ECODANE - Reducing Energy Consumption in Data Center Networks based on Traffic Engineering

Truong Thu Huong*, Daniel Schlosser†, Pham Ngoc Nam*, Michael Jarschel†, Nguyen Huu Thanh*, Rastin Pries†
*Hanoi University of Science and Technology, School of Electronics and Telecommunications, Hanoi, Vietnam.
Email: {huong.truong,pnnam-fet,thanhnh}@mail.hut.edu.vn
†University of Würzburg, Institute of Computer Science, Würzburg, Germany.
Email: {schlosser,michael.jarschel,pries}@informatik.uni-wuerzburg.de

I. Introduction

Nowadays, global data centers are growing rapidly to satisfy the tremendous traffic demand driven by the exponential development and popularity of the Internet. Consequently, data centers consume a huge amount of energy and emit a lot of greenhouse gases that turns to be a big concern for data center owners and managers as well as for policy-makers. Thus, the focus is on building green data centers. Therefore, many researches addressed this issue including designing smart cooling systems, migrating virtual machines across physical machines, optimizing power consumption of servers, optimizing power consumption of network components, etc. Our ECODANE project focuses on optimizing power consumption of network components by designing an intelligent network control system that dynamically adapts the set of active network components corresponding to the total traffic going through the data center. The optimizer module is accompanied by a load balancing routing module to guarantee the availability of a data center. The proposed system has been estimated by a virtual testbed built on Mininet environment with OpenFlow [1] switches and NOX controller, and will be tested on the hardware testbed using NetFPGA based OpenFlow switches. For the ECODANE project, OpenFlow helps us to evaluate our green networking ideas both on our real data center testbed and in emulation.

II. Data Center Management

For our data center management, we use the Elastic-Tree network [2] which is based on the Fat-Tree topology as shown in Figure 1. The fat-tree architecture [3], [4] was developed to reduce the oversubscription ratio and to remove the single point of failures of the hierarchical architecture. The Elastic-Tree was proposed for dynamically adapting the energy consumption of a data center network, i.e., its network topology adjusts to the traffic requirements. As similar switches are used on all layers of the architecture, the costs for setting up a fat-tree data center can be kept low. The architecture is not achieving complete 1:1 oversubscription in reality, but offers rearrangeable non-blocking paths with full bandwidth. The figure shows a 4-ary fat-tree which is build up of $k = 4$ Performance Optimized Data Centers (PODs), each containing two layers of $k/2$ switches.

The switches in the edge layer are connected to $k/2$ servers and the remaining ports of the edge switches are connected to the aggregation layer, cf. Fig. 1. The core layer consists of $(k/2)^2$ k-port core switches where each of them is connected to each of the k PODs [3]. A Fat-Tree data center architecture built with k-port switches support $k^3/4$ servers. Thus, when using 48-port switches, up to 27,648 server can be supported.

In our emulation to build the Elastic-Tree system, we developed some logical modules which are described in the following and are shown in Fig. 2.

(1) **Data center network**: The network is emulated with the Fat-Tree topology. In our implementation, the number of ports of each switch are k=4 or 6, corresponding to the number of servers (16 or 54). Mininet [5] is used as the emulation tool which is able to emulate a real network with switches, servers, links. Mininet enables to turn links, switches, and servers on and off.

(2) **Optimizer**: The Optimizer's role is to find the minimum power network subset (minimum numbers of switches and links) that satisfies current traffic conditions, while still offering good Quality of Service (QoS). The module is developed using the NOX controller [6], being able to provide network traffic statistics via OpenFlow messages. Its needed inputs are the network topology, traffic matrix, a power model for each switch, and the desired fault tolerance properties. Traffic statistics are gathered with the port-counter field of the OpenFlow switch. Fault tolerance is handled with defining spare switches or spare capacity for a link. In our testcase,

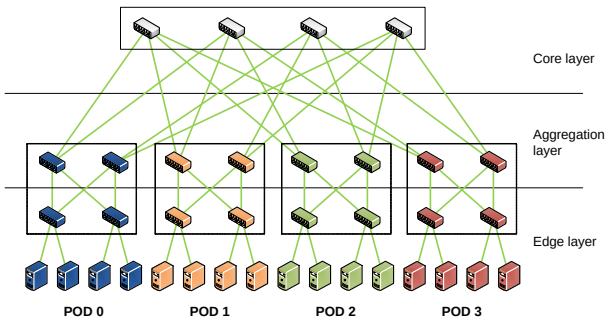

Fig. 1. Fat-tree data center architecture.

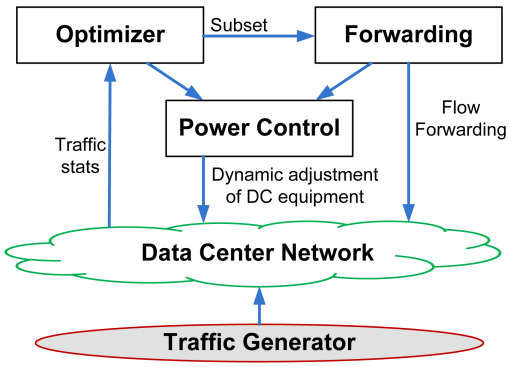

Fig. 2. The ECODANE modules.

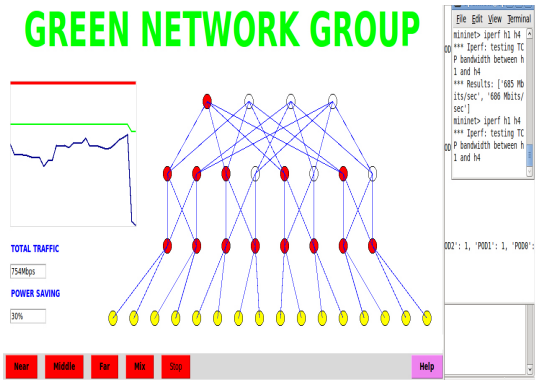

Fig. 3. Demonstration of Energy Reduction by turning off uneeded switches in a data center network.

a bandwidth threshold of traffic transmitted in a link is set 70% in order to spare 30% capacity of the link. Topology-aware heuristics [2] are developed for the Optimizer module to compute a set of active components (switches, ports, and links). Every 500ms, the Optimizer outputs a subset to the Power Control and Forwarding modules.

Based on our proposed power model of each commercial switch, port and link, the energy reduction in the whole network is estimated as numbers of switches, ports, and links can be turn off or be put into sleep mode. For testing the proposed optimizer on the hardware testbed, we run a number of experiments and proposed a power model for NetFPGA based switches.

(3) **Power Control**: The module toggles the power states of ports, linecards, and entire switches through OpenFlow messages and Python APIs of Mininet to ”tell” switches ”off or on” or change to an appropriate power saving mode. In order to support the power controller at switch level, we propose a hardware power management module for the NetFPGA platform. The module is also implemented in the NOX controller.

(4) **Forwarding**: The module is in charge of optimizing the routes in the data center. It is implemented in the NOX controller as a NOX module. In our implementation, a hierarchical load-Balancing routing algorithm is selected to guarantee the QoS requirements. The Forwarding module is implemented separately from the Optimizer. It fetches the Optimizer's outputs for its own routing calculation. However, if the Optimizer is, by accident, out of order, all network components are toggled On to the operating mode.

(5) **Traffic Generator**: The module is developed in D-ITG [7] to generate network traffic from servers within a data center network. The traffic pattern is gathered from Bensons et al. [8] with a lognormal distributed flow interarrival time.

The ECODANE tool using the Fat-Tree topology (k=4) is shown in Fig. 3. First results gathered with ECODANE show energy savings between 10% and 35%, depending whether most of the traffic is transmitted locally within a rack, within a POD, or globally transmitted within the whole data center. Furthermore, experimental results on NetFPGA OpenFlow switches have shown that when the switch is working at full speed, the power consumption increases only slightly as the network traffic increases. However, by reducing the working clock frequency of the switch, the power consumption of the switch can be reduced significantly.

III. Conclusion

In this paper, we demonstrated the ECODANE project in which we successfully studied and implemented the Elastic-Tree idea of Heller et al. [2] in an emulation framework. Moreover, we implemented a Load Balancing Routing Heuristic to see its impact on the network performance. Our first emulation results have shown that between 10% and 35% energy can be saved by dynamically adjusting the number of active switches, ports, and linecards. In future work, we will also adjust the link rates of the switches as well as migrate virtual machines to be able to switch off servers.

Acknowledgments

The authors would gratefully thank the Vietnamese Ministry of Science and Technology as well as the International Bureau of the BMBF for their support on this paper.

References

[1] “Openflow switch specification, version 1.1.0,” February 2011.
[2] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “Elastic tree: Saving energy in data center networks,” in *7th USENIX Symposium on Networked System Design and Implementation (NSDI)*, San Jose, CA, USA, April 2010, pp. 249–264.
[3] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, Seattle, WA, USA, August 2008, pp. 63–74.
[4] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, “Portland: a scalable fault-tolerant layer 2 data center network fabric,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.
[5] Stanford. (2011) Mininet. [Online]. Available: http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet
[6] (2011, 5). [Online]. Available: http://www.noxrepo.org/
[7] [Online]. Available: http://www.grid.unina.it/software/ITG/
[8] T. Benson, A. Akella, and D. Maltz, “Network traffic characteristics of data centers in the wild,” in *Internet Measurement Conference (IMC)*, Melbourne, Australia, November 2010.