DRA: Duplication Resolver Algorithm for Power Conservation Utilizing Software Defined Network (SDN)

Mohammad Masoud[1], Yousef Jaradat[1], Ismael Jannoud[1] and Hong Huang[2]
[1] Computer and Communication Engineering Department
Al-Zaytoonah University of Jordan Amman 11733, Jordan
[e-mail: {m.zakaria, y.jaradat, ismael.jannoud}@zuj.edu.jo]
[2] Klipsch School of Electrical and Computer Engineering
New Mexico State University, Las*Cruces, NM, USA*
[e-mail: hhuang@nmsu.edu]
*Corresponding author: Mohammad Masoud

Received November 21, 2016; revised March 11, 2017; accepted April 17, 2017; published July 31, 2017

Abstract

In recent years, datacenters, network devices and computers have proliferated. The power consumed by information and communication technology (ICT) devices has inflated in an extraordinary manner. Green communication has emerged as a new approach to reduce and optimize power consumption in ICT sector. Many methods and protocols have been proposed and implemented to achieve green communication. Nevertheless, the increase of power consumption remains a problem.

In this work, we attempt to reduce and optimize power consumption of network devices in datacenters environment utilizing software defined network (SDN) paradigm. To gain more insight of the power consumption requirements of network switches, a power measurement system is constructed to measure power consumption levels of network devices. Subsequently, we propose a duplication resolver algorithm (DRA) to power off/on switches reactively. DRA algorithm reduces the required time by switches to construct their flow tables after rebooting. To this end, DRA-based external circuit has been constructed utilizing Ethernet module and an Arduino kit to control power supplies of network devices. To facilitate our work, a testbed has been constructed utilizing Ryu SDN controller, HP2920-24G switches and Arduino kits. Our results show that DRA algorithm can reduce both the power usage and start-up time delay of network switches after **failures.**

Keywords: Software Defined Network (SDN), Duplication Resolver Algorithm (DRA), Ryu Controller, Power Consumption

1. Introduction

Power consumption in information and communication technology (ICT) sector is rapidly growing. It was claimed that 1.8% of the total power consumption worldwide, which is estimated to be 350 billion Kwh, is used in ICT sector [1]. From 2008 to 2012, this number increased 10% annually. It was also reported that network switches consume the highest power amount among all network devices in datacenters and offices' networks [2]. In 2012, network switches in offices' networks consumed approximately 28.2 Twh compared to 18.1 Twh in 2008 in America. Moreover, in 2008, datacenters consumed 54GW worldwide [3]. Typical power consumption in datacenters depends on its scale. Small size datacenter (100-500 servers) consumes about 50kW [4]. Large size datacenter farm (5000+ servers) consumes more than 50MW [5]. These numbers are predicted to increase more in the future for two reasons. First, the number of Internet users is growing 20% annually. Second, the number of datacenters around the world is going to reach 8.6 billion in 2017 according to the International Data Corporation (IDC) report [6]. These facts require countermeasures to reduce and to optimize power consumption levels in ICT field.

Green communication [7] has been emerged to optimize power usage in all communication aspects; datacenters, wireless and mobile communication. Many researches were conducted in this area to reduce power usage in network protocols [8]. Nevertheless, power usage in ICT sector is still growing. The question is how to reduce power consumption in existing datacenters with minimal cost?

Software defined networking (SDN) has emerged as a new network paradigm to tackle network management and configurations issues. In this paradigm, network devices are controlled by a central unit called SDN controller. This controller communicates with network devices to configure and control them over Openflow protocol [9]. SDN is used to tackle many network challenges, such as security [10] and QoE [11]. In addition, SDN can be used to manage and optimize power usage in datacenters' networks [1].

In this work, SDN approach is used to reduce power consumption in datacenters. A duplication resolver algorithm (DRA) is proposed to power off/on switches in datacenters and to reduce their start-up delays after reboots. DRA algorithm shuts down duplicated ports and switches. To this end, a power measurement system has been constructed utilizing Arduino kit [12] and current sensor for measuring power consumed by network switches. In addition, Arduino has been used to build a DRA based testbed that can be controlled by SDN controller to power off/on switches. To facilitate our work, HP2920-24G Openflow enabled switches have been used. Ryu controller [13] is used as the SDN controller. The contributions of this paper are as follow:

- We propose DRA algorithm that selects which switch to turn off or to keep on in datacenters' network. Moreover, DRA algorithm reduces start-up time delay of the off switches by embedding their flows in a flow table in the SDN controller.
- We constructed two Arduino based testbeds: measure-circuit (M-C), and control-circuit (C-C). M-C circuit is used to measure power consumption of SDN

switches for optimization purposes. C-C circuit is utilized to control switches power. The SDN controller and DRA algorithm control C-C circuit.
- We constructed a power model based on the harvested data from M-C system.
- We constructed a testbed utilizing HP2920-24G SDN enabled switches to measure the performance and accuracy of DRA algorithm and compare it to the upper and lower theoretical power consumption bounds of a hierarchical network model

The rest of this paper is organized as following. Section 3 overviews the related works in this area. Section 4 demonstrates power measurement technique used. Section 5 overviews proposed system and demonstrates DRA algorithm. Section 6 demonstrates the conducted experiment and simulation. Finally, the paper concludes in section 7.

2. Related Work

ICT power reduction and optimization in datacenters and server farms have attracted researchers over years. Many statistical studies have been conducted to estimate power consumption of datacenters and to predict future usage [14, 15]. Moreover, Many methods and techniques have been proposed to reduce this usage, such as, optimizing cooling system [16], controlling temperature of datacenters to reduce power [17], storage power optimization [18], virtual machine migration, load balancing and network devices operation optimization [19, 20].

In this work, we focus on the underlying network devices' operation and their impact on power consumption. Many works have been proposed and conducted in this area. ElasticTree [21] is one of the popular power reduction algorithms based on powering off unused network devices. ElasticTree monitors network traffic in datacenters. Then, it decides which switch to shut down or to keep up based on some performance and fault tolerance measures. The authors utilized openflow switches and NOAX SDN controller to implement ElasticTree. However, the authors did not discuss the mechanism by which switches power on/off. Moreover, the authors assumed the existence of new special version of openflow protocol that supports power information 'power of openflow' which is be hard to implement.

Correlation-aware power optimization [22] is similar to ElasticTree. It dynamically finds a set of network devices that can be turned off without affecting the performance. It does this by consolidating traffic flows into a subset of network devices based on whether or not flows are correlated. However, this method requires a complex computation to correlate traffic flows, which will increase the overload on network controller and increases the delay. Honeyguide [23] is similar to [22] in it procedure. However, it utilizes virtual machines interfaces and migration processes. It does not power switches on/off. In [24], the authors proposed four algorithms for energy optimization based on dynamic traffic. The proposed algorithms attempted to switch on the minimum number of network elements to support traffic. The proposed algorithms are general that can be used in scenarios other than datacenters settings. Their results showed that during low-use times, 45% of energy consumption can be reduced. The proposed algorithms are complex since they require modifications in dataflow from source to destination. Moreover, authors did not show the implementation mechanism to power on/off network devices.

In [25], REsPoNse algorithm was proposed. This algorithm pre-computes energy-critical paths in a topology by analyzing its historical traffic. Subsequently, parts of the network enter a low-power mode. REsPoNse pre-computes three sets of paths; always-on, on-demand and fail-over paths. This algorithm is hard to implement. Moreover, it only works on predictable traffic with known historical statistics. In [26], authors proposed datacenter's power reduction scheme by load balancing traffic over virtual machines in the datacenter network. Authors demonstrated that constructing multi-virtual machines on the same servers could reduce power consumption more than constructing virtual machines on different servers. This study did not take into account the power usage of network devices, such as, network switches and routers.

Finally, in [1], the author has constructed a system to measure power consumption of different network devices utilizing SDN. The utilized system required a complex method to measure power consumptions of network devices. In this work, we have implemented a simpler circuit utilizing costless equipment. Subsequently, we used the harvested data to construct total power consumption scheme of a hierarchical network model.

Our work differs from these works in four main points. First, we attempt to construct a central measurement system to measure power consumption of network devices and derive a total power consumption model of hierarchical network configuration. Second, our algorithm is easy to implement and attempt to reduce power usage by disabling network devices ports if shutting devices off are impossible. Third, an external circuit has been implemented to control powering off/on switches without modifying openflow protocol. Finally, SDN control in this work has three purposes. First it controllers SDN switches. Second, it is used to configure switches through telnet. Finally, it controls external power circuitry.

3. Power Measurement-Circuit (M-C) Design

To measure power consumption of a network switch, M-C circuit was constructed. The circuit consists of a precision miniature current transformer module (PMCT), a Bluetooth module HC-06 and an Arduino UNO kit. PMCT sensor measures AC current that passes through a cable by surrounding the cable from outside, see **Fig. 1**. In other words, PMCT sensor can be placed in any location without disconnecting any circuit components.

To validate the operation of M-C system, DT-2700 clamp meter was used. We have connected the switch to the power outlet and then turned it on with both M-C system and clamp meter attached to its main power wire. M-C system recorded values automatically. However, clamp meter readings were recorded manually. This made it hard to compare real-time values of M-C and clamp. To reduce the comparison error, the values harvested from M-C system were averaged every 10 seconds. Subsequently, a clamp meter reading was recorded every 10 seconds. We harvested power usage data for 30 minutes. **Fig. 2** shows the error differences between the harvested data from M-C system and the clamp meter. This figure utilizes eqn. (1) for error calculation. **Fig. 2** shows that the recorded data from M-C sytem and clamp meter are approximately identical. The maximum-recorded error value was less than 1mA.

$$Error\ (A) = Clamp\ recorded\ value - M\text{-}C\ harvested\ value \quad (1)$$

Fig. 1. M-C system

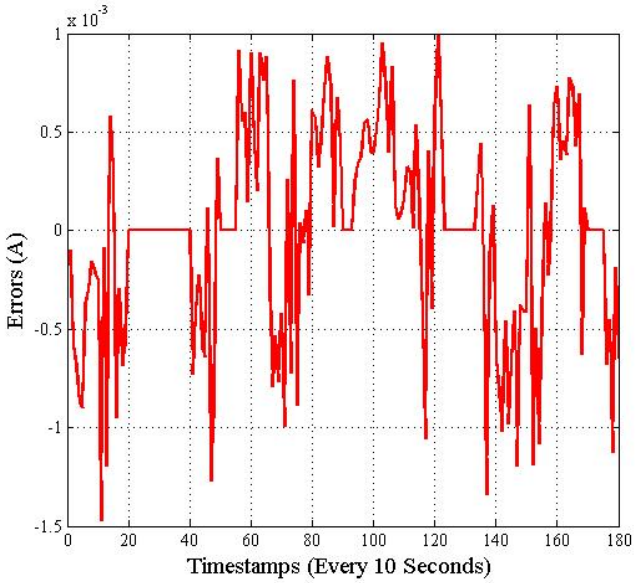

Fig. 2. Differences between M-C and clamp meter readings

After validating M-C system, it was placed at the main switch AC power cable. The HP2920-24G openflow enabled switch was used in our experiment setting. The switch ports can be configured for 10/100/1000 Mpbs half/full duplex.

To get all switch ports up and running, another HP switch was used. Twenty cat5 UTP crossovers cables were connected between the two switches. The last four ports were connected through direct UTP cables to three laptops and a wireless access point. One of these laptops has a Bluetooth connection to receive data from M-C circuitry. A Processing script [27] was written to record Bluetooth received data in a file for later analysis. M-C system sends an AC current measurement every 2 seconds. The consumed power was measured in three different scenarios. In the first scenario, we measure power consumption of the core switch components alone during the start-up; start-up time was recorded also; switch's ports in this

scenario were disabled. In the second scenario with all ports enabled and fully utilized power consumption readings were recorded. In the third scenario with increasing switch port speed from 10 to 100 to 1000 Mpbs power consumption readings were recorded.

Fig. 3 shows power consumption in the first scenario. We can observe that the switch uses the maximum power at start-up time. This power is used to test all internal components of the switch. The start-up time is 38 seconds. We also can observe that the measured power readings oscillate. The oscillation is due to the noise from PMCT sensor. However, we can observe that power readings oscillate about an average value of 212 watts.

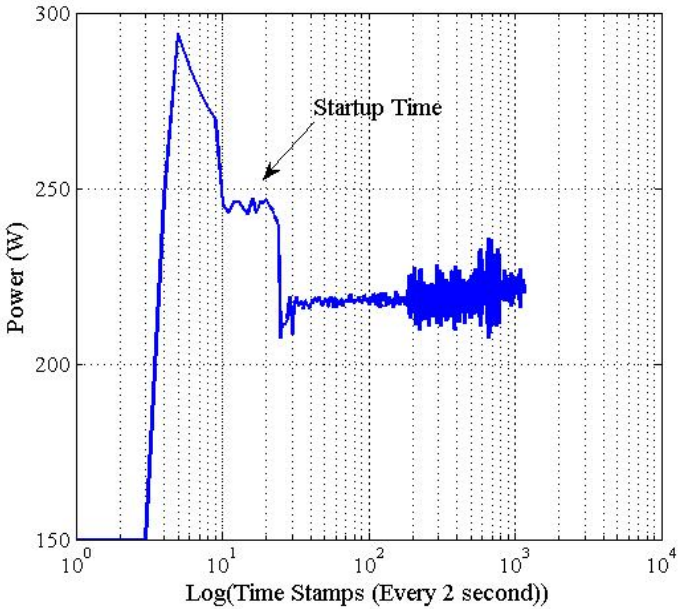

Fig. 3. Power Consumption in the first scenario

Fig. 4 shows power consumption in the second and the third scenarios. We attempted to measure the power increase of each switch port alone after we enable it. However, the increment in power usage of each port is small. To overcome this issue, we measured power consumption of 12 ports (half of switch ports) collectively after we enabled them. We can observe from the figure how power consumption had a pulse like increase when we enabled the 12 ports with a speed of 100 Mbps. Power consumption oscillates about 241 watts average value. Subsequently, we enabled the other 12 ports to obtain a fully enabled 24 ports with 100Mpbs speed. The power consumption jumped to 259 watts. This shows that approximately 1.8 watts is consumed when any port is enabled and connected to a device. It worth mentioning that enabling a port does not consume power; however, attaching an enabled port to a device consumes power.

In the third scenario, we kept 12 connected ports and reduced the port speed to 10Mpbs then increased it to 1000Mpbs. We can observe that average power consumption has decreased to approximately 222 watts. However, when we changed speed configuration into 1000Mpbs, three ports were disconnected; access point and the two laptops. The reason is that these devices have a maximum port speed of 100Mpbs. Nine ports stayed connected and the power

consumption jumped to approximately 260 watts. This shows that a 1000Mpbs port utilizes approximately 2.8 watts. **Table 1** summarizes these numbers.

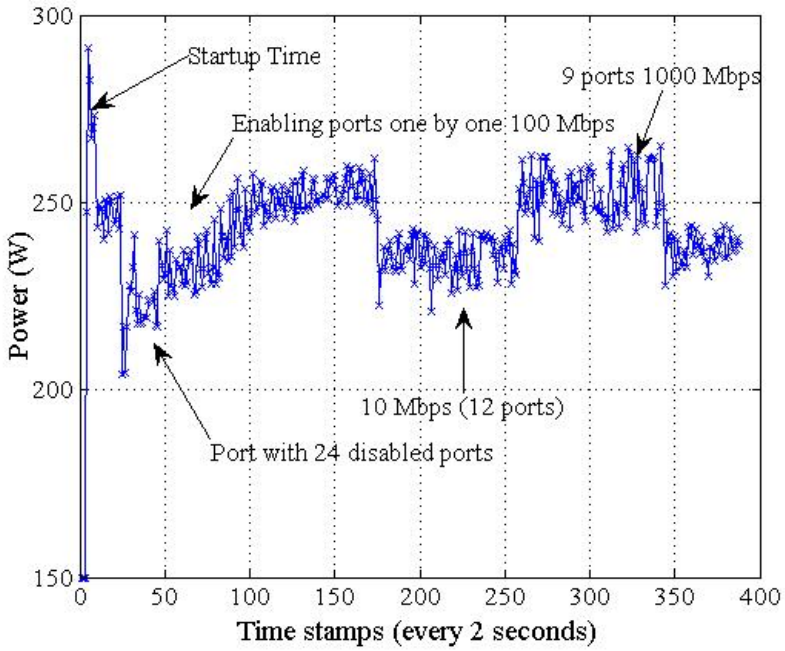

Fig. 4. Power consumption in the second and third scenarios

Table 1. Power consumption Summary

Parameter	Value
100 Mbps Port Power Consumption	1.8 Watts
1000Mbps Port Power Consumption	2.8 Watts
Startup time delay	38 Seconds

To measure power consumption of each port under traffic load, FTP server was deployed. A folder of 10 GB of data was shared through this server. Three clients requested the folder from the server. No changes in power consumption were recorded using M-C device.

We conclude four main points. First, PCMT sensor is a useful method to track devices power usage. Second, switch utilizes maximum power at start-up time for a short time period. Third, connected ports utilize more power even if they have no traffic load. Finally, more port speeds require more power since electronic components in physical layer uses more power. These conclusions facilitated DRA algorithm design.

4. Duplication Resolver Algorithm (DRA)

In this paper, SDN controller is utilized to reduce power consumption in two different ways. First, SDN controller will have the ability to turn off/on network switches; it also will have the ability to reduce switch start-up time by embedding flow table of the suspended switch in the new one. Second, SDN controller will have the ability to turn switches' ports off or disable them to reduce power usage.

SDN controller in this work is not only a controller of openflow-enabled switches that transfer commands over Openflow protocol. The SDN controller has also the ability to reconfigure a switch to disable its ports. Moreover, the controller controls an external circuit that is responsible of turning switches on and off. In the following sections, we will overview each one of these tasks.

4.1 Switch Power control

In this section we will introduces the procedure of powering a switch on and off. Next section explains the mechanism and algorithm to do so. Turning switch on/off cannot be handled from configuration CLI or GUI. It only can be done through power line and the on/off button. The only configuration command that control power is the reload command. Moreover, switch does not support sleep or hibernate modes. To control the power of a switch in a simple way, an external circuit has been constructed utilizing Arduino kit. A relay module, Ethernet shield and Arduino kit were used to build the controller circuit (C-C). The relay module is used to turn on/off the network switch according to the received command through its Ethernet interface. The SDN controller is responsible of sending these commands to the C-C system. The C-C circuit has a server socket that listen for a secure TCP connection from a client socket implemented in SDN controller. The C-C circuit can control up to 13 switches. This number can be increased by utilizing other variants of Arduino kits, such as, Arduino mega and decoders. Arduino mega can handle up to 50 switches. Using decoders, this number can be hundreds. **Fig. 5** shows the overall power control system. The C-C's server socket receives a message with three variables. The first variable is controller ID, second variable is switch ID and the last variable is a logical value for on/off command. **Fig. 6** shows the structure of this message.

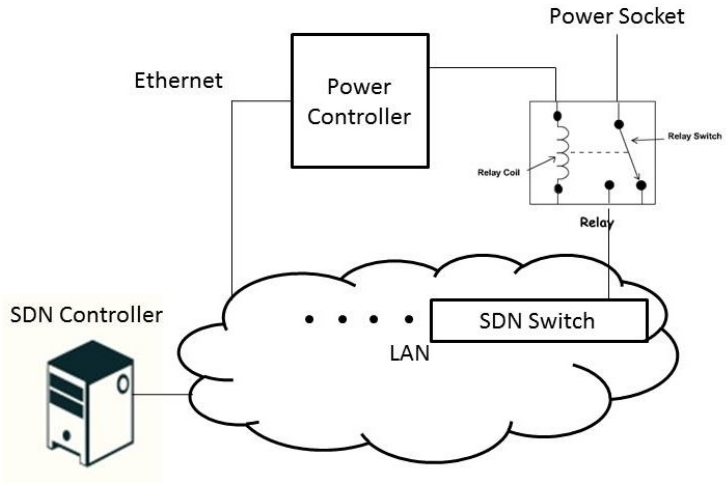

Fig. 5. Power Control Model

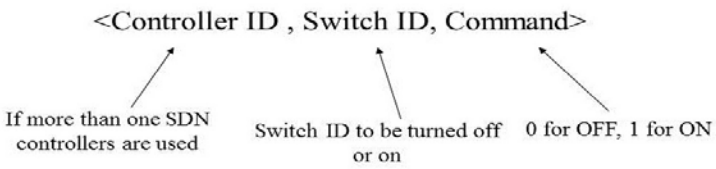

Fig. 6. Command Message

4.2 DRA Implementation

DRA is responsible for selecting which switch to turn off and which one to keep on. The algorithm requires SDN controllers in the network to have a centralized cache server to save some required fields. In our work, a single SDN controller is used as a controller and as a cache server. However, any number of SDN controllers can be used and one of them is selected as the cache server.

The algorithm consists of three steps. In the first step, SDN controller is required to disable all switches' ports that are not connected to any host or network device. This can be done by opening a telnet session with each switch and configure these ports to a disable state. Subsequently, the controller will create a list for each switch indicating the status of its ports. Each list will have the ID of the switch as its identifier.

In the second step, SDN controllers will save a tuple of data for each received packet or flow in the cache server. The tuple consists of; switch device ID, incoming port number, IP address and source hostname. The controller is responsible of mapping source IP address to its name by inquires the network. Many methods can be used for this mapping; nbstate, nslookup, ping or Nmap [28] to name a few. Nmap is utilized in the proposed system. One thing to be mentioned is that this step adds no delay to the network since it is done after the flow is inserted into the switch flow table.

In the last step, the controller runs DRA on the cache's registered tuples. The DRA algorithm searches for any duplicated hostname with different IP addresses in the harvested tuples. If duplication is found, one of the host's connected ports must be disabled or shut down. DRA selects the port of the switch with smallest ID and disable it. The SDN controller will open a telnet connection with the selected switch and disable the selected port. Subsequently, DRA will changes the status of this port in the switches lists. Finally, after the search process of DRA finishes, DRA searches switches lists for any switch with a complete list of disabled ports. If this switch is found, SDN controller will send a shutoff command to Arduino kit to power off this switch. One thing to be mentioned is that SDN controller runs DRA algorithm ones every 10 minutes on the collected tuples. Alg.1 shows pseudo code of DRA algorithm. We can observe that, DRA is an easy to implement algorithm. It has complexity of ***O(nlogn).*** Where n is the total number of ports connected to users in the network.

4.3 Failure Recovery

Shutting down duplicated switches and ports can lead to network breakdown in case of switch failure. Since the suspended switches need to start-up and build their forwarding table again. Network breakdown can be mitigated if we reduce the switch start-up time, enable

hibernate and sleep modes. Unfortunately, switches' vendors do not support these functions. In this work, we cannot configure devices to reduce their start-up delay. However, zero-time flow table construction can be achieved. The saved tuples that have been utilized by DRA algorithm are also utilized here. When a switch fails, SDN controller turns on the duplicated suspended network switch. Subsequently, SDN controller adds all the flows of the failed switch to the new one and turns its ports on. In this case, the device will eliminate the time delay for constructing new flowtable. Moreover, switches' ports will be configured as access ports and the Fastport property will be set to reduce port startup time to zero. With this property, the new started port will not go into any initializing process and will not be included in span tree protocol (STP) calculation.

Finally, sometimes the new switch ports will have bandwidth less than the failed switch port. In this case, DRA will configure multi-ports into one virtual port group. This group will act as a single port. The group will have a bandwidth equal to the bandwidth of the failed port. One thing to be mentioned here is that Fastport and virtual group properties are supported in many of the new commercial switches.

5. The Experiment

Our experiment consists of two parts. The first part is a simulation experiment utilizing MATLAB to demonstrate how DRA algorithm reduces power consumption to achieve a theoretical lower bound. The second part of the experiment is to implement DRA algorithm utilizing our HP2920 openflow enabled switches and Arduino kit to turn off/on switches. The following sections demonstrate the two parts.

5.1 DRA Simulation

In this section, we will try to answer the following question: How much power reduction can be achieved? To answer this question, a network model should be used. Network hierarchical model [29] is utilized in this work. Hierarchical model consists of three layers: access, distributed and a core layers. Each layer is responsible of different functionalities. Each layer can be implemented utilizing layer two and multi-layers switching devices. **Fig. 7** shows an example of this model. This model duplicates switching devices at each layer. Moreover, each computer, end node has two network interface cards to connect with the model.

In this model, two modes of operations are used: normal mode, in which all network devices and ports are enabled. This setting represents the theoretical power consumption upper bound. In half-normal mode, duplicated switching devices and ports are disabled or turned off. This setting represents the theoretical power consumption lower bound. **Fig. 8** shows an example of the same network in half-normal mode. In the following subsections, this model will be utilized in the conducted simulation.

Algorithm 1 Duplication Resolver Algorithm

$Port(i)$: Status of port i
SWT: total number of switches
$Prt(j)$: total number of ports of switch j
ara: array of Switches IDs
$LST(ID, Port(i))$: A directory or 2 dimensional array, i port number
$Hostname(j)$: hostname of machine with tuple address j
$tuples$: data saved in cache server which has three attributes

```
for all n in ara do
  for all j in Prt do
    if Port(j)==disconnected then
      Port(j)=disable
      LST(n,j)=disable
    end if
  end for
end for
for all i in tuples do
  temp=Hostname(i)
  tempIP=IP(i)
  for all j in tuples do
    if temp==Hostname(j) then
      if tempIP==IP(j) then
        LST(ID,j)=disable
      end if
    end if
  end for
end for
for all n in SWT do
  if all Prt(n)==disable then
    Turn_off(n)
  end if
end for
```

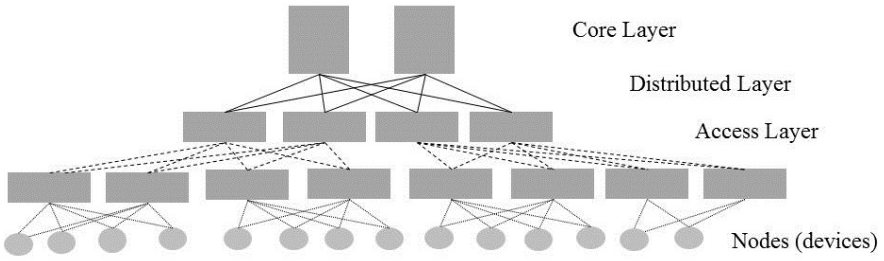

Fig. 7. Example of hierarchical model with all devices turned on

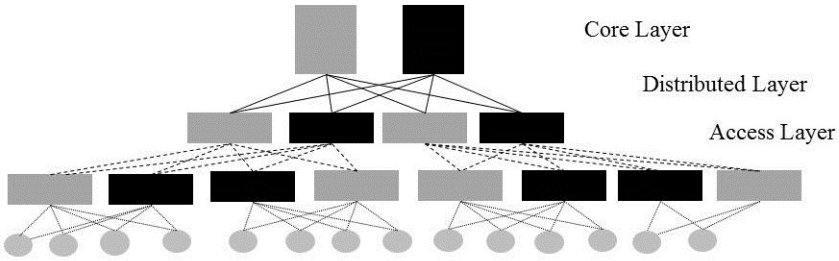

Fig. 8. Example of hierarchical model with Black devices turned off

5.1.1 Power Reduction Boundaries

The power boundary model utilized in the simulation was constructed based on the results harvested from M-C system. We have observed from M-C system readings that each port with 100Mbps of the switch consumes 1% more power if it is enabled and approximate 1.5% if its 1000Mpbs port. By utilizing the collected data in the hierarchical model, a power consumption model can be derived. **Table 2** shows variables used in the model.

Table 2. Variables Definitions

Variable Name	Variable Definition
CP	Total consumed Power
P_s	Power consumption of switch with all ports disabled
$d(i)$	Number of devices in layer i
$d(0)$	Number of computers and end devices in the network
P_P	Power percentage consumed by a Port
$Np(i)$	Number of ports in layer i

The number of devices in each layer can be calculated using eqn. 2. This number can be utilized to calculate the total power consumption of the tree layers as in eqn. 3. The negative term in the equation compensates for the uplink ports of the core switches since they have no connections with other upper devices. This term has small impact and can be neglected.

$$d(i) = \frac{2*d(i-1)}{N_p(i)-2} \qquad (2)$$

$$CP = -\left(P_s P_p * d(3) * 2\right) + \sum_{i=1}^{3} (P_s * d(i) + P_s P_p * N_p(i))$$

$$CP = -\left(P_s P_p * d(3) * 2\right) + \sum_{i=1}^{3} P_s * \left(\frac{2*d(i-1)}{N_p(i)-2} + P_s P_p * N_p(i)\right) \qquad (3)$$

MATLAB was utilized to generate a hierarchical network model with different number of devices. Subsequently, DRA algorithm was implemented and run on the generated model. All switches in our simulation are identical with the same number of ports (24 ports). We have calculated power usage for the upper bound (normal), lower bound (half-normal) and DRA algorithm modes, see **Fig. 9**. The figure shows that DRA mode attempted to reach lower boundary. However, to reach this boundary, end nodes should be connected to the network in a

configured and arranged settings with all devices connected to a switch are connected to the second duplicated switch. P_s in our work is 200 watts as observed in the measurement section.

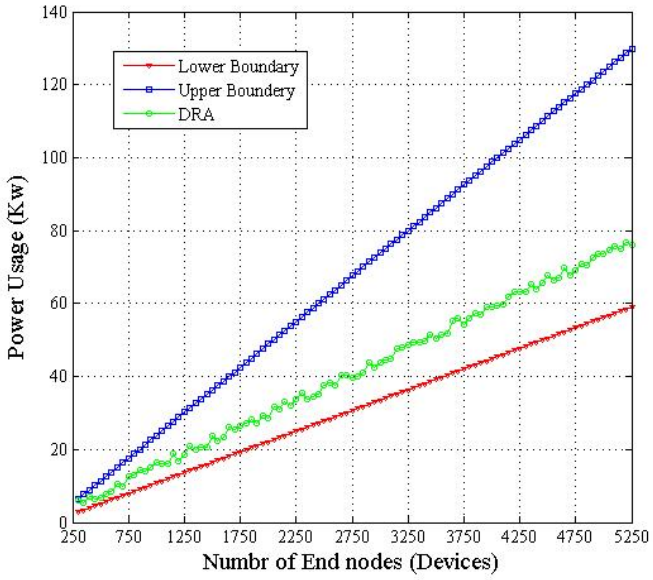

Fig. 9. DRA Boundaries

5.1.2 Startup power Impact

Fig. 3 and **Fig. 4** showed that the startup power consumption of network switches is higher than power consumption during normal operation. This consumption lasts for short time, namely, startup time. When DRA algorithm turns switches on/off, network switches consume more extra power at startup. To measure the impact of power consumption at startup time, device failure probability and redundancy percentages should be measured from a real datacenter. Measurement data from research work in [31] were used. In [31], the author claimed a failure percentage of 5% to 10% per year for network switches. Moreover, a full redundant topology was shown. Fig.10 shows power consumption reduction utilizing different switch redundancy percentage with a 10% failure probability. We can observe from the figure that power reduction occurs with all redundancy percentages. We can also observe that this reduction decays with the number of times switches powered up. Fig. 10 was generated by varying the number of times switches powered up from 1 to 200 in the redundant group. This show that even with the impact of startup power consumption, DRA will be capable of reducing power usage.

Finally, **Fig. 11** shows this impact with different failure percentages. We can observe that higher failure percentage reduces power reduction. However, the reduction is remarkable in all scenarios.

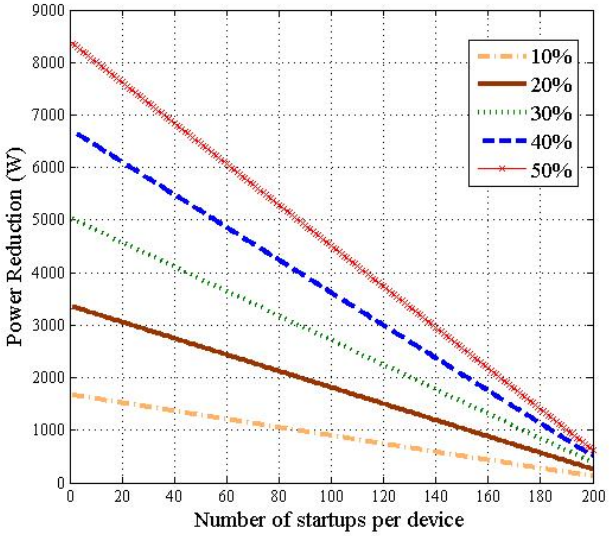

Fig. 10. Power Reduction with Different Redundancy Percentage

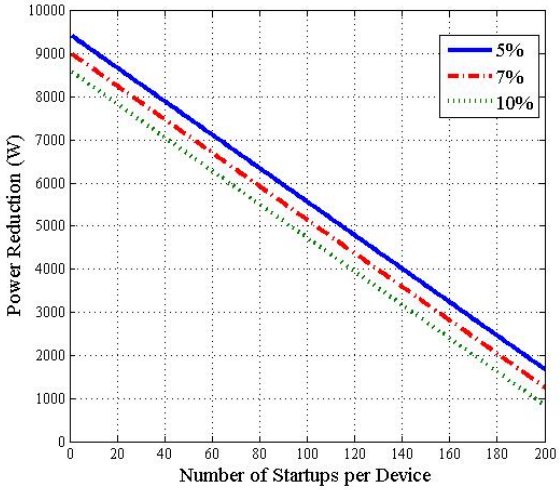

Fig. 11. Power Reduction with Different Failure Percentages

5.1.3 Tradeoff between Packet Loss and Power Reduction

Redundancy in network devices is used in network design and implementation to enhance network availability and reduce failure impact when it occurs. However, with the frequent powering up and down of switches, packets will be lost during startup delay time. For example, utilizing our switches, 30 seconds of packets will be lost. However, three main facts reduce this impact. First, network devices are reliable with failure probability of 5% per year [31]. Second, redundancy is not effective in all cases as reported in [31] that 40% of failure scenarios did not benefit from device redundancy. Finally, DRA does not only turn off switches, it also turns ports off and on. This means that switches may still running with multi-ports turned off to reduce their power consumption. We attempted to measure packet loss of single port in the transit state. An FTP server was deployed and a client started downloading data. Subsequently, the port was turned off then on. The lost packets duration

was less than 1 second. This can be provided by configuring the switch ports in access mode with fast-port startup command.

5.2 DRA Implementation

In this part, DRA-perform-testbed system has been constructed with two HP2920-24G switches and one Ryu SDN controller [13] installed in Intel i7 computer with 4G RAM and Ubuntu 15.4 operating system. Each switch has one VLAN connected to the controller with 12 ports. Python has been used to program the DRA algorithm and run it in Ryu controller since this SDN controller is a python based controller. A telnet client has been written in python to open a session with the two switches and configures their ports properties upon requested from DRA. In addition, a client TCP socket has been written to send messages to the C-C system that control power supply of these switches. The Arduino kit has two relays to control power of these two switches. Two client laptops have been connected to the two switches. However, we only have one NIC card on each laptop. To add another interface for each device, a USB Ethernet cards have been installed in each device. A DHCP server has been configured on the SDN controller laptop and a local DNS server. **Fig. 12** shows the configuration of the DRA-perform-testbed.

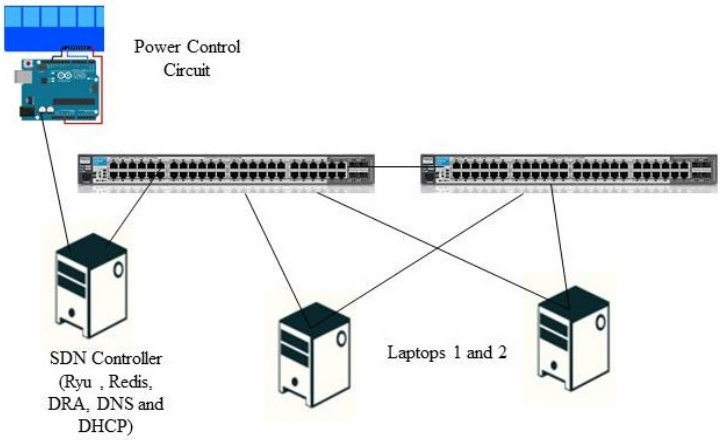

Fig. 12. Testbed configuration

To save data tuples received from switches, Redis caching server [30] has been downloaded and installed in the SDN controller. Redis was used for three reasons, first, it is free. Second, it is faster than a database management system (DBMS) in saving temporary data. Finally, it has a python driver, which simplifies its connection with DRA algorithm. Finally, to reduce delay and interruption in DRA, the algorithm runs multiple threads; one thread for openflow and tuple insertion; another thread for DRA calculations. In this way, no delay will be added to SDN controller normal operation. Finally, DRA algorithm was scheduled to run every 10 minutes.

We ran the DRA-perform-testbed and waited for 10 minutes before DRA runs for the first time. In the first attempt, DRA turned off ports that are connected to the second Ethernet of each device. However, the selection of ports did not help in turning off one switch. The selection of port numbers depended on the smallest ID number. We selected MAC address as an ID of each card. To solve this issue, we changed the wiring of one laptop of the two laptops

and waited 10 minutes. When the wires have been disconnected and re-connected again, DRA runs all ports and collect data again before it turns duplicated ports off again. This time, one of the switches turned off. Both of the laptops were connected to the network without any disconnection.

To generate a failure, the up and running switch was turned off manually; SDN control received the status of the switch and DRA turned all the switches on again. SDN controller waited start-up time delay of the switch before sending flows to switch. We compared the time required to add these flows with and without DRA. Wireshark was used to record the required time. **Table 3** shows this time. We can observe that DRA can reduce the complexity of building flow tables by saving them and embedding them upon request.

Finally, it worth mentioning that large-scale experiments with more network devices and nodes have to be conducted to gain more insights of power reduction and DRA behavior in large-scale networks. Moreover, DRA should be studied with SDN environment with multi-controllers. We attempt to conduct these experiments in the future.

Table 3. DRA Flowtable Embedding Time

Method	**Time (s)**
DRA	1
NO DRA	Without DRA from 30 s to unknown depends on traffic

6. Conclusion

Power consumption in ICT sector is rapidly growing especially in datacenters. In this work, SDN paradigm has been utilized to control power consumption in datacenter through turning on and off network middle boxes. Two electrical circuits were implemented and deployed in this work. The first one is the M-C (measurement circuit) circuit, which is a simple Arduino based circuit that is capable of measuring AC current in switches' wires externally. This circuit was utilized to measure power consumption in SDN switch in three different scenarios. The measurement was used to derive an equation of total power consumption for hierarchical network model. The second one is the C-C (controller circuit) circuit, which was implemented to control power supply of network devices by powering them up and down. To this end, DRA algorithm has been proposed to control the C-C circuit to reduce power consumption by selecting which switch to turn off and which to keep on. A DRA-perform-testbed has been implemented utilizing Ryu SDN controller and two HP2920-24G switches to test performance of DRA algorithm. Our results show how DRA successfully shutdown duplicated switch. Moreover, the results show how DRA can construct flow-tables in a fast manner.

7. Acknowledgment

This research was supported in part by Al-Zaytoonah University of Jordan fund (2/11/2014). We would like to thank the University for the equipments and tools they provided for this work.

References

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

[2] S. Lambert, W. Van Heddeghem, W. Vereecken, B. Lannoo, D. Colle, M. Pickavet, "Worldwide electricity consumption of communication networks," *Optics express*, vol. 20, no. 26, 2012. Article (CrossRef Link)

[3] M. Pickavet, W. Vereecken, S. Demeyer, P. Audenaert, B. Vermeulen, C. Develder, D. Colle, B. Dhoedt, P. Demeester, "Worldwide energy needs for ICT: The rise of power-aware networking," in *Proc. of IEEE 2nd International Symposium on Advanced Networks and Telecommunication Systems*, pp. 1–3, 2008. Article (CrossRef Link)

[4] G. Schomaker, S. Janacek, D. Schlitt, "The energy demand of data centers," in *Proc. of ICT Innovations for Sustainability*, Springer, pp. 113–124, 2015. Article (CrossRef Link)

[5] R. Hintemann, "The Impact of the Changing Structure of Data Centers on Total Electricity Demand," *ICT Innovations for Sustainability*, Springer International Publishing, pp. 125–136, 2015.

[6] D.C.K. Report (2014). URL http://www.datacenterknowledge.com/archives/2014/11/11/idcamountof-worlds-data-centers-to-start-declining-in-2017/

[7] W. Vereecken, W. Van Heddeghem, D. Colle, M. Pickavet, P. Demeester, "Overall ICT footprint and green communication technologies," in *Proc. of IEEE 4th International Symposium on Communications, Control and Signal Processing (ISCCSP 2010)*, 2010. Article (CrossRef Link)

[8] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright, "Power awareness in network design and routing," in *Proc. of IEEE 27th Conference on Computer Communications (INFOCOM)*, 2008. Article (CrossRef Link)

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no.2, 2008. Article (CrossRef Link)

[10] M.Z. Masoud, Y. Jaradat, I. Jannoud, "On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm," in *Proc. of IEEE Applied Electrical Engineering and Computing Technologies (AEECT)*, pp. 1–5, 2015. Article (CrossRef Link)

[11] H. Kumar, H.H. Gharakheili, V. Sivaraman, "User control of quality of experience in home networks using SDN," in *Proc. of IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS 2013)*, pp. 1–6, 2013. Article (CrossRef Link)

[12] M. Banzi, "Getting Started with Arduino," O'Reilly Media, Sebastopol, CA, 2008

[13] Tomonori FU, "Introduction to ryu sdn framework," Open Networking Summi, Apr, 2013.

[14] J.G. Koomey, "Estimating total power consumption by servers in the us and the world," 2007.

[15] D. Economou, S. Rivoire, C. Kozyrakis, P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Proc. of IEEE International Symposium on Computer Architecture*, 2006

[16] E. Pakbaznia, M. Pedram, "Minimizing data center cooling and server power costs," in *Proc. of ACM/IEEE international symposium on Low power electronics and design (ACM, 2009)*, pp. 145–150, 2009. Article (CrossRef Link)

[17] E. Pakbaznia, M. Ghasemazar, M. Pedram," Minimizing data center cooling and server power costs," in *Proc. of the Conference on Design, Automation and Test in Europe (European Design and Automation Association)*, pp. 124–129, 2009.

[18] B. Battles, C. Belleville, S. Grabau, J. Maurier, "Reducing data center power consumption through efficient storage," *Network Appliance*, 2007

[19] Q. Huang, F. Gao, R. Wang, Z. Qi, "Power consumption of virtual machine live migration in clouds," in *Proc. of IEEE Third International Conference on Communications and Mobile Computing (CMC) 2011*, pp. 122–125, 2011. Article (CrossRef Link)

[20] C. Dupont, T. Schulze, G. Giuliani, A. Somov, F. Hermenier, “Future Energy Systems: Where Energy,” in *Proc. of Third International Conference on Computing and Communication Meet (e-Energy)*, pp. 1–10, 2012.
[21] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown, “ElasticTree: Saving Energy in Data Center Networks,” *NSDI*, vol. 10, pp. 249–264, 2010.
[22] X. Wang, Y. Yao, X. Wang, K. Lu, Q. Cao, “Carpo: Correlation-aware power optimization in data center networks,” in *Proc. of INFOCOM*, pp. 1125–1133, 2012. Article (CrossRef Link)
[23] H. Shirayanagi, H. Yamada, K. Kenji, “Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks,” *IEICE TRANSACTIONS on Information and Systems*, vol. 96, no.9, pp-2055, 2013. Article (CrossRef Link)
[24] S.H. Wang, P.P.W. Huang, C.H.P. Wen, L.C. Wang, “EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks,” in *Proc. of IEEE The International Conference on Information Networking (ICOIN2014)*, pp. 220–225, 2014. Article (CrossRef Link)
[25] N. Vasi´c, P. Bhurat, D. Novakovi´c, M. Canini, S. Shekhar, D. Kosti´c, “Identifying and using energy-critical paths,” in *Proc. of the Seventh Conference on emerging Networking Experiments and Technologies*, ACM, p. 18, 2011. Article (CrossRef Link)
[26] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, “A power benchmarking framework for network devices,” in *Proc. of International Conference on Research in Networking*, Springer, pp. 795–808, 2009. Article (CrossRef Link)
[27] Processing programming language. URL http://Processing.org
[28] Nmap tool. URL https://nmap.org
[29] C.N. Academy, Connecting Networks Companion Guide (Pearson Education, 2014)
[30] Redis cahcing server. URL http://redis.io
[31] Gill, Phillipa, Navendu Jain, and Nachiappan Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, ACM, 2011. Article (CrossRef Link)

Mohammad Masoud received his B.S. in computer engineering from Mu'tah University, Karak, Jordan, in 2006, and the M.S. in electrical and computer engineering from NYIT, Amman, Jordan, in 2008. He obtained his PhD in communication and information engineering from Huazhong University of Science and Technology, Wuhan, China in 2013. He is currently an assistant professor in electrical and computer engineering at Al-Zaytoonah University of Jordan. His research includes work in the design and measurement of computer network and their applications, P2P networks, ad hoc networks, WSN and embedded systems.

Yousef Jaradat received the B.S. in electrical and computer engineering from Jordan University of Science and Technology, Irbid, Jordan, in 2000, and the Ph.D. in electrical and computer engineering from New Mexico State University, Las Cruces,New Mexico, USA, in 2012. He is currently an assistant professor with the department of electrical and computer engineering at Al-Zaytoonah University of Jordan. His research includes work in the capacity of wireless networks, Obstacles analysis in the network area, and simulation of wireless networks. He is a member of IEEE.

Ismael Jannoud is an Associate professor in electrical and computer engineering at Al-Zaytoonah University of Jordan, Jordan, and Damascus University, Syria. His research includes work in the digital image processing, computer vision, pattern recognition, multimedia processing, and computer networks.

Hong Huang received his B.E. degree from Tsinghua University, Beijing, China, and M.S. and Ph.D. degrees from Georgia Institute of Technology in 2000 and 2002, respectively, all in electrical engineering. He is currently an associate professor with the Klipsch School of Electrical and Computer Engineering at the New Mexico State University. His current research interests include wireless sensor networks, mobile ad hoc networks, network security, and optical networks. He is a member of IEEE.