


Article

Improving Energy Efficiency on SDN Control-Plane Using Multi-Core Controllers

Tadeu F. Oliveira ^{1,*}, Samuel Xavier-de-Souza ² and Luiz F. Silveira ²

¹ Federal Institute of Education, Science, and Technology of Rio Grande do Norte, Parnamirim, Rua Antonia de Lima Paiva, 155 Bairro Nova Esperanca, Parnamirim 59143-455, Brazil

² Computer Engineering and Automation Department, Federal University of Rio Grande do Norte, Natal 59078-970, Brazil; samuel@dca.ufrn.br (S.X.-d.-S.); lfelipe@dca.ufrn.br (L.F.S.)

* Correspondence: tadeu.ferreira@ifrn.edu.br

Abstract: Software-defined networks have become more common in data centers. The programmability of these networks is a great feature that allows innovation to be deployed fast, following the increasing number of new applications. This growth comes with a cost of more processing power and energy consumption. Many researchers have tackled this issue using existing routing techniques to dynamically adjust the network forwarding plane to save energy. On the control-plane, researchers have found algorithms for positioning the controller in a way to reduce the number of used links, thus reducing energy. These strategies reduce energy consumption at the expense of processing power of the controllers. This paper proposes a novel approach to energy efficiency focused on the network's control-plane, which is complementary to the many already existing data-plane solutions. It takes advantage of the parallel processing capabilities of modern off-the-shelf multicore processors to split the many tasks of the controller among the cores. By dividing the tasks among homogeneous cores, one can lower the frequency of operations, lowering the overall energy consumption while keeping the same quality of service level. We show that a multicore controller can use an off-the-shelf multicore processor to save energy while keeping the level of service. We performed experiments based on standard network measures, namely latency and throughput, and standard energy efficiency metrics for data centers such as the Communication Network Energy Efficiency (CNEE) metric. Higher energy efficiency is achieved by a parallel implementation of the controller and lowering each core's frequency of operation. In our experiments, we achieved a drop of 28% on processor energy use for a constant throughput scenario when comparing with the single-core approach.

Keywords: software-defined network; energy efficiency; control-plane; multicore; controller



Citation: Oliveira, T.F.; Xavier-de-Souza, S.; Silveira, L.F. Improving Energy Efficiency on SDN Control-Plane Using Multi-Core Controllers. *Energies* **2021**, *14*, 3161. <https://doi.org/10.3390/en14113161>

Academic Editor: Brian D. Fath

Received: 15 April 2021

Accepted: 12 May 2021

Published: 28 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Data center networks are increasing in size, data processing capacity and throughput to attend to the high demand of computing services required by: the growing demand on the cloud [1], smart cities, the advent of the Internet of Things (IoT), big data applications and deep neural networks among other technologies. The demand from DC for energy and its related environmental implications are increasing accordingly. A study by Van Hedeghem et al. [2] shows that the energy consumption of Information and Communication Technology (ICT) equipment was responsible for 4.6% of all energy consumption by 2012. In addition, the network interconnection equipment is accountable for 15% up to 50% [3] of the energy used in these facilities. In a recent review, Masanet et al. [4] showed that total energy consumption of data centers worldwide has increased from 95 TWh in 2010 to 130 TWh in 2018, and it is expected to reach 155 TWh by 2023.

The equipment present on a Data Center (DC) can be divided into three categories: infrastructure, servers and network. The energy-efficiency of a data center has been measured as the portion of overall energy consumption that is delivered to ICT equipment,

i.e., how much of the energy measured on the power source of a DC is used by the ICT equipment.

The most common metric for a DC energy's efficiency is the Power Usage Effectiveness (PUE), which is defined as the ratio between the facility's total power and the power consumed by the ICT equipment. The PUE is an important metric, but it lacks granularity on where the energy is consumed, observing the three aforementioned categories. Thus, Fiandrino et al. [5] proposed new metrics for each category, the Network Power Usage Effectiveness (NPUE):

$$NPUE = \frac{\text{Total ICT power}}{\text{Power consumed by network equipment}} \quad (1)$$

which is the equivalent to the portion of network equipment on the PUE, and the Communication Network Energy Efficiency (CNEE), which may be defined as "energy to deliver a single bit of information" [5]:

$$CNEE = \frac{\text{Power Consumed by Network Equipment}}{\text{Effective Network Throughput Capacity}} \quad (2)$$

These are the ones that relate to network equipment and energy efficiency.

1.1. SDN on Data Center Networks (DCN)

New ways to control and manage a DC network emerged to allow its programmability. The SDN is a well-accepted [6] approach to achieve better control of all the equipment on a modern network by separating control from data-planes, as shown in Figure 1. As shown in Figure 2, the control-plane is responsible for forwarding decisions using the southbound interface, as well as providing an API to the applications running on the network using the northbound interface. This way, a controller can change the network behavior (routing, queue policies, sleep states, etc.) according to applications running in the controller and out of the controller connected by the northbound interface. The separation also creates opportunities to understand better and improve each part of the network's energy efficiency.

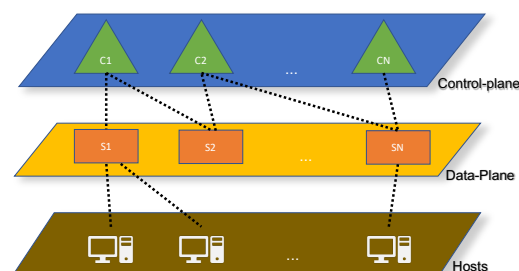


Figure 1. Software-defined network architecture.

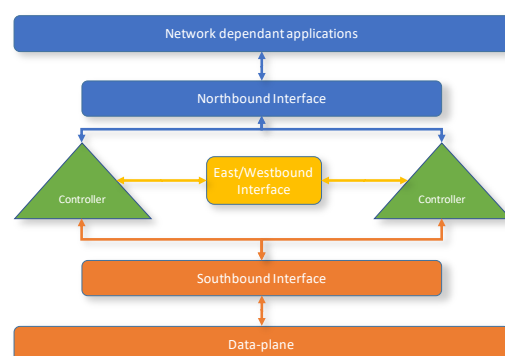


Figure 2. Software-defined network components roles.

The software-defined network paradigm brought programmability to networks, with the capacity to adapt to changes in traffic demand and to add new applications to the network. Many works used this new feature to improve energy-efficiency on data center networks. The current research mainly focuses on data plane efficiency, using novel routing algorithms or virtual machine migration to turn off ports or entire switches during low usage periods.

In a recent survey, Assefa and Özkasap [7] categorized energy efficiency on SDN into two groups: hardware and software based. The software-based solutions are divided into three main categories: traffic-aware, end system-aware and rule placement.

The traffic-aware solutions consider the global view of the network, enabled by the centralized controller. The controller gathers statistical information from the switches to decide to put in sleep mode or even turn off forwarding ports or entire devices.

The end system-aware solutions use the ability to change traffic sources and targets on a data center. In a data center, many end systems are virtual machines running on different servers. The server consolidation allows moving functions from many servers to one considering the load on those platforms. With virtual machines, it is also possible to move between physical devices to coalesce traffic, allowing to turn off physical end systems.

The last category on energy efficiency research falls on rule placement. SDN switches work by consulting a dynamic rule table to forward packets. This table is commonly stored on a Ternary Content-Addressable Memory (TCAM) that accounts for the most energy-consuming part of the switch exempting the link ports. To minimize the energy consumption of the TCAM, those works try to put as few as possible rules on each switch's TCAM; this is obtained by using wildcards rules [8]. These rules may force packets to take longer paths but allow switches to be turned off, trading packet delay for energy efficiency.

The second group classified by Assefa and Özkasap [7] comprises hardware-based solutions that consist of techniques to reduce the Ternary Content Addressable Memory (TCAM) use on forwarding switches. This is achieved by compressing data on the TCAM or changing the way a flow is represented on the table. Typically, an entry on an openflow switch table has five fields: source, destination, protocol, port and action. The techniques may compress the rules or the content of TCAM. The compression of rules works by detecting different rules with the same action with one bit of difference and merging them or using tags (sequence of bits that acts as a label, smaller than the actual five-field rule) that are added to the packet header and used to forward packets on switches. Other techniques uses dynamic programming to create an optimal decision diagram of paths on the network, and then creating the minimum set of rules to apply this diagram on switches discarding redundant rules. It is important to notice that all of those solutions imply in more processing load on the controller and a need to full knowledge of network topology.

Many works have used network programmability to enhance energy efficiency [9–14]. Most works use already known or new routing algorithms that optimize for energy-consumption by turning off links or entire switches [9,13–15]. Another common strategy is to distribute the load of switches among many controllers [10,12]. Ruiz-Rivera et al. [11] defined the best controller location to minimize energy consumption, while Son et al. [16] used the northbound interface to migrate applications and virtual machines to better meet Service Level Agreements (SLA) while minimizes link usage.

Despite the existence of many solutions for energy-efficiency on SDN, only a few of those consider the control-plane consumption. The justification for this is that, in the early days of SDN, it was common to have one controller tied to many switches so the data-plane energy consumption was much higher than the control-plane, thus deserving more attention; however, now, as many works have demonstrated [17–19], the recommendation is to have many controllers load-balancing the control of switches. Furthermore, many solutions add tasks to the controller without accounting for the impact on energy consumption of the now much more busy components.

Most of the research is focused on data-plane efficiency and neglects the energy cost of high demanding algorithms on the controller such as genetic algorithms [12] or

new routing algorithms [13]. In addition, many of the control-plane solutions consider a single controller [9,13,14] or for multi-controller all the controllers must have full and synchronized knowledge of the network topology.

Since the controller's role is crucial to the network operation, it must be protected by redundancy. The parallel implementation is capable of dealing with many instances of the controller running including a physically distributed manner.

An important innovation presented here is a deeper look at controllers' energy consumption beyond the traffic and network interfaces considering the processor power consumption. The novel parallel implementation allows leveraging the energy efficiency on current off-the-shelf processors already present on DCN.

1.2. Contributions

This paper investigates SDN controllers' energy consumption and how to improve their energy efficiency therefore improving CNEE metric (2) of any DC that employs SDN multicore controllers. It controls the number of available cores and the processor core frequency to distribute the load in a multicore environment. Many experiments were executed to demonstrate the approach's feasibility with different scenarios, reducing energy consumption while keeping latency and throughput. In addition, the results point to future works on load balancing capabilities of a distributed multicore controller scenario.

The main contributions of this paper are:

- The evaluation of processor power on controller energy consumption.
- A parallel implementation of the Ryu controller that improves multicore processors performance.
- A novel energy-reducing solution for SDN controllers not tied to any topology or knowledge of the current topology and that may be applied to any number of controllers in a hierarchical or centralized organization, with no special hardware.

Subsequent sections are organized as follows. Section 2 shows a brief overview of the literature on SDN energy efficiency. Section 3 describes some fundamental concepts and the approach proposed to save energy on the control plane. Section 4 presents the experiments executed to validate our solution. Finally, Section 5 provides a further discussion on the results and points to next steps on the energy efficiency for SDN control plane.

2. Related Work

The literature on energy efficiency for the network has many examples of routing algorithms applied to wireless software-defined networks and, more recently, to software-defined networks on DC, Cloud, 5G and Edge-computing.

About two decades ago, Heinzelman et al. [20] presented the algorithm called Low-Energy Adaptive Clustering Hierarchy (LEACH), one of the earliest works on energy-efficiency in networks to use routing. It proposes an algorithm to balance energy usage on wireless sensors networks alternating between clusters heads to avoid a single path to consume all energy resources distributing this load over the many nodes on many paths a packet can take to reach the base station. Today, the LEACH algorithm is still used as a benchmark by many works on energy-consumption and routing. Variations of this approach are used to balance energy consumption over the SDN, allowing the routing of packets to change dynamically using the processing capabilities of the controller.

The programmability of controllers was explored by Aujla et al. [21] to change the network routing after migrating workloads inside the DCN. They created a model to evaluate the best placement of tasks, moving Virtual Machines (VM) intra-DCN or to edge devices. The solution is divided in two steps: First, the model migrates the workloads of VMs evaluating their deadline and demand of computing resources. Second, the SDN controller defines the best path for energy-efficiency. The energy-aware flow-scheduling scheme implemented on the controller increases the energy consumption of the controller, but energy savings on the data-plane counterbalance this. This solution needs full knowledge of the network topology; to overcome this limitation, the authors used one main controller,

which has full knowledge of the network topology and creates many virtual controllers to do load balance among them.

The controller placement problem (CPP) was tackled by Hu et al. [22] from an energy-savings perspective. A model is defined, and the CPP is modeled as a Binary Integer Program (BIP) to minimize energy consumption. Due to the computational complexity of the BIP model, it cannot be applied to large networks. A heuristic genetic algorithm, called Improved Genetic Controller Placement Algorithm (IGCPA), is used to find a sub-optimal solution for large networks. The combined algorithms, namely BIP and IGCPA, use the GreCo [11] solution to address the allocation of switches to controllers. BIP and IGCPA are offline approaches since both algorithms need previous knowledge of the network controllers, nodes and applications to determine the best placement for energy efficiency.

A different approach by Son et al. [16] uses the ability to migrate virtual machines and virtual network functions to consolidate traffic while lowering SLA violations. It uses the online load to calculate the future need of server and network resources. The strategy is northbound-based and considers that the controller has access to the applications and evaluates its historical traffic patterns to overbook VM based on resource utilization. This solution adds complexity to the control-plane and the northbound applications running, but the paper does not account for this in energy measures.

The links usage is the metric used by Xu et al. [14], who developed a routing algorithm to use the maximum bandwidth of available links since switches' energy is not proportional to the traffic but is related to the number of active links. The solution tries to schedule the flows to keep below a maximum delivery time but use the minimum number of links.

On the dense 5G SDN, Fernández-Fernández et al. [15] defined network energy-optimal routing as an NP-Hard problem, meaning that computation time grows exponentially with network size. To address this issue, they proposed two heuristics: one offline and one online. The offline one, called SNetCa, models the controller switch connections as a graph and produces the minimum number of links to guarantee connection and some redundancy. The second heuristic, online, called DESRA, defines the flow rules to add to the switches when a new flow arrives this is done selecting the path with the minimum number of links thus consuming the minimum energy.

For wireless sensor SDN in which the energy is often a scarce resource, since most of this equipment is battery powered, Usmanyounus et al. [13] proposed an algorithm to route packets on a software-defined wireless sensor network. The objective is to increase the network lifetime and improve the average packet delay. A novel routing protocol called Energy-aware Software-Defined Network (EASDN) is used, and the controller must know all the network topology, also for a single controller scenario.

A more comprehensive approach to energy consumption on SDN, which accounts also for the control-plane consumption, was developed by Priyadarsini et al. [10] using a load balancing between many controllers to lower the demand on links by coalescing traffic turning off switches and controllers reducing energy on all planes. The algorithm is executed before the network is operational; it selects the most energy-efficient among various known routing algorithms and picks the most energy-efficient for that particular topology.

One of the most recent works on SDN energy-efficiency was by Assefa and Ozkasap [9], who presented a new metric for energy efficiency on SDN concerning the link utilization on the data-plane. Using this metric, the authors proposed Maximizing Ratio for Energy Saving in SDN (MaxRES DN) an algorithm to minimize the number of hops, thus turning off links reducing energy. The algorithm is executed by the single controller, which needs full knowledge of the network topology.

Table 1 summarizes some of the most recent and recognized works on energy efficiency for software-defined networks. The column Multi-Controller indicates if the approach applies to networks with more than one controller; Controller's Energy points if the work considered the impact of controller energy on overall energy consumption; Fault-Tolerant registers if the approach considers some redundancy on control-plane; and Partial

Topology indicates if the solution works with controllers that have partial knowledge of the network topology.

Table 1. SDN energy-efficiency related works.

Description	Multi-Controller	Controller's Energy	Fault-Tolerant	Partial Topology	Strategy
[20]	n	y	y	y	routing
[21]	y	y	y	n	load balancing
[22]	y	y	y	n	controller-placement
[16]	y	n	y	n	application-aware
[14]	n	n	n	n	routing
[15]	y	n	y	n	routing
[13]	n	n	n	n	routing
[10]	y	y	y	n	load balancing
[9]	n	n	n	n	routing
Our proposal	y	y	y	y	parallel-processing

As presented in Table 1, the main characteristics of our approach improve control-plane energy efficiency and can be associated with current state-of-the-art and commercial data-plane solutions, improving SDN overall energy efficiency. We use an innovative strategy based on parallel processing to achieve better energy efficiency in the SDN's control plane with no conflict with common data-plane strategies.

Using a parallel implementation, we can employ more processor cores with lower frequencies to any number of controllers on the network, filling the gap on multi-controllers scenarios. This approach also allows for easy fault-tolerance since the controllers do not need to have full knowledge (knowing a partial topology) of the network to benefit from the trade-off between operation frequency and performance.

3. Materials and Methods

Our proposed solution aims to reduce energy consumption by splitting controller's load among many cores on a processor, while lowering the frequency of operation, compared to the single-core high-frequency scenario. In this section, we discuss a few fundamental key concepts for our proposal and define the new method for energy efficiency. First, a model for energy consumption on SDN is presented showing the impact of processing packets on the controller. Then, the rationale behind multicore processors on energy savings is discussed. We also describe the hardware and software used to real-time control and measure: frequency, availability and energy consumption of processor's cores.

3.1. Energy Consumption on SDN

Energy consumption on SDN can be divided into three parts: the controllers, the switches and the links on a network. The work of Priyadarsini et al. [10] shows how each part interacts and models this consumption based on packet flows on the network. The comprehensive network energy model:

$$E = \sum_{i=1}^n (e_{1i} * E_{swi} + e_{2i} * E_{ci} + e_{3i} * L) + E_{link} \quad (3)$$

presented by the authors comprises the controllers E_{ci} , switches E_{swi} and links E_{link} energy consumption, along with the queued packets caused by latency L . The e_1, e_2, e_3 constants are derived from simulation, while the total network E consumption is defined as the summation of all n elements (switches, controllers and latency queues) on the network.

Although the switch and link take the biggest part of the energy consumption of a SDN network, controller consumption can also be minimized to contribute to the aggregated energy efficiency of the network. It is important to notice that those improvements on

controllers energy efficiency can be associated with other techniques for switch and link level energy consumption.

In this paper, we focus on controller consumption, which is defined as a function of packets processed, as described by:

$$E_c = \beta * E_c^A + (1 - \beta) * E_c^N \quad (4)$$

where E_c is total controller energy consumption and β is 1 when the controller is awake and 0 when it is in sleep mode. Thus, $E_c = E_c^A$ when the controller is awake and $E_c = E_c^N$ when the controller is in sleep mode, with

$$E_c^A = E_{SA} + \sum_{i=1}^n W_i * CPUUtil_i + f(T) * E_s \quad (5)$$

and

$$E_c^N = E_{SA} + \sum_{i=1}^n W_i * CPUUtil_i + E_{beacon} \quad (6)$$

where E_{SA} is the conversion energy from sleep to awake state, n is the number of packets treated by the controller during a time interval, W_i is the weight parameter for the various types of packets and their need for processing power is represented by $CPUUtil_i$, $f(T)$ is a function related to packet arrival during simulation time and E_s is the sensing energy necessary to treat one packet arriving. The expected value of this function $E(f(T))$ is, assuming an ergodic system, equal to the average packets per time interval of the simulation, which can be estimated by network observation, E_{beacon} is the energy used to send a beacon packet necessary to show the controller is still available even in sleep mode.

3.2. Energy Consumption on the Control Plane

As SDN's main feature, the separation of control from forwarding packets brought the ability to increase the applications that a network can run, but this increases the energy consumption on control-plane and the complexity to measure the network's overall consumption.

Most energy-related studies on networks focus on measuring link usage, which on SDN accounts only for the data-plane. As data centers networks grow in size, complexity and redundancy, the control structure, the SDN controllers, now also must be taken into account for energy consumption measuring.

3.3. Impact of Controller Energy Consumption on a Data Center Network

A data center that uses the SDN paradigm to structure its core network will have many controllers. This is necessary for redundancy and load balancing reasons. With the SDN paradigm, the separation between network and servers is blurry since the SDN controllers may be implemented on virtual machines running on Servers. These servers account for most energy consumption on a data center, as reported by Shehabi et al. [23]. The amount of energy consumed by SDN controllers poses severe limitations on previous works that centered energy measures on data-plane only. This paper focuses on the control-plane to add this variable to the energy-aware solutions.

3.4. Multi-Core and Single-Core

SDN controllers are composed of software and a hardware platform. Most of the controllers on the market are developed using multipurpose languages and are available for many operating systems. Similar to any application, they may take advantage of the multiprocessing capacity of those platforms. Since most operating systems are developed for different tasks, it is possible to optimize the hardware/software to a specific job such as the one executed by the SDN controller.

s Amdahl's law [24] divides problem workloads into two fractions: the part that is infinitely parallelizable P and a fraction $S = 1 - P$ that is strictly serial, thus the performance R obtained by adding new parallel capabilities is limited by the fraction S as defined by

$$R = R_S * \frac{S + P}{S + \frac{P}{M}} \quad (7)$$

where R_S is the performance of a computer that can only execute the problem in a totally sequential manner, M is the number of parallel units of execution (cores or processors).

Amdahl's law assumes a fixed problem size which for SDN controllers is not the case since the load a controller receives is related to network load that changes along the time. For this type of problems, where the size of parallel fraction grows as more resources are available, Gustafson's law may be applied [25]. It states that larger problems would benefit from more processors if the parallel fraction of those problems also grow accordingly, leading to a more optimistic speedup on parallel systems; this has been proven true by the emergency and prevalence of multicore processors on the market.

On multicore hardware, it is possible to take advantage of parallelism to decrease processing time, i.e., deliver faster results when compared to single-core processors. However, it is also possible to reduce energy consumption by reducing frequency of operation of each core such that, when compared to a single core (with higher frequency), the results are delivered in the same time [26]. For the SDN controller, this is equivalent to comparing a faster (higher frequency of operation) single-core controller and a slower (lower frequency of operation) multicore controller; both could handle packets keeping latency and throughput nearly the same, but saving energy on lowering frequencies. This solution to improve energy efficiency is especially useful on SDN controllers because the deadlines for delivering results (packet handling or throughput) do not need to be as fast as possible but may lie within a range of pre-defined values, commonly described on a Service Level Agreement (SLA).

3.5. Comparing Multi-Core and Single-Core Energy Consumption

The work by Barros et al. [26] shows that it is more energy-efficient, for applications with a measurable speedup, to have a multicore low-frequency processor than to have a single-core high-frequency processor keeping the deliver time constant. This assumption can be verified for SDN controllers as this application is very I/O bound but can also leverage the parallel capacity of modern multicore processors.

3.6. Proposed Method—Parallel SDN Controller for a Multicore Platform

An SDN controller is a very complex piece of software used as a platform to develop new applications and to provide APIs for the northbound software. There is an interface for the southbound, usually Openflow [27], and an interface for the northbound, usually some REST API. This dual function (southbound and northbound) by itself is a solid indication to leverage the parallel nature of multicore architectures. The task of reading and processing as many packets as possible can, in turn, be divided into sub-tasks that may run in parallel. One strategy to implement this separation of tasks is the dispatcher/worker, as shown in Figure 3.

In this paper, we use the Ryu [28] controller's default Openflow 1.3 simple-switch implementation with some improvements to have parallel tasks divided by many parallel processes that may run on different cores. The improved version diagram is shown in Figure 4 and its general inner workings described in Algorithm 1.

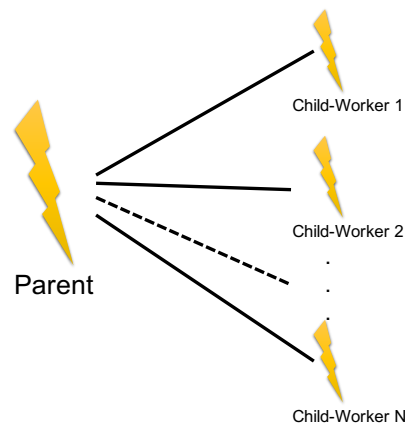


Figure 3. Multicore tasks implementation on Ryu.

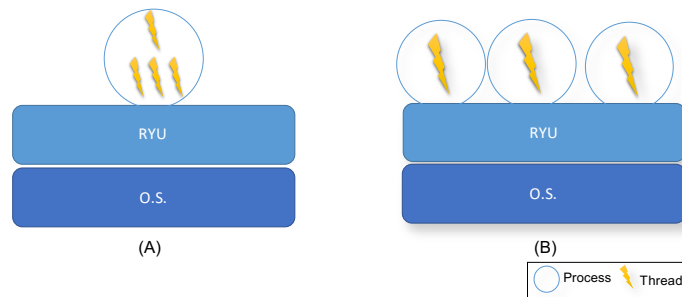


Figure 4. Ryu Simple switch implementation (A); and our multicore Ryu Implementation (B).

Algorithm 1: Simple multicore switch

```

createParentProcess;
i ← 0;
while i < CoreCount do
    createChildProcess(i);
    i++;
end
while hasPacketOnQueue do
    findIdleChildProcess;
    copyPacketToChildProcess;
    waitNextPacket;
end

```

As exposed, the SDN controller should be implemented in a way that takes advantage of the multicore nature of current processors. The implementation presented in Algorithm 1 has this property, as can be seen in Figure 5.

In Figure 5, we see that as the number of cores grows, the CPU utilization is divided between the available cores. CPU utilization on the graph is shown as reported by the Linux kernel in the `/proc/stat` file. In our implementation, as the number of available cores increases, the task is better divided by every core leading to a decrease in CPU utilization by core; in addition, this behavior is the same, independent of CPU operation frequency.

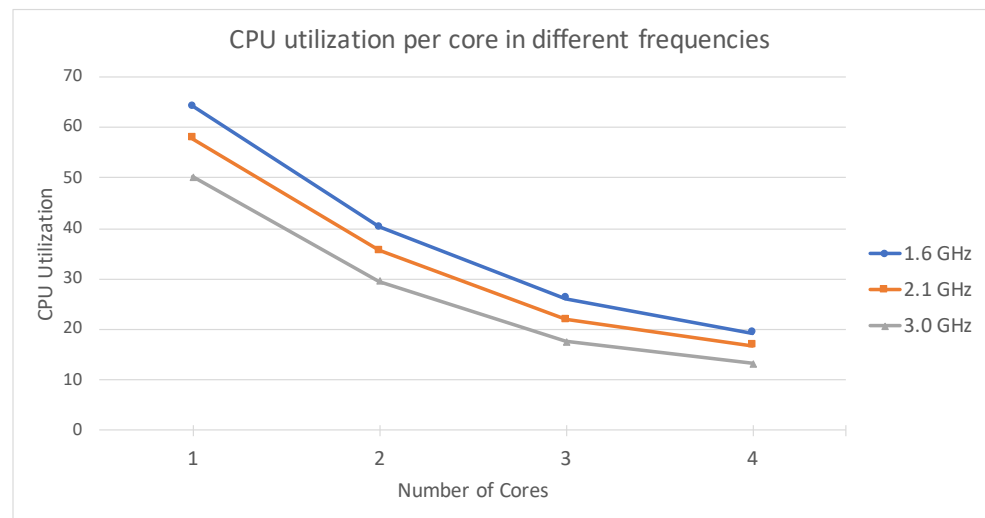


Figure 5. Average utilization per core on different frequencies for a high demand task on a SDN controller.

3.7. Working Frequency versus Parallelism

SDN controllers are responsible not only for the packet forwarding decisions but also for tasks related to security, accountancy and administration. The very nature of the multiple functions a controller has to do in a real-world scenario implies the need and leverage of parallel multicore processors. It is better to deliver these tasks as fast as possible, but this may imply higher energy use. The network has its bandwidth limits, and there is a maximum frequency when a faster processor will be idle waiting for the network limitation on delivering new packets, thus wasting energy.

In an optimized solution, the processor should adapt itself to the need of the network, lowering its energy consumption while keeping with the expected parameters of the network, namely throughput, latency, packet drop, jitter, etc. To demonstrate the impact of the processor frequency on the throughput of an SDN controller, an experiment was conducted: the same load was applied to a single-core controller with changing operation frequency from 1.6 to 3.0 GHz.

Figure 6 clearly shows that processor frequency and throughput have an almost linear relation for a single-core controller when the controller needs to process many packets per second. This conclusion that higher processor frequency leads to higher energy consumption, as stated by Barros et al. [26], leads to an analysis into the right balance between throughput and energy consumption to fulfill a well-defined level of service for an SDN.

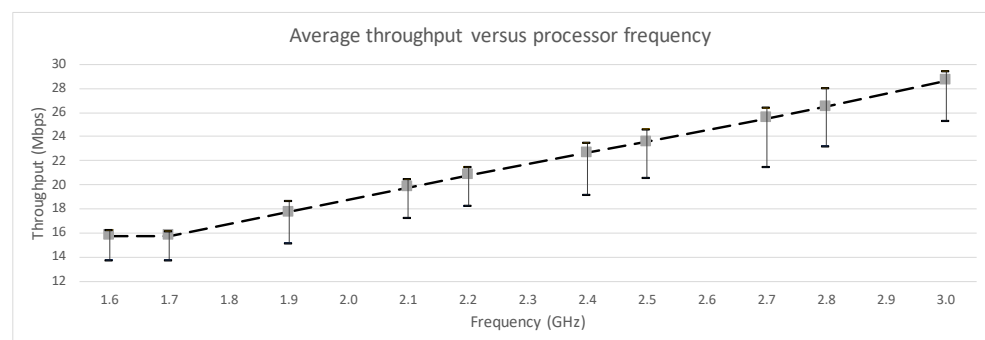


Figure 6. Average, min and max throughput on a single-core controller for various frequencies.

3.8. Capping Processor Frequency

On a Linux system, the processor frequency can be set using utilities such as the `cpupower` [29], changing every core's current frequency on a CPU package. To set the

frequency is necessary to choose the CPU frequency governor. The Linux kernel supports four governors; each governor implements a strategy to use the CPU while balancing energy efficiency and performance [30]. The four governors are: userspace, performance, schedutil and ondemand.

The performance governor causes the processor frequency to be set to the highest available value for the processor, independent of any external factor. It is a policy that compromises energy efficiency over a guaranteed maximum number of instructions over time.

The schedutil governor uses information from the Kernel scheduler to measure the current load of the CPUs and set the frequency accordingly. This governor is aware of the load, increasing the frequency of high load systems and decreasing it in a more idle system. It also raises the frequency of cores for processes blocked for too much time waiting on I/O.

The ondemand governor uses data gathered from its own context to estimate the current CPU load and set the frequency. The worker is invoked frequently (by default every 10 ms) to compute the time the CPU was idle since the last execution. This metric is used to estimate CPU load and derives the right CPU frequency within min and max frequency limits.

The userspace governor allows a root user to set each processor's core operation frequency to a static value; we used this governor to statically set the frequency of every core on all of our experiments described in Sections 4.1 and 4.2.

To further improve energy consumption reduction, the idle CPU cores are turned offline by issuing a kernel syscall. This is possible using the Linux CPU Hotplug [31], allowing a user to turn on and off processor cores as the demand grows. The control can be achieved by the filesystem tree mounted under `/sys/devices/system/cpu`.

All the measurements are also done from the `/sys` filesystem under the folder `/sys/class/powercap/intel-rapl`; these files are an interface to Intel's Running Average Power Limit (RAPL) [32] and Intel's Model-Specific Registers (MSR). The file `energy_uj` reports the energy in μ Joules.

4. Simulations and Results

In this section, a set of experiments are presented to show the reduced energy use on multicore SDN controllers compared to their single-core counterparts. To validate our proposed method, many experiments were conducted. The main idea was to measure the controller's processor energy consumption during heavy load.

All experiments were conducted using two physical hosts linked by an Ethernet 1000baseT-FD cat5e cable. The server is a quad-core Dell PowerEdge R230 with an Intel Xeon E31220 processor, while the client is a Dell Inspiron notebook with an Intel T6670 Core2 Duo processor, both running Debian OS 9.0. Figure 7 describes the hardware installation diagram.

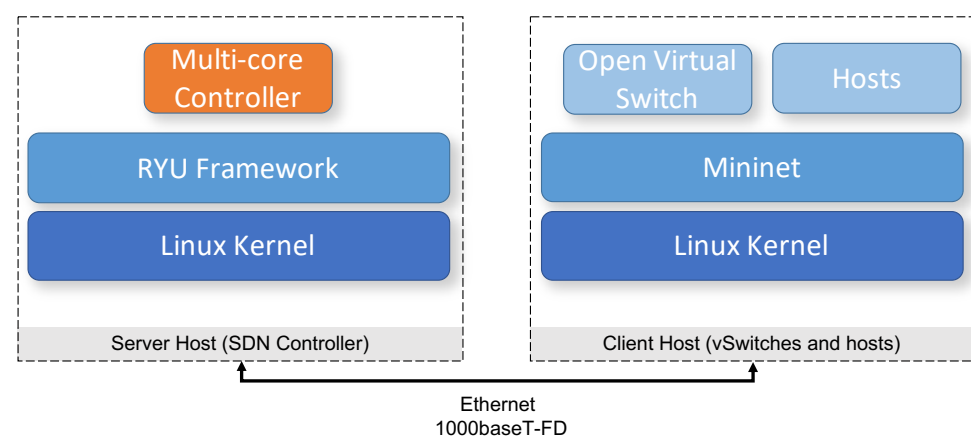


Figure 7. Block diagram of experiment.

To validate our proposal, we evaluated two scenarios: (1) a scenario with a constant throughput; and (2) a latency-aware scenario. The first scenario described in Section 4.1 uses one vswitch, two hosts and the Iperf [33] utility to generate constant traffic between the hosts passing through the controller. The second scenario described in Section 4.2 uses thirty hosts attached to two vswitches and ICMP-Echo requests to test latency keeping an average of 5 and 6 ms latency. Those scenarios were designed to put the controller under controlled load, testing for the two most critical metrics for a network: throughput and latency.

Network real traffic patterns change during a day and over the week, mostly related to user applications demands. Part of the traffic on a DCN flows through intra-data-center connections. As stated by Popoola and Pranggono [34], there is an explosion of intra-data-center data traffic in web communication and a drift toward software-defined data center. Thus, the real traffic patterns can be: (i) of many machines on the same network with request-response, such as the HTTP protocol, which we simulate using the latency-aware scenario described in Section 4.2; or (ii) big files transfers to or from storages and databases that we simulate using the throughput-aware scenario described in Section 4.1.

4.1. Evaluation for Constant Throughput (Iperf)

The Iperf utility was used to create a constant throughput flow from one host to the other passing through the controller. This scenario puts the controller under a steady and heavy but controlled load, a common pattern for applications such as video streaming. The Iperf tool was chosen as the traffic generator because it is a well known and widely used tool to generate and measure network throughput. Many researchers [35–37] consider it a good model for evaluating performance of various network elements. The use cases range from energy-efficiency tests in SDN [35] to statistical analysis of SDN management [36] including security strategies [37]. The Iperf tool is used to validate experiments and creates test traffic that correlates closely to real patterns.

The controller used for the experiment was a Ryu [28] based multicore controller running on a quad-core server (Dell PowerEdge R230) with an Intel Xeon E31220 processor. The vswitches and hosts were simulated using Mininet [38] utility on separate hardware (Dell Inspiron); the energy consumption measures were done on the server running the controller.

The first measurement in Figure 8 was done keeping a constant throughput of 30 Mbps over a 600 s period. The same experiment was done for 1–4 cores. The single-core instance keeps an almost constant energy consumption near 10.2 J, while the multicore versions run an average as low as 7.1 J. The few dips shown on the graphs represent moments where the processor was idle, waiting for I/O to happen.

The other two experiments were done increasing the load on the controller: 40 Mbps depicted in Figure 9 and 50 Mbps in Figure 10. In these two experiments, only 2–4 cores instances were used since the single-core instance could not deliver the throughput expected even on the highest available frequency (3 GHz). The energy consumption could still be optimized from the two-core instance to the three- and four-core ones, from 9.6 J to an 8.0 J on average. Another important finding is that the dips in the graphs for higher throughput are fewer on the 40 Mbps than the 30 Mbps one and even fewer on the 50 Mbps one, which means that the processor has much less idle time, pushing the energy consumption higher.

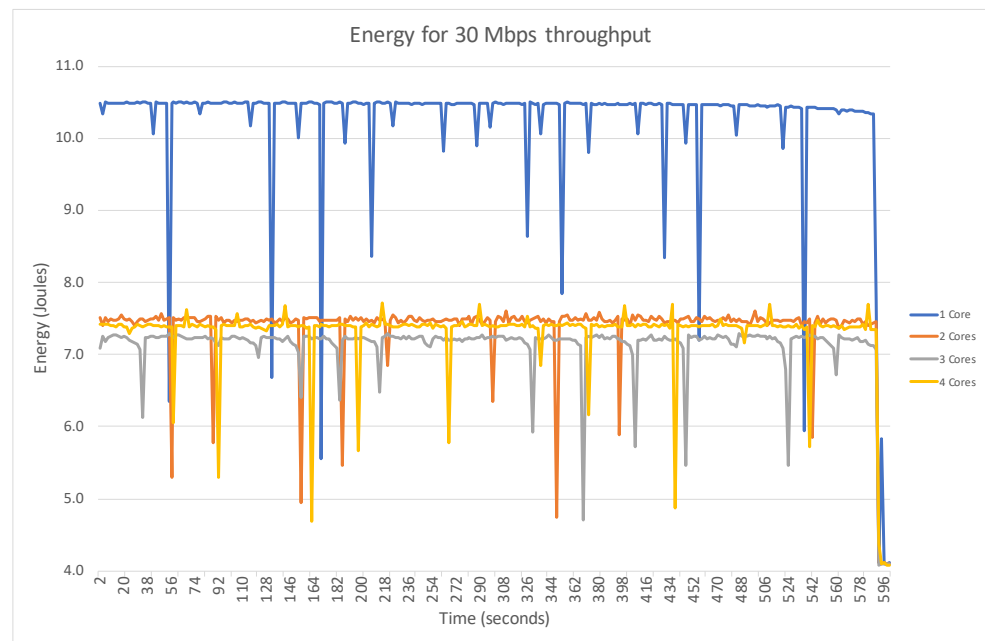


Figure 8. Energy consumed during the last second that precedes the measurement for different number of cores during a 30 Mbps traffic.

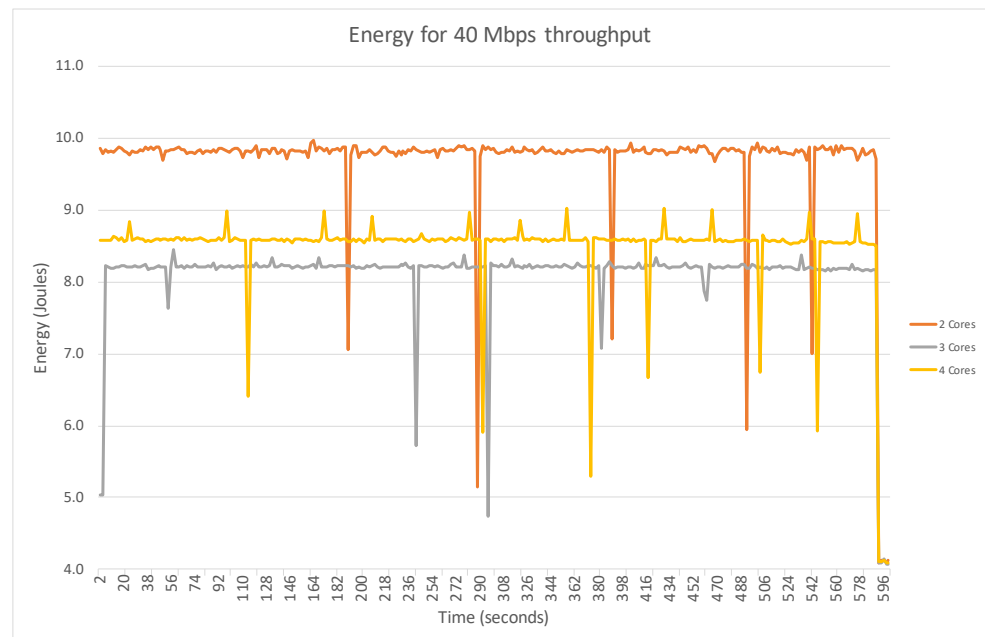


Figure 9. Energy consumed during the last second that precedes the measurement for different number of cores during a 40 Mbps traffic.

Figure 11 shows a comparison of the CNEE metric, defined in (2); even though a single core may deliver the agreed throughput, more cores with lower frequencies bring more efficiency. On the other hand, as shown by Barros et al. [26], there are limitations on efficiency gain related to the application's characteristics, namely the serial and parallel fractions, as discussed in Section 3.4. The gain is limited by the serial fraction of the problem, by the Amdahl's law.

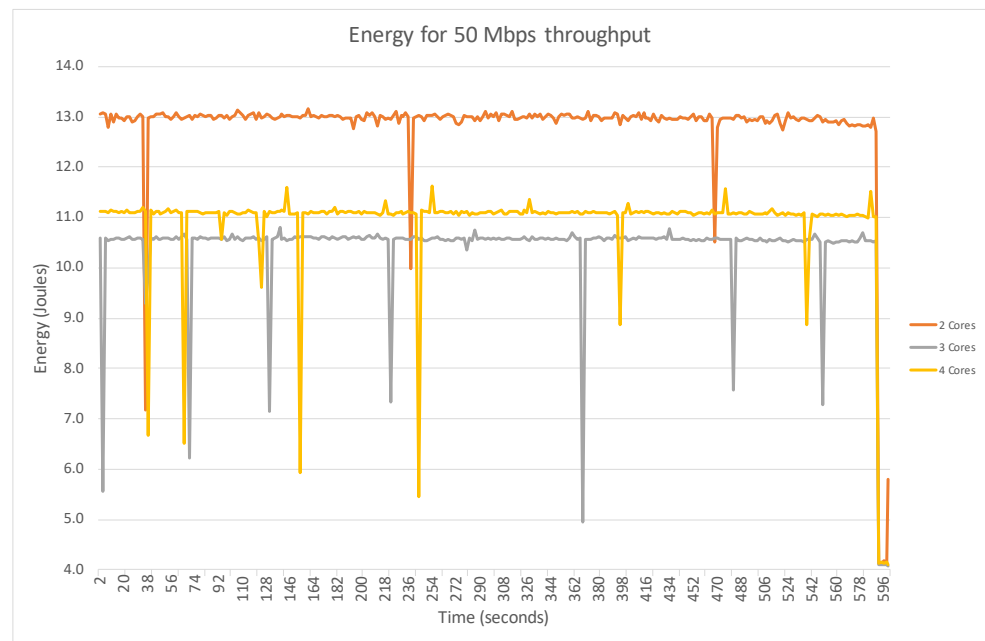


Figure 10. Energy consumed during the last second that precedes the measurement for different number of cores during a 50 Mbps traffic.

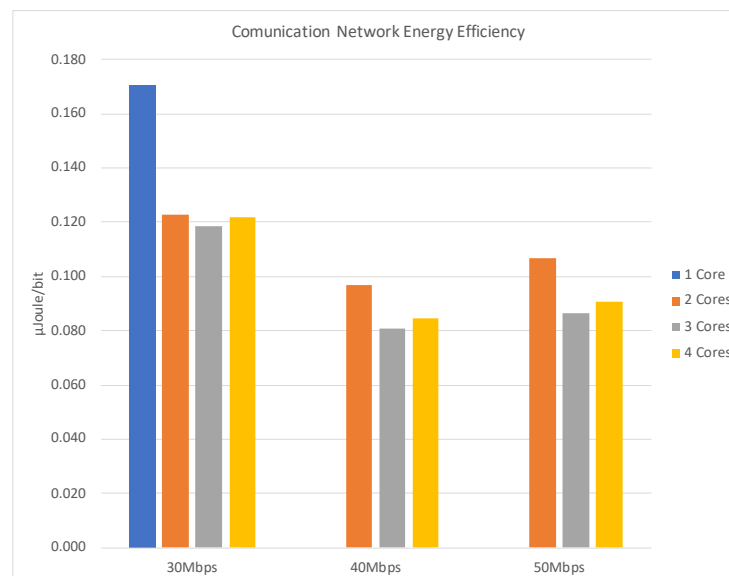


Figure 11. Communication network energy efficiency for different multicore scenarios.

As can be seen, the 40 and 50 Mbps scenarios could not be delivered by a single-core controller even on the highest available frequency (3 GHz). In our scenarios, the three-core configuration shows better efficiency. This is not a general rule as the best configuration will depend on the controller's implementation and the network's demand from the southbound and northbound interfaces. As stated by Gustafson [25] and discussed in Section 3.4, larger problems would benefit from more processors if the problem's parallel fraction also grows accordingly. Here, the behavior shown in Figures 5 and 6 supports the assumption that, with higher throughput, the parallel fraction will be distributed evenly by the cores, hence leading to better energy efficiency on the four-plus-core instances; not only the throughput but also as the number of users of the northbound controller interface grows.

4.2. Evaluation for Maximum Latency Threshold (ICMP)

The second scenario used the same hardware but with latency-aware traffic to simulate a full network with randomly small packets flowing. Each of the 30 hosts sends many (100) ICMP ping requests to every other host on the network, using the pingFull function from Mininet. The simulated hosts were attached to two vswitches with 15 hosts on each switch. The controller was kept the same software and hardware configuration as the first scenario.

This scenario aims to reproduce the load pattern created by request–response network applications such as HTTP or database access. Each host sends a request and waits for a response of every other host on the network. The packets are received and response is sent in an arbitrary time, which simulates requests from applications services on the network.

Internally, the Mininet software uses the native ping utility. Mininet pingFull function was edited to set the number of ICMP packets to send and allow ping flood mode, sending as many packets per second as possible. Our experiments were repeated 100 times for every combination of pair of hosts and controller configuration (frequency, number of cores), increasing the confidence of the results presented here.

Figure 12 shows the energy (Joule) consumed during the latency test for 30 hosts and 2 vswitches with a max average latency of 6 ms. In addition, the total energy used during the test is shown in Figure 13. Figures 14 and 15 show the same tests in a more stringent SLA where max average latency is 5 ms.

It is important to notice the different nature of the traffic between the latency and throughput tests. While Figures 8–10 show an almost constant energy consumption, with few dips caused by the I/O delays, Figures 12 and 14 show random intervals in which ICMP packets arrive. These two scenarios cover common stress situations on the network, namely the constant high throughput and the random time arrival of many packets. The real-world traffic will be located between the two presented scenarios: sometimes as bursts of traffic and sometimes as various smalls packets.

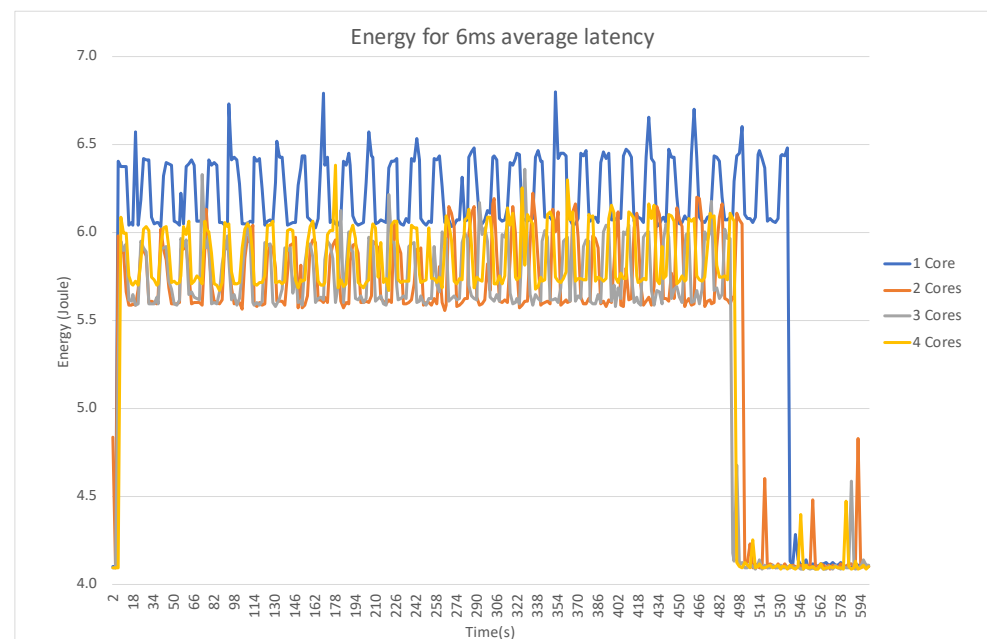


Figure 12. Energy consumed during the last second that proceeds the measurement for different number of cores during network average latency test with 6 ms limit.

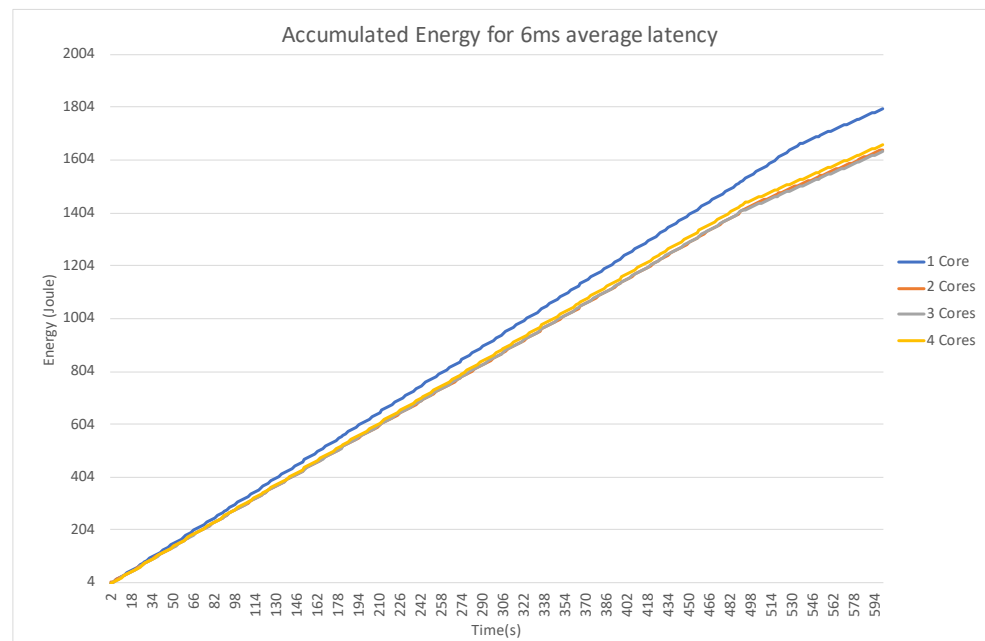


Figure 13. Accumulated energy consumption for different number of cores during network average latency test with 6 ms limit.

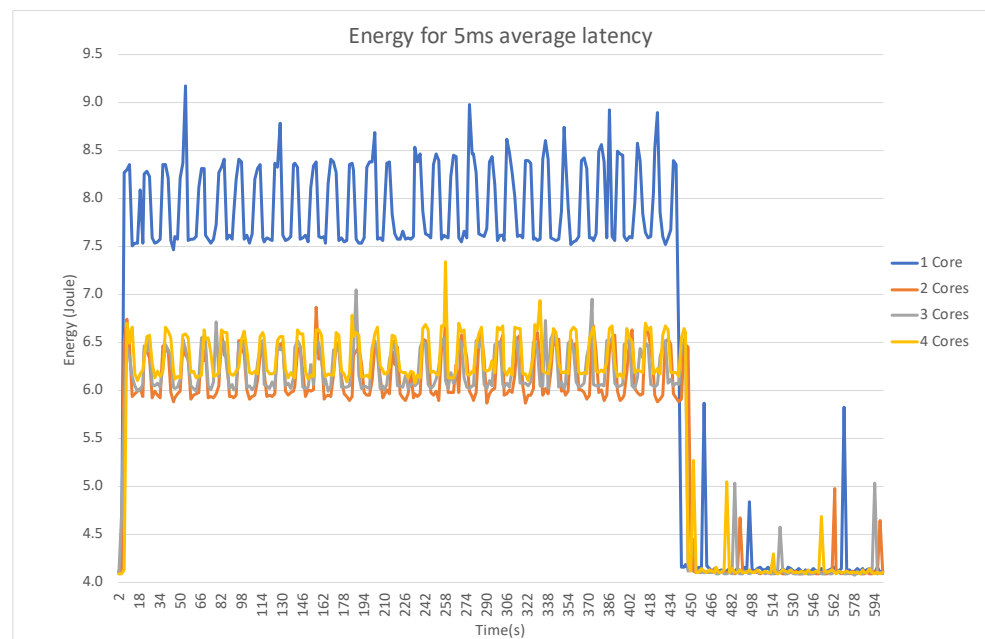


Figure 14. Energy consumed during the last second that proceeds the measurement for different number of cores during network average latency test with 5 ms limit.

Google has recently published an overview of its SDN control-plane called Orion [39], deployed globally in Google's data center called Jupiter. Each Orion instance runs on dedicated regular server-class platforms, where each controller uses up to 23 cores and 24 GB of memory.

CISCO's SDN solution, as described in [40], uses the Application Policy Infrastructure Controller (APIC) as the controller. This appliance is a centralized cluster with a minimum of three controllers. Although the APIC is an appliance and may also be virtualized on a CISCO-provided cloud or other cloud providers, such as Amazon and Azure, the hardware used is just the same as Google's, a regular server-class platform. The APIC appliance

has two configurations L3 and M3, which use two Xeon processors of 1.7 and 2.1 GHz, respectively.

These commercial products use the same platforms as used in our experiment but on a much larger global scale, which indicates that our finds may also apply to those larger network instances.

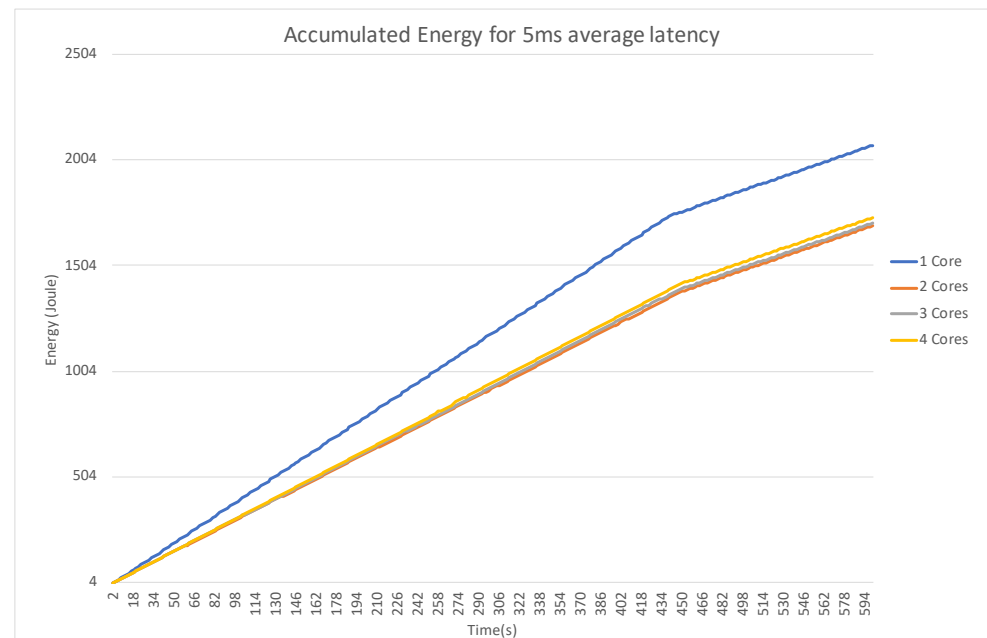


Figure 15. Accumulated energy consumption for different number of cores during network average latency test with 5 ms limit.

5. Conclusions and Future Work

This study shows that an SDN controller can have an improved energy efficiency by lowering its operating frequency using the parallel nature and availability of current multicore processors. The results obtained for the Communication Network Energy Efficiency (CNEE) (see Figure 11) show that the single-core or the dual-core scenarios use more energy to deliver the same number of bits. This improves the DC overall energy efficiency.

The reduction of energy consumption, as shown in Figures 8–10, is due to the lower operating frequency and parallel implementation. This indicates that a focus on leveraging parallelism on network applications, especially SDN controller, can improve not only performance but also energy-efficiency.

It is shown that a well-implemented parallel controller can leverage its efficiency by distributing tasks, keeping a guaranteed level of service, such as the industry-standard service level agreement.

The reduction of energy consumption on the control plane is a novel strategy for energy-aware SDN, where most researchers have focused on the data plane. As the number of applications using network programmability start to increase, the energy demand on controllers will follow. Exploiting the parallel nature of the controller is key to keep this component's energy consumption in balance.

Essentially, this solution can be applied to any SDN controller with an off-the-shelf multicore processor with no conflict with others solutions focused on data-plane, improving the overall efficiency of the network.

Future Works

To extend the findings in this paper, we shall evaluate how replacing the constant CPU_{util} in (5) by a function of the number of cores and their frequency, i.e., $f_{cpu}(ActiveCores, Frequency)$, can further improve energy efficiency. This may help to decide in which traffic

load scenarios it is useful to have more cores with a reduced frequencies or fewer cores with higher frequencies.

An energy-aware controller may change its processor frequency to go up during the network's bursts or go down on low demand automatically. This can be measured as a function of a predefined network parameter such as minimum throughput, packets per second or any other metric on a user's service level agreement.

Openflow protocol defines ways for a switch to deliver accountancy data to its controller. This information can fine-tune the processor frequency on the controller as the demand on the data-plane changes.

Author Contributions: All authors contributed equally to this work. Conceptualization, T.F.O., S.X.-d.-S. and L.F.S.; Formal analysis, T.F.O., S.X.-d.-S. and L.F.S.; Investigation, T.F.O., S.X.-d.-S. and L.F.S.; Methodology, T.F.O., S.X.-d.-S. and L.F.S.; Software, T.F.O., S.X.-d.-S. and L.F.S.; Writing—original draft, T.F.O., S.X.-d.-S. and L.F.S.; and Writing—review and editing, T.F.O., S.X.-d.-S. and L.F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES)-Finance Code 001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The software used on experiments in this study are openly available in Zenodo at doi:10.5281/zenodo.4653312 [41].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SDN	Software Defined Network
DC	Data Center
ICT	Information and Communications Technology
PUE	Power Usage Effectiveness
NPUE	Network Power Usage Effectiveness
CNEE	Communication Network Energy Efficiency
TCAM	Ternary Content-Addressable Memory
SLA	Service Level Agreement
API	Application Programming Interface
REST	Representational State Transfer
CPU	Central Processing Unit
RAPL	Running Average Power Limit
MSR	Model-Specific Register
OS	Operating System
ICMP	Internet Control Message Protocol

References

1. Lis, A.; Sudolska, A.; Pietryka, I.; Kozakiewicz, A. Cloud Computing and Energy Efficiency: Mapping the Thematic Structure of Research. *Energies* **2020**, *13*, 4117. [\[CrossRef\]](#)
2. Van Heddeghem, W.; Lambert, S.; Lannoo, B.; Colle, D.; Pickavet, M.; Demeester, P. Trends in worldwide ICT electricity consumption from 2007 to 2012. *Comput. Commun.* **2014**, *50*, 64–76. [\[CrossRef\]](#)
3. Abts, D.; Marty, M.R.; Wells, P.M.; Klausler, P.; Liu, H. Energy proportional datacenter networks. *ACM SIGARCH Comput. Archit. News* **2010**, *38*, 338–347. [\[CrossRef\]](#)
4. Masanet, E.; Shehabi, A.; Lei, N.; Smith, S.; Koomey, J. Recalibrating global data center energy-use estimates. *Science* **2020**, *367*, 984–986. [\[CrossRef\]](#)
5. Fiandrino, C.; Kliazovich, D.; Bouvry, P.; Zomaya, A.Y. Performance and Energy Efficiency Metrics for Communication Systems of Cloud Computing Data Centers. *IEEE Trans. Cloud Comput.* **2017**, *5*, 738–750. [\[CrossRef\]](#)
6. Kreutz, D.; Ramos, F.M.V.; Esteves Verissimo, P.; Esteve Rothenberg, C.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [\[CrossRef\]](#)

7. Assefa, B.G.; Özkasap, Ö. A survey of energy efficiency in SDN: Software-based methods and optimization models. *J. Netw. Comput. Appl.* **2019**, *137*, 127–143. [CrossRef]
8. Rodrigues, B.B.; Riekstin, A.C.; Janeiro, G.C.; Nascimento, V.T.; Carvalho, T.C.M.B.; Meirosu, C. GreenSDN: Bringing energy efficiency to an SDN emulation environment. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 11–15 May 2015; pp. 948–953. [CrossRef]
9. Assefa, B.G.; Ozkasap, O. RESDN: A Novel Metric and Method for Energy Efficient Routing in Software Defined Networks. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 736–749. [CrossRef]
10. Priyadarsini, M.; Kumar, S.; Bera, P.; Rahman, M.A. An energy-efficient load distribution framework for SDN controllers. *Computing* **2020**, *102*, 2073–2098. [CrossRef]
11. Ruiz-Rivera, A.; Chin, K.W.; Soh, S. GreCo: An Energy Aware Controller Association Algorithm for Software Defined Networks. *IEEE Commun. Lett.* **2015**, *19*, 541–544. [CrossRef]
12. Yonghong, F.; Jun, B.; Jianping, W.; Ze, C.; Ke, W.; Min, L. A Dormant Multi-Controller Model for Software Defined Networking. *China Commun.* **2014**, *11*, 45–55. [CrossRef]
13. Usmanyounus, M.; Islam, S.; Won Kim, S. Proposition and real-time implementation of an energy-aware routing protocol for a software defined wireless sensor network. *Sensors* **2019**, *19*, 2739. [CrossRef]
14. Xu, G.; Dai, B.; Huang, B.; Yang, J.; Wen, S. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. *Future Gener. Comput. Syst.* **2017**, *68*, 163–174. [CrossRef]
15. Fernández-Fernández, A.; Cervelló-Pastor, C.; Ochoa-Aday, L. Energy Efficiency and Network Performance: A Reality Check in SDN-Based 5G Systems. *Energies* **2017**, *10*, 2132. [CrossRef]
16. Son, J.; Dastjerdi, A.V.; Calheiros, R.N.; Buyya, R. SLA-Aware and Energy-Efficient Dynamic Overbooking in SDN-Based Cloud Data Centers. *IEEE Trans. Sustain. Comput.* **2017**, *2*, 76–89. [CrossRef]
17. Zhao, Y.; Iannone, L.; Riguiedel, M. On the performance of SDN controllers: A reality check. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 79–85. [CrossRef]
18. Yao, G.; Bi, J.; Li, Y.; Guo, L. On the Capacitated Controller Placement Problem in Software Defined Networks. *IEEE Commun. Lett.* **2014**, *18*, 1339–1342. [CrossRef]
19. Bannour, F.; Souihi, S.; Mellouk, A. Adaptive distributed SDN controllers: Application to Content-Centric Delivery Networks. *Future Gener. Comput. Syst.* **2020**, *113*, 78–93. [CrossRef]
20. Heinzelman, W.; Chandrakasan, A.; Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2000; Volume 1, p. 10. [CrossRef]
21. Aujla, G.S.; Kumar, N.; Zomaya, A.Y.; Ranjan, R. Optimal Decision Making for Big Data Processing at Edge-Cloud Environment: An SDN Perspective. *IEEE Trans. Ind. Inform.* **2018**, *14*, 778–789. [CrossRef]
22. Hu, Y.; Luo, T.; Beaulieu, N.C.; Deng, C. The Energy-Aware Controller Placement Problem in Software Defined Networks. *IEEE Commun. Lett.* **2017**, *21*, 741–744. [CrossRef]
23. Shehabi, A.; Smith, S.J.; Sartor, D.A.; Brown, R.E.; Herrlin, M.; Koomey, J.G.; Masanet, E.R.; Horner, N.; Azevedo, I.L.; Lintner, W. United States Data Center Energy Usage Report. *Berkeley Lab.* **2016**, *1*, 65.
24. Amdahl, G.M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 April 1967; Association for Computing Machinery: New York, NY, USA, 1967; pp. 483–485. [CrossRef]
25. Gustafson, J.L. Reevaluating Amdahl's Law. *Commun. ACM* **1988**, *31*, 532–533. [CrossRef]
26. Barros, C.; Silveira, L.; Valderrama, C.; Xavier-de Souza, S. Optimal processor dynamic-energy reduction for parallel workloads on heterogeneous multi-core architectures. *Microprocess. Microsyst.* **2015**, *39*, 418–425. [CrossRef]
27. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69. [CrossRef]
28. Ryu. Component-Based Software Defined Networking Framework. Available online: <https://ryu-sdn.org/> (accessed on 10 January 2021).
29. CPUPower. The CpuPower Software Package. Available online: <https://www.kernel.org/doc/readme/tools-power-cpupower-README> (accessed on 10 January 2021).
30. Linux. CPU Performance Scaling. Available online: <https://www.kernel.org/doc/html/v4.14/admin-guide/pm/cpufreq.html> (accessed on 10 January 2021).
31. Linux. CPU Hotplug in the Kernel. Available online: https://www.kernel.org/doc/html/latest/core-api/cpu_hotplug.html (accessed on 10 January 2021).
32. Intel. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide*, 1st ed.; Intel Corporation: Santa Clara, CA, USA, 2019; Volume 3, pp. 471–517. Available online: <https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-3a-3b-3c-and-3d-system-programming-guide.html> (accessed on 5 January 2021).
33. IPerf. iPerf—The Ultimate Speed Test Tool for TCP, UDP and SCTP. Available online: <https://iperf.fr/> (accessed on 10 January 2021).

34. Popoola, O.; Pranggono, B. On energy consumption of switch-centric data center networks. *J. Supercomput.* **2018**, *74*, 334–369. [[CrossRef](#)]
35. Li, D.; Shang, Y.; Chen, C. Software defined green data center network with exclusive routing. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; pp. 1743–1751. [[CrossRef](#)]
36. Najm, M.; Salman, A.; Kamal, A. Network Resource Management Optimization for SDN based on Statistical Approach. *Int. J. Comput. Appl.* **2017**, *177*, 5–13. [[CrossRef](#)]
37. Sahay, R.; Blanc, G.; Zhang, Z.; Debar, H. ArOMA: An SDN based autonomic DDoS mitigation framework. *Comput. Secur.* **2017**, *70*, 482–499. [[CrossRef](#)]
38. Lantz, B.; Heller, B.; McKeown, N. A network in a laptop: Rapid prototyping for software-defined networks. In Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, Monterey, CA, USA, 20–21 October 2010; ACM Press: New York, NY, USA, 2010; pp. 1–6. [[CrossRef](#)]
39. Ferguson, A.D.; Gribble, S.; Hong, C.Y.; Killian, C.; Mohsin, W.; Muehe, H.; Ong, J.; Poutievski, L.; Singh, A.; Vicisano, L.; et al. Orion: Google’s Software-Defined Networking Control Plane. In Proceedings of NSDI 2021: 18th USENIX Symposium on Networked Systems Design and Implementation, Boston, MA, USA, 12–14 April 2021.
40. Cisco. Cisco Application Policy Infrastructure Controller Data Sheet. Available online: <https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/application-policy-infrastructure-controller-apic/datasheet-c78-739715.html> (accessed on 25 April 2021).
41. Oliveira, T. Set of Scripts to Evaluate Ryu Multi-Core Energy Consumption. Available online: <https://doi.org/10.5281/zenodo.4653312> (accessed on 31 March 2021).