A Greedy Power-aware Routing Algorithm for Software-defined Networks

Mohamad Khattar Awad[†], Yousef Rafique[†], Sarah Alhadlaq[‡], Dunya Hassoun[§], Asmaa Alabdulhadi[§§], Sheikha Thani[◇]

[†] Computer Engineering Department, College of Computing Sciences and Engineering, Kuwait University

[‡] Ministry of Electricity and Water, Information Technology Department, Kuwait

[§] Australian College of Kuwait, Information Technology Department, Kuwait

[§§] Ministry of Higher Education, Information Technology Department, Kuwait

[◇] The Public Authority for Civil Information, Information Technology Department, Kuwait

E-mail: mohamad@ieee.org, yousef.rafique.kw@ieee.org, Sarah.S.Alhadlaq@eng.kuniv.edu.kw, d.hassoun@ack.edu.kw, aalabdulhadi@mohe.edu.kw, sheika@paci.gov.kw

***Abstract*—We consider the problem of minimizing the routing power consumption in software-defined networks. The network is composed of software-defined networking (SDN) nodes and a central controller where routing decisions are centralized. More specifically, the central controller minimizes the routing power consumption by routing flows on the minimum number of active links with the lowest discrete link rates. Thus, it maximizes the number of inactive links and the level of link rates, which reflects significant saving in power consumption. This problem is a mixed-integer programming problem and known to be NP-hard. Therefore, we propose a low-complexity greedy heuristic to minimize the number of active links and link rates by rerouting flows and aggregating them on common links. Numerical results show that the proposed algorithm achieves 17.18% to 32.97% power saving in real network topologies relative to a base shortest path algorithm. The savings are achieved with minimal increase in average path length that is less than 0.2 hops.**

I. Introduction

The explosive demand for diverse Internet applications and services have urged service providers to expand their network coverage and capacity. This has increased the operational cost and energy consumption. Recent studies have shown that 2% - 7% of global electricity consumption is consumed by Internet and Communications Technology (ICT) devices and services [1], [2]. These devices and services fall into three categories; data centers, personal computers, and communication networks, with communication networks being the most energy consuming category among the three [1], [2]. Moreover, studies have demonstrated that ICT embodies a rapid continuous growth in power consumption worldwide that rises about 3% annually [2]. These figures are expected to continue rising as the number of users increases.

In response to these monumental figures, many researchers have developed and are still exploring methods for reducing the power consumption of communication networks. Introducing new possibilities, the programmable interface of software-defined networking (SDN) has gained popularity among network administrators and operators allowing them to easily meet their network requirements and optimize its performance. In traditional networks, the control and data planes are coupled within the same networking device, which impedes innovation in creating new network services or improving existing ones. The static architectures of traditional networks make automatic response to events such as traffic shifts impossible [3], [4]. On the other hand, SDN shifts the intelligence to a central controller (CC) by decoupling the control plane and the data plane; therefore, forwarding devices become simple forwarding elements managed by a CC. SDN allows network administrators to add new protocols, manage the network easily, and control the forwarding elements using a single point of programming located at a logically centralized CC.

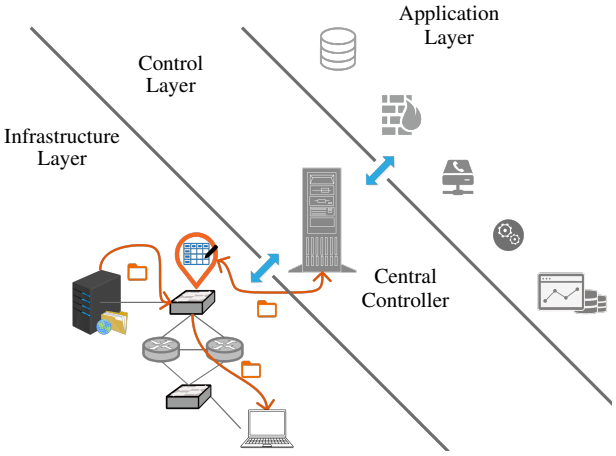


Figure 1. Illustration of a software-defined network and its layers.

The Open Networking Foundation (ONF), a nonprofit organization that promotes the use of SDN, proposed a paradigm for SDN that is composed of a three-layer architecture as illustrated in Figure 1. These layers include the infrastructure layer, the control layer, and the application layer. The infrastructure layer, also known as the data plane, is the lowest layer of the SDN architecture. It is comprised of the forwarding hardware accompanied with all of the hardware elements and software interfaces required. Unlike traditional static network devices,

these devices transform into simple packet forwarding devices as the embedded intelligence or control is removed. The control layer is the core of SDN that links the infrastructure with the application layer by using Application Programming Interfaces (APIs). The application layer is the top layer that consists of network programmable applications and services, such as routing, load balancing, access control etc. The control layer collects performance metrics from the infrastructure layer and relays it to the application layer. In addition, it receives network management decisions from the application layer and implements them at the infrastructure layer. The communication between the control layer and infrastructure layers is facilitated by ONF API called OpenFlow. Innovative network features, network configurations, and different forwarding schemes can be developed at the application layer to optimize the network performance [5], [6]. In SDN, the CC receives routing requests and computes the routing paths based on the application layer routing algorithm. Then CC installs these paths as forwarding rules at their corresponding forwarding elements as demonstrated in Figure 1. The focus of this work is on optimizing the routing application of SDN for power efficiency.

Several recent studies have addressed the issue of power efficiency in data centers and communication networks. In [7], authors proposed an energy-aware approach for flow scheduling using Exclusive Routing (EXR) in data centers. The SDN controller manages traffic loads for the flows, where traffic is distributed over different paths and thus ensures that each link is used by at most one flow at a time. EXR does not require bandwidth allocation, nor time synchronization between servers. It also allows the operator to decide the priority of each flow depending on its own constraint to save more energy. Rahnamay-naeini et al. [8] presented a heuristic to switch off unnecessary links during low demand periods to save energy. It is assumed that all links operate at their capacity and overlooks the discreteness of link rates. In [9] authors employ OpenFlow to perform energy-aware routing for backbone networks. They considered the finite size of flow tables and presented a greedy heuristic to compress flow rules and reduce the occupancy of flow tables. The algorithm routes flows through minimum number of forwarding elements and thus may lead to network congestion. Markiewicz et al. [10] also presented a greedy heuristic with four different strategies to save power during low traffic demand periods. The central controller of SDN was used to turn off as many networking devices as possible to reduce the power consumption. The strategies used include Shortest Path First (SPF), Longest Shortest Paths First (LPF), Smallest Demand First (SDF), and Highest Demand First (HDF). The work of [10] concluded that LPF strategy outperformed the other three strategies. It does traffic allocation for demands in a sequence where the best path is found from a collection of pre-calculated shortest paths. This design was observed to be beneficial as it functions well with large networks and it uses an arbitrary topology. Although simulations showed significant improvement in terms of complexity and energy saving during low-traffic periods, adaptation of link rates was not considered. Wong et al. [11] also presented a greedy energyaware routing heuristic. They focused on rerouting traffic in an unsplittable manner such that traffic passes through least number of forwarding elements. The network components that are not utilized are switched to sleep mode. The ease of implementation makes this approach desirable; however, link rate adaptation was not considered.

In this paper, we propose a power-aware greedy routing algorithm for the routing application of SDN. We extend the algorithm in [11] to support discrete link rates. In an earlier study [1], we showed the significance of considering link rate discreteness on the efficiency of energy-aware routing. The objective of the proposed algorithm is to route flows in such a way that the number of active links is minimized without overloading links utilization. Unlike our previous work in [12], the algorithm proposed here is greedy in nature and hence less complex. Energy consumption can be greatly reduced by switching off some of the forwarding devices, such as routers and switches. However, switching off these devices require switching off all active ports [9] and may degrade network responsiveness to unexpected traffic bursts. This approach in return, saves both time and energy consumption. The power-aware routing problem with discrete rates for software-defined networks is modeled as mixed integer programming problem. The problem is known to be NP-hard [11]; thus, finding an exact solution is not feasible.

The rest of our paper is organized as follows: Section II introduces the network model and problem formulation. The proposed greedy algorithm is presented in Section III. The performance of the algorithm is evaluated in Section IV. Finally, conclusions are presented in Section V.

II. Network Model and Problem Formulation

A. Network Model

We consider a software-defined network consisting of a set of forwarding elements (FEs) and links. A set of data flows represented by the set $\mathcal{F} = \{1, , f, , F\}$ are to be routed from their sources $s(f)$ to destinations $d(f)$. We consider integral routing where each flow is rerouted on one path in an unsplittable way. The routing path of flow f is denoted by P^f and represents the sequence of links that flow f traverses to reach the destination $d(f)$. Let the software-defined network network be represented by a weighted directed graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \{1, , n, , N\}$ represents the set of nodes (i.e., routers, switches, hubs), and $\mathcal{L}$ represents the set of links connecting node i to node j, where i and j belong to $\mathcal{N}$, and i is not equal to j. A link represents a bi-directional connection between two nodes. For a given flow f, let the required data rate be x^f. When a flow passes through a link (i, j), i.e., (i, j) belongs to P^f, it introduces a rate on the link denoted by x_{ij}^f and is equivalent to x^f, otherwise x_{ij}^f is 0. Thus, the load on a link (i, j) can be written as follows:

$$x_{ij} = \sum_{\forall f:(i,j)\in P^f} x_{ij}^f. \tag{1}$$

In practice, links operate at one of the available discrete link rates $R_0, R_1, \cdots, R_{max}$ that satisfies their loads. The discrete link rates can be set by the following function

$$r_{ij}(x_{ij}) = \begin{cases} R_1, & 0 < x_{ij} \leqslant R_1 \\ R_2, & R_1 < x_{ij} \leqslant R_2 \\ \vdots & \vdots \\ R_{max}, & R_{max-1} < x_{ij} \leqslant R_{max}, \end{cases} \tag{2}$$

where $(i,j) \in \mathcal{L}$. Note, that R_{max} does not exceed the link capacity. The power consumption of a link is a function of the link rate x_{ij}. Therefore, the link power consumption is also discrete and can written as follows

$$\Omega_{ij}(x_{ij}) = \begin{cases} \alpha_1, & 0 < x_{ij} \leqslant R_1 \\ \alpha_2, & R_1 < x_{ij} \leqslant R_2 \\ \vdots & \vdots \\ \alpha_{max}, & R_{max-1} < x_{ij} \leqslant R_{max}, \end{cases} \tag{3}$$

where $(i,j) \in \mathcal{L}$. A link is turned OFF if it is not utilized, i.e., $x_{ij} = 0$, and does not consume power. In the following subsection, we present a formulation of the SDN routing problem.

B. Problem Formulation

The main objective of the power-aware routing is to minimize the energy consumption of the network by routing flows in such a way that the minimum number of links is ON with the lowest operating rate. The power-aware routing problem in SDN can be written as follows:

$$\min \sum_{\forall (i,j) \in \mathcal{L}} \Omega_{ij}(x_{ij}) \tag{4}$$

subject to

$$\sum_{(i,j)\in\mathcal{L}} x_{ij}^f - \sum_{(j,i)\in\mathcal{L}} x_{ji}^f = \begin{cases} \sum\limits_{\forall f: i \in s(f)} x_{ij}^f, \\ -\sum\limits_{\forall f: i \in d(f)} x_{ji}^f, \\ 0, \text{otherwise} \end{cases} \tag{5}$$

$$r_{ij}(x_{ij}) = \begin{cases} R_1, & 0 < x_{ij} \leqslant R_1 \\ R_2, & R_1 < x_{ij} \leqslant R_2 \\ \vdots & \vdots \\ R_{max}, & R_{max-1} < x_{ij} \leqslant R_{max} \end{cases} \tag{6}$$

$$\Omega_{ij}(x_{ij}) = \begin{cases} \alpha_1, & 0 < x_{ij} \leqslant R_1 \\ \alpha_2, & R_1 < x_{ij} \leqslant R_2 \\ \vdots & \vdots \\ \alpha_{max}, & R_{max-1} < x_{ij} \leqslant R_{max} \end{cases} \tag{7}$$

$$x_{ij} = \sum_{\forall f: (i,j) \in \mathcal{P}^f} x_{ij}^f \tag{8}$$

$$x_{ij} = x_{ji} \tag{9}$$

Equation (4) models the objective function of the power-aware routing problem, that is minimizing the network power consumption, where Ω_{ij} is the power consumed by link (i,j) given in Equation (7). Equation (5) is the flow conservation constraint; it guarantees that traffic flow arriving to a node is equivalent to the flow that leaves it unless it is either a source or destination. The constraint in Equation (6) forces the link to select one of the available link rates. Equation (9) ensures that links are bi-directional. The power-aware routing problem given in Equations (4) to (9) is non-convex problem and finding its optimal solution is intractable. This is due to the discontinuous step functions $r_{ij}(x_{ij})$ and $\Omega_{ij}(x_{ij})$ in constraints (6) and (7), respectively. In addition, the flow conservation constrain and integral routing add to the complexity of the problem. It becomes reasonable then to develop a low-complexity greedy algorithm to approximately solve the problem.

III. Proposed Algorithm

In this section we present a greedy algorithm for solving the power-aware routing problem given in Equations (4) to (9). The algorithm is outlined in Algorithm 1. The core of the proposed approach is to identify the set of flows with the largest impact on the network power consumption and reroute them on alternative paths to alleviate this impact.

At initialization, the shortest paths of the flows are computed based on Dijkstra algorithm, see Algorithm 1 lines 2 to 6. Given these shortest paths, the discrete link rates required to support the flows passing through links are evaluated and the network power consumption, i.e., $C^{Net} = \sum_{(i,j)\in\mathcal{L}} \Omega_{ij}(x_{ij})$, is computed. The computed value represents an upper bound to the network's power consumption, and an improvement in the routing solution is expected to reduce this consumption.

The flows are removed one by one to compute the impact of each flow on the network power consumption. The flow with the largest impact is identified as f^* (line 12) and added to the set of visited links $\mathcal{V}$ (line 12). In lines 13 to 20, we try to reroute the flow on links with sufficient spare capacity Δ_{ij} where a sufficient link rate is already active, and rerouting flow f^* on this link does not incur any additional power consumption. In this case, the weight of such links w_{ij} is set to zero. Alternatively, if the spare capacity is not sufficient to reroute the flow on a particular link, the link rate is switched to a higher level that is sufficient, and the corresponding increase of power consumption is considered to be the link weight $w_{ij} = \left[\Omega_{ij}(x_{ij}^* + x^{f^*}) - \Omega_{ij}(x_{ij}^*)\right]$. A weighted graph $W(\mathcal{N}, \mathcal{L}, w_{ij})$ representing the network nodes, links and their weights is formed.

Given the weighted network graph, we try to reroute the flow f^* on one of the K-shortest paths with the least power consumption, see lines 21 to 33. Specifically, we compute the network power consumption C_κ^{Net} for each of the K-shortest paths, κ. The κ^{th}-shortest path that reroutes f^* and achieves the minimum C_κ^{Net} is denoted by κ^*, see line 28. If the network power consumption after rerouting a flow f^* is less than the current network power consumption C^{Net}, the path of

flow f^* and C^{Net} are updated in line 30 and 31, respectively. Otherwise, the original shortest path is restored. The algorithm iteratively tries to find flows with largest impact on C^{Net} until all flows are visited. Then, it terminates and returns the routing solution consisting of all flows routing paths.

IV. Performance Evaluations

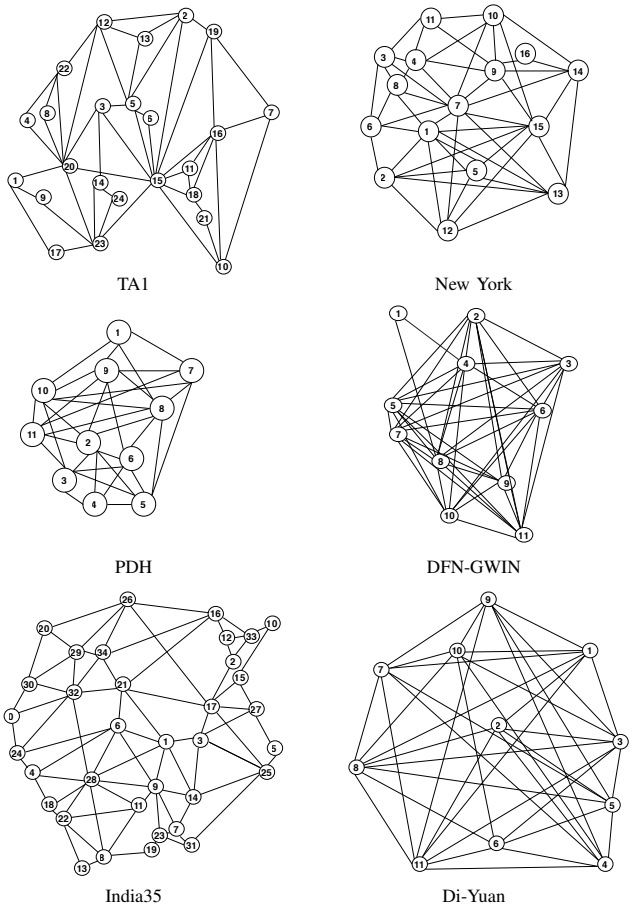


Figure 2. Network topologies considered in our performance evaluation experiments.

This section presents performance evaluation results of the proposed algorithm. To test the effectiveness of our solution, we compare the performance of our proposed solution with the shortest path algorithm. The results presented are averages of 10 runs of each experiment wherein flows sources and destinations are randomly generated. We consider 100 flows in each experiments with uniformly distributed flow size x^f in the range 50 to 200 Mbps. Our results are simulated using MATLAB and real-world networks called India35, TA1, New York, PDH, DI-YUAN and DFN-GWIN, which are shown in Figure 2. These networks are available at the survivable network design library [13]. The networks were selected to have diversity in characteristics as shown in Table I. The

Algorithm 1: Proposed Greedy Algorithm

Input: Bidirectional graph $G(\mathcal{N}, \mathcal{L})$, $\mathcal{F}$, x^f, $o(f)$, and $d(f)$ $\forall f \in \mathcal{F}$

1 $\mathcal{V} \leftarrow \emptyset$
/* Compute $\mathcal{P}^f$ $\forall f \in \mathcal{F}$ and C^{Net} */
2 **foreach** $f \in \mathcal{F}$ **do**
3 $\quad \mathcal{P}^f \leftarrow \text{Dijkstra}\,(G(\mathcal{N}, \mathcal{L}), f, o(f), d(f))$
4 **foreach** *link* $(i,j) \in \mathcal{L}$ **do**
5 $\quad x_{ij} \leftarrow \sum_{\forall f \in \mathcal{F}:(i,j)\in\mathcal{P}^f} x^f_{ij}$
6 $C^{Net} \leftarrow \sum_{(i,j)\in\mathcal{L}} \Omega(x_{ij})$
/* Reroute flows to reduce C^{Net} */
7 **while** $\mathcal{V} \neq \mathcal{F}$ **do**
$\quad$ /* Find flow with largest impact on C^{Net} */
8 $\quad$ **foreach** $f' \in \mathcal{F}$ **do**
9 $\quad\quad$ **foreach** $(i,j) \in \mathcal{L}$ **do**
10 $\quad\quad\quad x'_{ij} \leftarrow \sum_{f\in\mathcal{F}\setminus\{f'\}:(i,j)\in\mathcal{P}^f} x^f_{ij}$
11 $\quad\quad C^{Net\setminus f'} \leftarrow \sum_{\forall(i,j)\in\mathcal{L}} \Omega_{ij}(x'_{ij})$
12 $\quad f^* \leftarrow \arg\min_{f'\in\mathcal{F}} C^{Net\setminus f'}$ and $\mathcal{V} \leftarrow \mathcal{V} \cup \{f^*\}$
$\quad\quad$ /* Compute the residual network and form the weighted graph */
13 $\quad$ **foreach** *link* $(i,j) \in \mathcal{L}$ **do**
14 $\quad\quad x^*_{ij} \leftarrow \sum_{f\in\mathcal{F}\setminus\{f^*\}:(i,j)\in\mathcal{P}^f} x^f_{ij}$
15 $\quad\quad \Delta_{ij} \leftarrow (r_{ij} - x^*_{ij})$
16 $\quad\quad$ **if** $\Delta_{ij} < x^{f^*}$ **then**
17 $\quad\quad\quad w_{ij} \leftarrow \left[\Omega_{ij}(x^*_{ij} + x^{f^*}) - \Omega_{ij}(x^*_{ij})\right]$
18 $\quad\quad$ **else**
19 $\quad\quad\quad w_{ij} \leftarrow 0$
20 $\quad$ Form a weighted graph $W(\mathcal{N}, \mathcal{L}, w_{ij})$
$\quad\quad$ /* Try rerouting the flow f^* on one of the K-shortest paths */
21 $\quad \mathcal{K}^{f^*} \leftarrow \text{K-SP}\,(W(\mathcal{N}, \mathcal{L}, w_{ij}), f^*, o(f), d(f))$
22 $\quad$ **foreach** $\kappa \in \mathcal{K}^{f^*}$ **do**
23 $\quad\quad \mathcal{P}^{f^*}_{\text{temp}} \leftarrow \mathcal{P}^{f^*}$
24 $\quad\quad \mathcal{P}^{f^*} \leftarrow \kappa$
25 $\quad\quad$ **foreach** *link* $(i,j) \in \mathcal{L}$ **do**
26 $\quad\quad\quad x_{ij} \leftarrow \sum_{f\in\mathcal{F}:(i,j)\in\mathcal{P}^f} x^f_{ij}$
27 $\quad\quad C^{Net}_{\kappa} \leftarrow \sum_{\forall(i,j)\in\mathcal{L}} \Omega_{ij}(x_{ij})$
$\quad\quad$ /* Find the rerouting path that minimizes C^{Net} */
28 $\quad \kappa^* \leftarrow argmin_{\kappa} C^{Net}_{\kappa}$
29 $\quad$ **if** $C^{Net}_{\kappa^*} < C^{Net}$ **then**
$\quad\quad$ /* Update path and update C^{Net} */
30 $\quad\quad \mathcal{P}^{f^*} \leftarrow \kappa^*$
31 $\quad\quad C^{Net} \leftarrow C^{Net}_{\kappa^*}$
32 $\quad$ **else**
$\quad\quad$ /* Restore original path */
33 $\quad\quad \mathcal{P}^{f^*} \leftarrow \mathcal{P}^{f^*}_{\text{temp}}$
34 **return** C^{Net}, $\mathcal{P}^f$ $\forall f \in \mathcal{F}$

Table I
NETWORK TOPOLOGIES AND THEIR PROPERTIES

Network	N	L	Avg. Node Degree
India-35	35	80	4.57
TA-1	24	55	4.58
New York	16	49	6.12
PDH	11	34	6.18
DI-YUAN	11	42	7.64
DFN-GWIN	11	47	8.55

link rates are assumed discrete and have the following power consumption [14] function

$$\Omega_{ij}(x_{ij}) = \begin{cases} 3.2\text{W}, & 0 < x_{ij} \leqslant 100\text{Mbps} \\ 4.27\text{W}, & 100\text{Mbps} < x_{ij} \leqslant 1\text{Gbps} \\ 7.7\text{W}, & 1\text{Gbps} < x_{ij} \leqslant 10\text{Gbps} \end{cases} \quad (10)$$

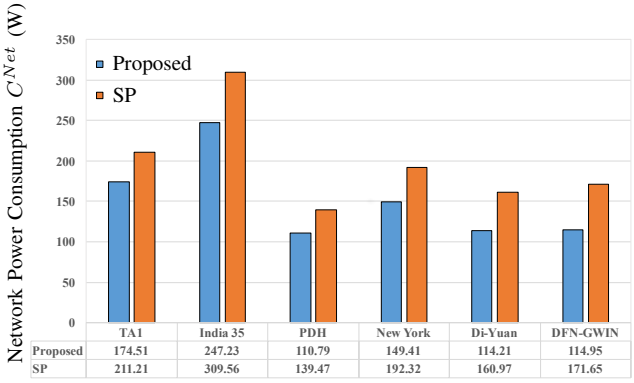


Figure 3. Average network power consumption of the considered topologies under the proposed and SP algorithms.

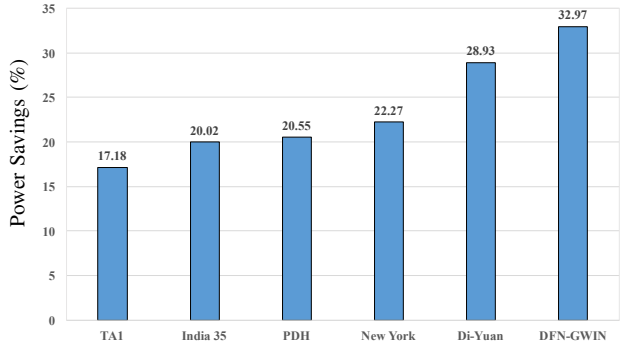


Figure 4. The percentage of power savings achieved by the proposed algorithm in comparison to SP.

Figure 3 presents the average power consumption of all networks when operating under the proposed algorithm and shortest path algorithm (SP). It is clear that the proposed algorithm achieves substantial power savings in comparison to SP, which ranges from 17.18% to 32.97%, as shown in Figure 4. If we group the topologies in three groups according to their average node degree, India-35 and TA1, New York and PDH, in addition to DI-YUAN and DFN-GWIN, we can see that the proposed algorithm savings generally increases with the increase in number of links for a given average number of node degree. The larger the number of links, the larger the chance of finding rerouting paths. The largest power saving (32.97%) is achieved in DFN-GWIN that is highly connected with 8.55 average node degree, while the least power saving (17.18%) is shown for one of the least connected networks TA-1 with 4.58 average node degree.

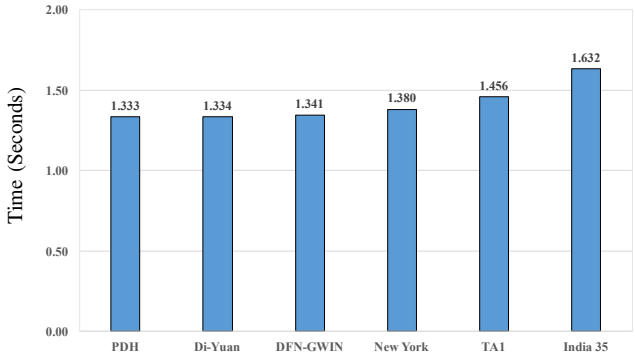


Figure 5. Computation time of the proposed algorithm.

Figure 5 shows the average computation time of the proposed algorithm. The computation time of the SP is negligible and not shown. Results demonstrate that the computation time increases slightly with the increase in number of links. Although, the number of links for India-35 (80 links) is almost double the number of links of PDH (34 links), the computation time increased by only 0.299 seconds.

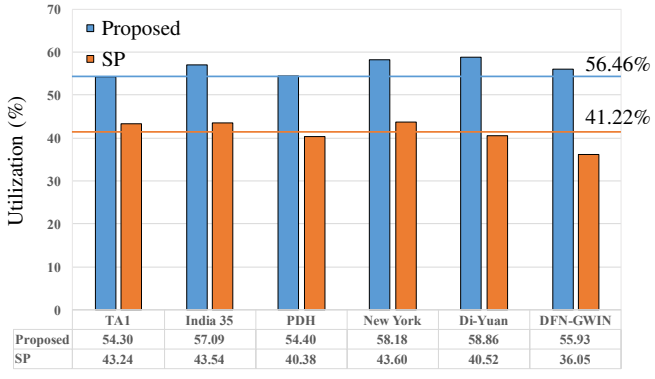
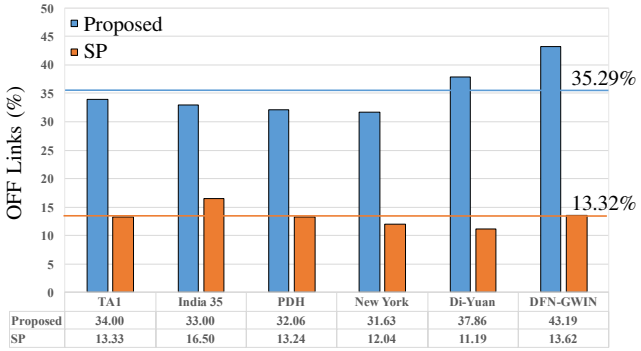


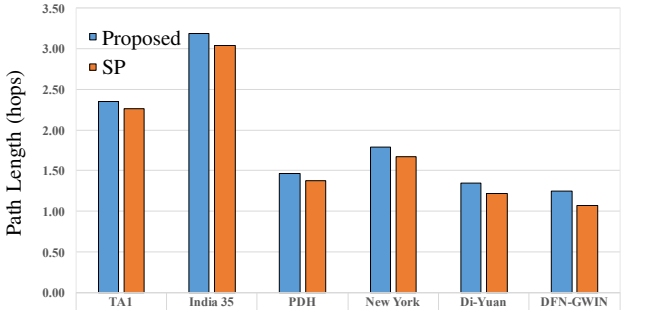
Figure 6. Average link utilization of the proposed and SP algorithms.

Figure 6 shows the link utilization in all considered networks for the proposed algorithm and SP. The proposed algorithm shows a higher average utilization than SP (56.46% vs. 41.22%) because it tries to reroute traffic on ON links and avoids OFF links, thus aggregates flows on ON links and increases the links utilization. This behavior results in an increase in the proportion of links switch OFF as shown in Figure 7, which reflects major impact on the power efficiency of the network.



	TA1	India 35	PDH	New York	Di-Yuan	DFN-GWIN
Proposed	34.00	33.00	32.06	31.63	37.86	43.19
SP	13.33	16.50	13.24	12.04	11.19	13.62

Figure 7. Percentage of links switched OFF by the proposed and SP algorithms.



	TA1	India 35	PDH	New York	Di-Yuan	DFN-GWIN
Proposed	2.35	3.18	1.47	1.79	1.35	1.25
SP	2.26	3.04	1.37	1.68	1.22	1.07

Figure 8. Average path length of routes generated by the proposed algorithm and SP.

The average number of hops traversed by flows is shown in Figure 8. Despite the significant power saving achieved by the proposed algorithm, the trade-off in path length is minimal. For all considered topologies, the average increase in path length is less than 0.2 hops.

V. Conclusions

We have proposed a greedy power-aware routing algorithm for software-defined networks with discrete link rates. To this end, we have formulated an optimization problem with the objective of minimizing the network power consumption subject to flow conservation and discrete link rates constraints. Because the problem is NP-hard, we designed a greedy algorithm to reroute flows on alternative paths computed using a K-shortest path algorithm, such that the number of OFF links and level of active link rates are minimized. Performance evaluation results indicate a significant energy saving ranging from 17.18% to 32.97% relative to SP in six different topologies. Furthermore, performance results show that the proposed algorithm increases link utilization by 15.25% and increases the percentage of OFF links by 21.97%, in comparison to SP. Despite the significant energy saving brought to the network, the trade-off in average path length was limited to 0.2 hops.

Acknowledgments

The project was funded partially by Kuwait Foundation for the Advancement of Sciences under project code: P314-35EO-01.

References

[1] M. K. Awad, G. Neama, and Y. Rafique, "The impact of practical network constraints on the performance of energy-aware routing schemes," in *IEEE International Conference on Service Operations And Logistics, And Informatics (SOLI)*, nov. 2015, pp. 77–81.

[2] W. V. Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ICT electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64 – 76, 2014.

[3] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *The second ACM SIGCOMM workshop on Hot topics in software defined networking (HostSDN)*, 2013, pp. 55–60.

[4] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, February 2013.

[5] Nishtha and M. Sood, "Software defined network – architectures," in *International Conference on Parallel, Distributed and Grid Computing (PDGC)*, Dec. 2014, pp. 451–456.

[6] W. Braun and M. Menth, "Software-defined networking using openflow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, p. 302, 2014.

[7] D. Li, Y. Shang, and C. Chen, "Software defined green data center network with exclusive routing," in *IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2014, pp. 1743–1751.

[8] M. Rahnamay-naeini, S. S. Baidya, E. Siavashi, and N. Ghani, "A traffic and resource-aware energy-saving mechanism in software defined networks," in *International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2016, pp. 1–5.

[9] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2014, pp. 2523–2529.

[10] A. Markiewicz, P. N. Tran, and A. Timm-Giel, "Energy consumption optimization for software defined networks considering dynamic traffic," in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct. 2014, pp. 155–160.

[11] R. Wang, Z. Jiang, S. Gao, W. Yang, Y. Xia, and M. Zhu, "Energy-aware routing algorithms in software-defined networks," in *IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2014, pp. 1–6.

[12] M. K. Awad, Y. Rafique, and R. A. M'Hallah, "Energy-aware routing for software-defined networks with discrete link rates: A benders decomposition-based heuristic approach," *Sustainable Computing, Informatics and Systems*, submitted.

[13] *SNDlib 1.0 - Survivable Network Design Library*, vol. 55, no. 3. New York, NY, USA: Wiley-Interscience, May 2010.

[14] Intel, "Intel ethernet controller x540 datasheet rev. 3.0," January 2016. [Online]. Available: http://www.intel.com/content/dam /www/public/us/en/ documents/datasheets/ethernet-x540-datasheet.pdf