Improving the energy efficiency of software-defined backbone networks

Radu Carpa, Olivier Glück, Laurent Lefèvre, Jean-Christophe Mignot

Improving the Energy Efficiency of Software Defined Backbone Networks

Radu CARPA, Olivier GLUCK, Laurent LEFEVRE, Jean-Christophe MIGNOT
Inria Avalon - LIP Laboratory - cole Normale Suprieure of Lyon, University of Lyon, France
Email: radu.carpa@ens-lyon.fr, olivier.gluck@ens-lyon.fr,
laurent.lefevre@inria.fr, jean-christophe.mignot@ens-lyon.fr

***Abstract*—The continuous growth of traffic and the energy consumption of network equipments can limit the deployment of large-scale distributed infrastructure. This work[1] aims to improve the energy efficiency of backbone networks by dynamically adjusting the number of active links according to network load. We propose an intra-domain Software Defined Network (SDN) approach to select and turn off a subset of links. The SPRING protocol (a.k.a. Segment Routing) is used to make our algorithms converge faster. The algorithms — implemented and evaluated using the OMNET++ discrete event simulator — can achieve energy savings of around 44% when considering real backbone networks. energy efficiency** backbone networks SDN traffic engineering SPRING segment routing MPLS

I. INTRODUCTION

Networks are essential to today's highly instrumented and connected society. Currently responsible for a fraction of the energy the IT sector consumes, networks rely on equipments that often use a constant amount of power regardless of their utilisation; a worsening scenario as traffic is expected to increase by a factor of three in the next few years [1]. Backbone or core networks for instance, employ devices that consume several kilowatts of power even when idle.

Several techniques have been proposed for reducing the energy consumed by backbone networks, including traffic rerouting and using low power consumption modes. By rerouting data over alternative paths it is possible to offload a subset of links, or make them completely idle since operators generally over-provision their networks in order to handle peak-demands and provide clients with high Quality of Service (QoS). Evidence shows that even during peak hours many links are rarely used more than 50%. Figure 1 depicts the link utilisation computed from traffic matrices of the Géant network[2], where most links are less than 25% utilised even at peak hours.

This work tackles the challenge of reducing the energy consumed by backbone networks by changing the status of router ports and transponders on the extremities of links. The status of these components is set to sleep mode whenever a link is not used for transferring data, and they are brought back to operational state when required. This process of activating and deactivating network links is also termed as switching links on and off respectively.

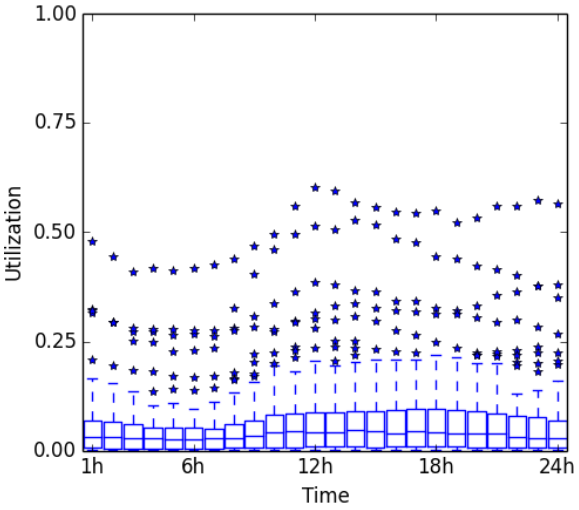


Fig. 1: Link loads in the Géant network.

Existing work seeks to adjust the number of active links according to the amount of traffic in the network, but they are mostly solutions at the conceptional level. They search to minimise the number of active links via solving Mixed-Integer Linear Programming (MILP) formulations or by employing heuristics without considering implementation details, and they often ignore race conditions. We perform SDN-based traffic engineering and take into account several aspects of designing energy-efficient traffic engineering in core networks. We implemented the proposed solution using the OMNET++ packet-based discrete event simulator. Unlike existing work that uses IP link weights or Multiprotocol Label Switching (MPLS) with Resource reSerVation Protocol Traffic Engineering (RSVP-TE) for rerouting traffic, we use a new protocol called Source Packet Routing in Networking (SPRING). In addition to making the implementation easier, SPRING is well suited for dynamic reconfiguration of a network.

Our solution is provided as a framework named SegmenT Routing based Energy Efficient Traffic Engineering (STREETE), which offers an online method for switching links off/on dynamically according to the network load [5]. This generic and extensible framework allows for incremental

[1]This work has been done as part of the CHIST-ERA STAR "SwiTching And tRansmission" European project (http://www.chistera.eu/projects/star).

[2]http://www.geant.net

enhancements, such as providing guarantees and more intelligent flow placement computations by simply updating the employed algorithms. Experiments considering real network topologies and dynamic traffic matrices allowed us to determine the trade-off between energy savings and the impact of our solution on network performance. This paper extends these initial results by providing a more extensive analysis of our framework considering dynamic aspects of a network and its evolution, and by relaxing some of the assumptions of our previous work. For instance, the number of interconnected data centres is expected to increase over time, we include experiments where we do not assume a diurnal traffic pattern and where the access nodes can be nearer the backbone.

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. Our solution is presented in Section IV and the result analysis in Section V. Section VI describes ongoing work and finally, Section VII concludes the paper.

II. Related Work

Gupta *et al.* [9] were among the first to consider setting network interfaces to sleep mode to save energy. In backbone networks, however, links cannot be switched off easily without incurring data loss. Bolla *et al.* [4] have shown that even micro-second interruptions are difficult to achieve due to the small packet inter-arrival time. To cope with this challenge, Nedevschi *et al.* [12] considered the transmission of data in bursts; nearly impossible to achieve under modern 100Gbps networks because very large buffers are needed to temporarily store packets.

A distributed network-wide solution has been provided by Vasić *et al.* [15] for changing flow paths and thus reduce link utilisation, allowing up to 21% of links of real networks to be put into sleep mode. Such a solution, however, is a concept whose implementation is not fully disclosed. Another distributed approach considers that every node locally monitors the utilisation of adjacent links and decides whether to switch them off [3]. Machine learning mechanisms are used to avoid switching off links that have previously caused congestion. Such a solution relies heavily on the Interior Gateway Protocol Traffic Engineering (IGP-TE) extensions, and after a detailed analysis, we observed that its communication overhead is highly underestimated. In a 30-node network authors estimate that Open Shortest Path First Traffic Engineering (OSPF-TE), the protocol used by their solution, floods the network six times per second. The proposed algorithm adds a couple more flooding per minute. However an analysis of OSPF-TE [13] revealed that flooding will happen only once every ten seconds even under stress traffic patterns with lots of variations. After correcting the assumptions, the obtained overhead of 0.52% will increase resulting in a 30% improvement of OSPF-TE flooding, which is non-negligible as flooding is costly both in number of required messages and processing demands.

Another proposed centralised solution using MPLS and RSVP-TE [16] relies on estimation of traffic matrices, which later was proven inefficient [8]. Authors also provide a hybrid solution where MPLS and RSVP-TE are used in parallel with shortest path routing. Our work uses the SPRING protocol with the same advantages, but with less complexity.

III. Assumptions and Background

We consider that the workload in a backbone network has slow variation and follows a diurnal pattern due to a high level of flow aggregation, as presented by Juva *et al.* [10]. Table I provides figures on the power consumed by backbone-network devices as described in previous work [14]. Our model considers that switching off a port corresponds to powering off part of its integrated circuits and the transponder. Switching off link hence means powering off its two extremities. For instance, if a 100Gbps link is switched off, $2 \times (135 + 150) = 570$ Watts are saved.

TABLE I: Power consumption of router ports.

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

* mathematical projection

In a core network with links of hundreds of Gbps, a short failure may incur the loss of hundreds of gigabytes of data. Although we do not address protection issues in this paper, our solution never disconnects the network and guarantees the availability of a working path between any pair of nodes. The term *Connectivity Constraint* is hereafter used to refer to this guarantee.

Our solution works at the electronic level and does not change wavelength paths in the optical domain. Only extremities of optical links are turned off/on; that is, the transponders and the port cards of the IP router.

A. Software Defined Network

In traditional networks nodes contain both control and data planes, and enforcing network-wide policies often requires access and configuration of all devices. SDN aims to simplify management by separating control and data planes. The control plane retains the network intelligence for calculating paths for data flows and for programming them into the data plane. In a network device, packets that arrive through an incoming port are forwarded by the data plane to the right outgoing port. Henceforth data-plane devices are referred to as SDN switches.

SDN controllers enable a lot of flexibility when it comes to traffic engineering. They maintain and exploit network-wide knowledge in order to execute all required computations and use a predefined API for applying changes to SDN switches. In this work, we use an SDN solution. The SPRING protocol is used as data plane by the SDN controller.

B. SPRING protocol

SPRING [7], also known as Segment Routing, is currently an IETF draft started in 2013 that aims to replace MPLS+RSVP-TE for traffic engineering. It combines the power of source routing, allowing for flexible traffic engineering, with shortest path routing, which requires less signalling or header overhead.

The data plane used by SPRING utilises the same concept of label switching of MPLS, but its control plane has been completely redesigned. The distribution of labels is done via an extension to the IGP instead of using special protocols such as LDP/RSVP-TE. Moreover, unlike MPLS, labels, called Segment IDentifiers (SID) in SPRING, have a global scope. In the present work, we are interested in two types of SIDs, namely "nodal" and "adjacency" as shown in Figure 2a.

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

After network discovery, sending a packet to node a through the shortest path requires encapsulating it into a packet with destination a. 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 $[b, L1]$ (Figure 2b).
- If h wants to send a packet to a via f (Figure 2c), it uses the header $[f, a]$.

Being a source routing protocol, SPRING enables fast flow setup and easy reconfiguration of virtual circuits with minimum overhead since changes must be applied only to the ingress devices. No time and signalling are lost re-configuring the midpoint devices.The policy state is in the packet header and is completely virtualized away from midpoints along the path. This means that a new flow may be created in the network by contacting only one network device: the ingress router. This comes in contrast with OpenFlow where the forwarding tables of all the devices along the path must be reconfigured. The agility in updating the paths of flows is essential in solutions that intend to perform link switch-off and improve energy efficiency.

Moreover, the label switching proved itself very efficient in backbone networks. For example, MPLS is the only forwarding protocol supported at full link speeds by Juniper PTX Series core routers.

IV. The STREETE Framework

This section provides a formal description of the energy-efficient traffic engineering problem. It divides the overall problem into sub-problems for easier discussion and then presents an analysis of centralised and distributed solutions. Finally, we present details of our solution.

A. Steps of STREETE Algorithm

The problem of energy-efficient traffic engineering can be divided into the following three steps:

- *Selecting links to switch off/on:* this step selects candidate links to be switched off/on and is referred to as *SelectLinksToOff/On*.
- *Computing new routes to avoid/reuse links:* prior to switching off/on the links selected by *SelectLinksToOff/On*, the algorithm must compute new paths for the flows traversing the network. Switching off a subset of links reduces network capacity, and thus flows must be intelligently rerouted to avoid congestion. In fact, congestion may occur even when a turned-on link provides a better path for a large number of flows. We refer to this step as *ComputeNewRoutes*.
- *Rerouting and switching off/on the links (RerouteAndSwitchOff/On):* once new routes for flows are computed, the decision must be enforced on the network devices. This step triggers rerouting and the actual link switch-off.

Algorithm 1 illustrates how the steps above fit together. The algorithm searches for links that can be switched on or off, and after that, executes *RerouteAndSwitchOn* or *RerouteAndSwitchOff*.

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

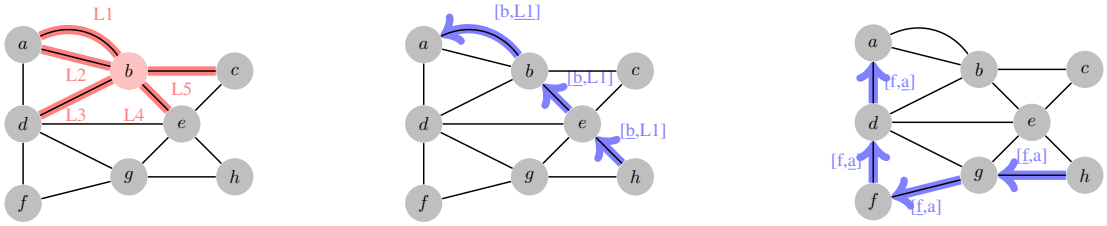
Algorithm 1: Main loop.

B. Centralised versus Distributed Approaches

There is often a divergence of opinions in the literature with respect to centralised and distributed approaches. Before implementing our solution, we compared the two approaches. We present here the conclusions after analysing each step of STREETE under centralised and distributed settings.

a) SelectLinksToOff/On::

Distributed approaches do not allow for optimising this step. There are essentially two ways to choose the links for switch-off; either each node independently decides to switch off a directly attached link, or a coordination between nodes is used to elect a link. Under the former, special measures must be taken by *RerouteAndSwitchOff* to ensure that the *Connectivity Constraint* is respected. As a result, coordination between nodes is necessary in both cases. Distributed N to N coordination means flooding the network with control messages, known to be costly in the IGP protocols. In contrast, a centralised solution can take complex decisions after collecting and analysing the traffic matrices, requiring only an N to 1 communication.



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

Fig. 2: SPRING protocol.

b) ComputeNewRoutes::

In a distributed solution every node must calculate its own routes. From a computational complexity perspective, this is better than having a unique entity calculating all routes for the entire network domain. However, missing information about global network state can limit the optimisation of route selections. Although it is possible to distribute this information across all nodes, the information can take minutes to converge [13], which can in turn result in long periods of network instability. Distributed solutions are limited to reactively rerouting the flows to avoid disabled links. A resource reservation protocol may be used to reserve new paths and avoid network congestion. However, the use of such protocol increases the time needed to setup the flows and reduces the network convergence speed.

Better energy efficiency and congestion avoidance are possible under centralised solutions. Centralised solutions can compute new routes by collecting traffic matrices and using them to solve either an approximation of the "Multicommodity Flow Problem" in order to optimise the distribution of load, or a K-shortest-path problem to optimise delay.

c) RerouteAndSwitchOff/On::

When considering a distributed scenario, nodes need to synchronise their actions in order to avoid switching off a set of links that can disconnect the network. This risk is minimised when a central entity takes all decisions on links to switch off; it does not risk to switch off simultaneously two badly chosen links and violate the *Connectivity Constraint*.

Networks are increasingly relying on centralised architectures, with SDN becoming commonplace in many data centres and its use being largely studied in optical networks [11]. Therefore, we focus on an SDN solution. Although we agree that a distributed solution has the advantage of avoiding a single point of failure, several techniques exist to make SDN more robust [6]. We are interested in using SDN for energy efficient traffic engineering, and the controller can be made optional for other operational activities. If the controller becomes unreachable, our solution can turn on the links and rollback to a fully functional distributed network without energy efficiency features.

C. Implementation Details of Each Step

To employ the proposed algorithm, an SDN controller must know the network status. During packet classification ingress routers compute the intensity of inter-domain flows. This data is used by the SDN controller to compute a network-wide traffic matrix.

We implement the "least congested link" technique for *SelectLinksToOff* as it is the approach against which most of existing work compares their results. First, links are sorted by instant transmission speed; a step whose complexity is $O(E \cdot log(E))$, where E is the number of links (edges). Then the algorithm tries to switch off links that transmit less data by testing, for each link, whether switching it off does not violate the connectivity constraint. It will take at most $O(E \cdot (E+V))$: E times breadth-first search to test the integrity of the network, where V is the number of nodes (vertices).

For *SelectLinksToOn*, our algorithm is simple. It tests the utilisation of the most utilised link and if it is above 75%, all links are switched on and the shortest path routes are recalculated.

ComputeNewRoutes uses a technique that computes the all-pairs shortest path routes and tests the possibility to route all the available traffic over these paths. The computational complexity is $O(V \cdot (E + V \cdot logV))$ if Dijkstra from every node is computed. The complexity may decrease if using an optimised all-pairs shortest path algorithm.

Network links are switched off one by one while no link reaches the threshold utilisation of 60%. If the controller detects that turning off a selected link can increase the utilisation of any other link above 60%, it skips the selected link. When the utilisation of any link exceeds 75%, the network is switched back on.

The steps executed by *RerouteAndSwitchOff* to actually switch off the links are illustrated in Algorithm 2, where the links between nodes A and B, C and D, etc, were selected for switch-off.

V. Performance Evaluation

In this section, we analyse experimental results from simulations using OMNeT++. First, the simulation parameters are described. Second, a detailed analysis considering a small network is presented. Third, we discuss results on a simulation taking into account two backbone networks with real traffic matrices. Finally, we stress our solution on the Germany 50 network under more aggressive traffic variations.

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 Every router applies the changes to the forwarding table to reroute traffic over the new paths;
4 The controller tells nodes adjacent to links (A,B,...) to turn them off;
5 Nodes A,B,C,... ACK to the controller ;
6 The SDN controller sends IGP flooding: link A-B is down `/* flooding is needed, because at line 2 we may not contact every node */`
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 2: *RerouteAndSwitchOff*

A. Experimental Scenario

The experimental results reported in this work consider two real networks, namely Géant and Germany 50, depicted in Figure 3 and summarised in Table II. The centralised SDN solution was implemented in the OMNet++ discrete-event simulator. The implementation includes a draft version of the SPRING protocol used to dynamically reroute data in the network avoiding switched-off links. We prefer SPRING over the commonly used MPLS with RSVP-TE solution because the latter faces convergence issues under a large number of LSPs [2]. With small numbers of virtual tunnels and the "refresh reduction" extension for RSVP-TE, the risk is small, but in our case we need a full mesh of virtual tunnels in order to reroute all flows when switching off links. Moreover, by switching off links we increase the load of a subset of routers and the number of LSPs they manage. SPRING is designed to be stable under a larger number of dynamically changing LSPs.

TABLE II: Evaluated network topologies.

Network	Number of nodes	Number of links
Géant (Figure 3b)	22*	36
Germany 50 (Figure 3a)	50	88

* the New York node is not shown in Figure 3b, but the two links connecting it to the network are visible.

We used the OMNet++ *UDPBasicApp* module to generate continuous traffic flows of a desired intensity. One flow is generated for each source-destination pair, hence resulting in V^2 flows, where V is the number of nodes. For real backbone networks, the intensity of flows corresponds to the real values found in the traffic matrices obtained from SNDLib[3], scaled up to represent today's faster network speeds. SNDLib contains snapshots of traffic matrices computed at fixed intervals of five minutes. As a result, the matrices lack detailed information about traffic variations within each five-minute period. As we are interested in traffic variations, we compressed every five-minute interval into two seconds of simulation. Hence, we are able to reduce the time needed to run the simulations and simulate high network speeds.

In the other two cases, namely the simulation of a small network and the stress test on the Germany 50 network, we inject V^2 flows of identical intensities into the network. We exponentially vary the intensity during the simulation in order to test the solution both on slow and fast traffic variations. We reduced the speed of each link to 100Mbps to reduce the time needed to run a simulation.

In all cases, the propagation delay is time needed for light to travel through the fibre. The distance between nodes is the bird-fly distance as given by Google Maps API. Queuing delay is simulated by OMNeT++.

The SDN controller implemented in OMNet++ uses a custom protocol to communicate with network nodes. All the communication between the controller and the data-plane devices passes through the network. A TCP connection is created between every node and the controller and is influenced by all the turn-on/off operations applied to the network.

B. Validation on a Small Network

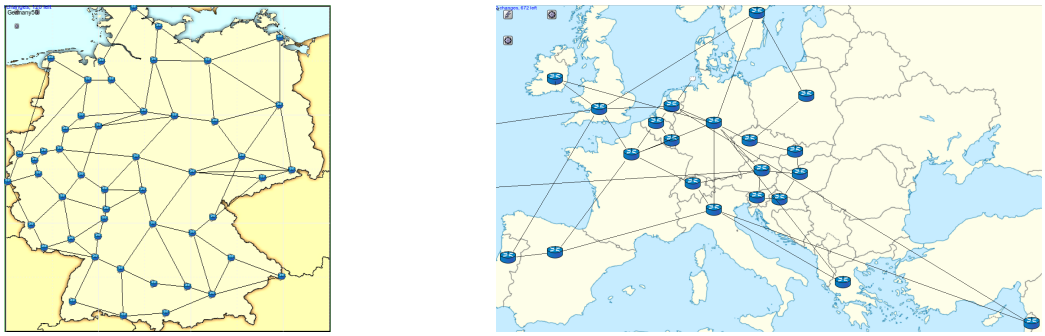
Figures 4 and 5 provide results on a small network of nine nodes and fifteen links. We observe that the network is smoothly turning the links off until $t = 5$, at which moment the *SelectLinksToOff* algorithm detects that it can no longer turn off links without causing congestion. The network is stable until $t = 11$, when given the decrease in the amount of data transmitted through the network, *SelectLinksToOff* finds more links to be switched off. At $t = 13$, the network reaches the state of a spanning tree and no more links can be switched off without violating the *Connectivity Constraint*. The network remains unchanged until more data is injected and it reaches a state close to congestion at $t = 24$. The *SelectLinksToOn* algorithm switches on links in order to offer more capacity in the network.

Performed in parallel with the process above, the *ComputeNewRoutes* algorithm tries to avoid congestion by rerouting traffic over alternate paths as presented in Section IV-C. Figure 5 shows the state of the router queue for the most congested link. Rerouting the flows does not saturate the network in the periods $t \in [1 \dots 5]$ and $t \in [11 \dots 13]$. The reason for saturation at $t = 24$ is the increase of intensity in the injected flows.

C. Results from Backbone Networks

A day of operation of the two considered backbone networks (*i.e.* Géant and Germany 50) was simulated. In Figures 6 and 7, green lines represent the percentage of links that are left switched on by our algorithm. Using the values of Table I for 100Gbps links, this translates to the energy savings shown in Table III.

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



(a) Germany 50 network

(b) Géant network

Fig. 3: Considered network topologies.

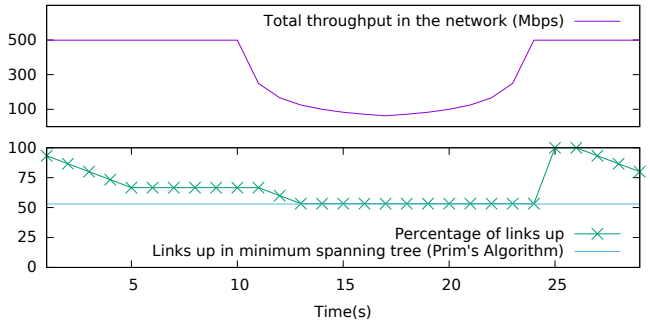


Fig. 4: Dynamic traffic scenario in a small network.

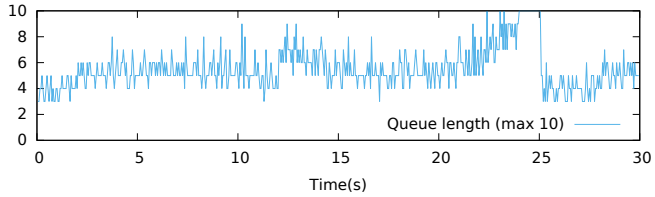


Fig. 5: Length of the router queue for the link that triggered the turn-on.

1) Energy Consumption: At the beginning of the day, when our algorithm starts to run, both networks switch off links one by one and converge quickly to a spanning tree. Further switch off is not possible without violating the *Connectivity Constraint*. In the Germany 50 network, we observe a turn-on at 23:00h, which is due to a large increase in traffic between two network nodes. We do not know the reason for this unusual increase in the original traffic matrix, but this demand was handled well by our algorithm, which switched on sleeping links and restored the full capacity of the network.

Rerouting data over paths that are not the shortest incurs in load overhead. The overhead of our solution can be viewed in Figures 6a and 6b, where the maximum overhead is 18.56%. Even with this overhead, the networks converge to a minimum energy state.

Moreover, we observe that even greedy techniques for *SelectLinksToOff* provide good energy savings due to high over-provisioning of the analysed networks.

TABLE III: Energy savings during the simulated day.

Network	**Average off time of links**	**Energy economy (kWh)**
Géant	14.88 out of 36	204
Germany 50	38.18 out of 88	522

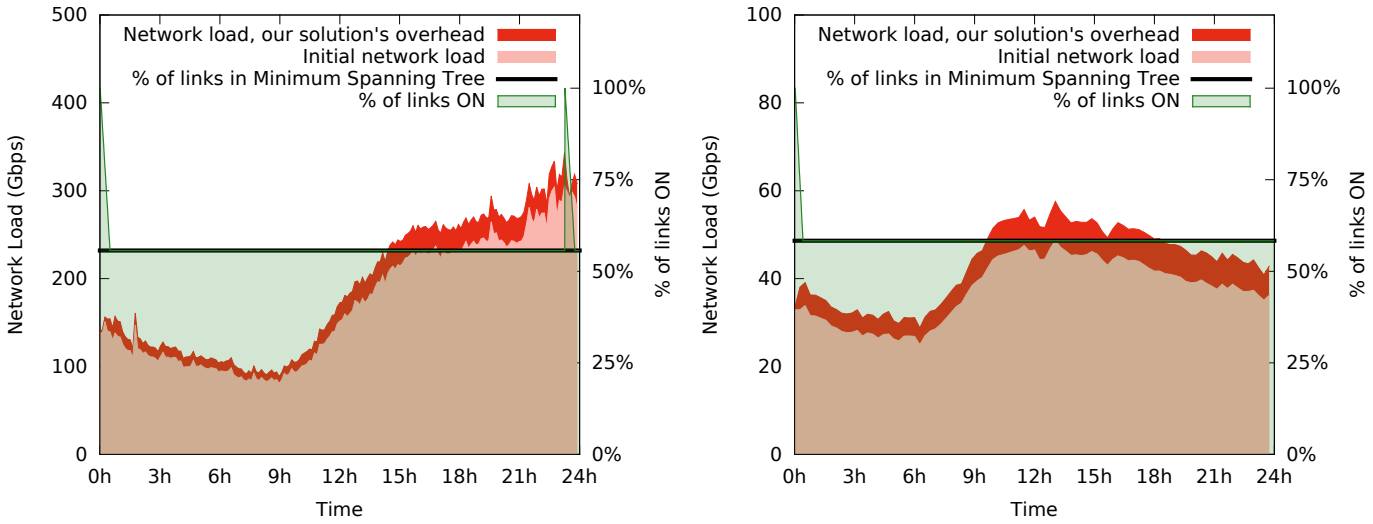
2) Impact on End-to-End Delays: Figures 7 shows the impact of our solution on end-to-end delay. For every source-destination pair, we calculated the mean end-to-end delay at every hour. The box plots represent the distribution of mean of all pairs during the previous hour. The blue ones describe the delays in the original network and the red ones are the delays in the network with switched off links. We also present the mean delay for cases with switch off and without switch off as continuous lines of the same colours.

In the Germany 50 network, the delay increases by 37% in the worst case, but usually by around 20%. A more interesting result can be seen in the Géant network. While in most cases the delay increases, it can be 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 this routing policy is not well suited. In some cases traffic between two European nodes passes through New York. Turning off the 2 links forced flows to pass through multiple hops, but they remain in Europe.

3) Impact on Packet Loss: The simulation revealed that our algorithms did not cause packet loss. In backbone networks, the aggregated traffic flows do not change fast enough to produce a congestion that can affect our solution since STREETE switches on all the links and serves the increased demand at the smallest sign of congestion.

D. Faster Traffic Variations and Improved Switch-On

In the previous section we observed that the original traffic varies very slowly and that STREETE has more than enough time to react to variations in network load. Moreover, the networks are too over-provisioned and a spanning tree is



(a) Germany 50 network

(b) Geant network

Fig. 6: Impact on link loads.

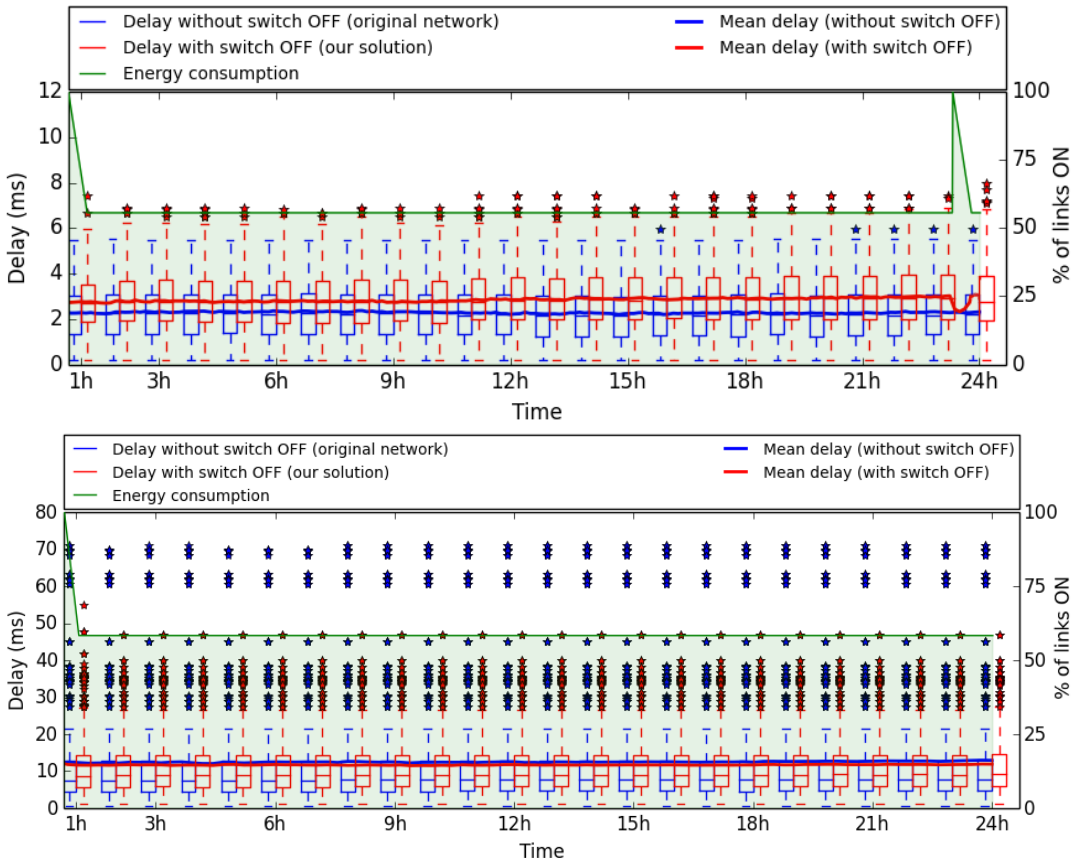


Fig. 7: Impact on end-to-end delay, Germany 50 (top) and Géant (bottom).

capable to route all traffic even during peak hours. STREETE hence converges to a stable spanning tree and is very unlikely to ever switch on links in these networks.

In order to stress STREETE, we generate a more aggressive traffic pattern in the Germany 50 network where we exponentially increase the amount of traffic over a forty-second interval. At $t = 40$, one of the links reaches utilisation close to 1, and then load decreases at the same pace. Figure 8 shows

the network load when subjected to the generated traffic.

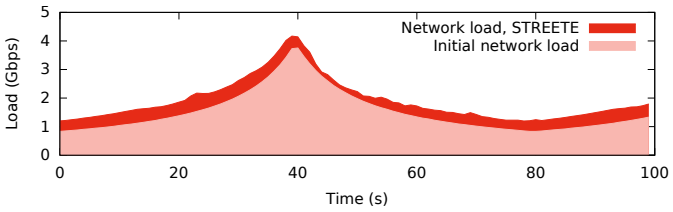


Fig. 8: Generated load and overhead of STREETE.

Preliminary tests revealed that an "all-on" strategy for the *SelectLinksToOn* step does not perform well in terms of energy consumption when subjected to big traffic variations. Among the links which are turned on, some are located far from the region of congestion. As a result, they have a high probability to be switched off again just after being turned on. A link-flap is produced, where links are continuously switched on and off. Therefore, we improved the *SelectLinksToOn* step of our algorithm by implementing a technique which selects, among the switched-off links, the link that would transport the most data in the all-on network.

Figure 9 summarises the execution with the improved *SelectLinksToOn* and the modified traffic pattern. It shows the utilisation map of each unidirectional link in the Germany 50 network, where the blue area has utilisation close to 0 and red points present utilisation close to 1. The white line in the right-hand side figure represents the limit between the links which are on an those that are off. As all links have the same speed, the percentage of energy savings follows the percentage of the switched off links, namely 39.7% on average.

Our solution progressively switches on links as load increases between $t = 0$ and $t = 40$. STREETE reacts with a small delay of approximately 2 seconds; the time needed for traffic matrices to be collected from all nodes and to detect the changes in network load. The algorithm switches off the links again when load decreases. Some links are left off even at the peak load. It is because they are located at the border of the network and transport a small number of packets. Turning them on does not have any impact on the load of the congested links (the links with utilisation greater than 75%).

Figure 10 shows the impact of STREETE on packet loss. The traffic injected into the network doubled between $t = 30$ and $t = 40$. There is a noticeable increase in the number of lost packets, especially at $t = 40$, when the initial network was already saturated. This increase is due to the two-second delay between the increase of network load and the moment STREETE reacts by switching on the links. With STREETE, the packet loss on the most congested link is 7.6% at $t = 40$.

VI. Outgoing Work

As described earlier, STREETE uses a very conservative approach to deal with congestion, which consists in switching all links back on at the smallest sign of congestion. Although this may look efficient from a QoS point of view — as it minimises the risks of overloading the network, increasing delays and losing packets — it may not be the best solution from an energy efficiency perspective. We are therefore working on algorithms for choosing links to switch on; algorithms that consider several aspects such as load decrease resulting from switching on a link, improvement of end-to-end delay, among other factors. Although the choice of links to switch on under congestion may look simple at first, it is challenging as it may require maintaining information about the network status during previous steps of link switch-off and, when possible, determining a previous stable state to which to roll back.

Moreover, we are working on implementing an SDN-based prototype using the Open Network Operating System (ONOS)[4] SDN controller for flow routing and switch on/off decisions. As data-plane devices we use both software switches in Mininet[5], and FPGA-based custom switches on a NetFPGA 10G infrastructure; whose cards are equipped with four 10Gbps Ethernet interfaces, a Xilinx Virtex 5 FPGA and additional hardware provided for prototyping and testing network protocols[6]. A subset of the SPRING protocol is being implemented.

VII. Conclusions

In this work, we designed an energy-aware traffic engineering technique for reducing energy consumption in backbone networks. Energy efficient traffic engineering was analysed 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.

Networks are progressively using centralised architecture and SDN is increasingly utilised in data centre networks. We believe that SDN may be extended to backbone networks. The implemented solution shows that SDN may also be a good means for reducing the energy consumption of network devices.

Compared to previous work, in this paper we used the SPRING 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 can be easily applied to SDN solutions.

Using simulations, we showed that as much as 44% of links can be switched off to save energy in real backbone networks. 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 is the connectivity constraint.

We performed a stress test of our solution under rapidly increasing traffic patterns and showed that more work must be

[4]http://onosproject.org/software/
[5]http://mininet.org/
[6]http://netfpga.org

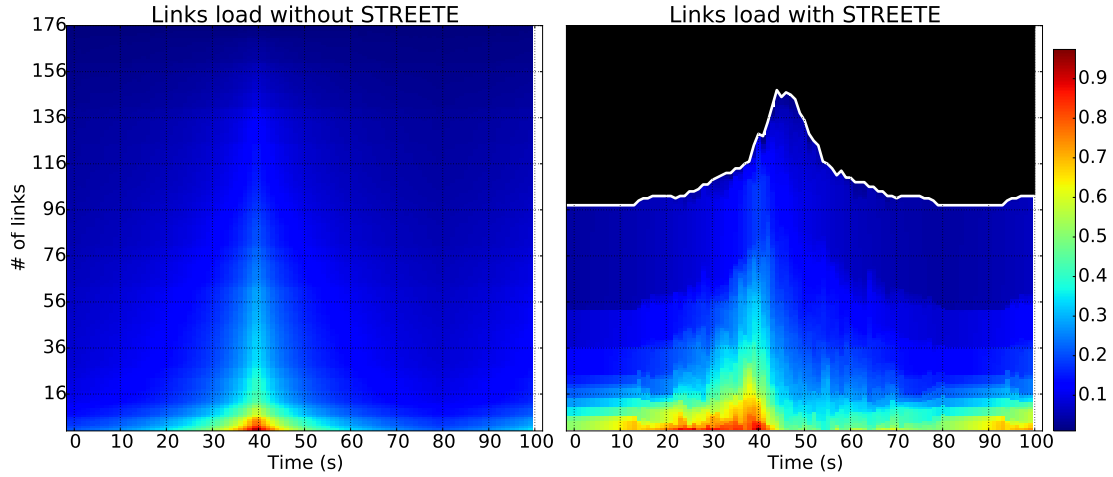


Fig. 9: Links utilisation without and with STREETE.

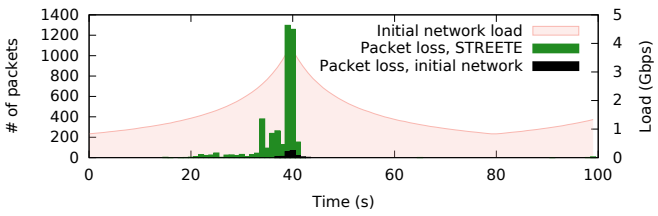


Fig. 10: Packet loss.

done in the domain of switching links back on; a field which has received little attention from the research community.

REFERENCES

[1] Cisco vni global mobile data traffic forecast, http://ciscovni.com/forecast-widget/index.html

[2] The ietf mailing list, motivation for benchmarking mpls-te convergence, http://www.ietf.org/mail-archive/web/bmwg/current/msg01502.html

[3] Bianzino, A., et al.: Grida: A green distributed algorithm for backbone networks. In: Online Conference on Green Communications (GreenCom), 2011 IEEE. pp. 113–119 (Sept 2011)

[4] Bolla, R., et al.: Energy efficiency in optical networks. In: Telecommunications Network Strategy and Planning Symposium, 2012 XVth International. pp. 1–6 (Oct 2012)

[5] Carpa, R., Glück, O., Lefevre, L.: Segment routing based traffic engineering for energy efficient backbone networks. In: ANTS2014. New Delhi, India (Dec 2014)

[6] Dixit, A., et al.: Towards an elastic distributed sdn controller. SIGCOMM Comput. Commun. Rev. 43(4), 7–12 (Aug 2013)

[7] Filsfils, C., et al.: Segment routing architecture", draft-filsfils-spring-segment-routing-02 (work in progress) (May 2014)

[8] Gunnar, A., 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. pp. 149–160. IMC '04 (2004)

[9] Gupta, M., et al.: Greening of the internet. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. pp. 19–26. SIGCOMM '03 (2003)

[10] Juva, I., et al.: Traffic characterization for traffic engineering purposes: analysis of funet data. In: Next Generation Internet Networks, 2005. pp. 404–411 (April 2005)

[11] Liu, L., Tsuritani, T., Morita, I., Guo, H., Wu, J.: Experimental validation and performance evaluation of openflow-based wavelength path control in transparent optical networks. Optics Express 19(27), 26578–26593 (2011)

[12] Nedevschi, S., et al.: Reducing network energy consumption via sleeping and rate-adaptation. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. pp. 323–336. NSDI'08, USENIX Association, Berkeley, CA, USA (2008)

[13] Salsano, S., et al.: Traffic engineering with ospf-te and rsvp-te: Flooding reduction techniques and evaluation of processing cost. Comput. Commun. 29(11), 2034–2045 (Jul 2006)

[14] Van Heddeghem, W., et al.: Power consumption modeling in optical multilayer networks. Photonic Network Communications 24(2), 86–102 (2012)

[15] Vasić, N., et al.: Energy-aware traffic engineering. In: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking. pp. 169–178. e-Energy '10, ACM, New York, NY, USA (2010)

[16] Zhang, M., et al.: Greente: Power-aware traffic engineering. In: Network Protocols (ICNP), 2010 18th IEEE International Conference. pp. 21–30