Linear Programming Approaches for Power Savings in Software-defined Networks

Moghaddam, F.A.; Grosso, P.

DOI
10.1109/NETSOFT.2016.7502448

Publication date
2016

Document Version
Final published version

Published in
2016 IEEE NetSoft Conference and Workshops : NetSoft 2016

Linear Programming Approaches for Power Savings in Software-defined Networks

Fahimeh Alizadeh Moghaddam
SNE & S2 groups
University of Amsterdam & VU University Amsterdam
Amsterdam, The Netherlands
Email: f.alizadehmoghaddam@uva.nl

Paola Grosso
System and Network Engineering group (SNE)
University of Amsterdam
Amsterdam, The Netherlands
Email: p.grosso@uva.nl

***Abstract*—Software-defined networks have been proposed as a viable solution to decrease the power consumption of the networking component in data center networks. Still the question remains on which scheduling algorithms are most suited to achieve this goal. We propose 4 different linear programming approaches that schedule requested traffic flows on SDN switches according to different objectives. Depending on pre-defined software quality requirements such as delay and performance, a single variation or a combination of variations can be selected to optimize the power saving and the performance metrics. Our simulation results demonstrate that all our algorithm variations outperform the shortest path scheduling algorithm, our baseline on power savings, less or more strongly depending on the power model chosen. We show that in FatTree networks, where switches can save up to 60% of power in sleeping mode, we can achieve 15% minimum improvement assuming a one-to-one traffic scenario. Two of our algorithm variations privilege performance over power saving and still provide around 45% of the maximum achievable savings.**

I. Introduction

Networking devices are one of the main contributors to power consumption in data centers. The deployment of large number of non-energy proportional switches and links, whose average utilization rate is only around 30% [1] leaks significant amount of energy. Software-defined networks (SDN) are replacing traditional networks due to their advantage of providing programmability and controllability of the underlying network. As Infonetics Research forecasts [2], there will be 87% growth in SDN deployment in North American-based enterprises by 2016. Given SDNs wide range of applications in data center networks (DCN), they are often chosen as part of the power efficient solutions. In an earlier systematic literature review [3], we observed a growing trend in using SDNs combined with energy awareness. However, the effectiveness of SDNs with respect to energy efficiency are not fully studied and it is not determined how the programmability of networks can be a added value in this matter. Besides, scalability and performance cost of energy efficiency optimization are still open questions.

Linear programming algorithms are recently deployed to perform as a scheduler for the incoming flows on the available paths [4]–[10]. Finding the optimum solution, e.g. mapping the flows to the paths, is an NP-hard problem. In this paper, we will show it is possible to reduce the complexity by splitting and adjusting the problem. We derive four scheduling algorithms that take into account a number of constraints to ensure their objective values. We implement a modular decision framework to evaluate our algorithm variations. We assess to what extent the scalability quality requirement is fulfilled by performing experiments in network of different sizes, different numbers of flows, and different characteristics of switches.

978-1-4673-9486-4/16/$31.00

II. Related Work

Power and energy efficiency of DCNs has often been a by-product of other optimization strategies. The main focus in data centers has been on energy-aware virtual machine (VM) placement, and the underlying network is treated as the side problem in terms of providing power savings. In this case, power savings in the network usually are achieved by traffic consolidation or traffic locality [4], [11]. Solely focused on the networking component, [12] introduced a distributed flow scheduling scheme suitable within a DCN. Still, there are no complete studies made on the trade-off between the optimization in the network component and the application related objectives. We instead focus specifically on this, with the goal of providing application developers that intend to program the network a clear overview of the pros and cons of their algorithms choices.

There are other studies that, like ours, deploy SDNs for energy-aware purposes. ElasticTree [7] designs a centralized decision framework to turn off the idle network devices. A disadvantage of this model is the multi-minute booting time of switches. Differently, we use in our simulation the sleeping mode of switches; this takes much shorter time (around 1s) to turn them back on. There are similar approaches that deploy heuristic scheduling algorithms [5], [13], [14]. They mostly provide the traffic matrix as an input to the scheduling algorithm, which is not always a realistic scenario. Contrarily, we focus on real time flow requests while keeping the scalability quality requirement in mind.

III. The Flow Scheduling Algorithm

We deploy the Integer Linear Programming model, which defines the problem as a linear function of different variables and the values selected for variables need to meet the limitations of pre-defined constraints. We aim to schedule

the incoming network traffic with power saving intentions. Reducing the volume of the traffic by intelligent endpoint placements is out of scope of this research. Therefore, we assume the endpoints are known beforehand.

A. Objectives

For clarity we summarized the notations used in our equations in Table I. There are three objectives that can be considered when scheduling traffic flows. Table II displays the equations for each objective.

TABLE I: The notations used in the equations

Notations	***Description***
PC	The sum of the power consumption of active switches
BW_i	The achievable bandwidth for the ith flow
m	The number of active switches
n	The number of requested flows
l	The number of *Combinations*
transition_degree	The number of switches that need to be turned on for the incoming load

To minimize the total power consumption, we place our focus on putting the idle network devices into the sleeping mode. Switches in the sleeping mode still consume energy but depending on the adopted technologies, the power savings will vary. Objective #2 prioritizes the already existing active switches over the sleeping ones. It shortens the duration of the scheduling/modification phase although the achieved bandwidths for the flows might be reduced due to link sharing. Objective #3 emphasizes on application performance. In this work, we assume that all the applications have the same priority. Therefore, we optimize the bandwidth requests with fairness.

TABLE II: Objectives taken into account in our algorithm variations

#	***Objectives***	***Equations***
1	Minimize the power consumption of the data center network	$Minimize(PC = \sum_{j=1}^{m} PC_j)$
2	Minimize the number of transitions from the sleeping mode to the active mode	$Minimize(transition\_degree)$
3	Maximize the bandwidth for the flows	$Maximize(\sum_{i=1}^{n} BW_i)$

B. Algorithm Variations

Variations in application-level quality requirements might privilege one or more of the objectives. However, it is important to note that all the variations are power efficient due to the objective #1. We identified four distinct algorithm variations and their objective variables in Table III. In order to avoid zero denominator in the fraction, 1 is added to *transition_degree* in the equations.

TABLE III: Our LP algorithm variations

Algorithm Variations	***Objectives***			***Objective Variable***
	#1	#2	#3	
LP-v1: *Full version*	✓	✓	✓	$\frac{\sum_{i=1}^{n} BW_i}{(transition\_degree+1)*\sum_{j=1}^{m} PC_j}$
LP-v2: *Without priority*	✓		✓	$\frac{\sum_{i=1}^{n} BW_i}{\sum_{j=1}^{m} PC_j}$
LP-v3: *Only throughput guaranteed*			✓	$\sum_{i=1}^{n} BW_i$
LP-v4: *Only energy efficient*	✓	✓		$(transition\_degree + 1) * \sum_{j=1}^{m} PC_j$

C. Algorithm Implementation

We define *combination* to model our linear programming problem. A *combination* consists of a set of selected paths for the requested flows. *CombinationList* is a list of combinations, differing in the selected paths and consequently in the achieved bandwidths and the total power consumption. *Combination* selection is done by comparing them based on their objective variable (See Eq. 1). We calculate the necessary metrics (the maximum power consumption, the maximum bandwidths or *transition_degree*) for each combination. In case of variations LP-v1, LP-v2 and LP-v3, the objective function is to maximize the objective variables, whereas it minimizes the objective variable in case of variation LP-v4. We define the constraints ($\sum_{k=1}^{l} x_k = 3$ and $\sum_{k=1}^{l} x_k = 1$) to ensure that first top 3 combinations are selected and then only 1 combination is selected.

$$\sum_{k=1}^{l} x_k * (\text{Combination objective variable}_k) \quad (1)$$

As new flows request arrive, the combinations will grow both in size and in number. Unlike the growth in size, the growth in number is more accelerated. Product of number of possible paths for each requested flow calculates the total number of combinations. We store only three combinations for each request arrival due to scalability reasons.

D. Scalability Analysis

The algorithm has two phases. First, combinations will be added based on the number of possible paths. Second, each new combination needs to be updated in terms of its performance metrics. The execution time for the first phase of the algorithm is $O(n)$, n being the number of possible paths for each step. Since n is usually a small number in (especially the FatTree) DCNs and does not grow rapidly along with the network size, it is reasonable to run the algorithm upon each flow request. The second phase of the algorithm updates the performance metrics of a subset of the active switches that will carry the incoming load and it executes in $O(1)$. For each new possible path, active switches can be updated/modified in constant time, which makes our algorithm very scalable with the number of flow requests and the size of the network.

IV. Design of Evaluation Decision Framework

We design an evaluation decision framework (See Fig. 1), whose task is to deploy the different variations of the scheduling algorithm. The decision framework consists of three main modules: 1) Scheduler, 2) Controller, and 3) Monitor.

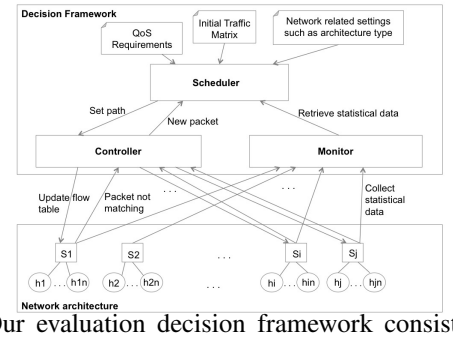


Fig. 1: Our evaluation decision framework consisting of the scheduler, the controller and the monitor components

Scheduler: All the information from other modules and external data sources are inputs to this module. The scheduler relies on two types of inputs: offline, and online. Monitoring data on power consumption and bandwidth of switches are online inputs. Offline input is given to the module at the initial time and will not change during the runtime. The offline inputs by the scheduler are:

1) ***QoS requirements***: An application will provide the minimum quality requirements, which will be added as constraints to our linear programming approaches.

2) ***Initial traffic matrix***: It is possible to provide the scheduler with the traffic setup at the initial time. We did not use this feature in our experiments.

3) ***Network architecture***: The scheduler is told at the beginning which network topology is deployed in the data center. In our experiments, we use the FatTree topology.

Controller: The controller component receives the requests for setting new flows from the underline devices. The controller later applies modifications to the flow tables of the switches for the requested flows.

Monitor: This component collects periodically statistical information from the network. Reports provided by the monitor are further used by the scheduler component. If realtime monitors are not present, it is possible to calculate the power consumption of the switches based on formalized assumptions. This is what we did in our experiments.

Realtime and online inputs are produced by the network devices and are passed to the scheduler through the controller and the monitor components during runtime.

V. Simulation Scenarios

We implemented a number of experiments in a simulation environment using our decision framework. To simulate the network architecture, we use Mininet[1]. For the controller component we use the open-source POX control software[2]. We implement a combination of the Gurobi Python optimizer[3] and POX as our optimization solver in the scheduler component. We adopted the FatTree topology in our simulations, which is the most widely deployed network architecture in DCNs [3]. We define network architectures of variable sizes namely, 20, 45 and 80 switches all connected with links of 1Gbps, summarized in Table IV. Further information can be found in an extended version at [15].

TABLE IV: The three simulation configurations of the FatTree network architecture used in our simulation

Switch ports	Switches	Hosts	PODs	Maximum Flows
4	20	16	4	8
6	45	54	6	27
8	80	128	8	64

A. Performance Metrics

Performance metrics help us evaluate through our experiments the achievable power savings. We focus on two performance metrics:

- *Power consumption*: We adopted this power model in our simulations, which is based on the model proposed by Mahadevan *et al.*[16]:

$$Power_{switch} = Power_{chassis} + num_{linecards} * Power_{linecard} + \sum_{i=0}^{configs} (Power_{configsi} * \sum_{j=0}^{numports} utilizationFactor_j) \quad (2)$$

  $Power_{linecards}$ represents the power consumption of the linecard and $num_{linecards}$ is the number of plugged-in cards. $Power_{configsi}$ is the power consumption of a port with the specified link rate i and $utilizationFactor_j$ is the utilization of port j of the switch. In our experiment, we assume that each flow sends data with the highest possible bandwidth. Therefore, it is always possible to calculate the utilization rates of the switch ports.
- *Time to complete (TTC)*: TTC is the time it takes for a host to send predefined number of bytes to another host in the network. It is a representative QoS requirement as it is aligned with response time and performance.

B. Traffic Patterns

We generate the traffic in the DCN based on *One-to-One*, having all the hosts in pairs [13], [17]. We implement the *Far* traffic study case, where nodes with longest distance transfer data. Since the Far traffic involves more number of switches and less switches can be turned to the sleeping mode, it is a more interesting scenario to validate our algorithms.

VI. Results

We focus on an extension of SP, which we call Smart SP to define the minimum and maximum values of our performance metrics. Smart SP is an implementation of a non-energy efficient version of our LP-v3. It does not put idle devices into the sleeping mode, but it maximizes the bandwidth for all the flows. Table V summarizes the power consumption of SP and Smart SP. As expected, they show a same range of power consumption. The little variations we observe is due to different utilization factors given the path chosen will not be always the same.

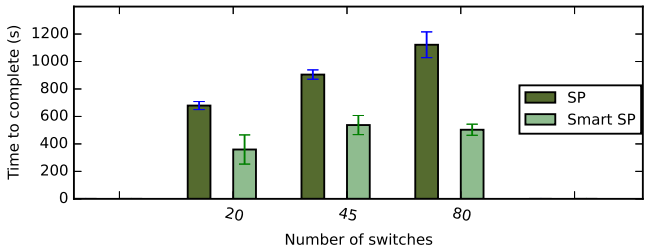


Fig. 2: Time to complete for SP and Smart SP scheduling algorithms in the three simulated network topologies (38GB of data)

Fig. 2 shows the TTC metric for the two candidate baseline scheduling algorithms. The TTC of Smart SP is lower than the one of SP. In fact Smart SP, which makes intelligent decisions regarding bandwidth achievements, effectively ends

[1]http://mininet.org
[2]http://www.noxrepo.org/pox/about-pox/
[3]http://www.gurobi.com/

TABLE V: Total power consumption of SP and Smart SP in the three simulated network topologies (20,45 and 80 switches)

Baseline candidates	Total power consumption (20 switches)	Total power consumption (45 switches)	Total power consumption (80 switches)
SP	3032W	6856W	12114W
Smart SP	3039W	6847W	12198W

up providing higher bandwidths to the requested flows. Given their equivalent power consumption and the better TTC of Smart SP, we adopt this algorithm rather than SP as baseline algorithm.

The achievable power saving in the sleeping mode will heavily influence the total power saving in the data center. To quantify this, we first examined the power consumption of the four scheduling algorithms under different power savings percentages in the sleeping mode. In the rest of this paper we will use 60% as the power saving of the devices in the sleeping mode[4]. Table VI shows the maximum power saving of each algorithm compared to Smart SP in the network of size 45 with 27 flows. In all cases, the maximum power saving is achieved by LP-v4, followed by LP-v1, LP-v2 and LP-v3. There is a nearly linear relation between the maximum achievable power saving and the maximum sleeping mode power saving. 20% improvement of power saving in sleeping mode results in 10% improvement for power sensitive algorithms (LP-v1 and LP-v4) and 5% improvement for performance sensitive algorithms (LP-v2 and LP-v3).

TABLE VI: Maximum power savings as function of sleeping mode power savings (45 switches with 27 flows)

Algorithm variations	Maximum sleeping mode power savings:			
	20%	40%	60%	80%
LP-v1	12%	23%	35%	46%
LP-v2	6%	12%	17%	23%
LP-v3	5%	11%	16%	22%
LP-v4	13%	24%	36%	48%

To assess the scalability quality requirement of the algorithms we investigated the power consumption of the four algorithm variations when increasing the flow numbers. Fig. 3 shows the results for a network of size 20, 45 and 80 respectively. In all cases LP-v4 and Smart SP present the lowest and highest values, identifying the power consumption bounds. The subfigures show that LP-v1 selects paths such that the total power consumption remains close to the optimum variation (LP-v4). Gradually, with higher number of flows, LP-v1 and LP-v4 diverge as LP-v1 takes into account the non-energy related quality requirements too. LP-v2 and LP-v3 show almost identical patterns when increasing the number of flows. Similar to square-root functions, they change rapidly in terms of power consumption for the small number of flows particularly in case of 45 and 80 switches. Their power consumption shows less acceleration for larger number of flows because there are no more switches to turn on from the sleeping mode. When the number of flows is small all the algorithm variations show the same power consumption; in this case it will not be possible to provide power savings from shaping the network traffic.

Fig. 4 shows the TTC measurements as function of the four scheduling algorithms when running with maximum number of flows. We configure applications to send 38GB of data to the receiver. Algorithm variations that perform better in power consumption exhibit higher TTC measurements, e.g. LP-v4 that focuses purely on the energy efficiency will produce a much larger TTC, 71% for the network of 20 switches and 502% for the network of 80 switches. LP-v1 also shows a considerable increase in TTC when the network size grows. Differently, LP-v2 and LP-v3 appear to be more performance-focused and stable.

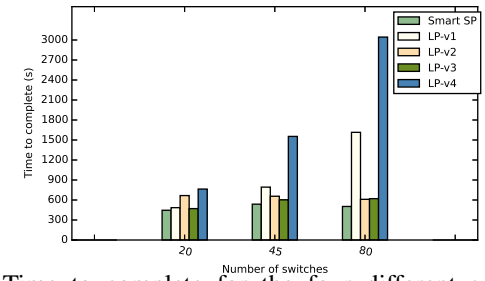


Fig. 4: Time to complete for the four different scheduling algorithm in the three simulated network topologies (38GB of data)

If performance is the decisive factor, LPv2 and LPv3 will be the likely choice. Fig. 5 shows the degradation in power saving compared to the max/min values (LP-v4 and Smart SP). We observed that LP-v1 is the most power efficient variation that shows around 95% improvement, which means only 5% degradation from the optimum power saving. LP-v2 with achieving 50% of maximum power saving outperforms LP-v3 with achieving 45% of maximum power saving, which provides more bandwidth for the running applications. It is interesting to see that all the variations remain with the same range of power saving for different network sizes.

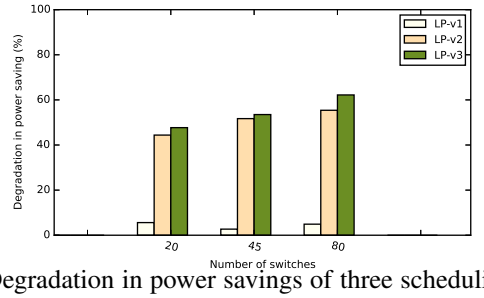
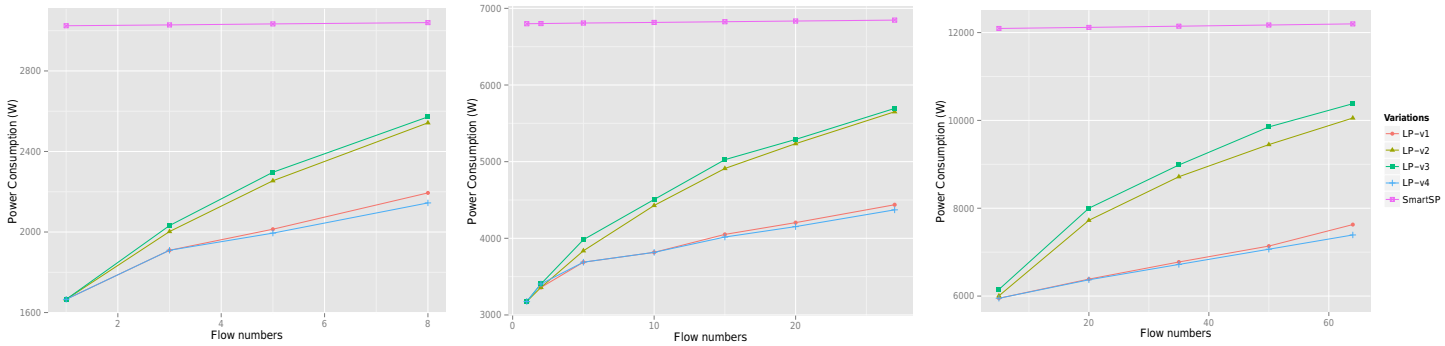


Fig. 5: Degradation in power savings of three scheduling algorithms namely, LP-v1, LP-v2 and LP-v3 in the three simulated network topologies. LP-v4 and Smart SP are considered as the baselines for calculation of power saving.

VII. Discussion

Applications running in data centers have different quality requirements. Our results can be used as inputs for them to decide on which algorithm to implement for flow scheduling in SDNs. As shown in section VI, our four algorithm variations show different behavior as function of the size of the network, the number of requested flows, and the expected power saving of the switches when in the sleeping mode. LP-v2 and LP-v3 provide higher bandwidth for applications and smaller time to complete, while still saving power compared to SP. This makes them a great choice for delay-sensitive applications. LP-v1 and LP-v4 focus on the energy efficiency quality requirement rather than the time to complete of applications, as such they can

[4]"Cisco catalyst 2960-x, 2960-cx, and 3560-cx platforms: The greenest catalyst switches ever," 2015, [Online; accessed 14-Mar-2016]



(a) FatTree network with 20 switches (b) FatTree network with 45 switches (c) FatTree network with 80 switches

Fig. 3: Power consumption of the DCN for different algorithm variations with variable number of flows based on the one-to-one traffic scenario for a fixed network size

be adopted for delay-insensitive applications. In a data center we will often see a combination of delay-sensitive and delay-insensitive applications. In this case, our framework can be easily extended to deploy multiple variations of LP algorithm concurrently.

Our findings show a nearly linear relation between the total power savings in DCN and the power saving of switches in sleeping mode. It is important to assess the switches power saving in their sleeping mode beforehand, which helps the scheduling algorithms adjust their decisions. Switches with lower sleeping mode power savings are prioritized to carry the incoming load either by performance sensitive algorithms (LP-v2 and LP-v3) or power sensitive algorithms (LP-v1 and LP-v4).

VIII. Conclusion

In this paper we presented four variations of linear programming scheduling algorithms that can optimize the power savings in SDNs. We follow three power saving approaches: "put the idle devices into sleeping mode", "increase the number of idle devices", "prioritize the existing active switches over the sleeping ones". Our simulation results show that two of the variations (LP-v2 and LP-v3) remain stable in terms of power saving and the time to complete metrics, as the network size grows. Two other (LP-v1 and LP-v4) provide the highest power savings in the network and they are suitable for delay-insensitive applications. Since many applications will exhibit a one-to-many traffic pattern, it would be interesting to identify power savings of our approaches in future. Furthermore, our current implementation of the scheduler treats all flows equally in terms of achievable bandwidth. We would run a combination of short- and long-lived flows.

Acknowledgment

This work has been sponsored by the RAAK/MKB-project "Greening the Cloud" and by the Dutch national program COMMIT.

References

[1] J. Liu, F. Zhao, X. Liu, and W. He, "Challenges towards elastic power management in internet data centers," in *Distributed Computing Systems Workshops, 2009. ICDCS Workshops' 09. 29th IEEE International Conference on*. IEEE, 2009, pp. 65–72.

[2] C. Grossner, "Sdn strategies survey," July 2014, [Online; accessed 14-Dec-2015].

[3] F. A. Moghaddam, P. Lago, and P. Grosso, "Energy-efficient networking solutions in cloud-based environments: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 64, 2015.

[4] B. Kantarci, L. Foschini, A. Corradi, and H. T. Mouftah, "Inter-and-intra data center vm-placement for energy-efficient large-scale cloud systems," in *Globecom Workshops (GC Wkshps), 2012 IEEE*. IEEE, 2012, pp. 708–713.

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

[6] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, and N. Pissinou, "Joint host-network optimization for energy-efficient data center networking," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*. IEEE, 2013, pp. 623–634.

[7] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks." in *NSDI*, vol. 10, 2010, pp. 249–264.

[8] B. Kantarci, L. Foschini, A. Corradi, and H. T. Mouftah, "Design of energy-efficient cloud systems via network and resource virtualization," *International Journal of Network Management*, 2013.

[9] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Energy aware networked cloud mapping," in *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*. IEEE, 2013, pp. 195–202.

[10] J. Buysse, K. Georgakilas, A. Tzanakaki, M. De Leenheer, B. Dhoedt, C. Develder, and P. Demeester, "Calculating the minimum bounds of energy consumption for cloud networks," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*. IEEE, 2011, pp. 1–7.

[11] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[12] R. Liu, H. Gu, X. Yu, and X. Nian, "Distributed flow scheduling in energy-aware data center networks," *Communications Letters, IEEE*, vol. 17, no. 4, pp. 801–804, 2013.

[13] M. Xu, Y. Shang, D. Li, and X. Wang, "Greening data center networks with throughput-guaranteed power-aware routing," *Computer Networks*, vol. 57, no. 15, pp. 2880–2899, 2013.

[14] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*. ACM, 2010, pp. 1–8.

[15] F. A. Moghaddam and P. Grosso, "Linear programming approaches for power savings in software-defined networks (the extended version)," *arXiv preprint arXiv:1603.04307*, 2016.

[16] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *NETWORKING 2009*. Springer, 2009, pp. 795–808.

[17] S. U. Khan and A. Y. Zomaya, *Handbook on Data Centers*. Springer, 2015.