An Online Power-Aware Routing in SDN with Congestion-Avoidance Traffic Reallocation

Adriana Fernández-Fernández*, Cristina Cervelló-Pastor*, Leonardo Ochoa-Aday*, Paola Grosso†
*Department of Network Engineering, Universitat Politècnica de Catalunya,
Esteve Terradas, 7, 08860, Castelldefels, Spain
Email: adriana.fernandez@entel.upc.edu, cristina@entel.upc.edu, leonardo.ochoa@entel.upc.edu
†System and Network Engineering group (SNE), University of Amsterdam,
Amsterdam, The Netherlands
Email: p.grosso@uva.nl

***Abstract*—Software-Defined Networks (SDN) can be seen as a promising alternative to achieve the long-awaited power efficiency in current communications systems. In these programmable networks a power-aware mechanism could be easily implemented leveraging the capabilities provided by control and data plane separation. For such purpose, this paper proposes a novel solution minimizing the number of active elements required in an SDN with multiple controllers and in-band control traffic. In order to provide a complete and fine-grained strategy, this proposal comprises two crucial modules: GrIS, a green initial setup and DyPAR, a dynamic power-aware routing. Besides being compatible with SDN environments without a dedicated control network, the proposed strategy is able to handle demanding traffic arrival without degrading the performance of higher priority traffic. Simulation results show that our heuristic approach allows to obtain close-to-optimal power savings with differences under 8%. Moreover, comparison with existing related methods using real topologies validates the improvements achieved by our solution in terms of power efficiency and performance degradation avoidance. For instance, after routing all the incoming traffic, a reduction of power consumption of up to 26.5% and an increase of allocated demands of up to 26.7% can be reached by our solution.**

I. Introduction

Energy consumption concern in communication systems has currently attracted a great deal of attention from research community due to the exponential demand growth and the ever-increasing number of connected devices [1]. According to [2] by 2025 the global Internet will be responsible for more than 10% of the world's electricity consumption. Given that in practice power consumption of network equipment is not in proportion with their traffic load, putting unused network elements into sleep mode (i.e. a low-power state) is an effective and widely accepted strategy to decrease the consumption of data networks [3].

In this context, Software-Defined Networking (SDN) is a very well-suited architecture to perform power-aware routing and manage the state of unused switch interfaces in a coordinated and centralized way. The basic idea of SDN [4] -control and data planes separation- makes network environments more manageable. The logically centralized control plane in SDN provides a global knowledge of the network state information. Moreover, it is responsible for managing network tasks and perform device programming. Meanwhile, interconnection devices only follow the rules set by the controller to forward the traffic. Therefore, the implementation of a power-aware solution in the control plane is a valuable opportunity to solve the power consumption problem in data networks, making it easier than with classical hardware-dependent standards.

Despite consistent efforts to improve the network power efficiency, power-aware techniques may lead to performance degradations. Inspired by this reality, this paper introduces a new power-aware strategy combining a control plane configuration with a dynamic routing for an SDN architecture. This solution dynamically reduces the number of active nodes and links required to manage changing traffic patterns. Instead of restricting the potential of power-aware solutions to low-loaded environments, this work proposes a more fine-grained strategy minimizing the power consumption while avoiding the performance degradation of higher priority traffic.

Throughout this work we consider an SDN architecture with multiple controllers and in-band control traffic [5]. In this operational mode, links are shared between data and control plane traffic. Hence, the proposed power-aware routing can be applied also in cases when implementing a dedicated control network is not feasible either for physical or cost-related restrictions. In backbone networks this is a more realistic scenario since additional links dedicated to directly connect controllers and forwarding devices, are impractical and cost-inefficient. Specifically, the major contributions of this work are as follows:

- An Integer Linear Problem (ILP) is formulated to optimize the number of active nodes and links in SDN with multiple controllers and in-band control traffic.
- A novel power-aware mechanism is proposed to allocate traffic demands in real time, reducing power consumption and performance degradation of higher priority traffic.
- Real topologies, as well as existing related proposals, are used to validate achieved improvements.

This work was supported by the Ministerio de Economía y Competitividad of the Spanish Government under project TEC2016-76795-C6-1-R and AEI/FEDER, UE and through a predoctoral FPI scholarship.
ISBN 978-3-903176-08-9

II. Related Works

Throughout recent years the power consumption of communication networks has been extensively treated and several solutions focused on reducing the number of active elements have been proposed. For instance, Bianzino et al. [6] aim to find the network configuration that minimizes the network energy consumption, modeled as the sum of the energy spent by all nodes and links carrying traffic. To achieve this, they formulated an optimization problem for finding minimum-power network subsets assuming the existence of traffic level with known daily behavior. Therefore, an accurate prediction of incoming traffic is required.

An energy-aware routing and traffic management solution is proposed in [7] to reduce the energy consumption, determined as the number of active Open-Flow switches in the network. For this, a low complexity algorithm is presented using, for each pair of endpoints, a pre-computed set of shortest paths to select the route that minimizes the number of switches that become active after allocating the flow. Although this proposal allows real-time operation routing flows sequentially, only low-loaded nighttime traffic is considered, failing to extensively examine the implications of more demanding scenarios.

The authors of [8] presented the design of an Energy Monitoring and Management Application (EMMA) to minimize energy consumption in SDN-based backhaul networks. They formulated this problem as a non-linear optimization model and proposed heuristic algorithms for the dynamic routing of flows and the management of the resulting link and switch activity. However, such algorithms were implemented in an SDN emulation environment with out-of-band control traffic, limiting their applicability to networks where dedicated links between the controller and forwarding devices are deployed.

In [9] authors proposed ElasticTree, a network-wide power manager to save energy in data centers using SDN. This solution dynamically finds the minimum set of network elements required by changing traffic loads, while satisfying performance and fault tolerance constraints. In this regard, three strategies were studied, namely Formal Model, Greedy Bin-Packing and Topology-aware Heuristic. While the first option presents scalability issues and the second saves less power, the best performance is obtained by the Topology-aware Heuristic. However, this approach is specifically conceived for FatTree networks.

Other approaches about power efficiency in software defined data center networks are presented in [10], [11]. The authors of [10] simultaneously optimize the power saving and the network performance, according to a pre-defined combination of quality requirements. In [11] different energy-aware routing strategies, combining common routing and scheduling algorithms, are evaluated and implemented as a OpenNaaS-based prototype. However, these strategies are only applicable in data centers and are also incompatible with environments without dedicated control networks.

Different from the aforementioned works, the aim of this paper is to provide a power-aware control plane configuration combined with a dynamic routing strategy considering an SDN architecture with multiple controllers and in-band control traffic. Our approach is also able to handle more demanding traffic patterns while reducing the performance degradation of higher priority traffic.

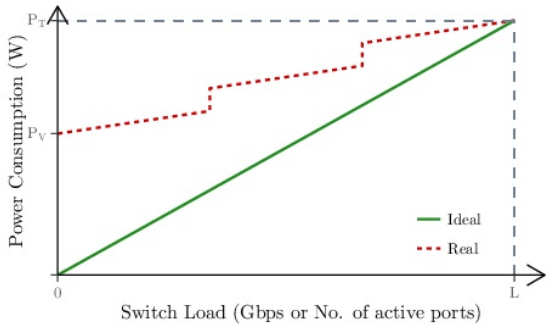


Fig. 1. Power Model (redrawn from [12]).

III. Power Model

Power consumption of networking devices is composed by a static component (due to power consumed by chassis, fans, line-cards, etc.) and a dynamic one, related to the rate of traffic flowing through their port interfaces. Ideally, the static part, also known as the idle component, which represents the power required by an unused switch, should be null. Then, in presence of an increasing traffic load, the power consumption should behave proportionally and linearly grow along with the traffic increase (line marked as Ideal in Fig. 1). However, this model differs considerably from the real one (line marked as Real in Fig. 1). In practice, whenever a device is active it will consume a fixed amount of power (P_n), irrespective of load conditions. Additionally, this baseline power is increased by the number of active ports and the utilization of each port.

In this regard, it has been previously measured that the amount of traffic handled by port interfaces does not have a significant effect on device's consumption [12]. Explicitly, while most of the power is consumed only by turning the device on, increasing the port utilization from zero to full load represents less than 8% of total power consumption [9]. Therefore, in this paper we consider that the power consumed by a network node depends on the baseline power and the number of active ports, both of which represent fixed contribution.

IV. Problem Statement

To formalize the power consumption optimization problem in SDN, in this section we present its mathematical formulation. The proposed model seeks to optimize the overall power consumption. To that end, the incoming traffic demands and the associated required control traffic will be routed minimizing the number of active network elements. In general, our model leverages preliminary works presented in [13]–[15] supporting that forwarding devices are put into sleep mode.

Being a general formulation, multiple controllers as well as SDN with in-band mode are supported by this proposal.

Given the controllers placement, our model also determines the optimal distribution of switches between controllers in terms of power efficiency and load balancing.

A. Network Model

In the proposed scheme the network topology can be modeled as a graph $G = (V, E, C)$, where V, E and C denote the set of switches, links and controllers respectively. Note that network nodes can only fulfill one role, i.e. controller or routing device. Additionally, we use $c_{i,j}$ to denote the capacity of a link $(i, j) \in E$. Considering F as the entire set of traffic flowing through the network between any pair of nodes, let D denote the subset corresponding to data plane communications. For the control plane, we use T to denote the subset of communications between controllers and switches, and H to denote the subset of communications between controllers. Each flow $f \in F$ from source s_f to destination t_f, has associated a throughput, denoted by d_f.

B. Formulation

Considering the entire set of demands fixed and known in advance, all the optimal control and data paths in terms of power efficiency can be computed jointly in a global optimization process. To formulate such optimization problem, the required variables, objective functions and constraints are defined as follows:

TABLE I
NOTATION OF BINARY VARIABLES

Name	Description
$x_{i,j}$	Indicates whether link i, j is active
y_v	Indicates whether node v is active
$t^f_{i,j}$	Indicates whether link i, j is selected to route flow f
$\lambda_{v,c}$	Indicates whether node v is associated with controller c

The objective function of our model seeks to reduce the overall power consumption considering the number of active nodes and links in the network. Consequently, both elements are integrated in the following expression, where P_p and P_n denote the power consumption of a port and a node, respectively.

$$\text{minimize} \quad 2P_p \sum_{(i,j)\in E} x_{i,j} + P_n \sum_{v\in V} y_v \tag{1}$$

A single controller must be selected to manage each active forwarding device in the network.

$$\sum_{c\in C} \lambda_{v,c} = y_v \qquad \forall v \in V \tag{2}$$

Looking to avoid congested controllers, we set the maximum number of forwarding devices that can be associated with each controller. In this way, active switches are evenly distributed and the load is balanced among controllers.

$$\sum_{v\in V} \lambda_{v,c} \leq \left\lceil \frac{\sum_{v\in V} y_v}{|C|} \right\rceil \qquad \forall c \in C \tag{3}$$

A node $v \in V$ is active if there is traffic in any of its incoming or outgoing edges, being $N(v)$ the set of neighbors of v.

$$y_v \geq \frac{1}{2\,|F|} \sum_{f\in F} \left(\sum_{u\in N(v)} t^f_{u,v} + \sum_{u\in N(v)} t^f_{v,u} \right) \quad \forall v \in V \tag{4}$$

To avoid additional traffic load through network controllers, data plane communications (i.e. $f \in D$) cannot be routed through these devices. Furthermore, control traffic between controllers and switches (i.e. $f \in T$) will not pass through any other controller that is not the source or target of the traffic. The same must hold true for communications between controllers (i.e. $f \in H$). In these constraints we use $N(c)$ to denote the set of neighbors of a controller $c \in C$ and v_f to identify the forwarding device involved in the source/target pair of traffic flow $f \in T$.

$$\sum_{n\in N(c)} t^f_{n,c} \leq \begin{cases} 0 & \forall f \in D, \forall c \in C \\ \lambda_{v_f,c} & \forall f \in T, \forall c \in C \\ 0 & \forall f \in H, \forall c \in C \setminus \{s_f, t_f\} \end{cases} \tag{5}$$

The routing of data plane communications and control traffic exchange between controllers, follows the traditional flow conservation constraints.

$$\forall v \in V, \forall f \in D \cup H: \tag{6}$$
$$\sum_{u\in N(v)} t^f_{v,u} - \sum_{u\in N(v)} t^f_{u,v} = \begin{cases} 1 & \text{if } v = s_f \\ -1 & \text{if } v = t_f \\ 0 & \text{otherwise} \end{cases}$$

Meanwhile, for the subset of communications between controllers and switches $f \in T$, these constraints are modified to assure that only active switches exchange control messages with its controller. Similarly, the forwarding device and the controller involved in the source/target pair of traffic flow $f \in T$, are denoted with v_f and c_f, respectively.

$$\forall v \in V, \forall f \in T: \tag{7}$$
$$\sum_{u\in N(v)} t^f_{v,u} - \sum_{u\in N(v)} t^f_{u,v} = \begin{cases} y_v \lambda_{v_f,c_f} & \text{if } v = s_f \\ -y_v \lambda_{v_f,c_f} & \text{if } v = t_f \\ 0 & \text{otherwise} \end{cases}$$

Finally, a link (i, j) is active if it is used by some traffic flow $f \in F$. Furthermore, the total traffic in each active link must be less than its assigned capacity.

$$\sum_{f\in F} t^f_{i,j} d_f \leq c_{i,j} x_{i,j} \qquad \forall (i,j) \in E \tag{8}$$

Although this model allows the attainment of optimal solutions for the power consumption problem in SDN, it becomes challenging to solve on large and even medium-scale topologies. This is because the difficulty of the power-aware routing problem is known to be NP-Hard [16], so the consumption of resources and time complexity grow exponentially with the network size. To overcome this issue, in the next section we develop a heuristic algorithm.

V. Heuristic Algorithms

To compute all the routes (i.e. for data and associated control traffic) using the global optimization model presented previously, the entire set of traffic demands need to be fixed and known in advance. Considering this as a limitation for current dynamic networking environments, in this section we propose a new approach to support time-variable traffic requirements. The key idea of this proposal is to fully take advantage of the high control flexibility given by the dynamic configuration capabilities and centralized network view of SDN without needing an accurate prediction of incoming traffic. In order to allow that nodes are put into sleep mode we assume network topologies with forwarding devices divided into two categories: edge nodes, which are connected to some traffic source/target and transit nodes, which merely route other nodes traffic.

A. Green Initial Setup (GrIS)

An initial control plane configuration, previous to the data traffic arrival, is required in order to support the in-band mode in SDN. This control plane setup is intended to establish the communication paths between switches and controllers, as well as between controllers. In this way, when new traffic flows arrive, switches can send to the controller routing requests through packet_in messages. To do so, in this section we propose an off-line solution denoted as Green Initial Setup (GrIS). This component will be statically activated at specific time instances as a planned operation.

The proposed strategy, shown in Algorithm 1, takes as inputs the original network topology G with controller placements, the subset of edge nodes $S \subseteq V$ and the control traffic requirements R^c. The outputs are a reduced graph with the initially active network elements $G^A = (V^A, E^A, C)$, an array keeping the controller-switch associations A and the initially required control paths P_c.

In the first step, the algorithm reduces the number of initially active nodes using the NET_PRUNING function, shown in Algorithm 2. This method aims to remove as many nodes as possible, considering as candidates the set of network nodes that will not be endpoints of incoming demands, denoted in the pseudocode as N. For each node inside this set of transit nodes, the function computes its betweenness centrality (B_n), as a measure of its intermediary role in the network. In the proposed approach, we use a simplified version of this metric considering only the shortest paths from each controller to every switch. In particular, after computing the shortest paths from each controller as single source, the B_n associated with a node n is increased for each path containing the node (lines 5-14). Using these values, transit nodes are sorted and stored in the list N'. At each iteration of this list the function attempts to increase the number of switched-off nodes. A new node is removed only when in the resulting graph forwarding devices remain being reachable by network controllers, i.e. at least one path exists between every controller-switch pair in the network.

Algorithm 1 GrIS Pseudocode

Require: G, S, R^c
1: $G' \leftarrow$ NET_PRUNING(G, S)
2: $O \leftarrow$ Get_All_Admissible_Control_Paths(G', R^c)
3: $S' \leftarrow S$ sorted by nodes priority criteria
4: $s \leftarrow$ First node in S'
5: $\left|V^A\right|, \left|E^A\right| \leftarrow \infty$
6: **repeat**
7: **for** $p \in O[s]$ **do**
8: Initialize $(V^{A'}, E^{A'}, P'_c, A', U')$
9: **for** $u \in p \setminus S$ **do**
10: Power_Aware_Path_Selection$(O[u])$
11: Update $(V^{A'}, E^{A'}, P'_c, A', U')$
12: **end for**
13: **for** $n \in L \setminus f$ **do**
14: $r =$ Power_Aware_Path_Selection$(O[n])$
15: Update $(V^{A'}, E^{A'}, P'_c, A', U')$
16: **for** $v \in r \setminus S$ **do**
17: Power_Aware_Path_Selection$(O[v])$
18: Update $(V^{A'}, E^{A'}, P'_c, A', U')$
19: **end for**
20: **end for**
21: **for** $(c1, c2) \in G'$ **do**
22: Power_Aware_Path_Selection$(O[c1, c2])$
23: Update $(V^{A'}, E^{A'}, P'_c, A', U')$
24: **end for**
25: **if** $\left|V^{A'}\right| \leq \left|V^A\right| \wedge \left|E^{A'}\right| \leq \left|E^A\right|$ **then**
26: $V^A, E^A, P_c, A, U \leftarrow V^{A'}, E^{A'}, P'_c, A', U'$
27: **end if**
28: **end for**
29: **if** $\left|V^A\right| = \infty \vee \left|E^A\right| = \infty$ **then**
30: **if** $s =$ last node in L **then** break
31: **end if**
32: $s \leftarrow$ Next node in S'
33: **end if**
34: **until** $\left|V^A\right| \neq \infty \wedge \left|E^A\right| \neq \infty$

To accomplish this, a temporal graph, denoted as G', is created. This graph is used to check the required connectivity between controllers and forwarding devices. After validating that the possibility of reaching every node in the network from any controller is not affected, the considered node is removed from the resulting graph. This means that these nodes together with their links are put into sleep mode in the original graph.

After pruning the network, the GrIS algorithm uses the reduced graph G' to find the overall set of admissible control paths which satisfy control traffic requirements R^c (line 2 in Algorithm 1). As previously stated, these paths do not pass through any other controller that is not the source or target of the traffic. Using these computed control paths, a sorted list of ingress and egress forwarding devices is stored in S'. This list is sorted in ascending order following two priority criteria:

1) the number of possible controllers to associate with,
2) the number of possible control paths.

Algorithm 2 NET_PRUNING

Require: G, S
1: $G' \leftarrow G$
2: $N \leftarrow V - S$ ▷ Transit nodes
3: $H \leftarrow$ NULL ▷ Removed nodes
4: $B \leftarrow$ NULL ▷ Array of betweenness values
5: **for** $n \in N$ **do**
6: $B_n = 0$
7: **for** $c \in C$ **do**
8: $SP_c \leftarrow$ Set of shortest paths from controller $c \in C$
9: **for** $p \in SP_c$ **do**
10: **if** $n \in p$ **then** $B_n = B_n + 1$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: $N' \leftarrow N$ sorted according to increasing order of B
16: **for** $n \in N'$ **do**
17: Remove from G' node n
18: **if** nodes are still reachable by controllers in G' **then**
19: Save n in H
20: **else**
21: $G' \leftarrow G$
22: Remove from G' nodes in H
23: **end if**
24: **end for**

Going through this list, the algorithm starts satisfying the most critical cases and the solution can be found with fewer iterations. The main loop of the Algorithm 1 determines for each possible control path of the selected node s, the number of active elements in the network after computing all control routes. The configuration of paths with fewer active elements is then selected in this process.

Inside this loop the algorithm first determines the paths between controllers and forwarding devices. Note that, for each forwarding device x, $O[x]$ contains admissible control paths to each controller available in the network. Precisely, paths selected in this step define one controller for each forwarding device. Additionally, any time a path between a switch and a controller is computed, nodes belonging to the control path but not in S are analyzed by the algorithm. Note that these nodes are the transit nodes that remained in the resulting graph after pruning the network. Since they are used to route some traffic, a control path is also established between them and some controller. After determining all switch-controller associations, the algorithm searches the paths between controllers.

In general, the power-aware path selected for every control pair is the best route between them in terms of minimizing the number of active elements in the network as long as it has sufficient link capacity to route the traffic volume, under the considered Maximum Link Utilization (MLU) constraint. Additionally, during the selection of one control path between each forwarding device and one controller, the number of devices already attached to the controllers is considered in order to keep a balanced load. Since the set of admissible paths obtained from the pruned network with a reduced number of elements is significantly smaller than in the original topology, the solution can be found with fewer iterations.

If after analyzing all control paths of node s, the algorithm cannot find a feasible configuration of paths to route all control and data plane communications, the main loop repeats this process for the next node stored in S'. This is done until the solution is found or until all forwarding nodes are analyzed, i.e. when the algorithm breaks without a solution. Note that this last option occurs when, given a controllers placement, an admissible configuration for controller-switches association could not be found or when the network has not sufficient capacity to meet the demand requirements under established constraints.

B. Dynamic Power-Aware Routing (DyPAR)

When a new traffic demand arrives, a routing request is sent from the input node to its associated controller using the previously computed path between both devices. Based on its global knowledge of the network topology, this controller calculates the required data path minimizing the number of elements that need to be activated for this connection request and creates the flow forwarding rules. The proposed dynamic power-aware routing, denoted as DyPAR and shown in Algorithm 3, performs in essence three crucial functions:

1) Power-aware data and control path selection;
2) Performance-aware data path selection;
3) Congestion-aware traffic reallocation.

For each incoming demand d, the algorithm starts trying to get the set of admissible data paths across the current active topology. This is done considering that admissible data paths do not pass through any controller in the network. If several paths were found, the one with the highest remaining available link capacity is selected. In this way, the number of future requests that can potentially be accommodated over the currently active paths is increased. Then, traffic is allocated and links utilization are updated.

On the other hand, if no admissible data path was found to route the incoming traffic across the currently active topology, the original network graph is then considered by the algorithm. Since the now determined candidate routes will imply the use of additional network elements, the data path minimizing the number of active network elements is selected to carry the demand. After updating the active topology and links utilization, a loop is used to establish the required control plane communications for each added node along the data path. In the same way, the algorithm first considers the currently active topology to set the required control path with some network controller and the entire network in case of failing the initial attempt.

In case of incoming traffic rates exceeding the remaining available network capacity (line 19 to 21), the algorithm considers all data paths in the original network without taking into account the capacity restrictions, but keeping that data plane communications cannot be routed through network controllers.

Algorithm 3 DyPAR Pseudocode

Require: G, G^A, P_c, A, U, d
$P_d \leftarrow$ Get_Admissible_Paths(G^A, d)
if $P_d \neq Null$ **then**
  $p_d \leftarrow$ Lest loaded path in P_d
  Update U after routing p_d
else
  $P_d \leftarrow$ Get_Admissible_Paths(G, d)
  if $P_d \neq Null$ **then**
    $p_d \leftarrow$ Power_Aware_Path_Selection(P_d)
    Update G^A, U after routing p_d
    for node n added to G^A by p_d **do**
      $P_c \leftarrow$ Get_Admissible_Paths(G^A, n, C)
      if $P_c = Null$ **then**
        $P_c \leftarrow$ Get_Admissible_Paths(G, n, C)
      end if
      $p_c \leftarrow$ Power_Aware_Path_Selection(P_c)
      Update G^A, U, A after routing p_c
    end for
  else
    $P_d \leftarrow$ Get_All_Paths(G, d)
    $p_d \leftarrow$ Performance_Aware_Path_Selection(P_d)
    Update U, T after routing p_d
  end if
  $BL \leftarrow$ Link with maximum load
  $F \leftarrow$ Demands established through BL
  CONGESTION_AWARE_REROUTING(G^A, F, BL, U)
end if

Then, the algorithm performs a data path selection based on reducing the performance degradation incurred. More in detail, the algorithm selects the data path inside this group whose congested links are less used by previously established demands. The reason is that, to allow the new traffic flow, the capacity remaining on those links, after allocating the QoS sensitive demands and control traffic, will be fairly divided between the involved lower-priority demands. Rates beyond this resulting throughput will be reduced and traffic will be handled on a "best-effort" basis. In this way, the proposed algorithm can efficiently handle bursty traffic and accommodate rates that exceed the remaining available capacity without affecting the QoS sensitive traffic if the network is not heavily loaded.

Every time a new network element is added to the active topology, the algorithm tries to alleviate the congestion level on the network. To accomplish this, after identifying the bottleneck link and the group of traffic flows going through this link, a CONGESTION_AWARE_REROUTING is performed. This function, described in Algorithm 4, starts creating a temporal graph G'' where the most loaded link is removed. Additionally, currently established demands sharing the most loaded link are sorted in decreasing order of rate requirements with the aim of reducing the congestion after rerouting the fewer number of connections. In order to avoid frequent reallocations of a traffic flow and mitigate related negative implications, a time threshold can be easily included to select

Algorithm 4 CONGESTION_AWARE_REROUTING

Require: G^A, F, BL, U
$Current\_MaxU \leftarrow U[BL]$
$G'' \leftarrow G^A$
Remove BL from G''
$F' \leftarrow F$ sorted by decreasing order of flow rate
for established demand f in F' **do**
  $P \leftarrow$ Get_Admissible_Paths(G'', f)
  $p \leftarrow$ Congestion_Avoidance_Path_Selection(P)
  $MaxU_p \leftarrow$ Maximum link utilization in p
  if $p \neq None \wedge MaxU_p < Current\_MaxU$ **then**
    Reroute f and associated control traffic
    Update U and $Current\_MaxU$
  end if
end for

only demands that have been allocated long enough over the current path. Using the residual graph a new set of admissible paths is obtained for each involved traffic flow. Then, the function looks for a path with lower load values trying to leave more resources available for future demands. A traffic flow is reallocated only when a feasible path is found and the MLU in the network is reduced. At the same time, the required control paths are updated.

Since the proposed approach is conceived for dynamic traffic environments, the set of established demands will be constantly checked. For those connection requests whose holding times have expired, the algorithm performs a demand removal, which means that their assigned paths are released and resources occupied by these routes become available again. Consequently, network elements used only by completed traffic demands will be then put into sleep mode.

VI. PERFORMANCE EVALUATION

To assess the performance of the ILP model, we used the linear programming solver Gurobi Optimizer [17]. Meanwhile, heuristic algorithms were implemented using the programming language Python. All computations were carried out on a computer with 3.30 GHz Intel Core i7 and 16 GB RAM.

A. Simulation Scenario

1) Network Topology: We conducted our simulations using real-world network topologies collected from SNDlib [18], considering each router in the network as an SDN node or as a controller placement. Specifically, in order to assess the effectiveness of the proposed scheme we use three topologies of different sizes. These networks are: Nobel-US (14 nodes, 21 links), Geant (22 nodes, 36 links) and Cost266 (37 nodes, 57 links). To allow the possibility of putting network nodes into sleep mode, different scenarios were considered varying T, which represents the percentage of forwarding devices that will not generate or receive traffic. According to this value, for each network topology we have selected as transit nodes the devices with the highest degree centrality as in [6].

2) Controllers Placement: Being the controller placements out of the scope of this paper we assume as preferred locations the ones minimizing the worst-case mean latencies. More precisely, we compute the mean propagation latency between each pair of nodes and associate each admissible location with the maximum average value involving it. Then, according to the number of controllers considered for each simulation instance, we place the controllers at node locations with smaller associated latency values. Note that a controller placement is admissible when the assumptions established in this proposal to avoid the routing of additional traffic load through network controllers can be kept (i.e. the network graph without any controller remains being strongly connected).

3) Traffic Patterns: Apart of the real static traffic matrices obtained from the topologies database in [18], we also consider a dynamic scenario where connection requests arrive with exponentially distributed inter-arrival and holding times taking different mean values from the sets [0.2, 1, 5] and [100, 150, 200], respectively. Accordingly, a traffic flow is generated between each pair of edge nodes (i.e. network devices which do not act as controllers or transit nodes). To evaluate the power savings and performance degradations considering increasing loads, for each network topology we considered every pair of edge nodes with an initial randomly assigned data rate and computed the associated shortest paths. We then identified the most loaded link from which we derived a scaling factor. Lastly, the initially assigned values were multiplied by this scaling factor to obtain the corresponding data rates for each incoming demand (see [19]). This was done considering different values of the over-provisioning factor (α) to further evaluate the implications of varying traffic load. We assume an average control traffic rate of 1.7 Mbps [20].

4) Power Values: Based on the power consumption behavior of data networks explained in Section II, we characterize the power consumption of a forwarding device using the 3:1 idle:active ratio given in [9]. This proportion, obtained from measurements on real switches, assigns 3W of power for each idle port of a switch and 1W extra when the port is active. Thus, power consumption P_n of a idle forwarding device n can be computed as $3D(n)$ where $D(n)$ denotes the node degree and P_p = 1W. Null power consumption is assumed when the node is put into sleep mode.

B. Optimal vs. Heuristic Solutions

To assess the suitability of the proposed solution we start evaluating the performance of the heuristic algorithms against the optimal ILP model, using the Nobel-US and Geant topologies with traffic matrices provided in [18]. This comparison is illustrated in Fig. 2 for different amount of controllers and percentage of transit nodes. Power savings are computed according to the expression $(Overall\_Pw - Pw\_X)/Overall\_Pw$, where $Overall\_Pw = \sum_{n \in V} P_n + 2P_p\,|E|$ and Pw_X is the power consumption achieved by the considered approach.

In Fig. 2 power savings of up to 35% can be reached by our optimization model in both topologies. Moreover, the heuristic approach allows to obtain close-to-optimal power savings with differences under 4% (Nobel-US) and 8% (Geant). It is also observed in both networks that lower savings are achieved when the percentage of transit nodes decrease from 50% to 10%. This behavior is expected given that with fewer transit nodes a smaller number of forwarding devices can be put into sleep mode, which yield the mayor contribution to the attained power savings. Additionally, with fewer transit nodes a higher number of demands are handled, thus more paths need to be established to accommodated such traffic. On the other hand, increasing the number of controllers implies in some cases a reduction in the power savings.

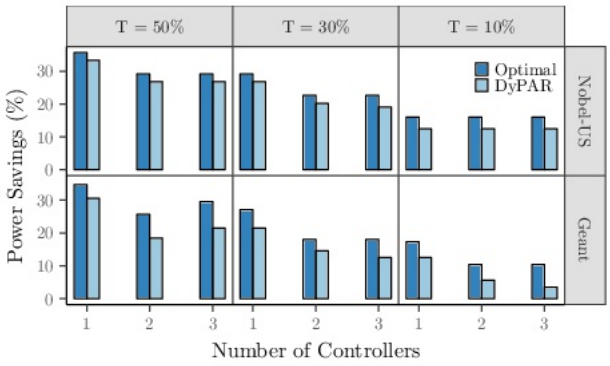


Fig. 2. Power savings in the Nobel-US and Geant topologies as a function of controllers amount, varying the percentage of transit nodes (T).

C. Assessment of Power Saving Potential

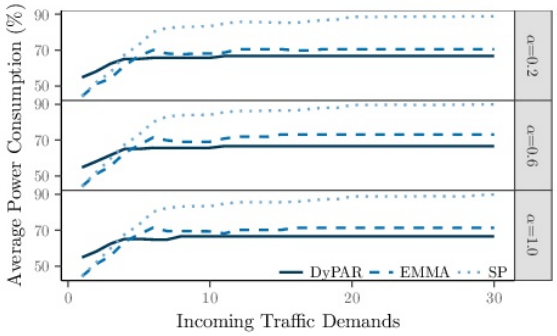
Due to the computational complexity of the exact model in networks similar in size or larger than Geant (see [15] for similar running time values), in what follows we use our heuristic algorithms. This is done taking into account a dynamic scenario with connection requests generated following the procedure previously explained. Several test were conducted and average values have been determined with a margin error less than 5.5% in the three considered networks, estimated by running our algorithm 10 times with different prime number seeds on each traffic configuration instance.

In terms of average running time of the algorithms, the off-line GrIS module requires around 39 ms (Nobel-US), 0.25 s (Geant) and 283 s (Cost266). Meanwhile, the DyPAR algorithm takes always less than 6.4 ms (Nobel-US), 16.5 ms (Geant) and 282.6 ms (Cost266), for all the considered traffic patterns. These values reveal the suitability of the proposed strategy for real-world deployments and its adequate scalability in terms of network size and traffic load. Due to space limitation, we may focus our attention on some specific traffic pattern configuration, but the general conclusions derived from performed evaluations hold for all the considered values.

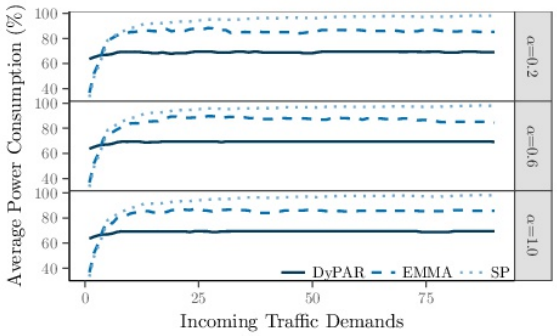
In addition, in order to evaluate the benefits of our proposal we compare the performance of the proposed algorithms with other two existing energy-aware routing approaches presented in related works [7] and [8], referred to here as SP and EMMA, respectively. As we are considering an in-band SDN, required control plane communications will be also established by these

two approaches. At the same time, shortest paths used by SP and EMMA are computed holding restrictions established to avoid additional traffic load through the network controller (i.e. data traffic cannot be routed through this device). On the other hand, we set the time threshold for demands reallocation (half of connection expected duration) and the number of transit nodes ($T = 50\%$) as in [8] for the three algorithms used in this comparison. Given the lack of support in SP and EMMA for network environments with multiple controllers we only consider the case of having one centralized network controller. However, the derived conclusions are general and a similar behavior is expected in case of having multiple controllers.

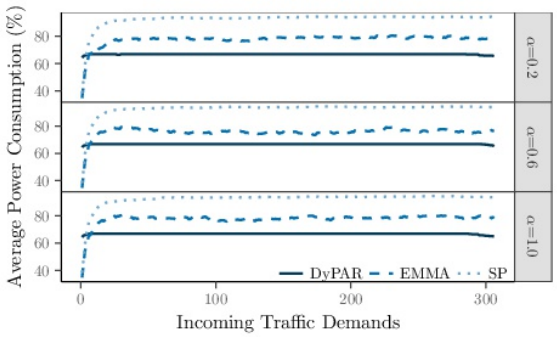
Fig. 3 shows the power consumption achieved by the three algorithms considering different topological scenarios and over-provisioning factor (α). These results correspond with an average arrival time of 0.2 demands/s and a mean holding time of 100 s, but similar values have been obtained for all the considered traffic patterns. Given the initial control plane configuration performed by the GrIS module, in the three considered topologies the other two methods exhibit a better behavior at the beginning of simulations. However, after allocating few demands more power can be saved by our approach. As it is shown, in terms of consumed power, DyPAR outperforms SP in all cases and it is generally better (in some cases just slightly better) than EMMA. For instance, after routing all incoming traffic, DyPAR attains power consumption reductions of up to 26.5% and 19.4% with respect to SP and EMMA, respectively. The reason is that SP only uses precomputed shortest paths to allocate the incoming traffic, while EMMA also performs a power-aware rerouting any time the active topology changes in order to find better paths for already allocated flows. On the other hand, power improvements achieved by our proposal are consequence of the combined GrIS/DyPAR operation where a minimum network subset is initially activated and new network elements (nodes and links) are only added when the incoming demand cannot be allocated on the currently active topology.



(a) Nobel-US topology.



(b) Geant topology.



(c) Cost266 topology.

Fig. 3. Power consumption in the three topologies with one controller as a function of traffic arrival, varying the over-provisioning factor (α).

D. Performance Degradation Avoidance

These power savings are only valid if the performance of QoS sensitive demands is not compromised. Moreover, to avoid overloaded networks a capacity reserve is typically set. So far, we had not considered this capacity margin, but now we analyze how the number of allocated demands is impacted when facing a more demanding traffic pattern and in presence of a MLU constraint. In this evaluation we set the average arrival time to 5 demands/s and the mean holding time to 200 s, while keeping the over-provisioning level equal to 1, since this represents the most demanding of the considered traffic patterns for the heuristics and the most critical from the performance degradation perspective.

Fig. 4 shows the percentage of demands that can be allocated by DyPAR, EMMA and SP in Nobel-US and Geant using different values of MLU. As it shown, DyPAR is able to reduce the blocking rate with respect to the other two approaches as a result of the CONGESTION_AWARE_REROUTING performed by this solution. In particular, while only negligible blocking rates are attained by our approach (less than 1.2%), up to 7 and 12 demands are blocked by SP and EMMA, respectively. SP performs better than EMMA given that in case of having more than one candidate route this algorithm selects the one leaving more available link capacity.

Intuitively, the capacity to successfully allocate the incoming traffic will not only be a result of the performed routing, since it is also related to the considered topology. In network topologies with more path redundancy a higher number of requests can potentially be accommodated. This difference can be noticed between Nobel-US and Geant, where an increase of allocated demands of up to 26.7% and 15.6% can be reached, respectively. Cost266 is not shown in Fig. 4, since a complete

(a) Nobel-US topology.

(b) Geant topology.

Fig. 4. Number of allocated demands with one controller as a function of traffic arrival, varying the MLU.

routing was always achieved in this topology by the three compared algorithms under the considered traffic patterns and MLU levels.

VII. Conclusion

In this paper we proposed a power-aware strategy that reduces the number of active nodes and links used to handle the incoming traffic suitable for SDN environments with in-band control traffic and multiple controllers. To achieve such goal, we first provided a link-based formulation of the optimization problem, integrating the routing requirements for data and control traffic. For large-scale topologies a heuristic approach is conceived combining a static control plane configuration with a dynamic power-aware routing. Besides being compatible with SDN environments without a dedicated control network, this strategy is able to handle demanding traffic arrival without degrading the performance of higher priority traffic. Through simulations using real-world topologies, we have validated that our heuristic approach allows to obtain close-to-optimal power savings, with differences under 8%. Furthermore, our proposal achieves better results in terms of power consumption and number of allocated demands than two existing related algorithms. For instance, after routing all incoming traffic, a reduction of power consumption of up to 26.5% and an increase of allocated demands of up to 26.7% can be reached by our solution. Lastly, it is important to emphasize that to exploit the reported benefits of our approach, fast switching-on technologies, allowing quick responses and low reconfiguration times between sleeping modes, are required for practical implementations. In the same way, additional criteria to ensure the capability of the network to quickly react in case of suddenly failures should be further analyzed. Therefore, the inclusion of restoration mechanisms in order to improve the fault tolerance capacity of our approach will be an important future task.

References

[1] "Cisco Visual Networking Index: Forecast and Methodology, 2016–2021," White Paper, Cisco, Sep. 2017.

[2] R. S. Tucker, "Energy Consumption in Telecommunications," in *Proc. Optical Interconnects Conference*, May 2012, pp. 1–2.

[3] M. Gupta and S. Singh, "Greening of the Internet," in *Proc. ACM SIGCOMM*, 2003, pp. 19–26.

[4] D. Kreutz, F. M. Ramos, P. Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[5] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Automatic Bootstrapping of OpenFlow Networks," in *Proc. IEEE LANMAN*, Apr. 2013, pp. 1–6.

[6] A. P. Bianzino, C. Chaudet, F. Larroca, D. Rossi, and J. L. Rougier, "Energy-Aware Routing: A Reality Check," in *Proc. IEEE GLOBECOM*, Dec. 2010, pp. 1422–1427.

[7] B. Özbek, Y. Aydoğmuş, A. Ulaş, B. Gorkemli, and K. Ulusoy, "Energy Aware Routing and Traffic Management for Software Defined Networks," in *Proc. IEEE NetSoft*, Jun. 2016, pp. 73–77.

[8] S. S. Tadesse, C. Casetti, C. F. Chiasserini, and G. Landi, "Energy-Efficient Traffic Allocation in SDN-basec Backhaul Networks: Theory and Implementation," in *Proc. IEEE CCNC*, Jan. 2017, pp. 209–215.

[9] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks," in *Proc. USENIX NSDI*, 2010.

[10] F. A. Moghaddam and P. Grosso, "Linear Programming Approaches for Power Savings in Software-Defined Networks," in *Proc. IEEE NetSoft*, Jun. 2016, pp. 83–87.

[11] H. Zhu, X. Liao, C. de Laat, and P. Grosso, "Joint Flow Routing-Scheduling for Energy Efficient Software Defined Data Center Networks: A Prototype of Energy-Aware Network Management Platform," *Journal of Network and Computer Applications*, vol. 63, pp. 110–124, 2016.

[12] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A Power Benchmarking Framework for Network Devices," in *Proc. IFIP-TC*, 2009, pp. 795–808.

[13] A. Fernández-Fernández, C. Cervelló-Pastor, and L. Ochoa-Aday, "Achieving Energy Efficiency: An Energy-Aware Approach in SDN," in *Proc. IEEE GLOBECOM*, Dec. 2016, pp. 1–7.

[14] ——, "Energy-Aware Routing in Multiple Domains Software-Defined Networks," *ADCAIJ*, vol. 5, no. 3, pp. 13–19, Nov. 2016.

[15] ——, "Energy Efficiency and Network Performance: A Reality Check in SDN-Based 5G Systems," *Energies*, vol. 10, no. 12, Dec. 2017.

[16] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy, "Minimizing Routing Energy Consumption: From Theoretical to Practical Results," in *Proc. IEEE/ACM GreenCom*, Dec. 2010, pp. 252–259.

[17] Gurobi Optimizer (Version 7.5). [Online]. Available: http://www.gurobi.com/

[18] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0-Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[19] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 463–476, 2012.

[20] J. Li, J.-H. Yoo, and J. W.-K. Hong, "Dynamic Control Plane Management for Software-Defined Networks," *International Journal of Network Management*, vol. 26, no. 2, pp. 111–130, 2016.