Towards Green SDN: An Approach Based on Graph Connectivity

[1]Eder J. Scheid, [1]Muriel F. Franco, [1]Matias A. K. Schimuneck, [2]Cristiano B. Both, [1]Juergen Rochol and [1]Lisandro Z. Granville

[1]Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil
[2]Federal Health Science University of Porto Alegre (UFCSPA)
Porto Alegre – RS – Brazil

[1]{ejscheid, mffranco, makschimuneck, juergen, granville}@inf.ufrgs.br
[2]cbboth@ufcspa.edu.br

Abstract. *The reduction of energy consumption is a key research topic area in computer networks. Green networking consists of selecting energy-efficient networking technologies and minimizing resource use whenever is possible. Software-Defined Networking (SDN) is a new networking paradigm that allows innovation and simplifies network management, which leads to the development of new approaches to reduce energy consumption. In this paper, we present a solution for energy efficiency in business networks, based on the shutting down switches identified as idle. Our proposed approach have been prototyped using a component-based SDN framework denominated Ryu, which has been used to create the control application. The evaluation results discloses the influence of switch usage to save energy. We can eliminate inactive switches and redundancy, and so, reduce energy consumption in a simulated environment.*

1. Introduction

Reduction of energy consumption is becoming an interesting research topic in computer networks [Bianzino et al. 2012]. In this context, Green networking arise as a new approach to rethink network design and implementation. Traditionally, networking systems are designed and dimensioned with a set of classical principles, but in the recent years, a new energy concern emerged together with technology evolution. The primary goal of Green networking is minimizing resource usage, by shutting down of inactive routers, and consequently, leading to energy saving.

Nowadays, many challenges related to power management emerge. Both Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) have high potential and in many ways are mutually supportive [Feamster et al. 2014]. SDN and Virtualization solve many problems that Network and Service Providers have to face for optimizing their infrastructures, such as reducing costs. Besides that, these paradigms emerge as an interesting tool to help the reduction of the network's energy consumption.

SDN is a new paradigm that allows innovations and simplifies network management. It is dynamic, manageable, cost-effective, and adaptable, making it ideal for dynamic systems. This architecture decouples the network into two planes, the control plane and the forwarding plane. This paradigm improves the capacity to design, deploy, and manage energy-efficient systems [Jammal et al. 2014].

In large networks, switches and forwarding devices can be turned off when inactive or dispensable, in order to save energy. We consider that a switch is inactive when there is no traffic and is dispensable when others switches that can sustain the traffic and support the connectivity of the network. It is essential that energy saving actions should not interrupt or produce a limitation in the network traffic.

In this paper, we present a prototype that discovers the minimum green routing path, where the traffic pattern is used to support its decisions. When network traffic is considered low, we use a customized topology to avoid the usage of some switches. Our algorithm is based on using a technique that eliminates redundant switches in the topology. An SDN controller can then find which switches can be removed and route the packets based on traffic observation patterns.

The remaining of this paper is organized as follows. In Section 2, we present a brief topic overview and related work. In Section 3, we introduce our controller prototype and describe our algorithm in details. In Section 4, we present a measurement environment, results, and discussions. Finally, in Section 5, we conclude the paper and list future work.

2. Background and Related Work

A significant challenge in green networking is finding a better way to guarantee full network operation while saving energy. Albers shows in his work [Albers 2010] that the two states advanced algorithm is a positive and simple way to reduce energy consumption. Grupta and Singh, in turn, was the pioneer on verifying the impact of network protocols on energy saving by putting network interfaces to a sleep state. These authors identify the problem of excessive energy consumption on the Internet and propose sleeping as an approach to save energy.

Bolla et al. proposed an energy-saving mechanisms based on topology optimization for the extension of traffic engineering. The basic approach is to reduce the network capacity, in terms of links and nodes, to match the actual traffic volumes. In other words, they try to route all traffic flows among network nodes using the minimum number of network resources [Bolla et al. 2011].

SDN can present an opportunity for the allocation of traffic flows and, at the same time, facilitating the management of power state of network nodes. As a consequence, recent Future Green Internet research efforts have been focusing on the SDN paradigm to ease energy management. Bruschi et al. propose a Green Abstraction Layer (GAL) to integrate the power management data and then it applies control strategies inside the SDN [Bruschi et al. 2014]. The results support the feasibility of allocating resources according to the incoming traffic characteristics while reducing the overall network energy consumption.

Other related work explores the SDN paradigm based on energy-aware flow scheduling. Li et al., for example, use exclusive routing for each flow in a data center to guarantee that it does not to compete for link bandwidths with others flows [Li et al. 2014]. Thanh et al. propose a new testbed architecture that combines hardware network devices with virtual emulation test environment to improve scalability, flexibility, and accuracy [Thanh et al. 2013]. This testbed enables the design and the experiment of new solutions

for an energy-efficient data center. It combines smart sleeping and power scaling mechanisms. This new algorithm proposed can save up to 35% of energy consumption in case of a 16-server data center. Almost all the work in the proposal was developed, concluding that the energy saving level can be improved as the size of the data center increases.

3. An Approach to Green SDN

In our approach, we seek to reach an optimized topology in terms of energy efficiency. Therefore, we expect to shutdown the largest number of switches as possible. The choice of turning on or off a switch is performed according to the existence of alternative routes. If there are other ways to route the traffic, the switch can be turned off. Thus, in a low traffic situation, this switch remains off, being activated only in the event of the increase of demand. Another concern is related to the energy waste for restarting network equipment. Considering that this process can be very costly, we define a waiting window of three hours before changing a switch state.

Algorithm 1 optimized_graph_discovery

```
Input: G-graph
Output: S-graph
for i in switches do
  edges_copy ← edges(s_graph, i)
  s_graph.remove_node(i)
  if is_connected(s_graph) is False then
    s_graph.add_node(i)
    s_graph.add_edge(edges_copy)
  end if
end for
```

In the first step of our implementation, we map the network topology in a G-graph. After, we find a G-subgraph that contains all connected hosts. This subgraph, will be our optimized graph S. Aiming to exclude unnecessary interfaces, we focus on a graph property called connectivity, which is one of the fundamental concepts of graph theory: it asks for the minimum number of elements (nodes or edges) that need to be removed to disconnect the remaining nodes from each other. Algorithm 1, is used to find a S-graph.

Algorithm 1 creates a copy of G-graph, which is called S-graph, and applies a greedy algorithm in A-graph to find nodes that can be removed. First, it removes $switch_n$ from S-graph and verifies the graph connectivity. If the connectivity is still true, $switch_n$ can be removed; otherwise, $switch_n$ is inserted in the S-graph again.

Algorithm 2 find_routing_path

```
Input: source, destination
Output: forward-path
if currently_traffic is "Low" then
  path ← shortest_path(s_graph, src, dst)
else
  path ← shortest_path(g_graph, src, dst)
end if
```

After calculating S-graph, we implement the controller algorithm. This implementation, consists in defining the best topology to use for a determined network state. For example, when the network is experiencing a high demand, we need to use G-graph to forward packets. Otherwise, when the network is experiencing low traffic, we set G-graph and remove switches that are not in use. Algorithm 2 is used to define current path of the incoming packets.

As Algorithm 2 denotes, we check the current period flag. If the flag has the "low" label, we find the shortest-path between packetsource and destination in S-graph, and with this path we set new rules in the switches to forward the packets. On other hand, when the flag has the "high" label, we use G-graph to calculate the route, since it is an indicative that the network traffic requires a large number of network resources.

Our proposed approach have been prototyped using component-based software-defined networking framework called Ryu [Team 2014], which has been used to develop the control application. This framework offers a platform for building SDN applications and provides useful libraries and a well-defined API to use all of OpenFlow (OF) versions. The controller waits for an *OFPacketIn* event to make a decision to forward the packet. If there is no rule for the destination, we add a flow rule based on the currently topology graph.

This approach has a good performance in networks with high-redundancy and multiple routes. Moreover, it is easy to maintain and add new functions. Also, it can be promptly activated in any network, with just a few configurations. For example, network administrators need to set traffic periods patterns, and the algorithm is responsible for removing switches and reporting to the administrator details about the current network status and switches.

3.1. Policy Monitor and Enforcer

To determine when to use an approach that reduces the energy consumption in the network, policies have been written. These policies are based on a traffic pattern measured on a link maintained by the RNP (the Brazilian research and education network) [Rede Nacional de Ensino e Pesquisa 2015].

The traffic pattern analyzed was the traffic over a week (Figure 1). In this graphic Axis Y represents the input (grey area) and output (black line) traffic on a link between two Brazilian states, while Axis X represents the day of the week. We can see that around midnight the traffic decreases from 7 Gbps to 1 Gbps and continues at this rate until the morning. The assumption inferred repeats itself, creating spikes between the 9th and the 12th. On the other hand, on the 13th and the 14th, which is Saturday and Sunday, there is minor traffic on the network, but on the 15th the traffic returns at its normal rate. Therefore, between these hours (midnight and 6-am) and on the weekends, the network can stay at a power conservative state.

Having those policies determined, a monitor was implemented within the controller. This monitor is responsible for enforcing the energy conservation policy if there is a match during the monitoring. Every fifteen minutes, it compares the current date and time with the hard-coded policy, and if the network (graph) is not optimized yet, it will optimize it. Also, when the current date and time do not match the policy, meaning that there is a need for the full network, the graph is restored to its full capacity.

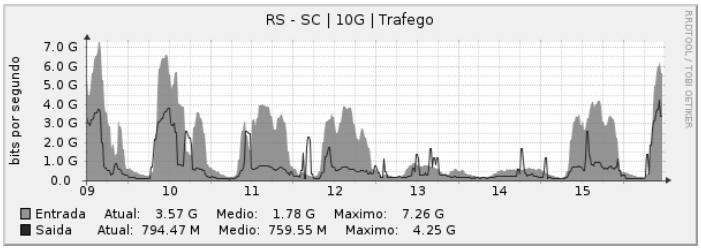


Figure 1. Traffic pattern over a 7-day period. [RNP 2015]

4. Experiments and Discussion

In order to demonstrate our controller actions in differents scenarios, we used two different topologies simulated in Mininet [Lantz et al. 2010]. Besides, we perfomed tests to measure our approach in terms of energy efficiency. In this section, we present and discuss our results and the tools that were necessary to achieve them. First, we present the techniques and tools that were used to create the simulated environment and the controller evaluation. Afterwards, we present our experiments and provide a discussion about the controller behavior.

4.1. Measurement environment and tools

The measurement environment was created and simulated in Mininet, which is a system for rapidly prototyping large networks on the constrained resources of a single desktop computer. It is able to create a realistic virtual network, running real kernel, switch and application code. We used the Mininet Python API to create our topology and carry out interactive performance tests. We proposed a scenario, which is presented along this subsection, to evaluate our approach.

To generate and analyse the network traffic, we use Iperf [Hsu and Kremer 1998] and Wireshark [Chappell and Combs 2010]. Iperf is applied to measure the maximum TCP bandwidth, allowing the tune of various parameters on UDP packets. It was used to generate UDP traffic in our simulated network. Also, we created a performance test that generates traffic between virtual hosts. During the experiment, the virtual hosts sent UDP packets across the network considering the network bandwidth limit. Wireshark was used to monitor the traffic between switches.

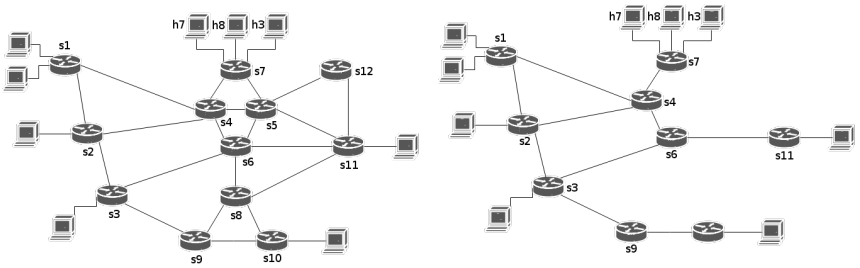
All the experiments were performed on two different machines. One of the machines running Debian 7.8, an Intel Core 2 Duo CPU @ 2.33GHz with 2GB of RAM, was responsible for executing the controller. The other machine, a Raspberry Pi Model B [Halfacree and Upton 2012], performed all the virtualization of the network with Mininet. The Raspberry Pi, also running Debian 7.8, contains a 700MHz ARM processor and 512MB of RAM, was capable of running the test scenario. The workload was generated through an ssh connection using an external terminal with the Iperf tool.

4.2. Scenario

We depict, in Figure 2 (a), a network topology modeled in Mininet, which was used to apply our controller tests and traffic measurements. This topology consists of twelve

switches and nine hosts. In order to perform the traffic measurement, we monitored UDP packets in a single switch, s_7. It is important to notice that all switches connections have a 1Gbps virtual bandwidth capacity.

For evaluation purposes, we considered three servers that receive requests and send responses: h_8, h_3 and h_7. All other hosts are clients. This topology was chosen because it provides redundancy and sufficient complexity to demonstrate our algorithm efficiency. Figure 2 (b) shows the optimized version of the topology computed by our algorithm.



(a) Network topology (b) Optimized topology

Figure 2. Scenario

4.3. Results

Considering that one single switch consumes 50Wh and that one port of the switch consumes 5Wh (10Wh per link), the topology consumes a total of 800Wh (12 switches + 20 links). When the optimization algorithm is applied to this topology, switches s_5, s_8, and s_{12} are removed along with their respective links (Figure 2 (b)). This removal represents a drop of 25% in the number of switches, and the energy consumption decreases from 800Wh to 550Wh (9 switches + 10 links), representing 31% of savings in one hour.

To demonstrate that the policies were being enforced correctly, a 7-day period was simulated in our environment. Every second was interpreted as a real-world hour. This representation allows quick tests over the topology and the controller. The controller, at every simulated hour, reports the number of switches that are present in the network and how much energy it is consuming.

In Figure 3, the number of switches that are being currently used in the network is presented in Axis Y. This plot follows the same pattern as the traffic flow presented in the Figure 1, which corresponds to the RNP link. We noticed that the number of switches corresponds to the traffic flow in the network; when the network is experiencing a low traffic demand, *e.g.*, on Saturday, Sunday, and Monday at dawn, the controller optimizes the graph and uses only nine switches.

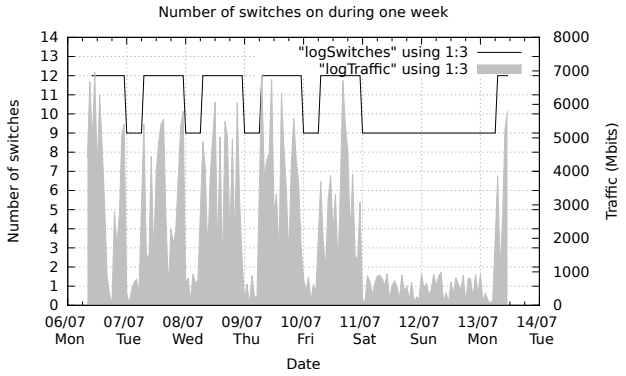


Figure 3. Network traffic analysis and daily active switches

With the policies being hard-coded within the controller, the number of switches being used will follow the same pattern weekly. It is visible that when the policies are enforced, the network will stop routing traffic through three switches. Therefore, these switches will be idle thus consuming less energy.

5. Conclusions and Future Work

The reduction of energy consumption is an important research topic in computer networks. In this context, Green Networking arises as a novel approach to rethinking the network design and deployment. The principal goal of green networking is to minimize resource usage, by turning off inactive routers, whenever possible, and with that, saving energy. To create dynamic and manageable systems, we leverage the SDN paradigm, which improves the capacity to design, deploy, and control energy-efficient networking systems.

In this work, we presented an SDN controller based on OpenFlow 1.0 for the management of packet forwarding and switch state. When network traffic is considered low, we use a customized topology to avoid using idle switches. Our algorithm consists of updating routing tables during idle periods while eliminating redundancy in the network. We reported on a prototype to find switches that can be removed based on the traffic observation pattern.

The evaluation results disclose the influence of switches usage to saving energy. We can eliminate inactive interfaces and redundancy, and so, reduce energy consumption. In the simulated solution, we decreased 31% of power consumption during the usage of the optimized solution and demonstrated the efficiency of our algorithm in terms of energy saving in networks with high redundancy. Our controller thus identifies low and high traffic periods based on policies, and can calculate new routing path, and with that avoid idle switches in low demand period.

As future work, we intend to: i) bring concepts from the Steiner-Tree problem to improve our controller solution, ii) implement a policy database with a more robusten-forcer and monitor (this database should allow the operator to create, remove and update

policies), iii) analyse other metrics (*e.g.*, delay) toevaluate our solution, and iv) compare our work to others Green SDN algorithms in heterogeneous environments.

References

Albers, S. (2010). Energy-Efficient Algorithms. *Communications of the ACM*, 53:86–96.

Bianzino, A. P., Chaudet, C., Rossi, D., and Rougier, J.-L. (2012). A Survey of Green Networking Research. *IEEE Communications Surveys Tutorials*, 14(1).

Bolla, R., Divoli, F., and Cucchietti, F. (2011). Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures. *IEEE Communications Surveys Tutorials*, 13(2):223–244.

Bruschi, R., Lombardo, A., Bolla, R., and Morabito, G. (2014). Green Extension of OpenFlow. *Teletraffic Congress (ITC)*, pages 1–6.

Chappell, L. and Combs, G. (2010). *Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide*. Laura Chappell University, 2 edition.

Feamster, N., Rexford, J., and Zegura, E. (2014). The Road to SDN: An Intellectual History of Programmable Networks. *ACM SIGCOMM Computer Communication*, pages 87–98.

Halfacree, G. and Upton, E. (2012). *Raspberry Pi User Guide*. Wiley Publishing, 1st edition.

Hsu, C.-H. and Kremer, U. (1998). IPERF: A Framework for Automatic Construction of Performance Prediction Models. *Workshop on Profile and Feedback-Directed Compilation*.

Jammal, M., Singh, T., Shami, A., Asal, R., and Li, Y. (2014). Software-Defined Networking: State of the Art and Research Challenges. *Computer Networks*, 72:201–213.

Lantz, B., Heller, B., and McKeown, N. (2010). A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. *ACM SIGCOMM Workshop on Hot Topics in Networks*.

Li, D., Shang, Y., and Chen, C. (2014). Software Defined Green Data Center Network with Exclusive Routing. *IEEE INFOCOM*, pages 1743–1751.

Rede Nacional de Ensino e Pesquisa (2015). Trafego Semanal RS-SC. `http://www.rnp.br/`. Acessed: 2015-06-16.

Team, R. P. (2014). *RYU SDN Framework*. Ryu Project Team.

Thanh, N. H., Cuong, B. D., and Thien, T. D. (2013). ECODANE: A Customizable Hybrid Testbed for Green Data Center Networks. *Advanced Technologies for Communications (ATC)*.