Minimizing Energy Consumption of Fat-Tree Data Center Networks*

Qing Yi
Department of Computer Science
Portland State University
Portland, OR 97207
yiq@cs.pdx.edu

Suresh Singh
Department of Computer Science
Portland State University
Portland, OR 97207
singh@cs.pdx.edu

ABSTRACT

Many data centers are built using a fat-tree network topology because of its high bisection bandwidth. There is a need to develop analytical models for the energy behavior of fat-tree networks and examine strategies to reduce energy consumption. The most effective strategy is to power off entire switches, if possible. In this paper, we derive formulas for the minimum number of *active switches* needed in a fat-tree data center network for arbitrary types of loading. We also derive expressions for the expected traffic loss when these networks are overloaded with external (Internet) traffic. Results of detailed simulations conducted using well-known traffic models for data center networks [4] closely match our derived formulas. We show that a fat-tree network incurs significant energy cost (approximately 45% of the energy cost of the data center) even when very lightly loaded. In order to further reduce energy consumption, we need to consolidate traffic into fewer switches. We derive expressions for energy cost versus load assuming traffic consolidation and show linear scaling. Finally, we observe that traffic patterns have a significant impact on energy consumption and this fact is evident in the analytical formulas.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network Communications

General Terms

Performance, Design

Keywords

Data center, simulation, fat-tree, analysis

1. INTRODUCTION

Networks in data centers tend to consume about 10%-20% of energy in normal usage [10], but it accounts for up to 50% energy [7] during low loads since at those times servers can be put into low power states. Unlike servers, which have many lower power states, powering off switches is not a good option unless the time between powering on and off is at least of the order of tens of minutes (due to the need for link and network initialization). Therefore, it is of interest to develop a better understanding of network energy consumption under varying types of loads with the eventual goal of designing more energy efficient data center networks.

This paper considers the fat-tree network which has been a popular choice for commercial data centers due to its full bisection bandwidth (which minimizes latency and boosts throughput). Unfortunately, the energy consumption of this or any other network is very dependent on the type of traffic, the traffic load and the selected routing algorithm. For instance, if most of the traffic is between servers located in the same pod (see Figure 1), the core switches are never used even at high loads, resulting in significant energy savings. On the other hand, if most of the traffic is between servers in different pods, then savings are small even at light loads since more switches in the network will need to be utilized for routing. Routing also plays an important part in the potential for energy savings. Thus, routing algorithms that seek to minimize only latency will distribute flows over unused paths when possible ensuring that a majority of switches are kept busy (albeit at very low loads). Alternatively, if paths can be consolidated into a few, there is potential to save energy at the idle switches.

The idea of pushing traffic to the "left" in a fat-tree was explored in ElasticTree [10]. In the present paper, we derive an analytical formula for energy consumption, and then explore additional savings made possible by use of a hardware device called a *merge network* [12], which *further consolidates the traffic* to fewer switches (See discussion in Section 3). The *merge network* pushes all the traffic to the *leftmost interface* of a switch so that even more switches can be put into low power modes.

1.1 Paper Summary

In this paper, we provide a systematic analysis of the energy efficiency of a fat-tree network using modeling and detailed simulations. The key question we ask is *how does energy usage scale with total load as well as with different types of loading.* To answer this question, we build a detailed analytical model that gives the *lower bound* on the *fraction of active switches* required for a given load and type of load. We show that fat-trees have a minimal cost of about 40-50% (i.e. about half the switches need to remain active at all times) but beyond that, the lower bound scales almost linearly with the total offered load. Next we develop a model for fat-tree networks with merge networks and show a 50% reduction in energy consumption relative to the case without

*This work was funded by the NSF under award No. 1217996.

merge networks at low loads. We conduct a detailed simulation of a fat-tree network where we use different types of load and different amounts of total load. We compute routing tables empirically every second for the next second and compute the fraction of needed active switches. The simulation demonstrates that the models we develop are accurate in predicting switch activity and by modifying the routing algorithms, we can potentially save significant amounts of energy in real networks.

The remainder of the paper is organized as follows. Section 2 presents a detailed derivation of our analytical model for the lower bound on energy consumption. The next section describes the experiment for a fat-tree enhanced with merge networks. The subsequent section presents results from our simulations. We describe related work in section 5 and conclude in section 6.

2. MODELING ENERGY USAGE

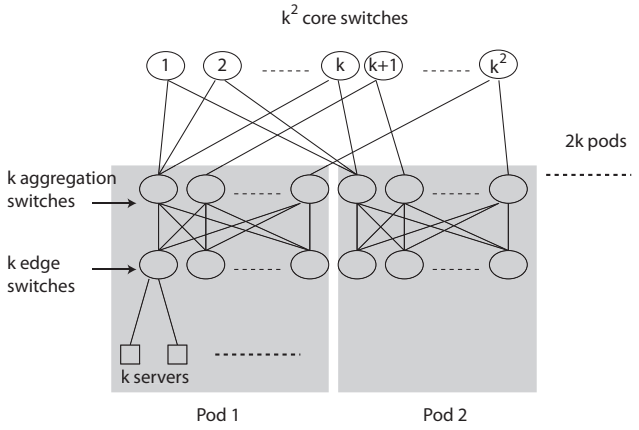

Figure 1: Fat tree network model.

As it is shown in Figure 1, a fat-tree is made up of $2k$ "pods" which are connected to k^2 core switches. Within each pod, there are k aggregation switches and k edge switches. Each edge switch is in turn connected to k servers. Therefore, each pod has k^2 servers and the data center network has a total of $2k^3$ servers. Each core switch has one link connected to each of the $2k$ pods. The ith port of a core switch is connected to an aggregation switch in pod i. The left-most k core switches are connected to the leftmost aggregation switch of each of the $2k$ pods. The next set of k core switches are connected to the second aggregation switch of each of the pods, and so on.

Our goal here is to derive analytical expressions for minimal energy consumption of fat-trees for different type of loading. The metric we use for energy consumption is *fraction of active switches*. We use three parameters to model different types of loading. A packet from a server goes to another server connected to the same edge switch with probability p_1, it goes to a server in the same pod but another edge switch with probability p_2, and with probability $p_3 = 1 - p_1 - p_2$ it goes to a server in a different pod. Thus p_1 of the traffic is never seen by either the core or the aggregation switches, while p_2 of the traffic is not seen by the core switches. Obviously, by varying p_1 and p_2, we can model very different types of traffic. Finally, we model external traffic (i.e. traffic going to/from the Internet) as the fraction q.

Let λ denote the average *internal* load offered by each server expressed as a fraction of link speed (which we normalize to 1). This load refers to packets that will stay within the data center. Thus, the *total offered load per server* is $\lambda + q$. For simplicity, we assume that load $\lambda + q$ is the same for all the servers in the data center. Thus, the total load in the data center is $2k^3(\lambda + q)$. We have the following equalities for total traffic at the level of edge switches, pod aggregation switches and core switches:

$$\text{Traffic per edge switch} = (\lambda + q)k$$
$$\text{Traffic for all aggregation switches in a pod} = ((1 - p_1)\lambda + q)k^2$$
$$\text{Traffic for all core switches} = ((1 - p_1 - p_2)\lambda + q)k^2 \times 2k$$

Note that traffic flow is symmetric and the numbers above correspond to both, traffic into and out of a switch or switches.

Let us assume that traffic coming into the data center is $2k^3 q_{\text{in}}$, which is equally distributed among all the servers, and traffic going out is $2k^3 q_{\text{out}}$ and is also equally generated by each server. It is easy to see that $\lambda + q_{\text{in}} \leq 1$ and $\lambda + q_{\text{out}} \leq 1$ since the normalized capacity of the link connecting each server to the edge switch is 1. Before proceeding with the derivations below, note that a switch interface is typically bi-directional. As a result, even if there is no traffic in one direction, the entire interface is functioning and running link layer protocols to maintain connectivity. Therefore, instead of considering q_{in} and q_{out} separately, we only need to consider the maximum of the two. Let

$$q = \max\{q_{\text{in}},\ q_{\text{out}}\}$$

And thus the total external traffic is thus $2k^3 q$.

2.1 Number of Active Switches

In large data centers, a small subset of core switches have external links providing connectivity to the Internet, and these switches are equipped with much higher rate links. Let us assume that of the k^2 core switches, C switches ($k \geq C \geq 1$) have external connectivity. *We assume that each of these C core switches is equipped with additional interfaces with a total normalized capacity of Q* and is connected to a border switch or router. Assume further that these C switches are connected to the aggregation switches using links of capacity $l \geq 1$. All remaining links in the network have a capacity of 1. Clearly, $Q \leq 2kl$ and $l \leq k$. The latter inequality makes sense since an aggregation switch is connected to k edge switches with capacity-one links and thus there is little point in connecting it to a core switch by a link of capacity greater than k. Without loss of generality, assume that the C core switches are $1, 1+k, 1+2k, \cdots, 1+(C-1)k$. Thus aggregation switches $1, \cdots, C$ in each pod are connected with a link of capacity l to these special core switches.

To compute the number of active core switches, we have two assumptions. First, each pod is assumed to be identical to other pods and generates an equal amount of external traffic. Second, in the computation of the number of core switches needed to support the external traffic, we assume that *all the external traffic is put into as few core switches as possible rather than spreading it out among all the core switches*. This design is more energy efficient since we can minimize the number of active switches.

As the total external traffic load is $2qk^3$ and the traffic is uniformly distributed among all the servers, the total number of externally-connected core switches that need to be active is thus given by

$$m = \frac{2qk^3}{Q}$$

Since m may be greater than C or have a fractional part, we obtain

$$m_{\text{core}}^{\text{ext}} = \min\{C, \ \lceil m \rceil\}$$

The $m_{\text{core}}^{\text{ext}}$ active core switches may not use all of their link capacity and thus they can be used for routing internal traffic as well. Each of the $2k$ interfaces of the active externally-connected core switches (facing towards the servers) has a capacity of l. Each of these switches has capacity of $2kl$ to handle traffic coming/going from/to the connected pods.

If $C < \lceil m \rceil$, then $m_{\text{core}}^{\text{ext}} = C$, and the external traffic exceeds the total external capacity. All $m_{\text{core}}^{\text{ext}}$ switches are using their full external capacity Q, leaving $f = (2kl - Q)C$ free capacity. Also, traffic losses occur since the external traffic exceeds the capacity. We will discuss the traffic loss later in Section 2.2.

If $\lceil m \rceil < C$, $m_{\text{core}}^{\text{ext}} = \lceil m \rceil$. $\lfloor m \rfloor$ of these core switches is using their full external capacity Q to handle $Q\lfloor m \rfloor$ external traffic, leaving $(2kl - Q)\lfloor m \rfloor$ free capacity for internal traffic. One additional externally-connected core switch will be using less capacity for external traffic of $(2qk^3 - Q\lfloor m \rfloor)$, leaving $(2kl - (2qk^3 - \lfloor m \rfloor Q))$ free capacity. The total *free* capacity for the $m_{\text{core}}^{\text{ext}}$ active core switches is thus $f = (2kl - Q)\lfloor m \rfloor + (2kl - (2qk^3 - \lfloor m \rfloor Q))$. The total internal traffic that needs to be forwarded by core switches is $2(1 - p_1 - p_2)\lambda k^3$. Therefore, the number of additional core switches we need is,

$$m_{\text{core}}^{\text{addl}} = \begin{cases} 0 & 2(1-p_1-p_2)\lambda k^3 \le f \\ \left\lceil \frac{2(1-p_1-p_2)\lambda k^3 - f}{2k} \right\rceil & \text{otherwise} \end{cases} \quad (1)$$

We divide the second term above by $2k$ because $2k$ is the degree of the additional core switches used. Since it is possible that the above number exceeds the available number of free core switches, we can write the final answer as

$$m_{\text{core}}^{\text{total}} = \min\left\{k^2, \ \ m_{\text{core}}^{\text{ext}} + m_{\text{core}}^{\text{addl}}\right\} \quad (2)$$

We then compute the number of aggregation switches required per pod. Within each pod, the total external traffic is qk^2 and this is forwarded to/from the externally-connected core switches using $m_{\text{core}}^{\text{ext}}$ aggregation switches. This is the case because of the way we are performing the minimization forces traffic from/to each pod to be identically routed. The total internal traffic that needs to be handled by the aggregation switches in a pod is $(1 - p_1)\lambda k^2$.

Consider the aggregation switches in a pod that are connected to the active externally-connected core switches. Say the high-capacity link (of capacity l) carries external traffic a. This traffic is evenly distributed over the k capacity-one links connecting one aggregation switch to edge switches. In other words, each of the edge switches can send up to $(1 - a/k)$ internal traffic to the aggregation switch. In all, the k connected edge switches can send $(k - a)$ total internal traffic to this aggregation switch. Consider the k links from this switch connected to the core switches. One of the links is capacity l while the other $(k - 1)$ links are capacity 1 each. Thus, the total available capacity of these links is $(l-a)+(k-1) = (k-a)+(l-1)$. Since $(k-a) < (k-a)+(l-1)$ the total internal capacity that can be handled by this aggregation switch is $(k - a)$.

In a pod, if $m_{\text{core}}^{\text{ext}} = \lceil m \rceil$ then there are $\lfloor m \rfloor$ aggregation switches where $a = Q/2k$ (corresponding to the $\lfloor m \rfloor$ core switches that run their external links at full capacity), and at most one switch $(\lceil m \rceil - \lfloor m \rfloor)$ where $a = (2qk^3 - Q\lfloor m \rfloor)/2k$. In total, these $\lceil m \rceil$ aggregation switches handle internal traffic equal to

$$\begin{aligned} t_{\text{aggr}} &= \lfloor m \rfloor (k - a) + (\lceil m \rceil - \lfloor m \rfloor)(k - a) \\ &= \lfloor m \rfloor (k - \tfrac{Q}{2k}) + \\ &\quad (\lceil m \rceil - \lfloor m \rfloor)(k - \tfrac{(2qk^3 - Q\lfloor m \rfloor)}{2k}) \end{aligned} \quad (3)$$

If $m_{\text{core}}^{\text{ext}} = C$, then all C aggregation switches have $a = Q/2k$. Thus

$$t_{\text{aggr}} = (k - Q/2k)C \quad (4)$$

In all these cases, traffic $(1 - p_1)\lambda k^2 - t_{\text{aggr}}$ is left to be handled by other aggregation switches. Therefore, the total number of aggregation switches needed in the entire network is written as

$$m_{\text{aggr}}^{\text{total}} = 2k \min\left\{k, \ \ m_{\text{core}}^{\text{ext}} + \left\lceil \frac{(1 - p_1)\lambda k^2 - t_{\text{aggr}}}{k} \right\rceil \right\}$$

Adding all these values together, we have

$$\text{Active Switches} = 2k^2 + m_{\text{core}}^{\text{total}} + m_{\text{aggr}}^{\text{total}} \quad (5)$$

In the discussion above, the $2k^2$ edge switches are always fully powered on because they are connected to servers at all times. Even if the servers have very light traffic going to the edge switch, the switch will still be fully powered on, albeit very lightly loaded. *Indeed, as Figure 2 shows, even at very low loads, more than* 60% *of switches are still active.*

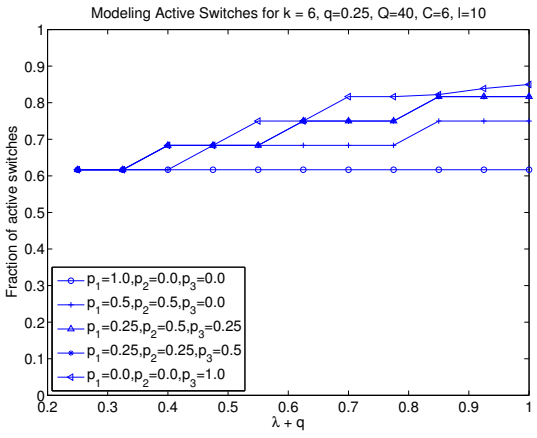

Figure 2: Modeling active switches for fat-tree.

2.2 Modeling Traffic Loss

Note that there is no traffic loss for the internal traffic λ since the fat-tree has full bisection bandwidth and $\lambda \le 1$. For the external traffic, on the other hand, loss happens if the external capacity Q is unable to handle the external

load. When $C < \lceil m \rceil$, the total external traffic that the network can handle is QC. The total external traffic load is $2qk^3 = \lceil m \rceil Q$. Since $QC < 2qk^3$, the total external traffic exceeds the external traffic capacity. This yields a loss of external traffic of,

$$\text{Loss}_{\text{ext}} = \max\{0, \ \ 2qk^3 - QC\}$$

Theorem 1: The total traffic loss is: $\text{Loss} = \text{Loss}_{\text{ext}}$.

Proof sketch: (We have not included the formal proof here for space reasons) The intuition behind this result is relatively simple. Consider traffic going up to the core first. Since $\lambda < 1$, there will be no losses seen by the internal traffic either in the pods, or in the core level switches. Traffic heading out to the Internet is limited by Q, and hence we will see packet drops if $2kq > Q$. Consider traffic coming into the network from the Internet as well as inter-pod and intra-pod traffic. At the core layer, this traffic will be $2(1 - p_1 - p_2)\lambda k^3 + 2k^3 q'$. The first term is the inter-pod traffic and the second term is the amount of external traffic that *was not lost* due to the limitation on Q. Clearly, $q' \leq q$ and hence, the total traffic flowing into the servers is below the link capacity (=1), and there will again be no losses. Therefore, we can write the total loss as $\text{Loss} = \text{Loss}_{\text{ext}}$.

3. MERGING TRAFFIC TO REDUCE ENERGY CONSUMPTION

Consider the case of an edge switch connected to k servers. Assuming each server offers a load of λ, then the total traffic to this switch from the servers is $k\lambda$. If $k = 4$, for $\lambda \leq 0.25$, one switch interface will suffice to handle the traffic from all four servers. In other words, if there is a way to *merge* the traffic from the four servers together, we can potentially power off three of the four switch interfaces connected to the servers.

3.1 Merge Network

In previous papers [12], we presented the idea of merging traffic via a merge network before feeding the traffic to a switch. A brief summary about the merge network is:

1. The merge network is a fully analog device with no transceivers, and as a result its power consumption is below one watt.
2. The merge network, by design, does not cause any packet loss or increase in delay.
3. Traffic merging is accomplished internally by sensing packets on links, and automatically redirecting them to the leftmost output that is free.
4. The merge network downlink 1-1 routing association is accomplished through special switch software as described in [12].

Consider a fat-tree pod with k^2 servers connected to k k-port switches. Normally, all the switch interfaces remain active even at very low link loads because they need to be able to forward traffic upstream and downstream. A $k^2 \times k^2$ merge network has k^2 downlink connections to the k^2 servers and k^2 uplink connections to each k interfaces of k edge switches, shown in Figure 3. The merge network consolidates the traffic from the k^2 servers to the leftmost of the k edge switches, and ensures switch with no active interface can be put to low power modes

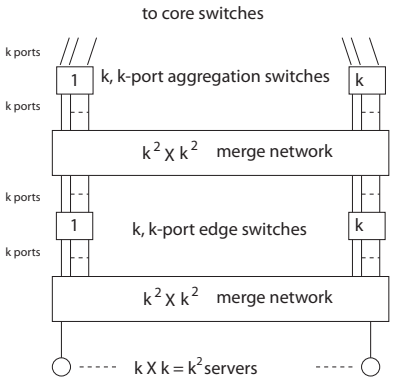

Figure 3: Merge network applied to pod in a fat-tree.

On the uplink from the servers to the merge network, all traffic coming into the merge network is output on the *leftmost* $q \leq k^2$ links connected to the q leftmost interfaces of the switch, where $q = \lceil k^2\lambda/2 \rceil$ (assuming a normalized unit capacity for links). This is accomplished internally by sensing packets on links and automatically redirecting them to the leftmost output from the merge network that is free. On the downlink to the servers, traffic from the switch to the k^2 servers is sent out along the leftmost $r \leq k^2$ switch interfaces to the merge network. The packets are then sent out along the m links attached to the servers from the output of the merge network. Because of the fact that we are breaking the 1-1 association of a switch interface to a server interface, several layer 2 protocols will break. In the previous paper [15], we have addressed this issue as well and show how the merge network and some additional switch software can overcome this limitation.

We apply merge networks to the fat-tree topology at two locations: between the servers and the edge switches, and between the edge switches and the aggregation switches. Figure 3 shows the edge and aggregation layer of a single pod of a fat-tree network after applying the merge network. As shown, we use one $k^2 \times k^2$ merge network to connect the servers to the edge switches, and another merge network to connect the edge switches to the aggregation switches.

3.2 Number of Active Switches with Traffic Merging

The consequence of applying merge networks is two-fold. First, traffic from servers is now sent to a merge network, and thus consolidated to the leftmost edge switches. The idle edge switches can be put into low power mode to save energy. The total active edge switches in one pod is therefore written as

$$m_{\text{edge}}^{\text{total}} = 2k \times \lceil \frac{(\lambda + q)k^2}{k} \rceil = 2k\lceil (\lambda + q)k \rceil$$

Obviously the number of active edge switches changes with traffic load $(\lambda + q)$.

Second, traffic $= \lambda p_2$ going to other subnets within the same pod is transferred directly with no necessity to go

through the aggregation level switches. Therefore, the parameters p_1 and p_2 become $p_1' = p_1 + p_2$ and $p_2' = 0$. Thus, the internal traffic to be handled by other aggregation switches will be $(1 - p_1 - p_2)\lambda k^2 - t_{\text{aggr}}$, where t_{aggr} is calculated as in Equation (3) and (4) and the number of active aggregation switches required in each pod is

$$m_{\text{aggr}}^{\text{total}} = 2k \times min\left\{k,\ \ m_{\text{core}}^{\text{ext}} + \left\lceil \frac{(1-p_1-p_2)\lambda k^2 - t_{\text{aggr}}}{k} \right\rceil \right\}$$

The total number of core switches is the same as the $m_{\text{core}}^{\text{total}}$ calculated in Equation (2). Therefore, the total number of active switches after applying merge networks is

$$\text{Active Switches} \quad = 2k\lceil(\lambda + q)k\rceil + m_{\text{core}}^{\text{total}} + m_{\text{aggr}}^{\text{total}}$$

Figure 4 shows the total number of active switches for the same topology parameters and traffic loads shown in Figure 2. It demonstrates saving around 30% of active switches at lighter loads. Even for the all near-traffic case ($p_1 = 1.0$), the number of active switches changes with the load and shows energy proportionality.

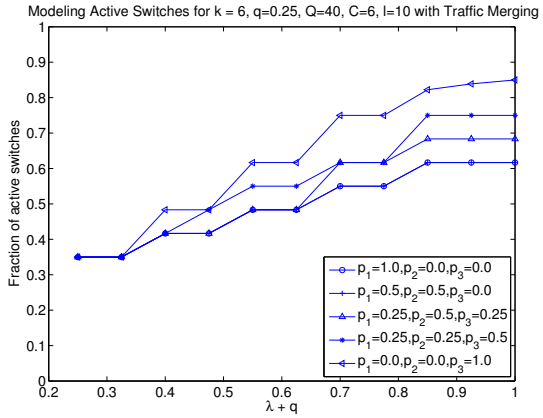

Figure 4: Modeling active switches with traffic merging (compare with Figure 2).

4. SIMULATIONS

We build a simulator for a fat-tree network with $k = 6$ and 1 Gbps link capacity. Each server generates traffic based on a two-state On/Off process in which the length of the On and Off periods follows a lognormal distribution. In the On state, packet inter-arrival times are also from a lognormal process [5]. The parameters selected for the lognormal processes are based on different types of traffic patterns as well as different loading patterns. For each packet, the destination is selected uniformly randomly from the set of all nodes based on probabilities p_1 and p_2. In the simulator, we read these trace files which are generated externally and forward packets based on routing tables computed *every second of simulated time*. The p_1 and p_2 of the traffic models we used are as follows:

1. $p_1 = 0.75, p_2 = 0.125$;
2. $p_1 = 0, p_2 = 0.75$;
3. $p_1 = 0, p_2 = 0$;

We assume that external traffic q is 10% in all three cases. The total traffic load is from 10-70% of the full bandwidth.

The routing algorithm is a modified version of Dijkstra's algorithm where we force flows to use routes that are already in use, thus packing flows together. In the algorithm we assign weights to edges as well as nodes. Edge weights are constant of 2, but node weights can be 0 or 1. If a node has been used for forwarding a flow, its weight changes from 1 to 0. Thus, flows are encouraged to reuse the same subset of nodes (or switches). We eliminate link with zero available capacity from further consideration in that round of routing computation.

We use $C = 1$ and designate the leftmost core switch as the externally connected core switch. For 10% external traffic, the link capacity l of the externally connected core switch has to be greater than 4. Therefore, we use $l = 4$ and let $Q = 2kl = 48$ to avoid traffic loss.

In Figure 5, we plot the fraction of active switches versus total load using simulation without merge networks and with merge networks. Figure 6 shows the same metric from the analytical models described in Section 2 and Section 3. It is easy to see that, our model is a very close match to the simulations. The minor difference between the simulations and model is due to the fact that we estimated the values of p_1 and p_2 from the simulations and then used them in the analysis. The estimated values for these probabilities are listed in the legend of Figure 6. *The implication of this is that the lower bound of energy efficiency can be achieved in practice by utilizing the simple routing algorithm described above.*

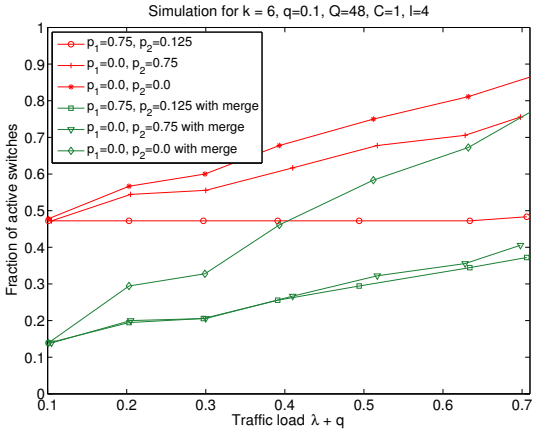

Figure 5: Simulation results of active switches for near and far traffic.

When we examine Figure 5, we observe that the type of loading has a significant impact on energy consumption. When it comes to allocating tasks to servers, the task manager should be mindful of the type of traffic that will be generated since we can obtain significant energy savings by careful scheduling.

5. RELATED WORK

Servers in a data center are interconnected using specific network topologies. A data center network is designed to have high scalability, low latency and high throughput. Energy consumption was not the primary consideration when designing data center networks in the past. But this issue has caught more attention recently.

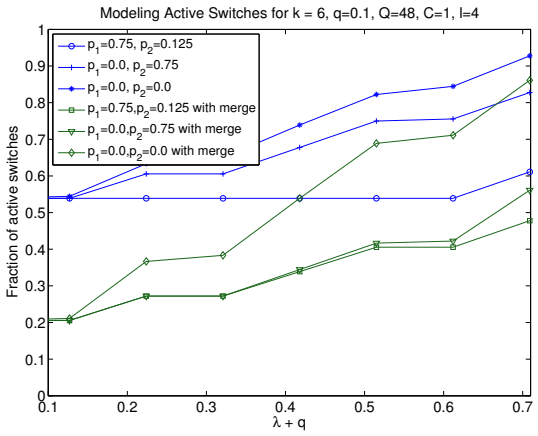

Figure 6: Modeling active switches for near and far traffic.

Several new data center topologies have been studied including fat-tree [4], Clos [6], DCell [8], BCube [9], Jellyfish [13] and flattened butterfly [11], to name a few. Generally, the common goal shared of the different designs is to maximize cross-section bandwidth at minimal link and switch cost. Related approaches consider building server-centric topologies with simple switches that push networking to servers [2].

In the past few years, there has been new work on reducing energy consumption of data center networks. For example, energy-proportional links [1] allow a fine-grained tuning of hardware energy consumption without changing network topology and routing. Besides, some other researchers propose dynamically right-sizing data center networks. For example, Heller *et al.* designed an ElasticTree[10] to compute a minimal subset of network elements for given traffic and power off unnecessary switches and links dynamically. *CARPO* [14] consolidates traffic flows by putting negatively-correlated flows onto one path and thus uses a smaller set of switches and links. More recently, Adnan and Gupta proposed an online algorithm to select most-overlapping paths to consolidate paths and right-size the networks [3]. Their method outperforms the *ElasticTree* approach when there is a large number of flows.

Our work develops analytical models for energy consumption and thus enables us to study fat-tree DCNs theoretically. A practical application of our work would be jointly optimizing task scheduling and flow assignment such that p_1, p_2 can be maximized for given job loads. We are currently studying this problem.

6. CONCLUSIONS

This paper analyzes the problem of energy consumption in fat-tree networks. We derive expressions for the fraction of active switches for arbitrary traffic loads and traffic losses. We show that merge networks can reduce the energy consumption by approximately 30% at light loads, and energy consumption can scale linearly by appropriately consolidating traffic flows.

An important conclusion of this paper is that the type of traffic has a big impact on the potential energy savings. This is clearly demonstrated in the simulations as well as in the analytical study. The key idea is to keep traffic local as much as possible. Another conclusion is that the topology enhancements for providing external connectivity can impact the overall energy consumption. Thus, it is better to have one or two core switches outfitted with very high bandwidth links to the Internet rather than having several core switches providing this connectivity via lower capacity links.

7. REFERENCES

[1] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy Proportional Datacenter Networks. In *ISCA*, 2010.

[2] Hussam Abu-Libdeh, Paolo Costa, Antonu Rowstron, Greg O'Shea, and Austin Donnelly. Symbiotic Routing in Future Data Centers. In *SIGCOMM*, pages 51–62, 2010.

[3] Muhhamad Abdullah Adnan and Rajesh Gupta. Path Consolidation for Dynamic Right-sizing of Data Center Networks. In *Proceedings IEEE Sixth International Conference on Cloud Computing*, 2013.

[4] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A Scalable, Commodity Data Center Network Architecture. In *SIGCOMM*, pages 63–74, 2008.

[5] Theophilus Benson, Aditya Akella, and David A. Maltz. Network Traffic Characteristics of Data Centers in the Wild. In *IMC*, 2010.

[6] Charles Clos. A Study of Non-Blocking Switching Networks. *The Bell System Technical Journal*, 32(2):406–424, March 1953.

[7] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The Cost of a Cloud: Research Problems in Data Center Networks. In *SIGCOMM CCR*, pages 68–73, 2009.

[8] Chaunxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. DCell: A Scalable and Fault-tolerant Network Structure for Data Centers. In *SIGCOMM*, pages 75–86, 2008.

[9] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *SIGCOMM*, pages 63–74, 2009.

[10] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. ElasticTree: Saving Energy in Data Center Networks. In *NSDI*, 2010.

[11] John Kim, William J. Dally, and Dennis Abts. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In *ISCA*, pages 126–137, 2007.

[12] Suresh Singh and Candy Yiu. Putting the Cart Before the Horse: Merging Traffic for Energy Conservation. In *IEEE Communications Magazine*. June 2011.

[13] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking Data Centers Randomly. In *NSDI*, 2012.

[14] Xiaodong Wang, Yanjun Yan, Xiaorui Wang, Kefa Lu, and Qing Cao. CARPO: Correlation-aware Power Optimization in Data Center Networks. In *INFOCOM*, pages 1125–1133, 2012.

[15] Candy Yiu and Suresh Singh. Merging Traffic to Save Energy in the Enterprise. In *E-Energy*, 2011.