An Energy Saving Mechanism Based on Vacation Queuing Theory in Data Center Networks

Emna Baccour[1(✉)], Ala Gouissem[1], Sebti Foufou[2,3], Ridha Hamila[1], Zahir Tari[4], and Albert Y. Zomaya[5]

[1] College of Engineering, Qatar University, Doha, Qatar
ebaccour@qu.edu.qa
[2] LE2i Lab, University of Burgundy, Dijon, France
[3] Computer Science, New York University, Abu Dhabi, UAE
[4] School of Science, RMIT University, Melbourne, Australia
[5] School of Information Technologies, University of Sydney, Sydney, Australia

Abstract. To satisfy the growing need for computing resources, data centers consume a huge amount of power which raises serious concerns regarding the scale of the energy consumption and wastage. One of the important reasons for such energy wastage relates to the redundancies. Redundancies are defined as the backup routing paths and unneeded active ports implemented for the sake of load balancing and fault tolerance. The energy loss may also be caused by the random nature of incoming packets forcing nodes to stay powered on all the times to await for incoming tasks. This paper proposes a re-architecturing of network devices to address energy wastage issue by consolidating the traffic arriving from different interfaces into fewer ports and turning off the idle ones. This paper also proposes to attribute sleeping and active periods to the processing ports to prevent them from remaining active waiting for random arrivals. Finally, we use the vacation queuing theory to model the packets arriving process and calculate the expectation of vacation periods and the energy saved. Then, we strengthen our work with a simulation part that validates the analytical derivations and shows that the proposed mechanism can reduce more than 25% of the energy consumption.

Keywords: Power consumption · Vacation queuing theory
Data center networks

1 Introduction

In the last few years, data centers witnessed a revolutionary growth caused by the increasing trend to migrate services, applications, computation and storage into more robust systems. Due to this rapid scaling of data center networks, the unavoidable increase of energy consumption became a challenging problem.

Studies on data center traffic statistics showed that, most of the time, the network operates only at 5% to 25% of its maximum capacity [3,11] depending

J. Hu et al. (Eds.): MONAMI 2017, LNICST 235, pp. 188–202, 2018.
https://doi.org/10.1007/978-3-319-90775-8_16

on the period. Also, the network devices have to be always powered on to wait for the unpredictable incoming jobs. However, when the load is low, the idle servers consume 70% of their peak power which causes a great waste of energy [10]. In this context, many research efforts such as ElasticTree [11] tried to make the power consumption proportional to the traffic workload by powering on only the active nodes. These efforts showed interesting results, most of them, however, are based on the traffic load, which is unpredictable. Also, since the traffic load is known to be bursty [11], energy conservation is not always significant especially when the load is high. In addition, defining the set of crucial nodes for the communication and the set of idle nodes to power off is a complex task that has an exponential time.

In order to conceive an energy-aware mechanism independent from the traffic, two facts should be considered. First, duplicating the critical components such as links, ports and servers (also known as *redundancies*) to backup the principal resources in case of failures and insure the network bandwidth may largely contribute in the energy wastage. In fact, maintaining many ports active waiting for packets in a network device can consume a large proportion of power. Although load balancing and fault-tolerance are important especially in heavy traffic loads, it is acceptable to make it optional as most of the time the network does not reach its maximum capacity. Second, the power sleep mode is the key point for energy efficiency. Thus, by limiting the active time of a device and fixing a vacation time, an important amount of energy can usually be saved.

The main contributions of this paper can be summarized as follows:

- Re-architecturing the network devices so that the incoming load is not treated necessarily by its input port but can be relayed to any available processing unit. In fact, we propose to separate the receiving interface from its processing part. In this way, any processing unit can treat the load incoming from any interface. Hence, in low loads, the traffic can be satisfied by only few units and the redundant ports are activated only in need.
- Proposing a packet scheduling algorithm to manage the distribution of tasks and vacation times between different processing units according to their availabilities. In particular, when the buffer of a particular unit is congested, the incoming packets are relayed to the next units. When the buffer is empty, the unit switches to energy efficient mode instead of idle mode.
- Analyzing this approach using the vacation queuing model to expect the vacation period of each unit, the load distribution between units according to the incoming packets, the energy gain compared to the always-on system (system without vacation) and the waiting time of packets in the queues.

The simulation results showed that the proposed approach can achieve the proportionality between the energy consumption and the traffic load. In addition, more than 50% of energy can be reduced in low loads and more than 25% in higher loads while respecting the system performance. Also, compared with the power aware algorithms proposed for data centers, our model owns a much better time and calculation efficiency.

The rest of the paper is organized as follows: Sect. 2 overviews some of the existing works to green data center networks. In Sect. 3, we describe the proposed approach to optimize the energy consumption. An experimental evaluation is provided in Sect. 4 and we conclude the paper in Sect. 5.

2 Related Work

Several industrial and academic investigations have been conducted to build a green data center. Some of them chose to use renewable sources of energy instead of brown power such as google data centers [2]. Others suggested to implement power efficient designs such as introducing wireless technology in the data center networks [6]. However, most of the efforts are focusing on saving the energy consumed by idle devices that are not included in the traffic but are still wasting energy. In fact, these works aim to consolidate the arriving traffic flows, restrict the network to a subset of devices and power off the idle nodes. Such approach is studied from different perspectives including routing and queuing perspectives. A short review of related works is summarized as follows:

2.1 Routing Level Power Aware Approaches

ElasticTree. [11] aimed to find the minimum subset of the network that must be kept active and the set of nodes that are unused and can be shut down. This approach consists of three modules: the optimizer that defines the devices contributing in the traffic communication, the routing module that calculates the packets routes and the power control module which is responsible to adjust the state of devices (on, off). Three optimizers are proposed: the first optimizer aims to find the optimal power saving solution by searching the optimal flow routes while respecting the traffic and performance constraints. However, searching the optimal solution is an NP-hard problem and needs a large computation complexity. Hence, two other optimizers are proposed where the optimal solution is not guaranteed. These optimizers either depend on a known network design or give a non-optimal routing paths to minimize the searching time.

Vital Nodes Approach. [5] suggested not to calculate the best routing paths in real time when receiving the traffic pattern. Instead, the network is abstracted to a graph and vital nodes between any two communicating clusters in this graph are calculated using different methods (betweenness, closeness, degree, etc.). These nodes are pre-calculated once when conceiving the network and used directly with a constant computation complexity. At a given time t, when receiving the traffic matrix, only the vital nodes are kept active.

2.2 Queuing Analysis for Power Aware Approaches

Queuing theory is a deeply established analysis that helps to predict the workloads, the performance change, the traffic volumes and scenarios. Few efforts use the queuing models in data centers including:

Task Managing Based on Vacation M/G/1 Queuing Model. [8] where the packet scheduling in a data center using an M/G/1 queuing analyze is modeled. The traffic is consolidated into the minimum set of servers and the others are switched off until the receiving of packets. Sejourn time of the packets in the system (time passed in the queue) is calculated and proven to be acceptable while gaining a large amount of energy. Still, since the modeled queue has an unlimited length, real data center scenarios can not be well studied (e.g. congestion, drop rate).

Task Managing Based on Vacation M/M/n Queuing Model. [12] proposes a threshold oriented model to reduce the energy consumed by servers in three-tier data centers. Specifically, authors fix an initial number of active servers and power off the others. Then, they keep examining the arriving jobs in the queue. If the queue size reaches a certain threshold, some extra servers must be activated. The optimal trade-off between the power saving and waiting time in the queue is determined by the M/M/n analytical model.

3 Proposed Approach

Most of the suggested solutions to green a given network propose to put the interfaces into a lower energy rate depending on the traffic load. In fact, based on the traffic matrix received at an instant t (servers sending and receiving the communication flows), the set of nodes to power off and the one to keep active are determined. However, the fundamental problem is the unpredictability of the incoming traffic. Also, because of the burst nature of the traffic, the energy is saved only for low loads. To overcome this challenge, we propose to re-architecture the network devices and to use a Sleep/Active algorithm to minimize the dependency on the traffic load and make the energy saving dependent only on the incoming packets at an instant t. The idea is to attribute vacation periods to the ports that are not receiving any job. To maximize the number of sleeping ports, the interface level (ports receiving the data) is separated from the processing level (units processing the routing decisions), in such a way, the packets received from different interfaces are processed by a minimum number of active units.

3.1 Re-architecturing the Network Devices

The objective of re-architecturing the network devices is that when a number of packets arrive to n interfaces (n is the number of ports per device), they can be treated by a smaller number of processing units. In an n-port device (switch, server,...), each input/output interface is connected to an intelligent unit that processes, extracts and decodes the packet header, looks up to the destination address and decides what to do with it (forward or drop) [9]. The key idea is to decouple each interface from its processing unit. The interfaces level will simply be responsible for receiving, gathering, and forwarding packets to the controller

level. A controller level is implemented to decide what is the available unit to process the packets by respecting the Sleep/Active algorithm described in Sect. 3.2. The Sleep/Active algorithm schedules the distribution of packets among different units based on the queue length and attributes sleeping periods to idle units. The processing unit level is responsible to process, and decode the packets. If the queue of one unit is congested, it notifies the controller to forward the incoming packets to the next unit. In this way, in low loads, the traffic incoming/directed to n ports can be handled by a lower number of processing units and the idle ones can be turned into sleep status (a) to save considerable amount of energy, (b) to reduce the dependency on the unpredictable load and (c) to reduce the computation complexity. Figure 1(b) shows the proposed re-architectured device. Unlike the conventional network device, presented in Fig. 1(a), where every interface has its own processing unit to run routing protocols and decide how to forward packets, routing decisions are stripped from interfaces.

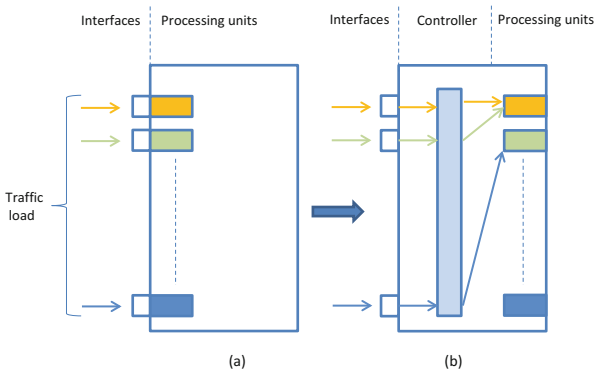

Fig. 1. Re-architecturing the network devices.

This new architecture requires new hardware design. In fact, a similar hardware has been proposed and implemented for the Software-Defined Networking (SDN) [13]. An SDN switch separates the data path (packet forwarding) from the control path (routing decisions). The data path portion resides on the switch; a separate controller makes routing decisions. As SDN becomes a trend for cloud computing, the industry is paying more attention to this decoupled hardware including Openflow controllers [1]. Therefore, the proposed re-architectured devices can be available to test our approach in a real network.

3.2 Sleep/Active Algorithm

Given the proposed re-architecturing described above, it now becomes possible to merge packets from multiple interfaces to be processed by few units. However, an algorithm to manage the distribution of tasks and vacation times between

different processing units is important to maximize the sleeping units and, hence, minimize the energy waste.

In the initial stage, all processing units are put into sleep mode. After a period of vacation, say V_1, their buffers levels are examined. If there are waiting packets, the related processor will be activated, otherwise, it remains in sleeping mode for another period, say V_2. The first packets communicated through different interfaces are routed automatically to the first processing unit. Whenever a congestion occurs, i.e. buffer level exceeds a congestion window K, packets will be routed to the adjacent processing unit.

Each processing unit can experience four states: *sleep*, *listening*, *recovery* and *active*. The transition between the *sleep* and *active* state is performed according to the vacation period (say V) and the result of the listening period (say T_l). When the vacation period elapses, the unit is switched to *listening* state where it examines the buffer status and listens in case of incoming or awaiting packets. If the buffer is empty and there is no appearing traffic load, the processing unit returns to *sleep* state. The *listening* state is performed at the end of every vacation period under a low rate. The passage to the *active* state is triggered when the listening indicates that there are waiting packets. To start serving packets, the unit passes by a recovery period (say T_w) where it warms-up between sleeping and active periods. In this paper, the active period lasts for a fixed period (say A) and then the unit passes automatically to sleep status, even if the buffer is not empty. The active period consists of multiple service times (times to process k packets), denoted $s_1, ..., s_k$. Figure 2 provides a summary of the transitions between different status of the processing unit.

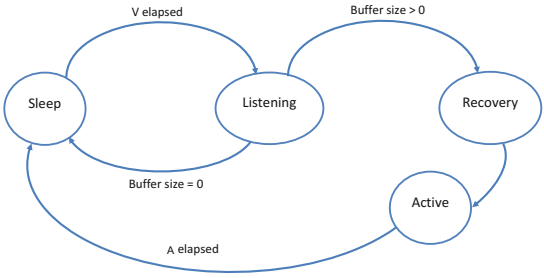

Fig. 2. Processing unit state diagram

3.3 Vacation Queuing Model

In this section, we will analyze our system from a queuing perspective. We will consider an $M/G/1/K$ queue in which the processing unit goes on vacation for predefined periods after a fixed active period A. Packets are assumed to arrive according to an independent Poisson process with a rate equal to λ and a distributed queue service time equal to μ.

Expectation of Sleeping Period: In this part, the expectation of the sleeping period S for one processing unit is computed. The sleeping period consists of N vacation periods denoted as $V_1, \cdots, V_N$, where V_N is the last vacation period, which means that the listening process reported the arrival of packets. Let E_i represent the event of no arrivals during the period V_i and E_i^c denote the complementary event. To calculate the expectation of the sleeping period S, we need first to compute the distribution of the number of vacations N. The expected value of N is given by the following expression:

$$E[N] = \sum_{i=0}^{\infty} iP(N = i). \tag{1}$$

Once we calculate the distribution of N, we can compute the expectation of S which is composed of N vacation periods $(V_1, \cdots, V_N)$. In this context, since we are dealing with a non-negative random value V_i, we can use the Laplace Stieltjes transform of V_i which is very useful to simplify calculations in the applied probabilities and queuing theory. The Laplace Stieltjes transform of V_i can be written as $L_{V_i}(s) = E[e^{-sV_i}]$. The probability of having a certain number of vacations can be calculated as follows:

$$\begin{aligned}
&P(N = 1) = P(E_1^c) = E[1 - e^{-\lambda V_1}] = 1 - L_{V_1}(\lambda).\\
&P(N = i) = P(E_1).P(E_2).P(E_3)\ldots P(E_i^c) = (1 - L_{V_i}(\lambda)).\prod_{j=1}^{i-1} P(E_j)\\
&\qquad = (1 - L_{V_i}(\lambda)).\prod_{j=1}^{i-1} L_{V_j}(\lambda).\\
&P(N \geq i) = P(E_1).P(E_2).P(E_3)\ldots P(E_{i-1}) = \prod_{j=1}^{i-1} P(E_j) = \prod_{j=1}^{i-1} L_{V_j}(\lambda).
\end{aligned} \tag{2}$$

Using Eq. (1), the expected *number of vacations* can be defined as:

$$E[N] = \sum_{i=0}^{\infty} iP(N = i) = \sum_{i=0}^{\infty} P(N \geq i) = \sum_{i=0}^{\infty} \prod_{j=1}^{i-1} L_{V_j}(\lambda). \tag{3}$$

We assume that the vacation periods are mutually independent and only depend on the no arrival of packets in the listening period. Hence, the expectation of the sleeping period can be calculated as follows:

$$S = \sum_{i=1}^{N} V_i = \sum_{i=1}^{\infty} V_i \mathbb{1}_{\{N \geq i\}}, \qquad E[S] = \sum_{i=1}^{\infty} E[V_i] \prod_{j=1}^{i-1} L_{V_j}(\lambda). \tag{4}$$

Where, $\mathbb{1}\{N \geq i\}$ is equal to 1 when $N \geq i$ and 0 when $N < i$.

Distribution of Load Between Units: Let P_B denote the probability to relay the packet to the next unit, if the queue is full. Consequently, $(1-P_B)$ represents the probability that an arrived packet is accepted. λ_U denotes the effective data rate of the system which is the number of packets that are actually served by the unit. We introduce also the offered load $\rho = \lambda\mu$ defined by Little's law [4] and similarly the effective carried load denoted by ρ_U. λ_U and ρ_U are given respectively by $\lambda(1-P_B)$ and $\rho(1-P_B)$ Hence, the probability P_B can be written as $P_B = \dfrac{\rho - \rho_U}{\rho}$.

The *effective carried load* ρ_U is also defined as the probability that the unit is busy at an arbitrary time in a long period of time P [7]. ρ_U can be written:

$$\rho_U = \lim_{P\to\infty} \frac{\sum \text{service periods in P}}{\sum \text{vacation periods in P} + \sum \text{service periods in P}} = \frac{P(E_s)\mu}{P(E_v)\bar{v} + P(E_s)\mu}, \tag{5}$$

where $\bar{v}$ is the mean vacation time, E_s is the event of being in the end of service and E_v is the event of being in the end of vacation which is expressed by:

$$P(E_v) = P(E_v^v) + P(E_v^s) = P(E_v^v) + P(E_v|E_s)P(E_s) = P(E_v^v) + \frac{1}{A}P(E_s),$$

where E_v^v is the event to pass from a vacation to another one and E_v^s is the event to pass from service to vacation. Since we know that $P(E_v) + P(E_s) = 1$, we can deduce $P(E_v)$ and ρ_U:

$$P(E_v) = \frac{P(E_v^v) + 1/A}{1/A + 1}, \rho_U = \frac{(1 - P(E_v^v))\mu}{(P(E_v^v) + 1/A)\bar{v} + (1 - P(E_v^v))\mu}. \tag{6}$$

To evaluate probability of passing from vacation to another $P(E_v^v)$, we will use the imbedded Markov Chain approach [7]. This approach is widely recognized as a powerful tool for the study of queues. It helps to predict the state of the unit queue at a random time t and to define the performance of the system (waiting time in the queue). The key idea of Markov Chain approach is to choose random points and calculate the probabilities of being in active or sleeping periods.

Markov points are chosen randomly from time instants when a vacation time ends or a service time ends. We assume here that the recovery period is small which is not enough to receive packets. We will consider the following probabilities:

- q_k: the probability of k jobs waiting when the vacation period ends ($k = 0, 1, \cdots, K$); Note that q_0 is equal to $P(E_v^v)$.
- π_k: the probability of k jobs waiting when the service period ends ($k = 0, 1, \cdots, K$-1); Note that after a service time, the queue size cannot be K because at least one packet was treated.
- f_j: the probability of receiving exactly j arrivals during a vacation period ($j = 0, 1, \cdots, \infty$).
- a_j: the probability of receiving exactly j arrivals during the service period ($j = 0, 1, \cdots, \infty$).

Since arrivals are assumed to form a Poisson process, we have:

$$f_j = \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t} v(t)dt, \quad a_j = \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t} s(t)dt, \tag{7}$$

where $v(t)$ is the probability density function of the vacation periods with a mean $\bar{v}$ and $s(t)$ is the probability density function of the service periods with a mean $\frac{1}{\mu}$. After an imbedded point, the system state can be defined as follows:

$$q_k = q_0 f_k + (\sum_{j=0}^{k} f_{k-j})P(E_v). \qquad k = 0, 1, \ldots, (K-1). \tag{8}$$

$$q_K = q_0 \sum_{k=K}^{\infty} f_k + (\sum_{k=K}^{\infty} \sum_{j=0}^{k} \pi_j f_{k-j}) P(E_v). \qquad k = K. \tag{9}$$

$$\pi_0 = q_1 a_0 + (1 - P(E_v))(p_0 a_0 + p_0 a_0 + p_0 a_1 + p_1 a_0). \tag{10}$$

$$\pi_k = \sum_{j=1}^{k+1} q_j a_{k-j+1} + (\sum_{j=0}^{k+1} \pi_j a_{k-j+1})(1 - P(E_v)). \qquad k = 1, \ldots, (K-2). \tag{11}$$

$$\pi_{K-1} = \sum_{j=1}^{k} q_j + (\sum_{j=0}^{k-1} \pi_j \sum_{k=K}^{\infty} a_{k-j+1})(1 - P(E_v)). \qquad k = K-1. \tag{12}$$

Since $\sum_{k=0}^{K} q_k$ and $\sum_{k=0}^{K-1} \pi_k$ are complementary, we have also $\sum_{k=0}^{K} q_k + \sum_{k=0}^{K-1} \pi_k = 1$. This system is solved using CVX toolbox of Matlab to compute all probabilities. To validate our theoretical derivations, we compared them to the simulated arrival process and Sleep/Active mechanism with the same parameters. Figures 3(a), (b) and (c) show that the theoretical derivations of q_0, q_k, π_k and $P(E_v)$ are very close to the simulation.

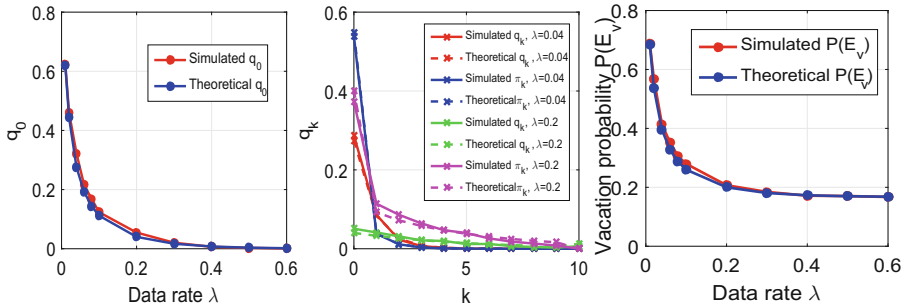

Fig. 3. Comparison between simulated and theoretical probabilities.

Until now, we analyzed the system $M/G/1/K$ for a single unit queue with a Poisson arrival process. However, a network machine has multiple processing units. Therefore, we need to generalize our study to an $M/G/n/K$ system, where n is the number of units per network device. As shown in Fig. 4, at a random time t, the unit can process only one packet and its queue has $(K-1)$ waiting positions, where the jobs can wait if they find the unit busy on arrival. This queue can have K waiting positions if the unit is on vacation. Packets arriving when the system is full are not allowed to enter the queue and will be relayed to the next unit.

Only a fraction $(1 - P_B)$ of the arrivals actually enters the queue of the first unit. The *effective arrival rate* of packets waiting in the queue is only $\lambda_U =$

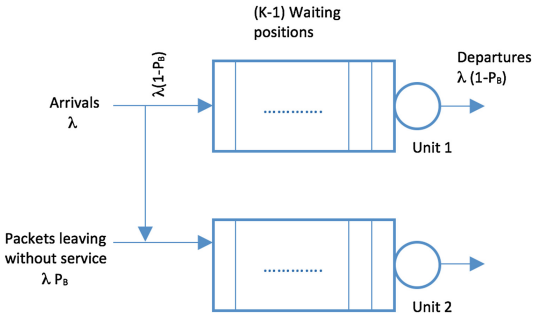

Fig. 4. Distribution of effective data rate between units.

$\lambda(1-P_B)$. Each unit experiences the rejection mechanism and relays the data to the next unit. Therefore, the traffic load is distributed among units with different rates. In this work, we assume that the blocked load leaving the first unit and entering the next unit follows a Poisson process with a parameter λP_{B_1} Following this assumption, we can write: $\lambda_{U_1} = (1 - P_{B_1})$ and $\lambda_{U_i} = \lambda \prod_{j=1}^{i-1} P_{B_j}(1 - P_{B_i})$, where λ_{U_i} is the *effective rate* of the i^{th} unit; $(i = 2 \ldots n)$.

Expectation of Energy Gain: To calculate the energy gain, we will compare the proposed system described in Fig. 5(a) to the always-on device presented in Fig. 5(b) (where conventional network devices are used and vacation queuing model is not applied).

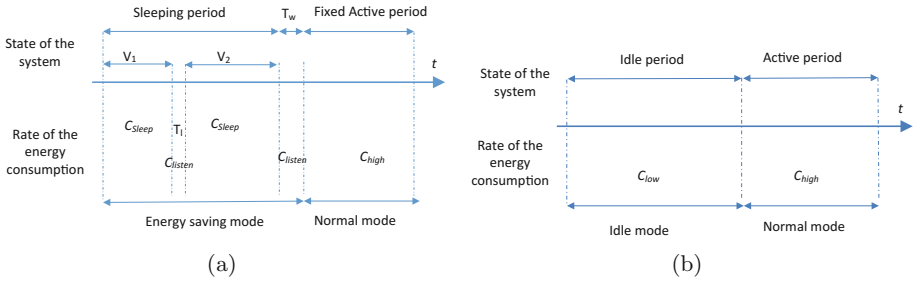

Fig. 5. Comparison between power-aware system and always-on system.

Since a processing unit can be in different states (i.e. *sleeping*, *active*, *listening* and *recovery*), we can distinguish between the following three possible energy levels which are from the highest to the lowest: C_{high} consumed during the process of packets (active period A); C_{listen} experienced when checking the state of the queue (listening period T_l) or in the recovery period T_w; and C_{sleep} consumed when the unit is inactive (sleeping period $E[S]$).

During sleeping period, we observe that there are $E[N]$ listening periods and one recovery period T_w. The energy consumption of the whole system per time unit can be defined as follows:

$$E_{Power-aware} = \frac{A \quad C_{high}}{(E[S] + T_w + A + T_l E[N])} + \frac{(T_l E[N] + T_w) C_{listen}}{(E[S] + T_w + A + T_l E[N])} + \frac{E[S] C_{sleep}}{(E[S] + T_w + A + T_l E[N])}. \quad (13)$$

If the power saving mechanism is not active (always-on system), the power consumption when there is a load is equal to C_{high} and it is equal to C_{low} in idle periods. So, its energy consumption can be calculated as follows:

$$E_{Always-on} = \rho C_{high} + (1 - \rho) C_{low}.$$

The economy of energy when using the power-aware mechanism comparing to the always-on mechanism is equal to $(E_{Always-on} - E_{Power-aware})$. Thus, we can define the relative energy gain as:

$$EG = \frac{E_{Always-on} - E_{Power-aware}}{E_{Always-on}}. \quad (14)$$

The mean energy gain per port of a network device is the average of energy gains of its units.

4 System Evaluation

Based on the metrics estimated in the Sect. 3.3 and by simulating the arriving process and the proposed queuing model on Matlab, we can calculate the efficiency and the performance of our system. But, first, some configurations need to be defined. Let $v(t)$ and $s(t)$ (the probability density functions of the vacation periods and service periods respectively) have an exponential distribution. We fix, then, the packet service time $\mu = 1$Tu (Time unit), $T_w = T_l = 0.5$Tu and the maximum number of packets in the queue $K = 10$. The mean duration of each vacation time is equal to $\bar{v_i}$; hence its related Laplace transform can be written as indicated in (15). In our simulation, we consider that all vacation times are equal and their mean size is equal to $\bar{v}$. In this way, the expression of $E[N]$ and $E[S]$ can be simplified as follows:

$$\begin{cases} E[V_i] = \bar{v_i} & i = 1, 2, \ldots \\ L_{V_i}(\lambda) = \dfrac{1}{1 + \bar{v_i}\lambda} & i = 1, 2, \ldots \end{cases} \Rightarrow \begin{cases} E[N] = 1 + \frac{1}{\bar{v}\lambda} \\ E[S] = \bar{v} + \frac{1}{\lambda} \end{cases} \quad (15)$$

The values of the energy consumed in different states are: $C_{high} = 5000^3 \times 10^{-6}$ watts; $C_{low} = C_{high} \times 0.7$; $C_{listen} = C_{high} \times 0.3$; $C_{sleep} = C_{high} \times 0.1$.

As a first step, we compared our theoretical results and simulation results and we proved the integrity of our calculations as we can see in Fig. 6. These results show that the network packets arrival rate has a great impact on the

energy consumption. As shown in Figs. 6(a) and (b), when increasing the network load, the power consumption of one processing unit increases and the power conservation degrades for different sizes of active period A and vacation time $\bar{v}$. This means that our system accomplishes more power saving at the lowest network rate. In addition, in the higher rates, the energy consumed is quasi-constant and the energy gain is always important. Our important energy gain in high loads is achieved due to the fixed active period imposed to the processing units and the switching to sleeping state even if the queue is not empty. In this way, our power-aware system achieves more than 15% of energy gain. The energy gain is also dependent of two other parameters which are the duration of the service period and the vacation time. As we can see in Fig. 6(a), the size of A has an impact on the energy for different values of λ and a mean vacation time equal to 10 Tu. More precisely, EG decreases by the increase of A. The contribution of A is more visible in high loads because in low loads the arrivals occur seldomly. So, the number of vacation periods is big and the service period is always small comparing to it. However, the sleeping period in high loads, generally, consists only of one vacation period which explains the bigger impact of A for these high rates. Figure 6(b) presents the impact of $\bar{v}$ on the energy gain when A is equal to 10 Tu. We can see that the energy increases greatly and monotonically with the increase of the mean vacation especially in high loads. The great impact of the vacation in high loads is explained by the fact that the energy is always high in very low loads due to the large number of vacation times which is not the case of higher loads composed generally of one vacation and one small service ($A = 10$ Tu). So, by increasing $\bar{v}$, A becomes insignificant.

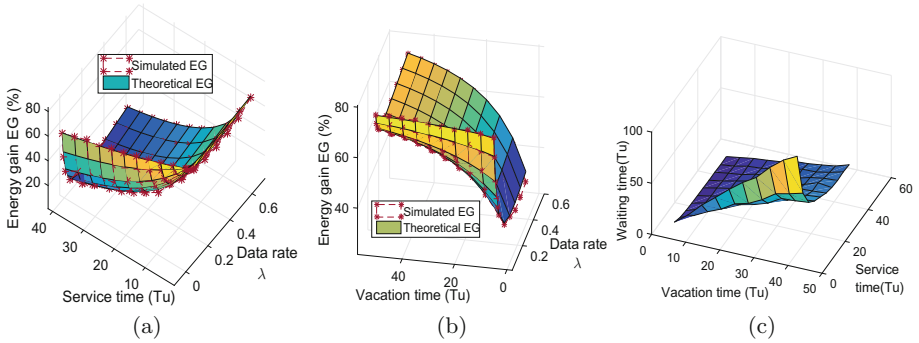

Fig. 6. Impact of the service and vacation on the energy gain and waiting time.

Therefore, to gain the maximum of energy, the active period A should be minimized and the mean vacation $\bar{v}$ should be maximized. However, the performance of the network should always be considered which is in our system the waiting time in the queue. Figure 6(c) presents the impact of the duration of vacation and service on the waiting time when $\lambda = 0.2$. We can see that the waiting time is smaller when the service is bigger and the vacation duration is

minimized. Since the maximization of the energy gain and the maintain of the system performance are in contrast, a trade off should be established. A waiting time threshold (WTT) should be chosen and the energy can be maximized while respecting the constraint. This constraint is chosen by the owner of the applications implemented in the data center as he knows the performance requirements of his services. Figure 7 presents the energy gain when fixing two WTT thresholds equal to 10 Tu and 30 Tu and choosing $(A, \bar{v})$ that maximize the energy. We can see that a bigger threshold contributes to gain more energy. Hence, the applications owners can sacrifice a little bit in terms of latency which is impacted by the waiting time in order to reduce the budget of the energy. Theoretical and simulated results are proved to be aligned.

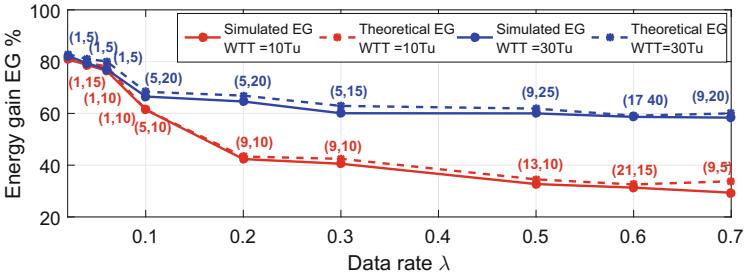

Fig. 7. Energy gain for different waiting time thresholds.

As explained in the previous section, after the re-architecturing of the network devices, the first processing units will accept higher load rates. However, the rest of the units will switch to sleep state or accept only low packets rates. Figure 8(a) shows the effective data rate processed by each unit in the new re-architectured device. In fact, the data rate λ is equal to the sum of all loads received by all the interfaces of the network device ($n = 6$). For example, when $\lambda = 1.2$, this means that each interface received a load equal to 0.2. As we can see, for lower loads, the received data is processed by only 2 or 3 units (see Fig. 8(b)). Hence, our approach proved its efficiency to maximize the number of sleeping units and save energy even in high loads, which is depicted in Fig. 8(c) without being dependent on the traffic matrix. The size of vacation and service must be well optimized while respecting the performance requirements of a data center to ensure better results and a good distribution of load between units (blocking probability depends on A and $\bar{v}$) which will be studied in future works in addition to the implementation of the solution in a data center architecture.

To conclude, comparing to the power aware algorithms described in the Sect. 2, the proposed power saving approach can be applied to any data center topology and it is not dependent on the architecture of the network since we only change in the network devices and not the interconnection. In addition, one of the biggest advantages of our re-architecturing and queuing approach

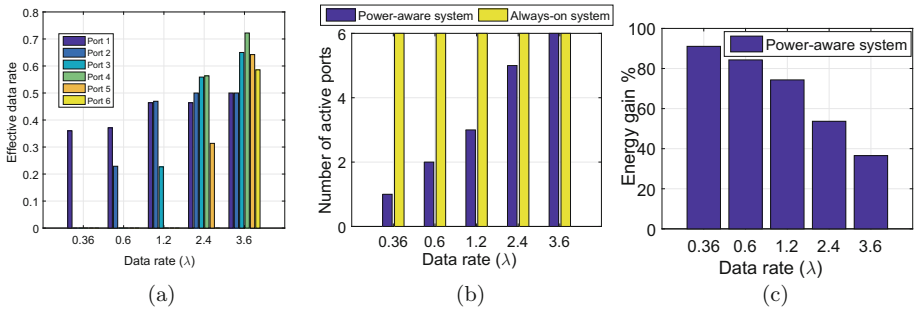

Fig. 8. Load distribution among different units.

lies in its negligible calculation complexity. Generally, the power aware routing algorithms have a bad exponential time complexity due to the huge searching space for communication flows routes. However, our approach is just relaying the incoming packets one by one to the available processing unit without any calculation complexity.

5 Conclusion

In this paper, we studied the idea of decoupling the network device interfaces from their processing units. In this way, the incoming loads from different interfaces can be handled by only few available units. The other ones will be switched into sleep state. To maximize the number of sleeping units and manage the distribution of incoming packets, a Sleep/Active algorithm is proposed. Then, an analysis following the M/G/1/K queuing model is conducted to estimate the energy gain and the sleeping periods. The simulation results aligned with the theoretical study proved the efficiency of the system.

Acknowledgments. This paper was made possible by NPRP grant 6-718-2-298 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Openflow (2011). http://archive.openflow.org/
2. Google green (2017). https://environment.google/
3. Abts, D., Marty, M.R., Wells, P.M., Klausler, P., Liu, H.: Energy proportional datacenter networks. In: Proceedings of the 37th Annual International Symposium on Computer Architecture, pp. 338–347. ACM (2010)
4. Adan, I., Resing, J.: Queueing Systems (2015)
5. Baccour, E., Foufou, S., Hamila, R., Tari, Z.: Achieving energy efficiency in data centers with a performance-guaranteed power aware routing. Comput. Commun. **109**, 131–145 (2017)

6. Baccour, E., Foufou, S., Hamila, R., Hamdi, M.: A survey of wireless data center network. In: Proceedings of CISS-15 International Conference on Information Sciences and Systems March 2015
7. Bose, S.K.: An Introduction to Queueing Systems. Heidelberg, Springer (2002)
8. Cheng, C., Li, J., Wang, Y.: An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing. Tsinghua Sci. Technol. **20**, 28–39 (2015)
9. Froom, R., Frahim, E.: Implementing Cisco IP Switched Networks (SWITCH) Foundation Learning Guide: (CCNP SWITCH 300–115). Pearson Education, Indianapolis (2015)
10. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. SIGCOMM Comput. Commun. **39**, 68–73 (2008)
11. Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., McKeown, N.: Elastictree: saving energy in data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, p. 17 (2010)
12. Schwartz, C., Pries, R., Tran-Gia, P.: A queuing analysis of an energy-saving mechanism in data centers. In: The International Conference on Information Network, pp. 70–75 (2012)
13. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software-defined networking, pp. 27–51 (2015)