Link Utility and Traffic Aware Energy Saving in Software Defined Networks

Conference Paper · June 2017

CITATIONS
0

READS
21

2 authors:

Beakal Gizachew Assefa
Koc University
5 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Oznur Ozkasap
Koc University
117 PUBLICATIONS **1,180** CITATIONS
SEE PROFILE

Link Utility and Traffic Aware Energy Saving in Software Defined Networks

Beakal Gizachew Assefa and Oznur Ozkasap
Department of Computer Engineering
Koc University, Istanbul, Turkey
{bassefa13,oozkasap}@ku.edu.tr

***Abstract*—Software Defined Networking (SDN) is an emerging network paradigm that gains increasing attention both from academia and industry. Energy saving aspects of network protocols, being studied for different network technologies, have also recently been addressed in SDN. In this paper, we address the traffic and energy aware routing problem in SDN and propose link utility based heuristic algorithms, namely NSP and NMU, that are not only general in their applicability but also balance the trade-off between energy saving and performance. We propose an IP formulation for traffic and energy aware routing problem based on link utility information, and evaluate the algorithms using real traces of low, medium and high traffic volumes and network topologies. NSP and NMU are shown to outperform the existing solutions in terms of average path length and achieve up to 37% energy saving.**

I. Introduction

SDN paradigm separates the control plane and the forwarding plane. In traditional networking, on the other hand, both the forwarding and controlling functions are confined in individual routers, where each router executes its routing algorithm to determine and update forwarding table entries. Changes in the network can happen due to several reasons including link failures, link cost updates, adding security policies and other network policy changes. Reflecting such changes in a network (i.e. autonomous system) is a complicated task since they need to be propagated to all routers in the network. In this context, the powerful feature of SDN is that it enables programming the network by a logically centralized controller, where the switches are simply forwarding devices. Despite the fact that changes in network protocols (e.g. IPv6) has taken more than a decade to be implemented fully, large-scale change of architecture by SDN is widely adopted by major companies (e.g. Yahoo, Google) in a relatively short period of time [1], [2].

Minimizing the global energy consumption has been studied by several disciplines. Regarding computing and networking, the ICT sector constitutes more than 10% of the global energy consumption out of which 2% is due to network components. The electricity cost of cloud data centers would increase by 63% in 2020 as reported in [3]. Networks are designed with resources (e.g., link bandwidth and device memory) with sufficient capacities to provide acceptable performance in the case of high traffic loads. The fact, however, is that the network devices and the links are utilized 30% to 40% most of the time [4]. On the other hand, network devices consume similar amount of energy even when the traffic volume is low, that leads to stable network energy consumption independent of the traffic load level. An important research question is that how the energy consumption of a network could be made proportional to the traffic volume. The flexible control approach of SDN can be used to deploy energy aware routing algorithms in an efficient and dynamic manner, unlike the traditional network routing. Energy efficiency in SDN can be provided with software based and hardware based solutions where we classify algorithmic solutions into traffic aware, end system aware and rule placement [4]. The power consumed by the switches and links constitute the energy cost of software defined networks. The practical approach is to minimize the number of active links and switches to accommodate the current traffic demand. That process involves turning off some switches and putting some links in an inactive state during low traffic demand, and doing the opposite when the traffic demand is high. However, there exists a trade-off since minimizing the amount of active network resources would adversely affect the network performance and quality of service. A problem that needs to be addressed is to minimize energy consumption at the same time maintain an acceptable performance. Energy efficient solutions mostly focus on the former objective [5]–[10].

In this work, we address the traffic and energy aware routing problem in SDN and propose link utility based heuristic algorithms which are not only general in their applicability but also balances the trade-off between energy saving and performance. The contributions are as follows. We propose an IP formulation for traffic and energy aware routing problem based on link utility information. We propose heuristics algorithms, namely NSP and NMU, with an objective of minimizing energy consumption and also maintaining acceptable network performance in terms of average path length. We evaluate the algorithms using real traces of low, medium, high traffic volumes and network topologies. Results show that NSP and NMU outperform the existing solutions in terms of average path length and also achieve 7 to 37% energy saving.

II. Related Work

Studies on traffic aware energy efficiency in SDN consider links [6]–[10], forwarding switches [11], [12] or combination of both [5], [13] as the energy saving components. Furthermore, queue engineering techniques consider the arrival of

packets and their waiting times to decide per-port power requirement [8]–[10], [12], [14], [15]. The classification of the approaches can also be done in terms of the network topology used in the experimental settings, where mesh, tree topologies, and real campus networks are examples. Fat-tree and Bcubic structures are used to organize end systems in data centers [5], [7], [16].

Several heuristic algorithms are proposed to solve the problem dynamically within short period of time. Table I presents various heuristics algorithms proposed in the literature. The heuristic types are either specific [16], [17] or general topology [13], [14], [18]. Elastic tree [5] proposes two algorithms namely greedy bin packing and fat-tree topology aware heuristics. An improvement on the greedy bin packing by applying link rate adaptation and correlation of flows is presented in [6]. Improved link rate adaptive approaches and resource awareness are proposed in [16], [17]. These approaches, however, are tailored to a particular topology like fat-tree that they cannot be applied to different structures of topology. While specific heuristics work best for a given scenario and cannot be generalized, the general heuristics cannot capture the energy capabilities fully.

TABLE I: Traffic aware, energy efficient routing heuristics for SDN

Heuristics	Type
Greedy bin packing, topology aware [5]	Tree based
Link rate adaptation based on bin packing [6]	Tree based
Rate-adaptive topology-aware [16]	Tree based
Resource aware heuristics [17]	Tree based
Dynamic traffic consolidation [13]	General
Based on residual capacity based [14]	General
Controller placement [18]	General

The general heuristics, on the other hand, assume the network as a bidirectional graph with vertices and edges along with their constraints. The approach used in [13] starts by formulating the problem with MIP, then proposes four heuristic algorithms. Each flow is assigned a path from its source to destination. The first flow is assigned its corresponding shortest path, the proceeding flows are assigned paths where the change in energy consumption is minimized. The efficiency of the algorithm depends on the order in which the flows are processed. Four variations of sorting criteria are proposed. The four variations of the algorithms are flow demands with Shortest Path First (SPF), Shortest Path Last (SPL), Smallest Demand First (SDF), and High Demand First (HDF). The algorithms are tested on a real campus network and HDF outperforms the rest in terms of energy saving. The heuristic in [14] removes the redundant paths from the network based on the residual capacity of the link. This approach, however, needs redundancy removal capable routers. Another recent work [18] formulates the controller placement problem and traffic awareness as a single problem, and provides a general heuristics.

III. Utility Based Formulation of the Energy Aware Routing Problem

The network is modeled as bi-directed weighted graph $\mathbb{G}$= ($\mathbb{Z}$,$\mathbb{E}$) where $\mathbb{Z}$ is the set of switches where $Z_i \in \mathbb{Z}$ represents switch i and $e_{ij} \in \mathbb{E}$ represents that there exists a link between switches Z_i and Z_j. The weight W_{ij} corresponds to the bandwidth of the link e_{ij}. Let binary variable S_i denote the status of switch Z_i such that

$$S_i = \begin{cases} 1, & \text{if switch } Z_i \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

Traffic in the network is represented by set of flows $\mathbb{F}$ where $f \in \mathbb{F}$ is defined as f=(sr, ds, λ_f). sr and $ds \in \mathbb{Z}$ are the source and destination switches and λ_f is the rate of flow f measured in bytes per second.

$$f_{ij} = \begin{cases} 1, & \text{if flow } f \text{ passes through edge } e_{ij} \\ 0, & \text{otherwise} \end{cases}$$

$\mathbb{U}$ is the set of the utilities of every edge in the graph $\mathbb{G}$ where $U_{ij} \in \mathbb{U}$

$$U_{ij} = \frac{\sum_{\forall f} f_{ij} * \lambda_f}{W_{ij}} \tag{1}$$

is defined as the ratio of the sum of the rates of the flows passing through the edge e_{ij} to the link bandwidth W_{ij}. Utility of a link is between 0 and 1, where 0 means no flow is passing through the link and 1 means the sum of the flow rates passing through the link is equal to the link bandwidth. Let $Umin$ be the minimum utility threshold to keep a link active. $Umax$ is the maximum utility of a link tolerated for performance reason. The network energy consumption is the total sum of the energy consumption by active links and active switches.

$$F(\mathbb{G}) = \sum_{\forall e_{ij} \in \mathbb{E}} L_{ij} * C_{ij} + \sum_{\forall Z_i \in \mathbb{Z}} S_i * CS_i \tag{2}$$

where

$$L_{ij} = \begin{cases} 1, & \text{if link } e_{ij} \text{ is on} \\ 0, & \text{otherwise} \end{cases}$$

and CS_i and C_{ij} are power consumption of switch Z_i and the link e_{ij} measured in watt. The objective function (3) minimizes the sum of the power consumption by active switches and active links.

$$\text{minimize } F(\mathbb{G}) \tag{3}$$

$$\text{subject to } \sum_{\forall f} f_{ij} * \lambda_f \leq W_{ij} \ , \ \forall e_{ij} \tag{4}$$

$$\sum_{\forall f} f_{ij} = \sum_{\forall f} f_{ji} \ , \ [Z_i, Z_j] \neq sr, \neq ds \tag{5}$$

$$f_{mj} = f_{in} \ , \ Z_m = sr \ , \ Z_n = ds \ , \ \forall e_{mj} \ , \ \exists e_{in} \tag{6}$$

$$L_{ij} \leq X_{ij} \ , \ \forall e_{ij} \tag{7}$$

where

$$X_{ij} = \begin{cases} 1, & Umin \leq U_{ij} \leq Umax \\ 0, & \text{otherwise} \end{cases}$$

The constraint in Equation 4 states that, the sum of the rates of flows between two switches should not exceed the link capacity. Constraint in equation 5 states that the number of flows entering and leaving for switches which are neither

destination nor sources of a flow should be equal. Constraint in equation 6 assures a flow entering from source switch should reach the destination switch. Constraint in 7 asserts that the a link needs to turned off X_{ij} is zero which means that the utility of the link is not between U_{min} and U_{max}.

IV. Algorithms for Energy Aware Routing

The intuition behind our heuristics is minimizing the energy consumption of links and switches (objective of the IP formulation) through identifying the underutilized links, redirecting selected flows to more utilized replacement path, turning off the underutilized links if all flows are redirected. We propose topology independent heuristic algorithms, Next Shortest Path (NSP) and Next Maximum Utility (NMU), based on the utility of links, which correspond to replacing the under-utilized links with next shortest path and with the next path in the direction of the maximum link utility, respectively.

Algorithm 1 ShortestPathInitialization($\mathbb{G}$, $\mathbb{F}$)

1: $\mathbb{U} \leftarrow Initialize(0)$
2: **for all** $f \in F$ **do**
3: $\quad path_f \leftarrow shortestpath(sr, ds, \lambda_f)$
4: $\quad$ **for all** $e_{ab} \in path_f$ **do**
5: $\quad\quad U_{ab} \leftarrow U_{ab} + \frac{\lambda_f}{W_{ij}}$
6: $\quad$ **end for**
7: **end for**
8: **return** $\mathbb{U}$

Algorithm 1 initializes utilities of the links using shortest path algorithm for each flow. It takes the graph $\mathbb{G}$, which represents the network topology and traffic demand $\mathbb{F}$. For a given link e_{ij}, if a traffic flow $f = (sr, ds, \lambda_f)$ passes through it, its corresponding utility U_{ij} is incremented by $\frac{\lambda_f}{W_{ij}}$. The algorithm returns the set $\mathbb{U}$ that includes the utilities of all links. Algorithm 2 takes the utility set $\mathbb{U}$ and threshold $Umin$, and returns the list of links with utilities less than or equal to the threshold.

Algorithm 2 ReturnCandidateList($\mathbb{U}$, $Umin$)

1: $CandidateList \leftarrow []$
2: **for all** $U_{ij} \in \mathbb{U}$ **do**
3: $\quad$ **if** $U_{ij} \leq Umin$ **then**
4: $\quad\quad CandidateList.append(e_{ij})$
5: $\quad$ **end if**
6: **end for**
7: **return** $CandidateList$

Algorithm 3 (NSP) identifies under-utilized links and re-routes traffic flows passing through them to the next shortest alternative path. The algorithm takes topology $\mathbb{G}$, set of flows $\mathbb{F}$ and threshold $Umin$ as inputs. It first initializes the link utilities based on the shortest path algorithm. It then selects the under-utilized links (invoking algorithm 2) and sets the $CandidateList$. The $shortestpaths_{ij}$ (line 4) returns the list of paths from Z_i to Z_j excluding those links in the $CandidateList$. The paths stored in $shortestpaths_{ij}$ are sorted according to the path length. The $shortestpaths_{ij}$ is set to $Path_{ij}$ (line 5). NSP algorithm then picks one of the shortest paths SP to replace the direct link. SP is replaced with the next $Path_{ij}$ until all the links in $\forall e_{ab} \in SP$, $U_{ab} + U_{ij} \leq Umax$ (lines 6 to 8). Replacing a direct link with alternative path needs redirecting the flows passing through it. Basically, the utilities of the hops composing SP are increased and the utility of the direct link are decreased (lines 11 and13). The status of a link is updated if the utility is 0 (all the flows passing are redirected to alternative path SP), hence updates the graph $\mathbb{G}$'s status (lines 15-18). Algorithm 3 returns the updated sub graph of $\mathbb{G}$ (line 20).

Algorithm 3 NSP: NextShortestPath($\mathbb{G}$, $\mathbb{F}$, $Umin$,$Umax$)

1: $\mathbb{U} \leftarrow ShortestPathInitialization(\mathbb{G}, \mathbb{F})$
2: $CandidateList \leftarrow ReturnCandidateList(\mathbb{U}, Umin)$
3: **for all** $e_{ij} \in CandidateList$ **do**
4: $\quad Path_{ij} \leftarrow shortestpaths_{ij}$
5: $\quad SP \leftarrow Path_{ij}[0]$ ▷ Pick one of the shortest paths
6: $\quad$ **while** $\exists\ e_{ab} \in SP$ where $U_{ab} + U_{ij} > Umax$ **do**
7: $\quad\quad SP \leftarrow \texttt{next}(Path_{ij})$
8: $\quad$ **end while**
9: $\quad$ **for all** f `passing through` e_{ij} **do**
10: $\quad\quad$ **for all** $e_{ab} \in SP$ **do**
11: $\quad\quad\quad U_{ab} \leftarrow U_{ab} + \frac{\lambda_f}{W_{ab}}$ ▷ increment U_{ab}
12: $\quad\quad$ **end for**
13: $\quad\quad U_{ij} \leftarrow U_{ij} - \frac{\lambda_f}{W_{ij}}$ ▷ decrement U_{ij}
14: $\quad$ **end for**
15: $\quad$ **if** $U_{ij} == 0$ **then**
16: $\quad\quad L_{ij} \leftarrow 0$ ▷ state of link is inactive
17: $\quad\quad \mathbb{G} \leftarrow \texttt{Turn off}(e_{ij})$
18: $\quad$ **end if**
19: **end for**
20: **return** $\mathbb{G}$ ▷ Subgraph

Algorithm 4 (NMU) replaces the candidate links with the path that has the link with maximum utility. After identifying the candidate links (line 2), list of replacement paths except for links in $CandidateList$, $minutilitypath_{ij}$ is assigned to $Path_{ij}$ (line 4). $Path_{ij}$ is sorted according to maximum utility link. UP is the first path in the $Path_{ij}$. NMU algorithm then picks one of the shortest paths UP to replace the direct link. UP is replaced with the next $Path_{ij}$ until all the links in $\forall e_{ab} \in UP$, $U_{ab} + U_{ij} \leq Umax$ (lines 6 to 8). The algorithm then updates the utilities of links and the graph $\mathbb{G}$ in the remaining part with similar procedure of Algorithm 3.

Figure 1 shows an example of how the algorithms work. As given in Fig1.a, utility of the link connecting nodes i and j is 0.05 and the minimum ratio to make a link active $Umin$=0.06. Thus, link e_{ij} is a candidate link to be turned off. $Path_{ij} = [i, G, j], [i, A, B, j], [i, C, D, E, j]$ where number of hops in each path is 2, 3, and 4 respectively. $SP = [i, G, j]$ and the number of hops is 2. According to NSP (Algorithm 3), the replacement path for the direct link, as illustrated in Fig.1.a,is to pick the next shortest path among $Path_{ij}$ which is $[i, G, j]$. The NMU algorithm on the

Algorithm 4 NMU: NextMaximumUtility($\mathbb{G}$, $\mathbb{F}$, $Umin$)

$\mathbb{U} \leftarrow ShortestPathInitialization(\mathbb{G}, \mathbb{F})$
$CandidateList \leftarrow ReturnCandidateList(\mathbb{U}, Umin)$
for all $e_{ij} \in CandidateList$ **do**
  $Path_{ij} \leftarrow maxutilitypaths_{ij}$
  $UP \leftarrow Path_{ij}[0]$
  while $\exists\ e_{ab} \in UP$ where $U_{ab} + U_{ij} > Umax$ **do**
    $UP \leftarrow \texttt{next}(Path_{ij})$
  end while
  for all f `passing through` e_{ij} **do**
    for all $e_{ab} \in UP$ **do**
      $U_{ab} \leftarrow U_{ab} + \frac{\lambda_f}{W_{ab}}$
    end for
    $U_{ij} \leftarrow U_{ij} - \frac{\lambda_f}{W_{ij}}$
  end for
  if $U_{ij} == 0$ **then**
    $L_{ij} \leftarrow 0$ ▷ state of link is inactive
    $\mathbb{G} \leftarrow \texttt{Turn off}(e_{ij})$
  end if
end for
return $\mathbb{G}$

other hand, selects the path which contains the link with the maximum utility $[i, C, D, E, j]$ with a path length of 4 hops. After the updates, the flows passing through the edge e_{ij} would be redirected to the replacement path. Assuming all links have equal bandwidth in this example, the utility ratio of the link would be added to each link of the replacing path. The outcomes of NSP and NMU algorithms applied to Fig.1.a are illustrated in Fig.1.b and c, respectively. The minimum threshold value $Umin$ directly affects the set of candidate links. As another example, in Fig.1.c, the threshold $Umin$ is set to 0.08. The candidate links are $e_{i,j}$, $e_{i,G}$ and $e_{G,j}$. The results of the NSP and NMU algorithms are shown in Fig.1.d and e, respectively.

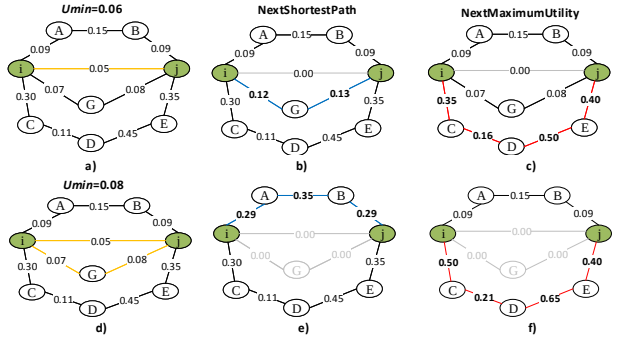

Fig. 1: Examples for NSP and NMU utility based heuristics

V. Experimental Results

The experimental platform is based on a POX controller and Mininet [19] network emulator installed on Ubuntu 16.04 64-bit. The topology is created on Mininet, and the heuristics are implemented on POX controller. We conducted our experiments using real traces from SNDlib [20], in particular the Geant dynamic traffic trace which is a European research network. Traffic demand matrices are pulled every 15 minutes from a 4-month duration trace. The number of nodes and bidirectional links are 22 and 36 respectively. Low traffic, medium traffic, and high traffic volumes correspond to 20.0% (98 flows), 50% (212 flows), and 80% (449 flows) of the total capacity of the underlying network. Parameters $Umax$ is set to 95%, and $Umin$ values are set to 20%, 50% and 53% for low, medium, and high traffic loads, respectively. The $Umin$ parameter is determined through experimentation by finding the value that maximizes the number of links saved. The performance metrics are the energy saving and the average path length. The power consumption of a switch is considered as three times the power consumption of a link, as reported in [21]. Then, the total energy consumption of a network $\mathbb{G}$ working with full capacity would be $|\mathbb{Z}| * 3 + |\mathbb{E}|$. For a given set of flows $\mathbb{F}$, the energy saving is calculated as $[1 - \frac{\sum_{\forall f, \forall e_{ij}} f_{ij} + 3 * \sum_{\forall S_i} S_i}{|\mathbb{Z}| * 3 + |\mathbb{E}|}] * 100$. The average path length is calculated as the sum of the path length of each flow divided by the total number of flows: $\frac{\sum_{\forall f, \forall e_{ij}} f_{ij}}{|\mathbb{F}|}$.

Table II presents the abbreviation and description of the algorithms used in this study. B is the algorithm with the best performance in terms of energy saving among the four algorithms proposed in [13] in the experiments. B+NSP and B+NMU correspond to the algorithms when NSP and NMU are applied on the best result obtained by SPF, SPL, SDF, and HDF. Furthermore, we also report the best combined result.

TABLE II: Table of Heuristics

Abbreviation	Description
Proposed heuristics	
NSP	Next Shortest Path
NMU	Next Maximum Utility
Heuristics of [13]	
SPF	Shortest Path First
SPL	Shortest Path Last
SDF	Smallest Demand First
HDF	Highest Demand First
B	Best of SPF, SPL, SDF, and HDF
Combinations	
B+NSP	NSP on top of B
B+NMU	NMU on top of B
Combined	Best of combinations

Figure 2 shows the energy saving and average path length results as a function of load, medium and high traffic volumes. NSP and NMU exhibit the shortest path length and the least energy saving in all traffic volume scenarios. The least energy saving is 7.07% in high traffic and 37.5% in low traffic. B algorithm is always better in terms of energy saving but the worst in terms of average path length. The energy saving increases from high traffic to low traffic. For average path length NSP and NMU perform the best and algorithm B exhibits the worst average path length. This is because NSP and NMU initialize the network by the shortest paths for each traffic flow. However, initializing the network with the outputs of B and applying NSP and NMU on top provides slightly better energy saving and reduces the average path length. Our proposed solutions applied to any energy aware routing algorithm increase the energy saving and minimize the average path length at the same time.

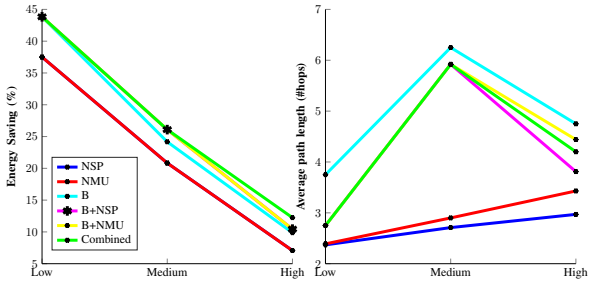

Fig. 2: Comparison of algorithms for different traffic volumes. a) Energy saving, b) Average path length

Figure 3 shows the energy saving and average path length for medium traffic. NSP and NMU are best in minimizing the average path length but less efficient in energy saving. B is 5% more energy efficient than NSP and NMU. Combining B with NSP, NMU shows an increase of 2% energy saving and decrease average path length by 0.32. Figure 4 shows that NSP and NMU are better in average path length and B is better in energy saving. The combination of the NSP and NMU which are B+NSP and B+NMU are 2.53% more energy efficient and 0.23 less in average path length as compared to B.

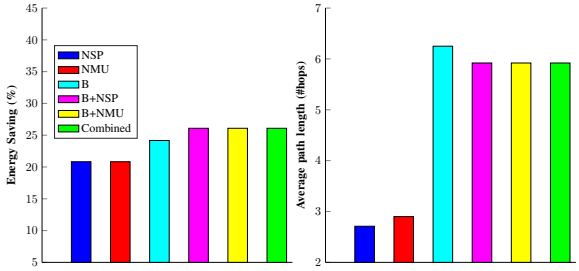

Fig. 3: Medium traffic load. a) Energy saving, b) Avg path length

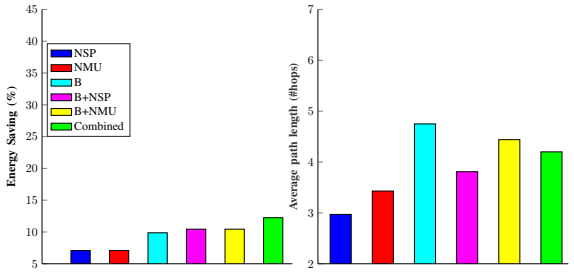

Fig. 4: High traffic load. a) Energy saving, b) Avg path length

VI. Conclusions

We present the formulation for traffic and energy aware routing based on link utility information in SDN, and propose link utility based heuristics (NSP and NMU) that balance the trade-off between energy saving and performance. Using real topology and traffic traces, comparative experimental results show that NSP and NMU outperform the existing heuristics in terms of average path length and also achieve up to 37% energy saving. Our approach applied on top of the other algorithms exhibit increase in energy saving and decrease in average path length. As future work, we aim to investigate the performance of our utility based heuristics on data center networks, analyze the effect of heuristics' $Umin$ and $Umax$ parameters, determine the optimal values that maximize the energy savings, and examine the scalability aspects.

References

[1] 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," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[3] GreenPeace. (2014) Clicking clean: How the Companies are creating the green internet. [Online]. Available: http://www.greenpeace.org/international/en/

[4] B. G. Assefa and O. Ozkasap, "State-of-the-art energy efficiency approaches in software defined networking," *ICN 2015*, p. 268.

[5] 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.

[6] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "CARPO: Correlation-AwaRe power optimization in data center networks," in *IEEE INFOCOM, 2012*, pp. 1125–1133.

[7] N. H. Thanh, P. N. Nam, T.-H. Truong, N. T. Hung, L. K. Doanh, and R. Pries, "Enabling experiments for energy-efficient data center networks on openflow-based platform," in *Communications and Electronics (ICCE), 2012 Fourth International Conference on*. IEEE.

[8] B. B. Rodrigues, A. C. Riekstin, G. C. Januário, V. T. Nascimento, T. C. Carvalho, and C. Meirosu, "Greensdn: Bringing energy efficiency to an sdn emulation environment," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May, pp. 948–953.

[9] H. Zhu, X. Liao, C. de Laat, and P. Grosso, "Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform," *Journal of Network and Computer Applications*, vol. 63, pp. 110 – 124, 2016.

[10] R. Ana C, J. Guilherme C., R. Bruno B, N. Viviane T, C. Tereza C.M.B., and M. Catalin, "Orchestration of energy efficiency capabilities in networks," *Journal of Network and Computer Applications*, vol. 59, pp. 74 – 87, 2016.

[11] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford, "Identifying and Using Energy-critical Paths," in *Seventh Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT, 2011. ACM.

[12] T. H. Vu, V. C. Luc, N. T. Quan, N. H. Thanh, and P. N. Nam, "Energy saving for OpenFlow switch on the NetFPGA platform based on queue engineering," *SpringerPlus*, vol. 4, no. 1, p. 64, 2015.

[13] 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.

[14] F. Giroire, J. Moulierac, T. K. Phan, and F. Roudaut, "Minimization of network power consumption with redundancy elimination," vol. 59. Elsevier, 2015, pp. 98 – 105.

[15] W. Rui, SuixiangGao, Y. Wenguo, and J. Zhipeng, "Bandwidth-aware energy efficient routing with sdn in data center networks," in *Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*, pp. 766–771.

[16] T. Nguyen *et al.*, "Modeling and Experimenting Combined Smart Sleep and Power Scaling Algorithms in Energy-aware Data Center Networks ," *Simulation Modelling Practice and Theory*, vol. 39, 2013.

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

[18] A. Fernandez-Fernandez, C. Cervello-Pastor, and L. Ochoa-Aday, "Achieving energy efficiency: An energy-aware approach in sdn," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec.

[19] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, p. 19.

[20] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007)*.

[21] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11 n power consumption," in *Proceedings of the 2010 international conference on Power aware computing and systems*, October.