilization. Backbone networks, for example, comprise devices that consume several KWh of energy even when idle.

Techniques to reduce the energy consumption of backbone networks include traffic rerouting and the use of low power consumption modes. It is possible to reroute data over alternative paths to offload a subset of links because operators over-provision their networks to handle demands during peak-periods and to offer high Quality of Service (QoS) to their clients. Evidence shows, however, that even during peak hours links are rarely used more than 50% [8]. Fig. 1c demonstrates the utilization of links computed from traffic matrices of the Géant network where most links are less than 25% utilized even, at peak hours.

In this work, we tackle the problem of energy consumption of backbone networks by changing the status of router ports and transponders on the two extremities of a link. The status of these components is set to sleep mode whenever a link is not required to transfer data, and brought back to operational state when needed. Hereafter we term this process as switching off/on links.

We analyzed the implementation issues of an energy-efficient SDN-based traffic engineering in core networks. Most works which seek to adapt the number of switched on links to the amount of traffic in the network rest at the conceptual level. They search to minimize the number of active links via solving MILP formulations or heuristics. Few went deeper into implementation details, but still miss-observed or ignored some race conditions. Our team fully implemented the proposed solution in the OMNET++ packet-based discrete event simulator. Experiments considering real network topologies (Germany50 and Géant, see Fig.1) and real dynamic traffic matrices allowed us to quantify the trade-off between energy saving and impact of our solution on network performance.

As mean to reroute the traffic we use a promising new protocol, SPRING. This comes in contrast with other works, which use classical IP link weights changes or MPLS+RSVP-TE for this purpose. SPRING proved itself well suited for dynamic reconfiguration of the network. As a bonus, the usage of SPRING makes the implementation easier. To the best of our knowledge, SPRING has never been considered in the literature for reducing the energy consumption of networks.

The implemented solution is proposed under a form of a framework which we name STREETE (SegmenT Routing based Energy Efficient Traffic Engineering). It represents an online method to switch some links off/on dynamically according to the network load. The generecity of our framework allows gradual enhancements, like adding protection and more intelligent flow placement computations by a simple update of its composing algorithms.

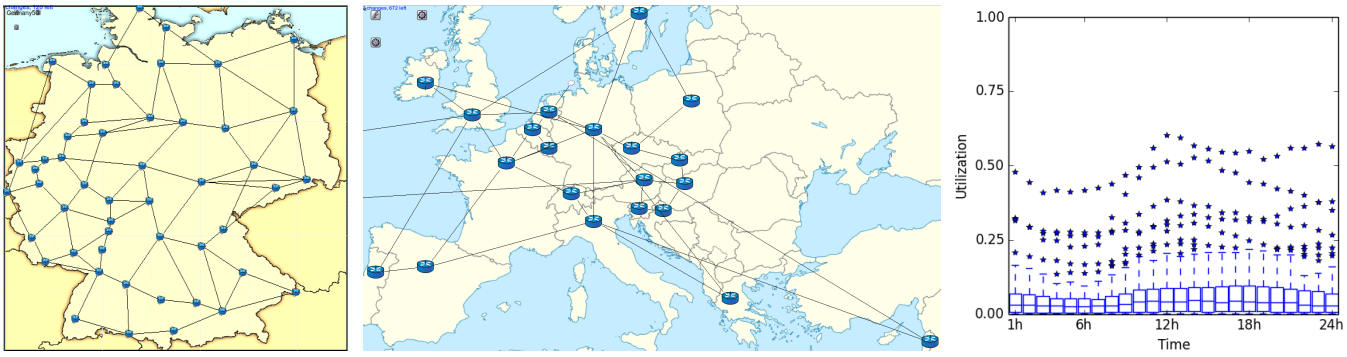
The rest of this paper is structured as follows. Section II discusses related work, whereas Section III describes our assumptions and concepts such as SDN and segment routing/SPRING protocol. After that, we present our solution in Section IV and analyze simulation results in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Gupta *et al.* [7] were among the first to consider putting interfaces into sleep mode to save energy. In backbone networks, however, links cannot be simply switched off without losing data. It has been shown that even micro-second sleeps

[1]The work is done as part of the CHIST-ERA STAR "SwiTching And tRansmission" European project (http://www.chistera.eu/projects/star), which aims to decrease the consumption of backbone networks by a factor of 100 until 2020.

[2]http://ciscovni.com/forecast-widget/index.html



(a) Germany 50 network (b) Géant network (c) Link loads in Géant network

Figure 1: Considered network topologies and Géant usage

are difficult to achieve due to the small inter-packet arrival time [2]. To cope with this challenge, data transmission in bursts has been considered [13]; a technique nearly impossible under modern 100Gbps speeds.

A distributed network-wide solution has been provided by Vasic et al. in order to change flow paths to reduce link utilization, enabling 21% of links of real networks to be put in sleep mode [17]. Such a solution, however, is a concept whose implementation details are not fully disclosed. Another distributed approach considers that every node locally monitors the utilization of adjacent links and decides whether to switch them off or not [1]. Machine learning mechanisms are used to avoid choosing links whose extinctions have already provoked a congestion. This solution heavily relies on the Interior Gateway Protocol Traffic Engineering extensions (IGP-TE) and the communication overhead is underestimated. After detailed analysis, we discovered that due to bad estimations of the frequency of the IGP-TE exchanges [14], the announced overhead of 0.52% is actually a 30% increase of IGP-TE flooding. This is not negligible because these floodings are costly both in number of messages and required processing power. Moreover, the authors used classical IP routing instead of MPLS, which can induce instability periods while the network is re-configuring.

Zhang *et al.* proposed a centralized approach based on MPLS+RSVP-TE [18] that relies on estimation of traffic matrices, proven not very efficient [6]. They also opt for a hybrid approach where MPLS+RSVP-TE is used in parallel with shortest path routing. Our work uses the SPRING protocol, which yields the same advantages with less complexity.

Existing work also concentrates on searching the best links to switch off assuming full knowledge of network state. The results on energy efficiency of the proposed solutions varies drastically from one work to another. For example, at times the performance of greedy approaches like "selecting the least congested link" is claimed to be close to that of optimal algorithms in terms of energy savings (Kamola *et al.* [10]). In contrast, stochastic optimization techniques, like genetic algorithms, are said to outperform greedy algorithms when considering stability of both network and energy savings (Bonetto *et al.* [3]). We attempt to explain this variation in our work.

[4] presents ESTOP, an algorithm which is similar to STREETE. They also test their solution on the Germany50 network. However, they do not exploit the knowledge of the network wise traffic matrices and obtain worst results in terms of energy efficiency.

III. Assumptions and Background

We consider that the variation of traffic in backbone network is slow and follows a diurnal pattern due to a high level of flow aggregation [9]. Table I provides figures on the power consumed by backbone-network equipments [16]. Our model considers that switching a port off corresponds to powering off part of its integrated circuit and the transponder. Switching a link off means to switch off the two ports at the link's extremities. Hence, if a 100Gbps link is switched off, $2 \cdot (135 + 150) = 570$ Watts are saved.

Table I: Router port consumption

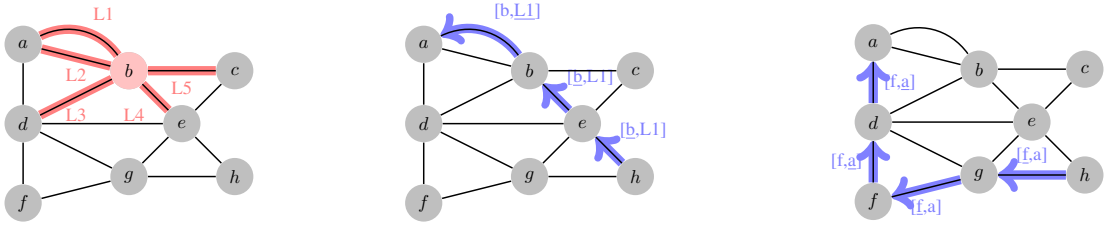
Port Speed (Gbps)	Consumption (W)	
	Port card	Transponder
10	10	50
40	35	100
100	135	150
400	335*	300*

* mathematical projection

A. SDN

In traditional networks, control and data planes are located on network nodes, and adjustments to network-wide policies often require access to, and configuration of, all devices. SDN aims to simplify management by separating the control and data planes. The control plane contains the network intelligence responsible for calculating paths to be followed by data flows and programming them into the data plane. The data plane is responsible for passing packets from an incoming port of a device to the right outgoing port. We refer to data plane devices as SDN switches.

SDN controllers exploit network-wide knowledge to execute all required computations and apply changes into the data



(a) Segment Identifiers (b) Force data over a link (c) Force data over a node

Figure 2: SPRING protocol

plane of network nodes through a predefined API. SDNs enable a lot of flexibility for energy efficient traffic engineering.

Compared to the data-center networks, the optical backbone networks face problems (such as lightpath provisioning), for which the current SDN implementation, OpenFlow, is not ready. However, researchers succeed to bring the SDN concept into the backbone [15]. In this paper we assume the availability of such a control plane, whose comportment we can change for our needs: for example, by modifying its source code.

We focus on a centralized SDN solution. While it is possible to chose the best links to be shut down in a distributed fashion [1], more complex and intelligent decisions can be taken if done in a centralized manner.

B. SPRING protocol

SPRING [5], also known as Segment Routing, is an IETF draft since 2013 and has the goal to substitute MPLS+RSVP-TE for traffic engineering. It combines the power of source routing (flexible traffic engineering) with shortest path routing (less signaling/header overhead compared to MPLS/Myrinet).

The data plane of the SPRING protocol uses the same concept of label switching of MPLS and can even use existing MPLS-capable devices without modification. The control plane though, suffered a complete re-design. Called Segment IDentifiers (SID), labels are global identifiers, unlike classical MPLS where they are local. Here, we are interested in 2 types of SIDs, namely "nodal" and "adjacency" (Fig. 2a).

- A nodal SID is globally unique and identifies a node $(a, b, c, ..., h)$.
- An adjacency SID is local to a node and uniquely identifies an outgoing interface (node b has the Adjacency SIDs $L1, L2, L3, L4$ and $L5$).

Compared to MPLS, the distribution of labels is done via an extension to the IGP instead of using special protocols (LDP/RSVP-TE). After the network discovery, sending a packet to the node a through the shortest path resumes to encapsulating it into a packet with destination a. However, unlike in IP, much more flexible traffic engineering is possible:

- If node h wants to send a packet to node a while forcing it over link $L1$, it adds the header $[b, L1]$ (Fig. 2b). As a result, the packet will take the shortest path to b, and b will force it through link $L1$ to a.
- In Fig. 2c, if h wants to send a packet to a via f, it will add the header $[f, a]$. As result the packet will take the shortest path to f, and afterward continue to a.

SPRING is a good choice for dynamic rerouting regardless of opting for an SDN approach or not. Being a source routing protocol, it enables fast flow setup and easy reconfiguration of virtual circuits with minimum overhead. Changes must be applied only on the ingress devices and no time and signaling is lost re-configuring the midpoint devices. This is important when frequent routing policy changes are needed.

We also use SPRING protocol because MPLS+RSVP-TE faces convergence issues in case of link failures[3]. With small numbers of RSVP-TE virtual tunnels and the refresh reduction extension added to RSVP-TE, the risk has been reduced. However, in our case, due to rerouting the virtual tunnels over a subset of paths and switching other links off, we increase the load on some routers and the number of LSPs per router. For this reason, SPRING suits better our work because it is more stable with a larger number of dynamically changing LSPs.

IV. The STREETE framework

A. Analysis

In this section we formalize the problem of the energy efficient traffic engineering. The goal is to avoid losing connectivity between any two nodes in the network. The term **Connectivity Constraint** is used to refer to satisfying the integrity of the network.

We divided our solution into the following three steps:

- Selecting links to switch off/on.
- Computing new routes to avoid/re-use these links.
- Pushing these routes to the network devices and switching off/on the links.

1) Selecting links to switch off/on: This step selects candidate links to be switched off/on. We implement the "least congested link" technique because it is the approach against which most of existing work compare their results. This step, quite straightforward, first sorts links by their instant transmission speed. The complexity of this step is $O(E \cdot log(E))$, where E is the number of links (Edges). It then tries to switch off links that transmit less data by testing, for each link, whether shutting it down does not violate the connectivity constraint. At most it will take $O(E \cdot (E+V)) : E$

[3]http://www.ietf.org/mail-archive/web/bmwg/current/msg01502.html

times breadth-first search to test the integrity of the network, where V is the number of nodes (Vertices). The algorithm that searches for good links to be switched off/on is hereafter referred to as **<SelectLinksToOff/On>**.

2) Computing new routes for data flows: Prior to actually switching off/on the links selected by <SelectLinksToOff/On>, we must compute new paths for every flow traversing them. It may happen that the network is not capable of routing all data flows if all the selected links are switched off. Moreover, letting every node recalculate new shortest paths throughout the network is not efficient and may produce congestion. In fact, congestion may occur even when a link is turned on, such as when the turned-on link provides a shortest path for a large number of flows.

To compute the new routes, the SDN controller exploits global knowledge about the network traffic to solve the "Multicommodity Flow Problem" which has been proven to be NP-complete, but for which approximation techniques exist [12]. As a prerequisite, the SDN controller has to know the used bandwidth between any head-end to any tail-end (the traffic matrix). The statistics must be collected on the ingress SDN switches. Fortunately, the availability of this kind of metric is already considered in the OpenFlow SDN specification. At this step, the algorithm may also compute protection paths [11].

In our simulation we implemented a technique that computes the all-pairs shortest path routes and tests the possibility to route all the available traffic over these paths. The complexity of the computation is $O(V \cdot (E + V \cdot logV))$ if Dijkstra from every node is computed. The complexity may decrease with the use of an optimized all-pairs shortest path algorithm.

In the case of switching on, our algorithm is let as simple as possible. It tests the utilization of the most utilized link, and if it is greater than 75%, everything is switched on the shortest path routes are recalculated. We refer to this algorithm as **<ComputeNewRoutes>**.

3) Rerouting and switching links off/on: Once new routes for flows are computed, the controller directly informs each node about the explicit routes to be introduced in the forwarding table and the links will be switched off or on.

The steps taken to switch off the links are described in the Algorithm 1. For example, here the links between nodes A and B, C and D, etc were selected for shutdown. The algorithm presents the steps to be executed to switch these links off.

The overhead of this solution in terms of exchanged messages is smaller than for distributed solutions. In distributed cases, the nodes need to synchronize their actions to avoid switching off a link which disconnects the network. In the centralized SDN case, if network failures do not occur, there is no risk to violate the connectivity constraint. The cases of network failures may be treated by adding protection constraints to <SelectsLinksToOff> and <ComputeNewRoutes>. We do not address this issue in this paper.

4) Main Loop: <MainLoop> demonstrates how the presented algorithms interact. This algorithm searches links that can be switched on or off. If it finds such links, it will actually execute <RerouteAndSwitchOn> or <RerouteAndSwitchOff>.

Data: List-Of-Links :[(A-B),(C-D),...] and the new routes
`/* given by <SelectLinksToOff> and <ComputeNewRoutes> respectively */`

1 **begin**
2 The controller tells every router the new paths to be used for virtual circuits;
3 The controller tells nodes adjacent to links (A,B,...) to turn them off;
4 Nodes A,B,C,... ACK to the controller ;
5 The SDN controller sends IGP flooding : link A-B is down
`/* flooding is needed, because at line 2 we may not contact every node */`
6 Every router applies the changes to the forwarding table to reroutes over the new paths;
7 Nodes A,B,C,... suspend the IGP HELLO messages ;
8 Let the links (A-B,C-D,...) go to a sleep mode / negotiate the shutdown ;
9 **end**

Algorithm 1: Centralized SDN: <RerouteAndSwitchOff>

1 **while** *Energy Efficient Traffic Engineering is active* **do**
2 List-Of-Links = <SelectLinksToOn>();
3 **if** <*ComputeNewRoutes*>*(All-links-that-are-ON* $\cup$ *List-Of-Links)* **then**
4 <RerouteAndSwitchOn>(List-Of-Links);
5 **end**
6 List-Of-Links = <SelectLinksToOff>();
7 **if** <*ComputeNewRoutes*>*(All-links-that-are-ON* $\setminus$ *List-Of-Links)* **then**
8 <RerouteAndSwitchOff>(List-Of-Links);
9 **end**
10 **end**

Algorithm 2: Main loop

V. Performance Evaluation

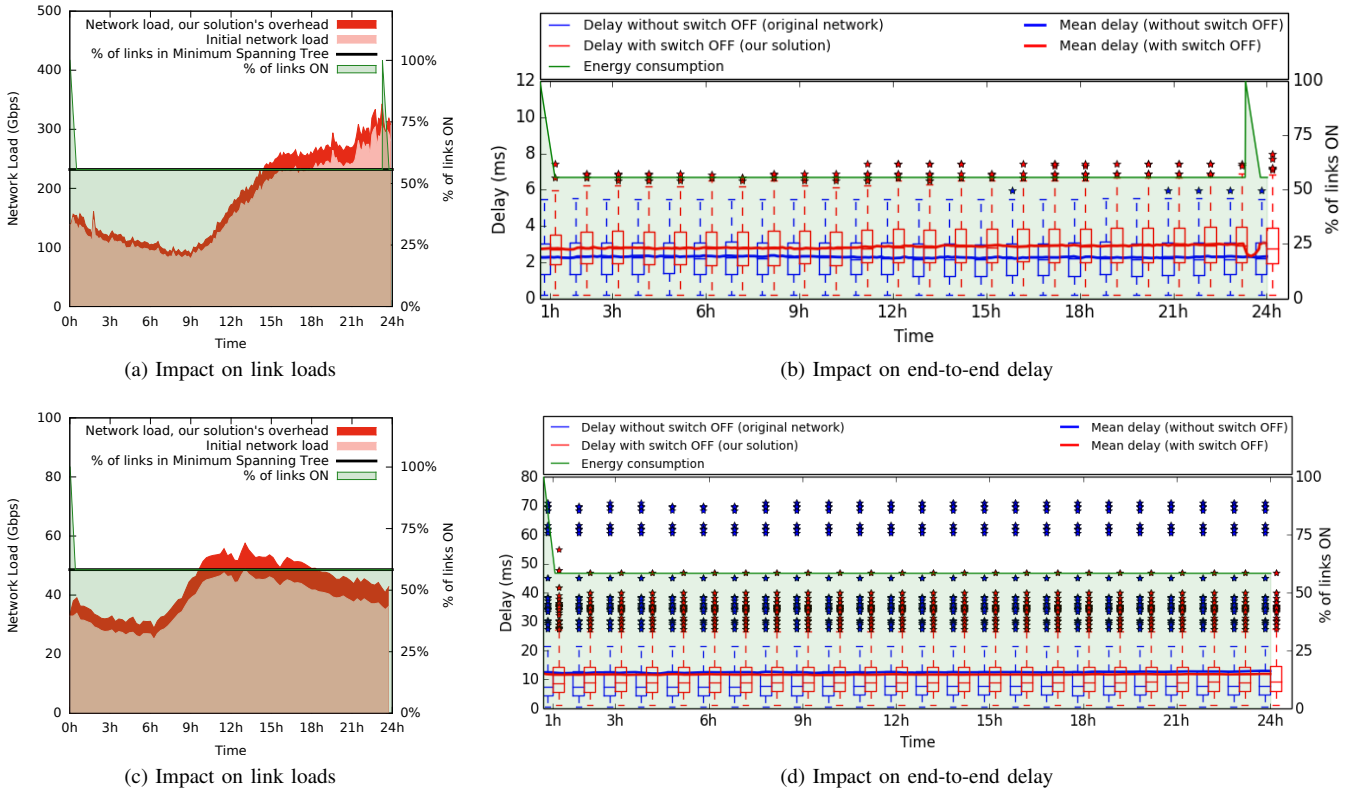
A. Environment Setup

The STREETE approach was implemented in the OMNET++ simulator. We also developed a draft version of the SPRING protocol to dynamically reroute data in the network to avoid turned-off links. At the border of the network domain, during the classification of each packet and assignment to a virtual circuit, the ingress nodes collect the statistics about the amount of data sent to all the other nodes inside the domain. The SDN controller retrieves this information from every router and generates the network-wide traffic matrix to be used by the algorithm presented in the previous section.

We tested our solution considering two different networks, summarized in Table II. The Germany50 network is a real backbone, *i.e.* the traffic is the aggregation of a large number of flows and is steady over the time. Although Geant is interconnecting research networks, the aggregation degree is high enough and the traffic pattern is similar to other backbone networks. We still observe slightly higher traffic variations, but that did not have any impact on our solution.

We tested STREETE by simulating real networks with real traffic matrices from SNDLib[4], which were scaled to represent

[4]http://sndlib.zib.de/



(a) Impact on link loads

(b) Impact on end-to-end delay

(c) Impact on link loads

(d) Impact on end-to-end delay

Figure 3: Simulation results in Germany50 (top) and Géant (bottom) networks

Table II: Evaluated network topologies.

Network	Number of nodes	Number of links
Géant (Fig. 1b)	22*	36
Germany50 (Fig.1a)	50	88

*the New York node is not visible in Fig. 1b, but the two links connecting it to the rest of the network are shown.

Table III: Energy savings during the simulated day.

Network	Average off time of links	Energy economy (kWh)
Géant	14.88 out of 36	204
Germany50	38.18 out of 88	522

today's faster network speeds. The links in the network are switched off one by one while no link reaches the threshold utilization of 60%. Over this value, we stop switching off links. When the utilization of any link goes beyond 75% for a couple of minutes, the network is switched back on.

The propagation delay was calculated as the time needed for light to travel through the fiber. The distance between nodes is the bird-fly distance. Queuing delay is simulated by OMNET++.

B. Results

1) Energy consumption: We simulated a day of operation for each network. In Fig. 3, green lines represent the % of links left on by our algorithm. Using the values of Table I for 100Gbps links, this translates into reductions in energy consumed during the simulated day as shown in Table III.

At the beginning of the day, when our algorithm starts to execute, both networks start to shut down links one by one every couple of minutes and converge very fast to a spanning tree. The further extinction of links is not possible without violating the connectivity constraint. In the case of the Germany 50 network, we observe a turn-on at 23:00h. It is due to a large increase in traffic demand between two network nodes. We do not know the reason for this unusual increase in the original traffic matrices, but this demand was handled well by our algorithm which turned on sleeping links and restored the full capacity of the network.

We discovered that even greedy techniques for selecting links to switch off provide good energy savings due to high over-provisioning of the analyzed networks. An *a posteriori* detailed analysis of the work which affirms the contrary ([3]) on the same traffic matrices showed that they compare a greedy approach acting at the physical link granularity with complex techniques acting on the light path granularity.

2) Impact on link loads: The fact of rerouting data over paths which are not the shortest possible, induces overhead in the network load. The overhead of our solution is visible in Figures 3a and 3c. The maximum overhead observed during these simulations was 18.56%. It may be noticed that even with this overhead, the networks converge to a minimum energy state.

At the beginning of the simulations and the moment when the network was turned on, in the case of Germany50 we

observe that the overhead is null, as expected.

3) Impact on end-to-end delays: Figures 3b and 3d show the impact of our solution on the end-to-end delay. For every source and destination node, we calculated the mean end-to-end delay per intervals of one hour. The box plots represent the distribution of the means of all pairs (source,destination) during the previous hour. The blue ones describe the delays in the original network and the red ones the delay in the network with switched off links. We also represent the mean delay for the cases with switch off and without switch off as continuous lines of the same colors.

In the case of the Germany50 network, the delay increases by 37% in the worst case, but usually increases by around 20%. A much more interesting result can be seen in the Géant network. While in most cases the delay still slightly increases, it can be clearly seen that the maximum outlier suffered a delay reduction. It is due to the minimum hop count shortest path routing. The Géant network has a complex physical topology and it occurs that the minimum hop count routing is not well suited for this network: in some cases traffic between two European nodes will pass through New York. Turning off the 2 links forced the flows to pass through multiple hops, but remaining in Europe.

4) Impact on packet loss: The simulation revealed absolutely no packet loss due to our algorithms. In the case of backbone networks, the aggregated traffic flows do not change fast enough to produce congestion implied by our solution. The algorithm wakes up the links and serves the increased demand.

VI. Conclusion

In this work, we implemented an energy-aware traffic engineering technique for reducing energy consumption in backbone networks. Energy efficient traffic engineering was analyzed in previous work, but none addressed implementation challenges of their solutions. We showed that ignoring to test the feasibility of techniques can lead to bad estimations and unstable solutions. We proposed and implemented a working prototype in the OMNET++ simulator.

Nowadays networks are changing towards a centralized architecture and SDN is increasingly utilized in data center production networks. We believe that the SDN philosophy may be extended to the backbone networks. The simplicity of the implemented solution shows that SDN may also be a good path for reducing the energy consumption of the network devices.

Compared to previous work, in this paper we used the SPRING transport protocol to improve the stability of energy efficient traffic engineering solutions. To the best of our knowledge, this is the first work proposing the use of SPRING to improve the energy efficiency of backbone networks. The flexibility of this routing protocol is well suited to frequent route changes that happen when we switch links off and on. Moreover, this protocol is very SDN-friendly.

Via simulations, we showed that as much as 44% of links can be switched off to save energy. We also discovered by using real traffic matrices that backbone networks are very over-provisioned. Even greedy techniques can easily approach the maximum reduction in the amount of energy consumed. In fact, the bottleneck in terms of energy efficiency in energy aware traffic engineering solutions is the connectivity constraint. While this may not hold true in all networks, the analyzed ones confirmed the assumption.

In future work, we will evaluate the solution on a real NetFPGA testbed simulating a backbone network.

References

[1] A.P. Bianzino, L. Chiaraviglio, et al. “GRiDA: A green distributed algorithm for backbone networks”. In: *Online Conference on Green Communications (GreenCom), 2011 IEEE*. Sept. 2011, pp. 113–119.

[2] R. Bolla, R. Bruschi, et al. “Energy efficiency in optical networks”. In: *Telecommunications Network Strategy and Planning Symposium, 2012 XVth International*. Oct. 2012, pp. 1–6.

[3] E. Bonetto, L. Chiaraviglio, et al. “Algorithms for the multi-period power-aware logical topology design with reconfiguration costs”. In: *Optical Communications and Networking, IEEE/OSA Journal of* 5.5 (May 2013), pp. 394–410.

[4] Francesca Cuomo, Antonio Cianfrani, et al. “Network pruning for energy saving in the Internet”. In: *Computer Networks* 56.10 (2012). Green communication networks, pp. 2355–2367.

[5] C. Filsfils, S. Previdi, et al. *Segment Routing Architecture", draft-filsfils-spring-segment-routing-02 (work in progress)*. May 2014.

[6] Anders Gunnar, Mikael Johansson, et al. “Traffic Matrix Estimation on a Large IP Backbone: A Comparison on Real Data”. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. IMC ’04. Taormina, Sicily, Italy: ACM, 2004, pp. 149–160.

[7] Maruti Gupta and Suresh Singh. “Greening of the Internet”. In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM ’03. Karlsruhe, Germany: ACM, 2003, pp. 19–26.

[8] A. Hassidim, D. Raz, et al. “Network utilization: The flow view”. In: *INFOCOM, 2013 Proceedings IEEE*. Apr. 2013, pp. 1429–1437.

[9] I. Juva, R. Susitaival, et al. “Traffic characterization for traffic engineering purposes: analysis of Funet data”. In: *Next Generation Internet Networks, 2005*. Apr. 2005, pp. 404–411.

[10] M. Kamola and P. Arabas. “Shortest path green routing and the importance of traffic matrix knowledge”. In: *Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*. Sept. 2013, pp. 1–6.

[11] Victor Yu Liu and David Tipper. “Spare Capacity Allocation Using Shared Backup Path Protection for Dual Link Failures”. In: *Comput. Commun.* 36.6 (Mar. 2013), pp. 666–677.

[12] Aleksander Madry. “Faster Approximation Schemes for Fractional Multicommodity Flow Problems via Dynamic Graph Algorithms”. In: *Proceedings of the Forty-second ACM Symposium on Theory of Computing*. STOC ’10. Cambridge, Massachusetts, USA: ACM, 2010, pp. 121–130.

[13] Sergiu Nedevschi, Lucian Popa, et al. “Reducing Network Energy Consumption via Sleeping and Rate-adaptation”. In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. NSDI’08. San Franc., CA: USENIX Association, 2008, pp. 323–336.

[14] Stefano Salsano, Alessio Botta, et al. “Traffic Engineering with OSPF-TE and RSVP-TE: Flooding Reduction Techniques and Evaluation of Processing Cost”. In: *Comput. Commun.* 29.11 (July 2006), pp. 2034–2045.

[15] M. Siqueira, J. Oliveira, et al. “An optical SDN Controller for Transport Network virtualization and autonomic operation”. In: *Globecom Workshops (GC Wkshps), 2013 IEEE*. Dec. 2013, pp. 1198–1203.

[16] Ward Van Heddeghem, Filip Idzikowski, et al. “Power consumption modeling in optical multilayer networks”. English. In: *Photonic Network Communications* 24.2 (2012), pp. 86–102.

[17] Nedeljko Vasić and Dejan Kostić. “Energy-aware Traffic Engineering”. In: *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. e-Energy ’10. Passau, Germany: ACM, 2010, pp. 169–178.

[18] Mingui Zhang, Cheng Yi, et al. “GreenTE: Power-aware traffic engineering”. In: *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. Oct. 2010, pp. 21–30.