Energy-efficient Traffic Allocation in SDN-based Backhaul Networks: Theory and Implementation

Energy-efficient Traffic Allocation in SDN-based Backhaul Networks: Theory and Implementation

Senay Semu Tadesse, Claudio Casetti, Carla Fabiana Chiasserini
Politecnico di Torino, Italy

Giada Landi
Nextworks, Italy

***Abstract*—5G networks are expected to be highly energy efficient, with a 10 times lower consumption than today's systems. An effective way to achieve such a goal is to act on the backhaul network by controlling the nodes operational state and the allocation of traffic flows. To this end, in this paper we formulate energy-efficient flow routing on the backhaul network as an optimization problem. In light of its complexity, which impairs the solution in large-scale scenarios, we then propose a heuristic approach. Our scheme, named EMMA, aims to both turn off idle nodes and concentrate traffic on the smallest possible set of links, which in its turn increases the number of idle nodes. We implement EMMA on top of ONOS and derive experimental results by emulating the network through Mininet. Our results show that EMMA provides excellent energy saving performance, which closely approaches the optimum. In larger network scenarios, the gain in energy consumption that EMMA provides with respect to the simple benchmark where all nodes are active, is extremely high under medium-low traffic load.**

I. Introduction

The 5G-Infrastructure Public-Private Partnership (PPP), created to design and deliver architectures, technologies and standards for 5G communication, has set the following Key Performance Indicators (KPI) for energy management [1], [2]: (i) energy efficiency improvement by at least a factor of 3 and (ii) reduction of energy cost per bit by a factor of 10. Of course, no single solution can achieve such ambitious goals. Instead, they should be achieved through orchestrated actions, involving a fully-unified, automated control and management plane, that oversees radio resources as well as computation and transport resources in the the fronthaul and backhaul. A key role is played by Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), tasked with the control and coordination of hundreds of nodes that need to be reconfigured on the fly in order to optimize utilization and QoS, in view of rapidly changing traffic flows. Among the coordinated actions that can be taken are the de-activation or decommissioning of scarcely used network portions, including links and switches, and the flexible re-routing of existing flows so as to jointly address energy saving and QoE requirements.

In this paper, we address the latter action, by designing and evaluating an Energy Monitoring and Management Application (EMMA), that can minimize energy consumption of the backhaul network. Consistently with the pervasive use of SDN solutions expected in 5G networks, EMMA natively interacts with the Northbound interface of ONOS [3], a popular carrier-grade network operating system, whose Southbound interface is used to control OpenFlow switches. The design of EMMA hinges upon heuristic algorithms for the dynamic routing of flows and the management of the resulting link and switch activity. These algorithms represent a heuristic solution to a non-linear integer problem that aims at minimizing the instantaneous power consumption of nodes and links. Performance evaluation has been done by comparing the optimum obtained through the above optimization formulation, with practical results derived by implementing the algorithms in an SDN network emulation environment.

The rest of the paper is organized as follows. Section II introduces the power model for OpenFlow switches that we adopt and formalizes the problem under study. Our heuristic scheme is presented in Section III, while its implementation on top of ONOS, as well as the required interactions between ONOS and the underlying network, are described in Section IV. Emulation results and the comparison between EMMA and the optimum solution are shown in Section V. A detailed discussion of previous work and of our novel contribution with respect to that, is provided in Section VI. Finally, Section VII draws some conclusions and highlights future research directions.

II. System Model and Problem Formulation

Energy consumption of the backhaul network can be minimized by limiting the number of active links and nodes, i.e., by *(i)* turning off link drivers whenever possible, resulting in proportional (possibly non-linear) changes, and *(ii)* turning off those nodes whose links are inactive.

Both approaches can be studied by building a directed network graph whose vertices represent the network core switches, and edges correspond to links connecting the switches. Let us then consider that the network includes N core switches and L links and denote by $\mathcal{N}$ and $\mathcal{L}$ the set of switches and links, respectively. Let a link $(i,j) \in \mathcal{L}$, with $i,j \in \mathcal{N}$, have a capacity $C(i,j)$ bits/s. Let $\mathcal{F}(t)$ denote the set of flows at time t, with each flow, $f^{\alpha\omega} \in \mathcal{F}(t)$, characterized by the pair of end[1] core switches α and ω, and by QoS constraints that in our case correspond to the required data rate $R(f^{\alpha\omega})$.

Let $x_{ij}(t)$ be a binary variable indicating whether link $(i,j) \in \mathcal{L}$ is "on" ($x_{ij}(t) = 1$) or "off" ($x_{ij}(t) = 0$), at time t. Likewise, $y_i(t)$ is a binary variable indicating whether node

[1]Since we do not consider edge switches, the end core switches are those where the flow, respectively, starts and ends in the backhaul.

TABLE I
MODEL NOTATIONS

N, L	No. of core switches and links	$\mathcal{N}$, $\mathcal{L}$	Set of switches and links
$(i,j) \in \mathcal{L}$	Link from switch i to switch j	$C(i,j)$	Capacity of link (i,j)
$\mathcal{F}(t)$	Set of active traffic flows at time t	$f^{\alpha\omega} \in \mathcal{F}(t)$	Active flow between end core switches α and ω
$R(f^{\alpha\omega})$	Rate requirement for flow $f^{\alpha\omega}$	$\boldsymbol{\pi}$	Generic path as an ordered sequence of links
P_{idle}	Power consumption of an idle switch	$P_{sw}(i,j,t)$	Power consumption due with link (i,j) at t
$x_{ij}(t)$	Takes 1 if link (i,j) is on at t, 0 else	$y_i(t)$	Takes 1 if switch i is "on" at t, 0 else
$z_{\boldsymbol{\pi},f^{\alpha\omega}}(t)$	Takes 1 if $f^{\alpha\omega}$ is routed through path $\boldsymbol{\pi}$ at t, 0 else	$\tau_{ij}(t)$	Traffic flowing over link (i,j) at t

$i \in \mathcal{N}$ is active at time t ($y_i(t) = 1$) or not ($y_i(t) = 0$). Also, let a path $\boldsymbol{\pi}$ be an ordered sequence of links. We indicate by the binary variable $z_{\boldsymbol{\pi},f^{\alpha\omega}}(t)$ whether flow $f^{\alpha\omega} \in \mathcal{F}(t)$ is routed through path $\boldsymbol{\pi}$ at time t ($z_{\boldsymbol{\pi},f^{\alpha\omega}}(t) = 1$) or not ($z_{\boldsymbol{\pi},f^{\alpha\omega}}(t) = 0$).

We consider that the generic core switch i has zero power consumption when "off", and P_{idle} when "on" but idle. The power consumption associated with a link, (i,j), at time t linearly depends on the traffic that flows over the link. It follows that the total power consumption of an active core switch i is given by:

$$P(i,t) = P_{idle} + E_{sw} \sum_{j \in \mathcal{N}, j \neq i} \tau_{ij}(t) \qquad \text{(II.1)}$$

where the second term on the right hand side of the above equation represents the power consumption at i due to traffic switching. In particular, E_{sw} is the energy consumption per bit due to traffic switching and $\tau_{ij}(t)$ is the total traffic (expressed in bit/s) flowing over link (i,j) at time t. We have:

$$\tau_{ij}(t) = \sum_{f^{\alpha\omega} \in \mathcal{F}(t)} \sum_{\boldsymbol{\pi}:(i,j)\in\boldsymbol{\pi}} R(f^{\alpha\omega}) z_{\boldsymbol{\pi},f^{\alpha\omega}}(t)\,; \qquad \text{(II.2)}$$

Note that in Eq. (II.1) we accounted only for outgoing links since we consider core switches, which cannot be source or destination of traffic flows. Furthermore, incoming traffic equals outgoing traffic and only one of them should be considered since the switch processes that traffic only once.

Below, we first present the power consumption model we adopt in order to determine realistic values for P_{idle} and E_{sw}, for OpenFlow switches. Then, we formalize the problem under study by using standard optimization.

A. Power Model

The power consumption of an IP router or an Ethernet switch that is "on" is the sum of the power consumed by its three major subsystems [4]: $P_{ctr} + P_{evn} + P_{data}$, where P_{ctr} accounts for the power needed to manage the switch and the routing functions, P_{evn} is the power consumption of the environmental units (such as fans), and P_{data} indicates the data plane power consumption. The latter can be decomposed into (i) a constant baseline component, and (ii) a traffic load dependent component. In other words, when a switch is powered on but it does not carry any data traffic, it consumes a constant baseline power. When a device is carrying traffic, it consumes additional load-dependent power for header processing, as well as for storing and forwarding the payload across the switch fabric. Combining the power model in [4] with that for OpenFlow switches in [5], we can write P_{idle} as the sum of P_{ctr}, P_{evn} and the base line component of P_{data}. The load-dependent component of P_{data} instead contributes to the per-bit energy consumption due to traffic switching. In particular, we have:

$$E_{sw} = \frac{E_{lookup}}{\sigma} + E_{rx} + E_{xfer} + E_{tx} \qquad \text{(II.3)}$$

In the above expression,

- E_{lookup}/σ is the energy consumed per bit in the *lookup* stage of a switch. E_{lookup} is the energy consumption due to searching the TCAM for the received flow-key and retrieving the forwarding instructions, while σ is the packet size;
- E_{rx} is the energy consumed per bit in the switch *reception* stage, which involves receiving a packet from the physical media, extracting important fields to build a flow-key and streaming the packet into the input memory system;
- E_{xfer} is the energy consumed per bit in the *xfer* stage, which involves reading a packet from the inbound memory, all of the logic required to initiate a transfer across the fabric, driving the fabric connections and crossbar, as well as writing the packet into the remote outbound memory;
- E_{tx} is the energy consumed per bit in the switch *transmission* stage, which involves reading a packet from the outbound memory and transmitting it on the physical media.

In the following, we set: $E_{rx} = E_{tx} = 0.2$ nJ/bit, $E_{xfer} = 0.21$ nJ/bit, $E_{lookup} = 17.4$ nJ/bit, and $P_{idle} = 90$ W [6].

B. Minimum-energy Flow Routing

The problem of energy-efficient flow allocation can be modeled similarly to what done in [7], [8]. However, we stress that, using the accurate power model introduced above, optimal flow routing becomes a non-linear integer problem (see our discussion at the end of this section). Furthermore, in order to achieve the minimum energy expenditure, we aim at minimizing the *instantaneous* power consumption of the network, i.e., the flow routing problem should be solved whenever a new flow starts or an existing flow ends. Typically, this is impractical in real-world networks but our goal here is to set the best performance that could be achieved. We report

below the formulation of the optimization problem adapted to our scenario.

Objective. The goal is to minimize the instantaneous power consumption, which is the sum of the power consumption due to the nodes being "on" and to active link drivers:

$$\min \sum_{i \in \mathcal{N}} \left[y_i(t) P_{idle} + E_{sw} \sum_{j \in \mathcal{N}, j \neq i} x_{ij}(t) \tau_{ij}(t) \right] \quad \text{(II.4)}$$

where recall that $\tau_{ij}(t)$ depends on the binary variable $z_{\boldsymbol{\pi}, f^{\alpha\omega}}(t)$, as reported in Eq. (II.2).

Constraints.

- Flow conservation constraint, for any $j \in \mathcal{N}$:

$$\sum_{i \in \mathcal{N}, i \neq j} \tau_{ij}(t) x_{ij}(t) - \sum_{k \in \mathcal{N}, k \neq j} \tau_{jk}(t) x_{jk}(t) = \sum_{f^{\alpha\omega} \in \mathcal{F}(t): j = \omega} R(f^{\alpha\omega}) - \sum_{f^{\alpha\omega} \in \mathcal{F}(t): j = \alpha} R(f^{\alpha\omega}) \quad \text{(II.5)}$$

- The total traffic flowing on a link must not exceed the link capacity:

$$\tau_{ij}(t) \leq C_{ij} \,, \quad \forall (i,j) \in \mathcal{L} \,. \quad \text{(II.6)}$$

- A link between two nodes, i and j, can be active only if i and j are both active:

$$\sum_{j \in \mathcal{N}, j \neq i} [x_{ij}(t) + x_{ji}(t)] \leq M y_i(t) \,, \quad \forall i \in \mathcal{N} \quad \text{(II.7)}$$

  where M is an arbitrary constant s.t. $M \geq 2(N-1)$.

The input parameters of the above problem are the set of nodes, links and traffic flows, along with their characteristics, while the decision variables are: $x_{ij}(t)$, $y_i(t)$, and $z_{\boldsymbol{\pi}, f^{\alpha\omega}}(t)$. Thus, the problem is an integer non-linear problem, due to the product of $x_{ij}(t)$'s and $z_{\boldsymbol{\pi}, f^{\alpha\omega}}(t)$'s in the objective function and in the flow-conservation constraints, which appears when $\tau_{ij}(t)$ is expressed as a function of $z_{\boldsymbol{\pi}, f^{\alpha\omega}}(t)$ (see Eq. (II.2)). Also, it is akin to the bin packing problem[2], which is a combinatorial NP-hard problem. Thus, obtaining the optimal problem solution in large-scale scenarios is not viable. Below we propose a heuristic algorithm that has low computational complexity and whose performance results to be very close to the optimum.

III. EMMA: A Heuristic Approach

In order to design an efficient algorithm to solve the above problem, we first observe that an efficient heuristic for the solution of the bin packing problem is the First Fit algorithm [9], [10]. Given the set of items to be inserted into the bins, the First Fit algorithm processes an item at a time in arbitrary order and attempts to place the item in the first bin that can accommodate it. If no bin is found, it opens a new bin and puts the item in the new bin.

[2]In the bin packing problem, objects of different volumes must be packed into a finite number of bins, each of a given capacity, in a way that the number of used bins is minimized.

We leverage the First Fit algorithm and design a heuristic scheme, named Energy Monitoring and Management Application (EMMA), which: (i) monitors the network status, (ii) efficiently allocates traffic flows as they come, and (iii) and re-routes the existing flows when necessary and possible. Flows are (re-)routed by EMMA with the aim to minimize the length of the flow path and the energy consumption of the overall network. In particular, upon the arrival of a new flow, EMMA first tries to fit the flow into the current "active network" while meeting the flow traffic requirements. It then turns on other links and/or nodes only if no suitable path is found. Every time a new link and/or node are added to the active network, EMMA looks for a better alternative path for flows that have been (re-)allocated long enough (more than half their expected duration) ago. If a more energy-efficient allocation is found, then a flow is diverted on the new path, provided that its traffic requirements are still met. Note that EMMA differs from the First Fit algorithm since it tries to find a better path for already allocated flows whenever any change in the active topology occurs.

More in detail, the EMMA scheme is composed of two algorithms. Algorithm 1 presents the sequence of actions to be taken whenever a new flow is activated in the network. Input to the algorithm is the network topology and the information on the new flow to be allocated, the power state of the network devices and the traffic crossing every link. Initially, the computation of the possible paths for the incoming flow is done considering the nodes and links that are currently active (namely, the active network) (line 2). Then if there exists a path that meets the flow traffic requirements, the flow can be successfully allocated (lines 3-5). Otherwise, the whole network should be considered and the search for a suitable path repeated (lines 6-7). If a path is found, the links and nodes that need to be added are activated (lines 8-11).

Algorithm 1 New flow allocation

```
Require: Topology, new flow, network power state, traffic load
 1: for each new flow do
 2:     Compute all shortest paths across active topology
 3:     if suitable path is found then
 4:         % if more than one, select one at random
 5:         Allocate the flow
 6:     else
 7:         Compute shortest paths considering the whole network
 8:         if suitable path is found then
 9:             % if more than one, select one at random
10:             Turn on the selected links and nodes that are off
11:             Allocate the flow
12:             installation_time ← current_time
13:             Run Algorithm 2
14:             % It moves previous flows to a better path if any
```

Algorithm 2 states the steps followed during re-routing of existing traffic. This algorithm is run whenever there is a change in the active network topology, i.e., if nodes and/or links become active while finding a path for a new flow.

Input to the algorithm are the network topology and the information on the current flows. The algorithm selects all

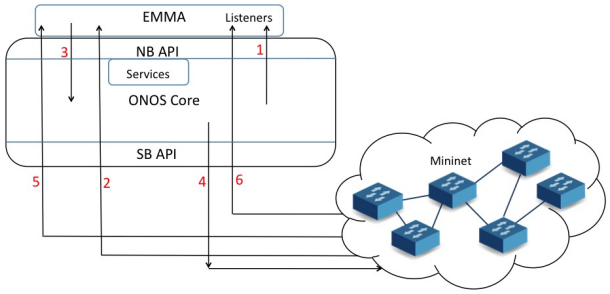

Fig. 1. Interaction between EMMA and ONOS, and between ONOS and Mininet. The interactions are labeled with numbers corresponding to the sequence of events described in the text.

flows that have started or re-allocated at least T_a time ago (line 1). For each flow satisfying the time hysteresis T_a, it computes a path on the current topology starting with the flows with higher rate requirements (lines 2-4). If the cost of the computed path is less than that of the flow current path, it diverts the flow to the new path and updates the flow installation time (lines 5-7). If the process of moving flows to a different path results in some links and/ or nodes being idle, those links and/or nodes are turned off (lines 8-9). Similarly, upon the termination of a flow, nodes and links which are idle will be switched off to save energy.

Algorithm 2 Move flows to a better path

Require: Topology, information on current flows
1: $\mathcal{S} \leftarrow \{$flows s.t. installation_time $\geq T_a\}$
2: Order flows in $\mathcal{S}$ with decreasing rate requirements
3: **for** each flow in $\mathcal{S}$ **do**
4: Search a path on the active topology
5: **if** suitable path exists $\wedge$ (new path cost $<$ old path cost) **then**
6: Move flow from old to new path
7: installation_time $\leftarrow$ current_time
8: **if** there are links and/or nodes no longer carrying traffic **then**
9: Turn them off

IV. EMMA Implementation in ONOS

We consider a system architecture including:

- a network of OpenFlow switches and links interconnecting them;
- an SDN controller;
- an application on top of the SDN controller that implements EMMA.

The system has to:

- keep track of the set of active network elements, and the set of current traffic flow routes;
- route traffic (i.e., check routing paths for traffic flows and push routes into the network);
- monitor and toggle the power state of the network elements.

In SDN, traffic forwarding rules are created by the controller and are installed into the switches. The flow rule installation could be either proactive, e.g., flows rules are installed into the switches prior to the actual flow arrival, or reactive, e.g., flows rules are created upon every new flow coming into the switches. In the latter case, the switches at the network edge will need to be always active, in order to receive the traffic that a host may generate. When an edge switch receives a packet for which it has no rule, it directly sends the packet to the controller. Then the controller gets from EMMA a path for such kind of packets and installs a flow rule into the switches. Afterwards, all packets of that kind will be forwarded based on the flow rule just installed.

In the implementation of our algorithms we followed a reactive approach. The algorithms are tested using the Mininet SDN network emulation environment [11] as forwarding plane, and a controller whose functions are supported by the Open Network Operating System (ONOS) [3]. The algorithm is developed based on ONOS built-in Reactive Forwarding application, which uses the shortest path to route incoming traffic. However, we recall that the topology considered for the shortest path search is not always the whole topology. Indeed, EMMA first tries to find a path on the nodes/links which are already carrying traffic thus concentrating all the traffic in the active infrastructure whenever possible. If no path is found using the active network, EMMA considers the whole network to route the traffic.

Also, it is worth mentioning here that EMMA implements the *PacketProcessor* interface of ONOS, running below it in the architecture, in order to process packets, i.e., we replicated the processing function at the *PacketProcessor* interface of ONOS within EMMA. In this way, EMMA can distinguish between traffic and control packets, as well as between unicast and multicast packets, thus further processing only unicast traffic packets.

Finally, we consider that initially all network nodes and links are "on" and that ONOS provides EMMA with the whole network topology. Specifically, EMMA gets the network topology via the *TopologyService* application of ONOS, to which EMMA has registered. EMMA can then use the network topology for path computation and flow provisioning. If there are idle core switches, EMMA will instruct ONOS to turn them off.

Figure 1 highlights the main components, chain of events and interactions in the implemented system. The description of the set of actions taken by EMMA and the list of events happening in the network are detailed below (the labels associated with the arrows in the figure correspond to the numbers of the listed events/actions).

1) Whenever a switch receives a packet for which it does not have a forwarding rule, it sends the packet of the flow directly to ONOS. EMMA intercepts the packet using the *requestPacket* method of the *PacketService* interface of ONOS.
2) EMMA processes the packet and, if it carries unicast traffic, it computes a flow path according to Algorithm 1. Then it creates a forwarding rule using a set of ONOS applications. Initially, it uses *TrafficSelector.builder*, to form the part of the rule specifying the pair of source and destination IP addresses to be matched when pro-

cessing the packet at the switch. It then resorts to *TrafficTreatment.builder* in order to express a set of instructions, namely, forwarding the packet toward the intended output port and decrementing the packet TTL. Finally, it invokes *ForwardingObjective.Builder* to create the forwarding rule based on the *TrafficSelector* and *TrafficTreatment* outputs.

3) The ONOS application *FlowObjectiveService* is used to send the rule and install it in the switch.
4) A FLOW_ADDED event will be generated by the switch after the flow rule has been installed. Following the blueprint of the *FlowRuleListener* interface of ONOS, we built a *FlowListener* interface within EMMA. Such an interface allows EMMA to detect a FLOW_ADDED event, after which EMMA starts accounting for the power consumption due to the newly added flow.
5) A FLOW_REMOVED event will be generated by a switch whenever a flow finishes, or the application removes a flow from a switch when a better path is found. As for the FLOW_ADDED event, EMMA is able to detect a flow removal through the *FlowListener* interface we developed and, hence, to correctly compute the energy consumption of the active network. Note that when a FLOW_REMOVED event corresponds to a flow termination (i.e., it has not been triggered by an EMMA re-routing action), EMMA will check whether existing flows can be re-routed, according to Algorithm 2, thus performing a further step similar to step 2).

We remark that any change in the network topology or related to new/finishing traffic flows will be detected by the south bound ONOS providers, which will notify the Manager residing in the ONOS core about such events. Then EMMA will become aware of an event thanks to the TopologyListener and the FlowListener interfaces that we developed on top of ONOS.

We conclude by remarking that EMMA keeps track of the power usage in the network by computing the power at the FLOW_ADDED and FLOW_REMOVED events sent from nodes. In our implementation, EMMA computes the active network power consumption based on the packet size and the flow rate, as explained in Section II-A. Such values are obtained as the ratio of, respectively, the number of bytes to the number of packets, and the number of packets to the flow duration. The number of packets, number of bytes and flow duration are provided to EMMA by ONOS. Specifically, EMMA exploits the *getDevice* method of *DeviceService* in ONOS to get the list of available switches, and the *getFlowEntries* method of the *FlowRuleService* of ONOS to get information (i.e., number of bytes, number of packets and flow duration) about each flow handled by a given switch.

V. Experimental Results

We evaluated the EMMA performance against the optimal solution, as well as versus the simple case where no power saving strategy is adopted and the whole network is always active (hereinafter referred to as No Power Saving). The performance of EMMA and of the No Power Saving schemes are obtained by emulation, in the system we implemented and that is described above. The solution of the optimization problem in Eq. (II.4) is instead obtained using the Gurobi solver, considering the same network as that emulated in our experiments with Mininet.

TABLE II
Default settings

Parameter	Value
Flow arrival rate	0.1 flows/s
Average flow duration	20 s
Number of core switches	12
Number of edge switches	half the no. of core switches
Link Capacity	10 MB/s
Hysteresis (T_a)	10 s
P_{idle}	90 W [6]
E_{sw}	0.644 nW [6]
Number of hosts per edge switch	10
Link prob. b/w switches	0.5
Packet size	1500 bytes
Experiment duration	500 s

We derived the results assuming a default number of core and edge switches equal to 12 and 6, respectively; 10 hosts are connected to each edge switch. Links between any two core switches are set with probability 0.5 and the link capacity is set to 10 Mbytes/s. TCP traffic flows are generated using the Iperf tool, using 1500-byte packets. For each traffic flow, source and destination are selected at random among all possible hosts. Note that this is a worst case assumption for EMMA, while it favours the No Power Saving strategy. The inter-arrival time of newly generated flows follows a negative exponential distribution with a default mean arrival rate of 0.1 flows/s. The traffic flow duration is also exponentially distributed with mean equal to 20 s. The complete list of default values that we adopted for the system parameters is reported in Table II.

In the following figures, we show the average power consumption per flow, as the flow arrival rate and the number of network switches vary. The results have been obtained by averaging over 20 experiments. Note also that power consumption is computed based on traffic statistics and nodes operational states, consistently in all cases.

Figure 2 compares the performance of EMMA to the optimum as well as to that of the No Power Saving scheme, as the flow arrival rate varies and for the default number of core switches (namely, 12). Observe that EMMA matches the optimum very closely, for any value of flow arrival rate. The power saving it provides with respect to the case where all network switches are always on is very noticeable. Clearly, the power gain tends to shrink when many flows have to be allocated (high flow arrival rate), i.e., as an increasing number of switches and links have to be used. Also, we observed that the energy saved by turning off the nodes is significantly larger than that saved by turning off the links.

The behavior of EMMA compared to the No Power Saving scheme, as the network size varies, is presented in Figure 3. Here we do not show the optimum performance, as we

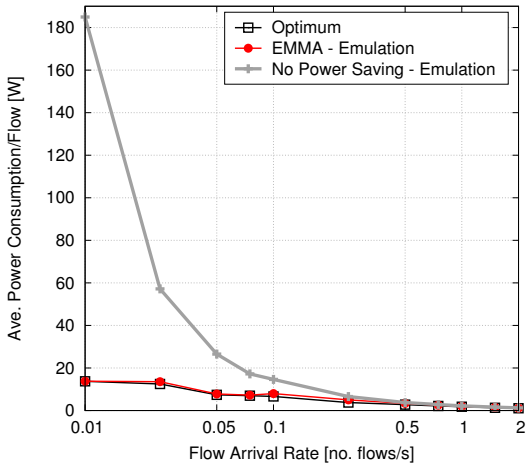

Fig. 2. Comparing EMMA against the optimum and the No Power Saving scheme: Average power consumption per flow as a function of the flow arrival rate (no. core switches = 12).

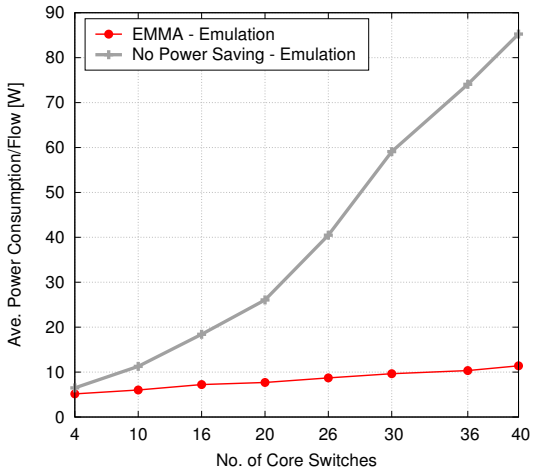

Fig. 3. Average power consumption per flow vs. number of core switches: comparison between EMMA and No Power Saving (flow arrival rate = 0.1)

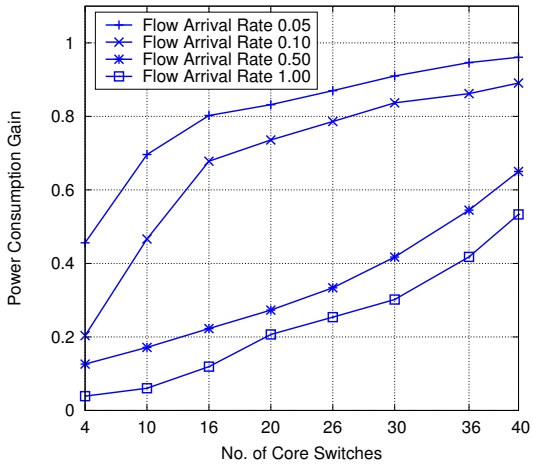

Fig. 4. Gain in average power consumption per flow (derived by emulation) provided by EMMA with respect to No Power Saving, as the number of core swicthes and the flow arrival rate vary.

could not solve the optimization problem for a number of core switches significantly larger than the default value. All results are therefore derived by emulation, under a flow arrival rate equal to 0.1. The plot confirms the excellent performance of EMMA: it reduces the power consumption per flow by a factor ranging from 2 (for 10 core switches) to 8 (for 40 core switches). As noted before, a smaller improvement is obtained only when the network size is small compared to the flow arrival rate (e.g., for 4-5 core switches).

Finally, Figure 4 depicts the gain that we can achieve with EMMA with respect to the No Power Saving strategy, as a function of the number of core switches and for a flow arrival rate equal to 0.05, 0.1, 0.5, 1. The gain is computed as the difference in power consumption between No Power Saving and EMMA, normalized to the power consumption of the former scheme. As expected, the gain that EMMA provides is higher for a lower value of flow arrival rate and a larger network size, since it is possible to aggregate more flows on the same links and there are more idle switches that can be turned off. Interestingly, the gain we obtain is always quite high, with peak values that approximate 1.

VI. Related Work

Energy-efficient traffic routing in wired networks has been largely addressed in the literature. Here, we limit our discussion to the studies that are most relevant to ours. In particular, at the end of the section we highlight our major contributions with respect to those that are closest to our study.

One of the first works to investigate energy-efficient management of nodes and links in backbone networks, can be found in [7]. The optimization problem they present to obtain an energy-efficient traffic allocation is an integer linear problem ILP, thus a heuristic is proposed too. Their algorithm first turns off nodes with the smallest traffic load and re-routes traffic consequently, then it tries to de-activate links. An opposite approach with respect to [7] is adopted in [12], [13], where the least congested links are turned off first.

In [9], both virtual machine (VM) placement and traffic flow routing are optimized so as to turn off as many unneeded network elements as possible. In particular, the authors use traffic-aware VM grouping to partition VMs into a set of VM-groups so that the overall inter-group traffic volume is minimized while the overall intra-group traffic volume is maximized. An approach based on the greedy bin-packing algorithm is proposed to route the traffic, and to put as many network elements as possible into sleep mode. A similar approach is adopted in [14], which focuses on the case where a sudden surge in traffic occurs after an off-peak period, during which most of the nodes have been turned off. The work tries to minimize the number of nodes/links that have to be activated and to reduce service disruption by avoiding turning off links that are critical to guaranteeing network connectivity.

Relevant to our work are also the studies in [8], [15], which minimize the power consumption of a data center network. In particular, [15] considers a hierarchical topology and proposes a hierarchical energy optimization technique: all edge switches

connecting to any source or destination server in the traffic matrix must be "on". Also, the network is divided into several pod-level subnetworks and a core-level subnetwork, and traffic is re-organized accordingly. [8] instead assumes that each traffic flow can be split over different paths. Interestingly, they have used OpenFlow to collect the flow matrix and port counters, which are used as input to their routing scheme. The work in [16] extends [8] by introducing a monitoring module that collects statistics, such as switch state, link state, active topology and traffic utilization.

Finally, physical characteristics of the links are accounted for in [17], [18]. The former considers network routers connected by multiple physical cables forming one logical bundled link, and it aims at turning off the cables within such links. The problem [17] poses accounts for the bundle size, besides the network topology and the traffic matrix, and it consists in maximizing the spare network capacity by minimizing the sum of loads over all links. Instead, in [18] the focus in on optical links and the minimum-energy traffic allocation is solved taking into account their peculiarities.

We remark that [7], [9] are the closest works to ours, nevertheless the study we present significantly differs from them. In particular, we recall that our problem formulation resembles that in [7], but it accounts for the instantaneous power consumption and for a more realistic model of the nodes power consumption, which changes the nature of the optimzation. As far as our heuristic is concerned, we leverage [9] but design an algorithm that, unlike [9], aims to find a better route for all existing flows whenever there is a change in the active topology. In addition, our focus is on the implementation of energy-efficient flow routing: our algorithms are implemented on ONOS, we define and implement the interactions that our application requires between ONOS and the network emulated through Mininet, and we derive emulation results by letting ONOS and Mininet interact.

VII. Conclusions

We addressed energy-efficient flow allocation in the 5G backhaul where traffic forwarding rules are created by an SDN controller, which can also turn switches on or off. We first formalized flow allocation by formulating an optimization problem whose complexity, however, results to be unbearable in large-scale scenarios. We therefore used a heuristic approach and developed an application, named EMMA. We implemented EMMA on top of ONOS and derived experimental results by emulating the network through Mininet. The comparison between EMMA and the optimal solution (obtained in a small-scale scenario) showed that the EMMA performance is very close to the optimum. Also, in larger scale scenarios, emulation results highlighted that EMMA can provide a dramatic energy improvement with respect to our benchmark where switches are always on.

Future research will address the problem of energy-efficient allocation and migration of virtual machines, and it will combine the solution with the EMMA scheme proposed in this paper for flow routing.

Acknowledgment

This work has received funding from the 5G-Crosshaul project (H2020-671598).

References

[1] The 5G Infrastructure Public Private Partnership, KPIs, https://5g-ppp.eu/kpis/.

[2] "5G Use cases and Requirements," *White Paper*, http://networks.nokia.com/file/28771/5g-white-paper.

[3] ONOS, http://onosproject.org

[4] A. Vishwanath, K. Hinton, R. Ayre, R. Tucker, "Modeling Energy consumption in High-Capacity Routers and Switches," *IEEE Journal on Selected Areas in Communications,* vol. 32, no. 8, 2014.

[5] P. Congdon, P. Mohapatra, M. Farrens, V. Akella, "Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches," *IEEE/ACM Transactions on Networking,* vol. 22, no. 3, 2014.

[6] Cisco Catalyst 3750 Series Switches Data Sheet, http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3750-series-switches/product_data_sheet0900aecd80371991.html.

[7] L. Chiaraviglio, M. Mellia, F. Neri, "Reducing Power Consumption in Backbone Networks," *IEEE ICC,* June 2009.

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

[9] W. Fang, X. Liang, S. Li, L. Chiaraviglio, N. Xiong, "VMPlanner: Optimizing Virtual Machine Placement and Traffic Flow Routing to Reduce Network Power Costs in Cloud Data Centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[10] A. Gyárfás, J. Lehel, "On-line and First-fit Colorings of Graphs," *Journal of Graph Theory*, vol. 12, no. 2, pp. 217-227, 1988.

[11] B. Lantz, B. Heller, N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," *ACM HotNets,* 2010, http://mininet.org.

[12] R. Carpa, O. Glück, L. Lefevre, "Segment Routing based Traffic Engineering for Energy Efficient Backbone Networks," *IEEE ICC*, 2014.

[13] F. Giroire, D. Mazauric, J. Moulierac, B. Onfroy, "Minimizing Routing Energy Consumption: From Theoretical to Practical Results," *IEEE/ACM ICC*, 2010.

[14] O. Okonor, N. Wang, Z. Sun, S. Georgoulas, "Link Sleeping and Wake-up Optimization for Energy Aware ISP Networks," *IEEE ISCC*, 2014.

[15] Y. Zhang, N. Ansari, "HERO: Hierarchical Energy Optimization for Data Center Networks," *IEEE Systems Journal*, vol. 9, no. 2, pp. 406–415, 2015.

[16] T. M. Nam, N. H. Thanh, N. Q. Thu, H. T. Hieu, S. Covaci, "Energy-Aware Routing based on Power Profile of Devices in Data Center Networks using SDN," *IEEE ICC*, 2015.

[17] W. Fisher, M. Suchara, J. Rexford, "Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links," *ACM SIGCOMM Workshop on Green Networking,* 2010.

[18] A. Coiro, M. Listanti, A. Valenti, F. Matera, "Reducing Power Consumption in Wavelength Routed Networks by Selective Switch Off of Optical Links," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 17, no. 2, pp. 428–436, 2011.