Optimizing Rule Placement in Software-Defined Networks for Energy-aware Routing

Frédéric Giroire
COATI team-project
INRIA and I3S (CNRS/ UNS), France
Email: frederic.giroire@cnrs.fr

Joanna Moulierac
COATI team-project
INRIA and I3S (CNRS/ UNS), France
Email: joanna.moulierac@unice.fr

Truong Khoa Phan
COATI team-project
INRIA and I3S (CNRS/ UNS), France
Email: truong_khoa.phan@inria.fr

***Abstract*—Software-defined Networks (SDN), in particular OpenFlow, is a new networking paradigm enabling innovation through network programmability. Over past few years, many applications have been built using SDN such as server load balancing, virtual-machine migration, traffic engineering and access control. In this paper, we focus on using SDN for energy-aware routing (EAR). Since traffic load has a small influence on power consumption of routers, EAR allows to put unused links into sleep mode to save energy. SDN can collect traffic matrix and then computes routing solutions satisfying QoS while being minimal in energy consumption. However, prior works on EAR have assumed that the table of OpenFlow switch can hold an infinite number of rules. In practice, this assumption does not hold since the flow table is implemented with Ternary Content Addressable Memory (TCAM) which is expensive and power-hungry. In this paper, we propose an optimization method to minimize energy consumption for a backbone network while respecting capacity constraints on links and rule space constraints on routers. In details, we present an exact formulation using Integer Linear Program (ILP) and introduce efficient greedy heuristic algorithm. Based on simulations, we show that using this smart rule space allocation, it is possible to save almost as much power consumption as the classical EAR approach.**

I. Introduction

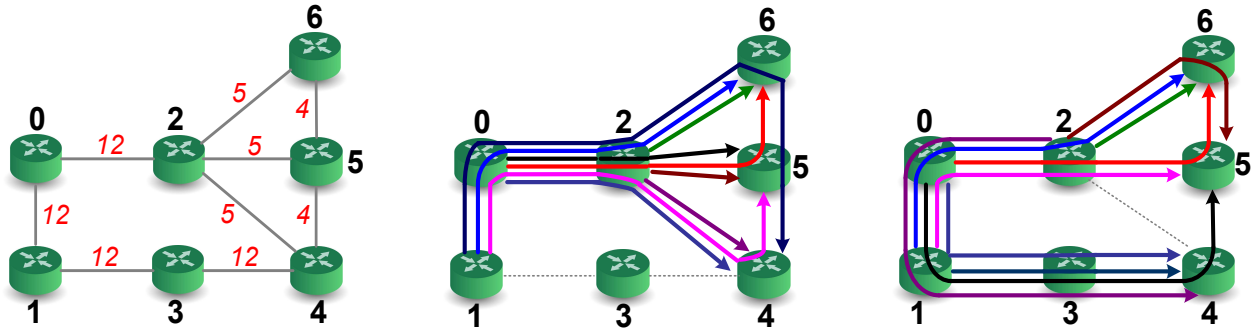
Recent studies have shown that ICT is responsible for 2% to 10% of the worldwide power consumption [6]. As estimation, energy requirement for European telecommunication can reach 35.8 TWh in 2020 [3]. Backbone ISP networks currently consume around 10% of the total network power requirements and it can increase to 40% by 2017 [15]. Therefore the green networking has been attracting a growing attention during the last years (see the survey [2]). While the traffic load has a marginal influence, the power consumption is mainly due to active elements on IP routers such as ports, line cards, base chassis, etc. [16]. Based on this observation, the energy-aware routing (EAR) approach aims at minimizing the number of used network elements while all the traffic demands are routed without any overloaded links [6][10]. In fact, turning off entire routers can earn significant energy savings. However, it is very difficult from a practical point of view as it takes time for turning on/off and also reduces life cycle of devices. Therefore, as in prior work [5][9], we assume to turn off (or put into sleep mode) only links to save energy.

Software-defined networking (SDN) in general, and OpenFlow in particular [17], has been attracting a growing attention in the networking research community in recent years. In traditional networks, network devices such as routers and switches act as "closed" systems. They work as "black boxes" with applications implemented on them. Users can only control them via limited and vendor-specific control interfaces. Moreover, since the data plane (forwarding function) and control plane are integrated, it is difficult for current network infrastructure to evolve (e.g. to deploy new network protocols). SDN is a new networking paradigm that decouples the control plane from the data plane. It provides a flexibility to develop and test new network protocols and policies in real networks. Over past few years, many applications have been built using the OpenFlow API [17].

In this paper, we focus on one application of the OpenFlow, that is to use OpenFlow to minimize power consumption for an Internet Service Provider (ISP). As shown in literature, many existing works have used OpenFlow as a traffic engineering approach to deploy EAR in a network [11][24]. In these works, the flow table of each switch is assumed to hold an infinite number of rules. In practice, however, this assumption does not hold, and rule space becomes a significant bottleneck for large-scale SDN networks. It is because the flow table is implemented using Ternary Content Addressable Memory (TCAM) which is expensive and power hungry. Therefore, commodity switches typically support just from few hundreds to few thousands of entries [13][14][22]. Taking this limitation into account, we show that the rule space constraints are very important in EAR. An inefficient rule allocation can lead to an unexpected routing solution, causing network congestion and affecting QoS. In summary, we make the following contributions:

- To our best knowledge, this is the first work that defines and formulates the optimizing rule space problem in OpenFlow for EAR using ILP.
- As EAR is known to be NP-hard [8], we propose heuristic algorithm that is effective for large network topologies. By simulations, we show that the heuristic algorithm achieves close-to-optimal solutions obtained by the ILP.
- Using real-life data traffic traces from SNDlib [20], we quantify energy savings achieved by our approaches. Moreover, we also present other QoS aspects such as routing length of EAR solutions.

The rest of this paper is structured as follows. We summarize related work in Section II. We present the ILP and heuristic algorithm in Section III. Simulation results are shown in Section IV. Finally, we conclude our work in Section V.



(a) Network and capacity on links (b) EAR - unlimited rule space (c) EAR - limited rule space

Fig. 1: Example of EAR with and without rule space constraints

II. Related Work

A. Limited Rule Space in OpenFlow Switches

To support a vast range of network applications, OpenFlow rules are more complex than forwarding rules in traditional IP routers. For instance, access-control requires matching on source - destination IP addresses, port numbers and protocol [4] whereas a load balancer may match only on source and destination IP prefixes [23]. These complicated matching can be well supported using TCAM since all rules can be read in parallel to identify the matching entries for each packet. However, as TCAM is expensive and extremely power-hungry, the on-chip TCAM size is typically limited. Many existing works in literature have tried to address this limited rule space problem. For instance, starting with OpenFlow v1.1 and beyond, rather than having one flow table, there is a pipeline made up of multiple flow tables. While this proposal can handle a large number of rules, it does add complexity to the standard [1]. The authors in [18] have proposed algorithms to reduce the number of rules needed to realize policies on a single switch. To the best of our knowledge, the closest papers to our work are [13] and [14]. These works present efficient heuristic rule-placement algorithms that distribute forwarding policies while managing rule-space constraints at each switch. However, they do not rely on the exact meaning of the rules and the rules should not determine the routing of the packet (e.g. access control rules) [14]. Therefore, these techniques cannot directly solve the rule-placement problem in EAR. In this work, we focus on EAR with OpenFlow where rules explicitly determine routing of traffic flows. Besides an exact formulation ILP, we also propose an efficient heuristic algorithm for large networks.

B. Energy Savings with OpenFlow

Starting from the pioneering work of Gupta [10], the idea of power proportionality has gained a growing attention in networking research area [2][6]. Since power consumption of router is independent from traffic load, people suggested putting network components to sleep in order to save energy. OpenFlow is a promising method to implement EAR in a network. Without setting entries manually, OpenFlow can collect traffic matrix, performs routing calculation and then installs new routing rules on routers. For instance, the authors in [11] have implemented and analyzed ElasticTree on a prototype testbed built with production OpenFlow switches. The idea is to use OpenFlow to control traffic flows so that it minimizes the number of used network elements to save energy. Similarly, the authors in [24] have set up a small testbed using OpenFlow switches to evaluate energy savings for their model. OpenFlow switches have also been mentioned in existing work as an example of the traffic engineering method to implement the EAR idea [7]. However, as we can see, the testbed setups with real OpenFlow switches are quite small. For instance, in [11], 45 virtual switches onto two 144-port 5406 chassis switches are used; or in [24], there is a testbed with 10 virtual switches on a 48-port Pronto 3240 OpenFlow-enabled switch. We argue that when deploying EAR in real network topologies, much more real OpenFlow switches should be used and they have to handle a large amount of traffic flows. In this situation, limited rule space in switches becomes a serious problem since we can not route traffic as expected. Therefore, we present in next Section a novel optimization method to overcome the rule placement problem of OpenFlow for EAR.

III. Optimizing Rule Placement

Routing decision of an OpenFlow switch is based on flow tables implemented with TCAM. Each entry in the flow table defines a matching rule and is associated with an action. Upon receiving a packet, a switch identifies the highest-priority rule with a matching predicate, and performs the corresponding action. A packet that matches no rule is processed using the *default rule*, which has the lowest-priority. Depending on applications of OpenFlow, the default rule can be "drop packets" or "forward packets to the controller" over the OpenFlow channel. In this work, to avoid delay communication between routers and the centralized controller, we consider that the *default rule* is "forward packets to a default port" (without contacting the controller), and each switch has exactly one default port [19].

TABLE I: Traffic demands and routing solutions

Traffic demand	Volume	Routing solution (Fig. 1b)	Routing solution (Fig. 1c)
(0, 4)	1	0 - 2 - 4	0 - 1 - 3 - 4
(0, 5)	2	0 - 2 - 5	0 - 1 - 3 - 4 - 5
(0, 6)	2	0 - 2 - 5 - 6	0 - 2 - 5 - 6
(1, 4)	1	1 - 0 - 2 - 6 - 5 - 4	1 - 3 - 4
(1, 5)	3	1 - 0 - 2 - 4 - 5	1 - 0 - 2 - 5
(1, 6)	3	1 - 0 - 2 - 6	1 - 0 - 2 - 6
(2, 4)	1	2 - 4	2 - 0 - 1 - 3 - 4
(2, 5)	1	2 - 5	2 - 6 - 5
(2, 6)	1	2 - 6	2 - 6

Rule	Action
(0, 4)	*Port-4*
(0, 5)	*Port-5*
(0, 6)	*Port-5*
(1, 4)	*Port-6*
(1, 5)	*Port-4*
(1, 6)	*Port-6*
(2, 4)	*Port-4*
(2, 5)	*Port-5*
(2, 6)	*Port-6*

(a) Simple routing table

Rule	Action
(0, 5)	*Port-5*
(0, 6)	*Port-5*
(1, 4)	*Port-6*
(1, 6)	*Port-6*
(2, 5)	*Port-5*
(2, 6)	*Port-6*
Default	*Port-4*

(b) With default rule

Rule	Action
(1, 5)	*Port-4*
(1, *)	*Port-6*
(*, 4)	*Port-4*
(2, 6)	*Port-6*
Default	*Port-5*

(c) With default rule and wildcards

Fig. 2: Routing table at router 2 for routing of Fig. 1b

We show in Fig. 1 how the limited rule space impacts EAR solution. Assume that there are 9 traffic demands with volumes as shown in Table I. The network topology and capacity on links are shown in Fig. 1a. For ease of reading, Table I also shows the routing of each traffic flow in Fig. 1b and Fig. 1c. These routing solutions are found by using the ILP in Section III.A. As the classical EAR approach, Fig. 1b shows an optimal solution since it satisfies capacity constraints and uses a minimum number of active links (7 links). It is noted that, as the objective of EAR is to minimize the number of used links, some traffic flows may be routed via long paths. For instance, the flow (1, 4) is routed via 5 hops while its shortest path is only 2 hops. One possible way to avoid this is to have some constraints that limit the stretch of the path of each flow.

Assume that the routing table of router contains rules which are the mapping of [(src, dest) : port-to-foward]. As the routing in Fig. 1b, the router 2 needs to forward 9 flows, hence a simple routing table can be as Fig. 2a. However, we can reduce the size of the routing table by using a default rule (Fig. 2b), or combining default rule and wildcards (Fig. 2c). Note that the rules [(0, 5): port-5] and [(0, 6): port-5] of Fig. 2b can not be combined as [(0, *): port-5]. Indeed, in this case, the flow $(0, 4)$ will go to port-5 when it should go to port-4. In Fig. 2c, as the rule $(1, 5)$ has higher priority than the rule (1, *), the flow $(1, 5)$ is forwarded to port-4 while the flows $(1, 4)$ and $(1, 6)$ are forwarded to port-6 with the rule (1, *). Assume that we implement EAR on SDN network where each router can install at most 4 rules. As a result, the router 2 can install only 3 distinct rules and 1 default rule. However, as we have shown, the minimum routing table contains 5 rules (Fig. 2c). Therefore, in this situation, some flows need to be routed using the default port. For instance, if the flow $(2, 6)$ in Fig. 2c goes to the default port-5, then the link $(2, 5)$ will be overloaded. It is also easy to check that, when the rule capacity is equal to 4 and with a set of active links as in Fig. 1b, it is not possible to find a routing solution that satisfies both capacity constraints on links and rule space constraints on routers. However, if we consider the rule space constraints as inputs of the problem, we can find a feasible solution as Fig. 1c. Actually, since we add more constraints, the EAR with rule space is able to save less energy with respect to the classical EAR. For instance, there is only 1 inactive link in Fig. 1c while we can turn off 2 links and save more energy in Fig. 1b.

As we have shown in this example, the limited rule space is very important in EAR. Inefficient rule placement can cause unexpected routing solution, and hence result in network congestion. To overcome this problem, we present in this section a precise formulation (Integer Linear Program) and heuristic approach for large networks. However, we note that the current algorithms can find optimal solution of energy consumption (or close-to-optimal if it is heuristic) if we consider the default rule but not the wild-card. If the rule space is scarce, we can apply the work of "compressing policy on a single switch" [18] as a post-processing step to further reduce the routing table size.

A. Integer Linear Program

We consider a backbone network as an undirected graph $G = (V, E)$. The nodes in V describe routers and the edges in E present connections between those routers. We denote by D^{st} the demand of traffic flow from node s to node t such that $D^{st} \geq 0, s, t \in V, s \neq t$. We assume that the capacity of links and the rule space at routers are constant. The objective is to find a feasible routing for all traffic flows, respecting the capacity and the rule space constraints and being minimal in energy consumption.

We first define the following notations and then formulate the problem as Integer Linear Program:

- $\mathcal{D}$: a set of all traffic demands to be routed.
- $D^{st} \in \mathcal{D}$: demand of the traffic flow from s to t.
- C_{uv}: capacity of a link (u, v).
- $\mu \in (0, 1]$: maximum link utilization that can be tolerated. It is normally set to a small value, e.g. $\mu = 0.5$.
- C_u: maximum number of rules can be installed at router u.
- $N(u)$: the set of neighbors of u in the graph G.
- x_{uv}: binary variable to indicate if the link (u, v) is active or not.
- f^{st}_{uv}: a flow (s, t) that is routed on the link (u, v) by a distinct rule. We call f^{st}_{uv} as normal flow.
- g^{st}_{uv}: a flow (s, t) that is routed on the link (u, v) by a default rule. g^{st}_{uv} is called default flow to distinguish from the normal flow f^{st}_{uv} .
- k_{uv}: binary variable to indicate if the default port of the router u is to go to v or not.

$$\min \sum_{(u,v)\in E} x_{uv} \tag{1}$$

$$\text{s.t.} \sum_{v\in N(u)} \left(f^{st}_{vu} + g^{st}_{vu} - g^{st}_{uv} - f^{st}_{uv}\right) = \begin{cases} -1 & \text{if } u = s, \\ 1 & \text{if } u = t, \\ 0 & \text{else} \end{cases} \quad \forall u \in V, (s, t) \in \mathcal{D} \tag{2}$$

$$f_{uv}^{st} + f_{vu}^{st} + g_{uv}^{st} + g_{vu}^{st} \leq 1 \; \forall (u,v) \in E, (s,t) \in \mathcal{D} \quad (3)$$

$$\sum_{(s,t)\in\mathcal{D}} D^{st}(f_{uv}^{st} + f_{vu}^{st} + g_{uv}^{st} + g_{vu}^{st}) \leq \mu C_{uv} x_{uv} \quad \forall (u,v) \in E \quad (4)$$

$$\sum_{(s,t)\in\mathcal{D}} \sum_{v\in N(u)} f_{uv}^{st} \leq C_u - 1 \quad \forall u \in V \quad (5)$$

$$\sum_{v\in N(u)} k_{uv} \leq 1 \quad \forall u \in V \quad (6)$$

$$g_{uv}^{st} \leq k_{uv} \quad \forall (u,v) \in E, (s,t) \in \mathcal{D} \quad (7)$$

$$x_{uv}, f_{uv}^{st}, g_{uv}^{st}, k_{uv} \in \{0,1\} \quad \forall (u,v) \in E, (s,t) \in \mathcal{D} \quad (8)$$

The objective function (1) minimizes the power consumption of the active links. The flow conservation constraints (2) express that the total flows entering and leaving a router are equal (except the source and the destination nodes). It is noted that a normal flow entering a router can become a default flow on outgoing link and vice versa. Constraints (3) ensure that a flow (s,t) on a link (u,v) cannot be both normal (f_{uv}^{st}) and default flow (g_{uv}^{st}) at the same time. Constraints (4) are capacity constraints. We consider an undirected link capacity model [21] in which the capacity of a link is shared between the traffic in both directions. This model is not common but it allows to reduce the number of variables without impacting the general idea of our proposal. Constraints (5) denote rule capacity constraints where we reserve one rule at each router to be the default rule. Constraints (6) and (7) are used to fix only one default port for each router.

B. Heuristic Algorithm

Since energy-aware routing problem is known to be NP-Hard [8], it is very challenging to find an exact solution. Therefore, we present in this section an efficient greedy heuristic for large networks. In summary, the heuristic algorithm works through two steps:

- Step 1: starting from the whole network, we compute a feasible routing which respects the capacity and the rule space constraints as described in *Algorithm 1*, *2* and *3*. For each router $u \in V$, we keep the two sets $\mathcal{F}_u$ and $\mathcal{G}_u$ containing normal and default flows, respectively. At the beginning of the algorithm, we are freely to assign distinct rules for flows (line 10 - *Algorithm 1*) until the routing table is full ($|\mathcal{F}_u| = C_u$). Then, we try to shrink it (line 9 - *Algorithm 1*) as Fig. 3 by setting the port that carries most of traffic flows as the default port. As a result, the number of installed rules is reduced and there is some more space to install new rules.

Rule	Action
(0, 2)	*Port-3*
(0, 3)	*Port-3*
(0, 4)	Port-2
(0, 5)	*Port-3*

Before shrinking

Shrink table →

Rule	Action
(0, 4)	Port-2
Default	Port-3

After shrinking

Fig. 3: Routing table at a router

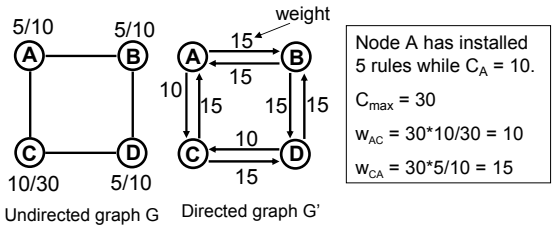


Fig. 4: Example of updating link weight

Algorithm 1: Finding a feasible routing
Input: An undirected graph $G = (V, E)$, link capacity $C_e \; \forall e \in E$, rule space capacity $C_u \; \forall u \in V$ and a set of demands $\mathcal{D}$.
Ouput: routing solution on graph G.

1 Residual capacity $R_e = C_e \; \forall e \in E$;
2 Initially, $\mathcal{F}_u = \emptyset$ and $\mathcal{G}_u = \emptyset \; \forall u \in V$;
3 Creating directed graph $G' = (V, E')$ from G where $\forall (u,v) \in E$, we add both directions (u,v) and (v,u) to E' (Fig. 4). Initial weight on link $w_e = 1 \; \forall e \in E'$;
4 **while** $D^{st} \in \mathcal{D}$ *has no assigned route* **do**
5 find the shortest path P^{st} on G' such that $R_e \geq D^{st} \; \forall e \in P^{st}$;
6 assign the routing P^{st} to the demand D^{st};
7 update $R_e := R_e - D^{st} \; \forall e \in P^{st}$;
8 update link weights proportionally to the size $|\mathcal{F}_u|$ as *Algorithm 2*;
9 if $|\mathcal{F}_u| == C_u$, shrink table at $u \; \forall u \in P^{st}$;
10 update $\mathcal{F}_u$ and $\mathcal{G}_u \; \forall u \in P^{st}$ using *Algorithm 3*;
11 **end**
12 return the routing (if it exists) assigned for $\mathcal{D}$

- Step 2: remove in priority links that are less loaded (*Algorithm 4*). The aim of this step is to turn off the low loaded links and to accommodate their traffic on other links in order to reduce the total number of active links.

An example of computing link weights (*Algorithm 2*) is shown in Fig. 4. The link weight is used to perform rule-balancing between routers. Assume that the next demand is (A, D), then the routing solution using shortest path is (A, C, D). It is better than (A, B, D) since node C still has much more available rule space.

Algorithm 2: Updating link weight
Input: An undirected graph $G = (V, E)$, a set of normal flows $\mathcal{F}_u \; \forall u \in V$ and a maximum value of rule space capacity $C_{max} = max(C_u) \; \forall u \in V$.
Ouput: weight setting on links of G'.

1 Create a digraph $G' = (V, E')$ as Fig. 4.
2 **for** $(u,v) \; in \; E'$ **do**
3 compute rule utilization at v: $U_v = C_{max} \times |\mathcal{F}_v|/C_v$;
4 update $w_{uv} = max(U_v, 1)$;
5 **end**

Algorithm 3: Updating $\mathcal{F}_u$ and $\mathcal{G}_u$
Input: An undirected graph $G = (V, E)$, the shortest path P^{st} found in Algorithm 1, the set $\mathcal{F}_u$ and $\mathcal{G}_u$ $\forall u \in P^{st}$ and the default port of each router $d(u)$ $\forall u \in P^{st}$.
Ouput: Updated sets of $\mathcal{F}_u$ and $\mathcal{G}_u$ $\forall u \in V$.

for $u \in P^{st}$ **do**
  for $v \in G.neighbor(u)$ **do**
    if $(u, v) \in P^{st}$ *and* $v == d(u)$ **then**
      $\mathcal{G}_u = \mathcal{G}_u \cup g^{st}_{uv}$
    else if $(u, v) \in P^{st}$ *and* $v \neq d(u)$ **then**
      $\mathcal{F}_u = \mathcal{F}_u \cup f^{st}_{uv}$
    end
  end
end

Algorithm 4: Removing less loaded links
Input: An undirected graph $G = (V, E)$, link capacity C_e and residual capacity R_e $\forall e \in E$.
Ouput: routing solution on a set of active links.

while *edges can be removed* **do**
  remove the edge e that has not been chosen and has smallest value C_e/R_e;
  compute a feasible routing with the Algorithm 1;
  if no feasible routing exists, put e back to G;
end
return the feasible routing if it exists.

IV. Simulation Results

We solved the ILP model with IBM CPLEX 12.4 [12]. All computations were carried out on a computer equipped with 2.7 Ghz Intel Core i7 and 8 GB RAM. We consider real-life traffic traces collected from SNDlib [20]. To compare the optimal and the heuristic solutions, we use a small network as Atlanta network ($V = 15, E = 22, |\mathcal{D}| = 210$). As mentioned in [13][14], the routing table can support from 750 to few thousands of rules. To show that this is a realistic problem, we use three of the largest network topologies in SNDlib: ta2 (Telekom Austria: $V = 65, E = 108, |\mathcal{D}| = 4160$), zib54 (Zuse-Institut Berlin: $V = 54, E = 81, |\mathcal{D}| = 2862$) and germany50 ($V = 50, E = 88, |\mathcal{D}| = 2450$). In our test instances, five traffic matrices (D1 - D5) are used to represent daily traffic pattern (Fig. 5). Since traffic load is low, the traffic matrix obtained from SNDlib is considered as $D1$. To achieve a network with high link utilization, we scale $D1$ with a factor of 1.5, 2.0, 2.5 and 3.0, and they form D2 - D5, respectively. Note that we can play with finer granularity traffic matrices by changing these factors.

A. Optimal vs. Heuristic Solutions

Assume that each router on the network has the same rule capacity represented by $C_u = (p \times |\mathcal{D}|)$ where $p \in (0, 1]$ and $|\mathcal{D}|$ is the total demands. The value of C_u is varied as we change the parameter p. Energy savings is computed as the number of links to sleep divided by the total number of links of the network ($|E|$).

As shown in Table 1 and Table 2, the ILP model can find solution for very limited rule space ($p = 5\%$) while the heuristic algorithm needs $p = 16\%$ (if p is less than this value, no feasible solution can be found). Similarly, the heuristic algorithm is able to save less energy when p is small (e.g. $p \leq 16\%$). However, when the rule space capacity is large enough, e.g. $p = 30\%$, the gaps between the optimal and the heuristic solutions are small. Moreover, the heuristic algorithm is better in computation time. For instance, the ILP model can take up to 9000 (s) to find solution while it is always less than 10 (s) for the heuristic algorithm.

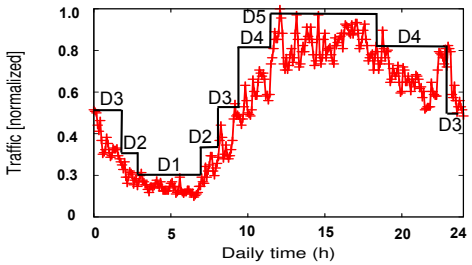


Fig. 5: Daily traffic in multi-period

TABLE II: Atlanta network (optimal solution)

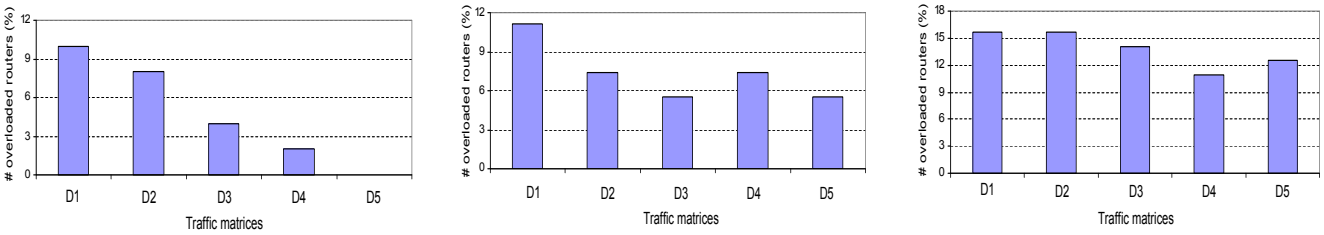
Rule capacity (p - %)	energy savings (%)	computation time (s)
5%	9	2200
10%	22.7	9000
20%	22.7	445
30%	22.7	540
40%	22.7	300
100%	22.7	400

TABLE III: Atlanta network (heuristic solution)

Rule capacity (p - %)	energy savings (%)	computation time (s)
16%	4.5	< 10
20%	13.6	< 10
30%	18.2	< 10
40%	18.2	< 10
100%	18.2	< 10

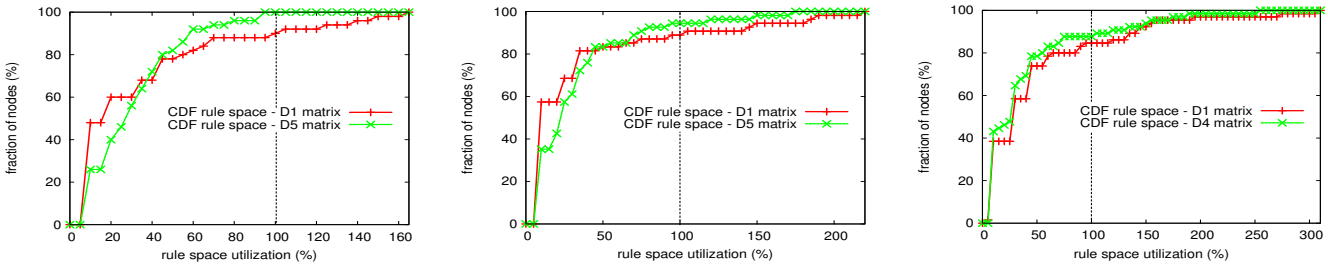
B. Heuristic Solutions for Large Networks

1) Rule allocation at routers: We assume that all the routers have the same rule space capacity $C_u = 750$ [14]. As the EAR approach, we use the heuristic algorithm in [8] to represent the case where there is no limit of rule space. We run this algorithm on germany50, zib54 and ta2 networks to see if the rule space constraints are violated or not. As shown in Fig. 6a, Fig. 6b and Fig. 6c, most of the cases (except $D5$ of germany50 network), there are routers that use more than 750 rules in their routing tables. In zib54 network (resp. ta2 network), from 6% to 11% (resp. 11% to 16%) of routers exceed their rule space capacities. In germany50 network, with the $D5$ traffic matrix, there is no router that uses more than 750 rules. For other traffic matrices, from 2% to 10% of the routers are overloaded in rule space (Fig. 6a). Therefore, the limited rule space is really a problem in real networks. This problem is extremely important since we cannot route traffic as expected. The number of routers overloaded in rule space depends on the traffic matrix. However, an accurate explanation is difficult (e.g. less overloaded routers when traffic load is high) since the algorithm in [8] does not care at all about the rule capacity.



(a) Germany50 network (b) Zib54 network (c) Ta2 network

Fig. 6: Number of overloaded routers in the three networks with unlimited rule-space algorithm



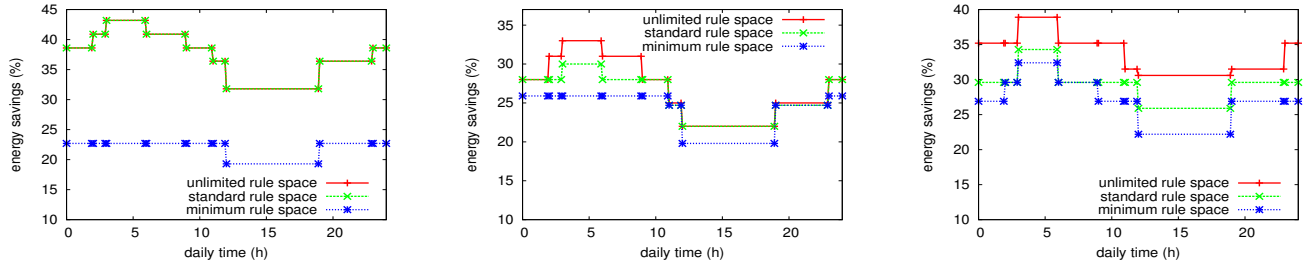
(a) Germany50 network (b) Zib54 network (c) Ta2 network

Fig. 7: CDF rule space utilization in the three networks with unlimited rule-space algorithm

To take a closer look at rule space allocation, we draw cumulative distribution function (CDF) for rule utilization at each router (Fig. 7a, Fig. 7b and Fig. 7c). When a rule space utilization is larger than 100%, it means that the router has used more than 750 rules. Based on Fig. 6a, Fig. 6b and Fig. 6c, we can find the traffic matrices that cause the maximum and the minimum number of routers violating their rule space constraints. We call these traffic matrices as the max-over-rule and the min-over-rule, respectively. For instance, in case of germany50 network, $D1$ is the max-over-rule and $D5$ is the min-over-rule. For each network, we draw two CDFs of rule utilization for the max-over-rule and the min-over-rule cases. As shown in Fig. 7a, Fig. 7b and Fig. 7c, the CDF of rule space allocation of the two cases are quite similar. However we can see in the max-over-rule case, more fractions of routers are overloaded. For instance, in Fig. 7b, only 89% of routers are less than 100% rule space utilization in the max-over-rule case (D1), meanwhile it is 94% of routers in the min-over-rule case (D5). In general, the larger the network is, the more rule space is needed at routers. For example, the maximum rule utilization of germany50, zib54 and ta2 networks are 165%, 220% and 310%, respectively.

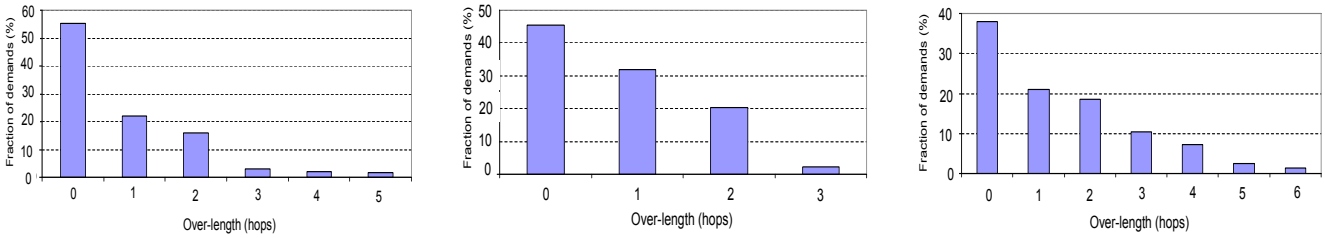
2) Energy savings: We collect energy savings for each network in three cases: the minimum, the standard and the unlimited rule spaces. In the standard case, each router can install at most 750 rules ($C_u = 750$). To find the minimum rule space, we reduce the value of C_u until we get a minimum value of C_u for which it is possible to find a feasible solution. The minimum values of C_u for germany50, zib54 and ta2 networks are 227, 670 and 695, respectively. The unlimited rule space case is equivalent to the classical EAR model in which we do not consider at all rule space constraints at routers. In general, as shown in Fig. 8a, Fig. 8b and Fig. 8c, the larger the rule space at routers is, the more flexible routing solutions we have and more energy can be saved for the network. The energy savings gap between the standard and the unlimited rule space cases is small. In some traffic matrices, both cases offer the same amount of energy savings. For instance in germany50 network, the standard and unlimited cases offer the same amount of energy savings. It means that if we use a smart rule allocation, it is possible to achieve equivalent energy savings as the classical EAR approach. The maximum energy savings gaps of the standard and the unlimited cases are 3% and 5.6% for zib54 and ta2 networks, respectively. As expected, less energy is saved in the minimum rule space case as we do not have enough installed rules to route traffic in a better way. As shown in Fig. 8a, there is a big gap of the energy savings between the minimum and the standard cases. This can be explained as the difference between the minimum and the standard values of C_u in germany50 network is large.

3) Route length: Intuitively, EAR would affect the length of routing flows as we redirect traffic flows to minimize the number of active links. In Fig. 9a, Fig. 9b and Fig. 9c, we evaluate the impact of EAR on routing length with respect to the shortest path routing. For each traffic demand, we collect the length of its routing flow and the corresponding shortest path. We use the notation *over-length* to denote the difference (in number of hops) between the length of the routing solution and the shortest path. When *over-length* is equal to 0, it means that the routing solution is exactly the shortest path. As shown in Fig. 9a, Fig. 9b and Fig. 9c, a large fraction of the demands follow their shortest paths. Indeed, in the heuristic algorithm, we use the shortest path to find routing solution. In germany50 and zib54, the maximum number of additional hops for a demand is 3 and 5 hops, respectively. The ta2 network is larger, up to 6 hops can be added to a demand, but it happens only for 1.4% of the demands. However, if latency is important, especially for sensitive delay applications such as voice or video streaming, we can add constraints to limit the route length so that it will not excess a predefined threshold value.



(a) Germany50 network (b) Zib54 network (c) Ta2 network

Fig. 8: Energy savings in daily time in the three networks



(a) Germany50 network (b) Zib54 network (c) Ta2 network

Fig. 9: Over-length of demands in the three networks

V. Conclusions

To our best knowledge, this is the first work considering rule space constraints of OpenFlow switch in energy-aware routing. We argue that, in addition to capacity constraint, the rule space is also important as it can change the routing solution and affects QoS. Based on simulations with real traffic traces, we show that our smart rule allocation can achieve high energy efficiency for a backbone network while respecting both the capacity and the rule space constraints.

References

[1] OpenFlow Switch Specication version 1.1, 2011 archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

[2] A. P. Bianzino, C. Chaudet, D. Rossi, and J. Rougier. "A Survey of Green Networking Research". In *IEEE Communication Surveys and Tutorials*, volume 14, pages 3 – 20, 2012.

[3] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh. "The Potential Impact of Green Technologies in Next-generation Wireline Networks: Is There Room for Energy Saving Optimization?". In *IEEE Communications Magazine*, 2011.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker. "Rethinking Enterprise Network Control". In *IEEE/ACM Transaction in Networking*, 2009.

[5] L. Chiaraviglio, A. Cianfrani, E. L. Rouzic, and M. Polverini. "Sleep Modes Effectiveness in Backbone Networks with Limited Configurations". In *Computer Networks*, pages 2931 – 2948, 2013.

[6] L. Chiaraviglio, M. Mellia, and F. Neri. "Minimizing ISP Network Energy Cost: Formulation and Solutions". In *IEEE/ACM Transaction in Networking*, pages 463 – 476, 2011.

[7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, and P. Yalagandula. "DevoFlow: Scaling Flow Management for High-Performance Networks". In *ACM SIGCOMM*, 2011.

[8] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. "Minimizing Routing Energy Consumption: from Theoretical to Practical Results". In *IEEE/ACM GreenCom*, 2010.

[9] F. Giroire, J. Moulierac, T. K. Phan, and F. Roudaut. "Minimization of Network Power Consumption with Redundancy Elimination". In *IFIP NETWORKING*, 2012.

[10] M. Gupta and S. Singh. "Greening of The Internet". In *ACM SIGCOMM*, 2003.

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

[12] IBM ILOG. CPLEX Optimization Studio 12.4.

[13] N. Kang, Z. Liu, J. Rexford, and D. Walker. "Optimizing the "One Big Switch" Abstraction in Software-Defined Networks". In *ACM CoNEXT*, 2013.

[14] Y. Kanizo, D. Hay, and I. Keslassy. "Palette: Distributing Tables in Software-defined Networks". In *INFOCOM Mini-conference*, 2013.

[15] C. Lange. "Energy-related aspects in backbone networks". In *35th European Conference on Optical Communication*, 2009.

[16] P. Mahadevan, P. Sharma, and S. Banerjee. "A Power Benchmarking Framework for Network Devices". In *IFIP NETWORKING*, 2009.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "Openflow: Enabling Innovation in Campus Networks". In *ACM CCR*, pages 69 – 74, 2008.

[18] C. R. Meiners, A. X. Liu, and E. Torng. "Bit Weaving: A Non-prefix Approach to Compressing Packet Classifiers in TCAMs". In *IEEE/ACM Transaction in Networking*, pages 488 – 500, 2012.

[19] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti. "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks". In *IEEE Communications Surveys and Tutorials*, 2013 (accepted).

[20] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. SNDlib 1.0 - survivable network design library. *Networks*, 55(3):276–286, 2010.

[21] C. Raack, A. M. C. A. Koster, S. Orlowski, and R. Wessäly. "On Cut-based Inequalities for Capacitated Network Design Polyhedra". In *Networks*, volume 57, pages 141 – 156, 2011.

[22] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. "PAST: Scalable Ethernet for Data Centers". In *ACM CoNEXT*, 2012.

[23] R. Wang, D. Butnariu, and J. Rexford. "OpenFlow-based Server Load Balancing Gone Wild". In *Hot-ICE*, 2011.

[24] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. "CARPO: Correlation-Aware Power Optimization in Data Center Networks". In *INFOCOM*, 2012.