Traffic Consolidation for Green Optical DCNs

Qing Yi
Department of Computer Science
Portland State University
Portland, OR 97207
yiq.qing@gmail.com

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

***Abstract*—Present-day datacenter networks (DCNs) are designed to provide full-bisection bandwidth in order to support high network throughput and server agility. However, many studies have shown average utilization of the DCN is very low even at high server loads. As a result, there is a significant waste of energy resources in modern datacenters. In this paper we describe the construction and evaluation of a hardware device that *consolidates* traffic going between servers and the top-of-rack (ToR) switch. The consolidator *operates entirely in the optical domain* and has very low power consumption. We evaluate the performance of the consolidator in a small network. Subsequently, using simulations, we show that for larger systems, the consolidator allows energy savings of almost 70% at light loads and scales with increasing loads.**

***Index Terms*—Optical network, power-aware, data center**

I. Introduction

Datacenters are the predominant mechanism by which virtually all internet applications are deployed today. Due the the rapid growth of the number and variety of these applications, their corresponding data, and user base, the number and size of data centers maintained by cloud providers has exploded. Within these data centers, to support the increasing volume of traffic, most DCNs (Data Center Networks) are now optical networks. Indeed, it is not unusual to see 10/40/100 GigE deployments in the larger datacenters [6].

Most DCNs utilize high-bisection bandwidth topologies like the fat-tree in order to support uniform high-capacity traffic between end-systems [12], [16] under the assumption that this is the predominant traffic pattern. However, various studies have shown that typically only a fraction of ToR (Top of Rack) switches run at capacity even at high server loads [14], [20] while other ToRs run at a fraction of their capacity. Furthermore, analysis of high-performance computing applications has shown that most interprocessor communication is fixed (i.e., the communicating sets are fixed) and only change slowly [10] implying long periods of stable traffic patterns. Finally, [4] studied ten production datacenter networks and shows that average link utilization is below 10% for 95% of the time. They also show that job scheduling ensures that on average 75% of MapReduce traffic remains within the same rack.

Network hardware, including switch CPUs, NICs, interconnection fabrics, memory, etc. consumes energy continuously when powered on, even if the loading is small or zero. This is done because traffic is unpredictable and therefore, unlike servers which can be put in standby, network equipment is always on. However, the studies we describe above show traffic in DCNs can be very light over large parts of the network even when datacenter loading is high. This naturally results in energy inefficiency of networking equipment.

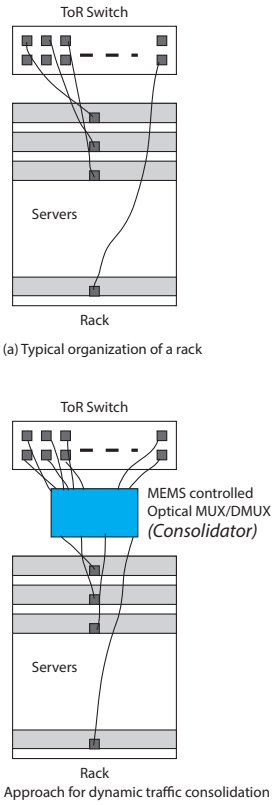

Fig. 1. Our approach for a Green DCN.

We propose a novel method for saving energy in DCNs by consolidating traffic from servers to ToR switches in a way that ensures few of the switch NICs are active, thus enabling large parts of the switch to be put in low-power state. Our method for traffic consolidation ensures that traffic latency, delay, and loss behavior is unchanged. The design relies on switching traffic at line speed from multiple servers to a subset of ports on the ToR using MEMS optical Multiplexer/Demultiplexers (MUX/DMUX) controlled by appropriate logic (we call this

This work was supported by NSF Award No. 1217996

device the *consolidator*). Figure 1(a) illustrates a typical topology where all the servers in a rack are connected to a ToR switch. In Figure 1(b) we show that servers and ToR ports are instead connected va a MEMS controlled device. This device consolidates traffic going to the ToR into few links allowing the idle parts of the ToR to be powered off. On the downlink side, the device demultiplexes traffic coming from a few ToR ports to all the servers. We show that this implementation saves significant amount of power and then demonstrate its feasibility via a small prototype and in-lab measurements.

In the next section we describe the design of the MEMS *consolidator*. Section III then describes the implementation of a prototype while section III-A describes measurements run to evaluate the implementation. We discuss related work in section IV and conclude in section V.

II. Traffic Consolidation

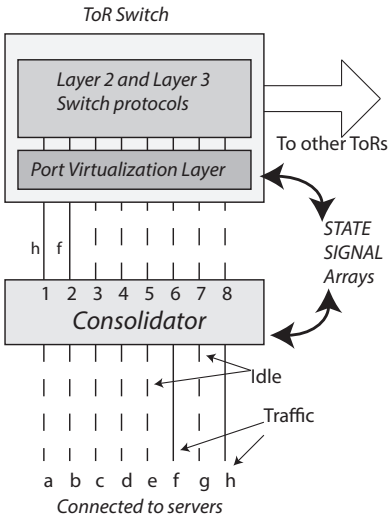

Fig. 2. Operation of the consolidator.

Figure 2 illustrates the operation of the consolidator in a DCN. Let us assume there are only eight servers (*a* thru *h*) in a rack. Instead of directly connecting the servers to ports in the ToR switch, we connect them instead to the consolidator. The consolidator, in turn, has eight links to the switch. Let us assume that only two of the servers *f* and *h* have traffic. The dotted links indicate no traffic is flowing. The consolidator combines the traffic to the *leftmost* two of the eight links (links 1 and 2) to the ToR leaving six links idle. Note that uplink and downlink traffic follow the same paths through the consolidator.

Since we are breaking the one-to-one physical layer mapping between server ports and switch ports, the various Layer 2 and 3 protocols will not run. To fix this problem, we implement a software layer called the *port virtualization layer* between the physical ports in the switch and the Layer 2 and 3 implementation in the kernel. This layer essentially demultiplexes the traffic on the uplink and sends it out via virtual ports to the Layer 2 and 3 software in the switch. In other words, this layer recreates the one-to-one port mapping between server ports and switch ports. On the downlink, the reverse is done.

Consolidating traffic is done using a simple greedy algorithm. Let us assume that initially all servers are idle and no traffic is present. At this point all links to ToR are marked idle. After this, say traffic flow *h* starts. This flow is sent out to the ToR via the leftmost link (link 1) by the consolidator. Next, assume that flow *f* becomes active. If the first link is free (i.e., flow *h* is not continuous) then packets from flow *f* starts going out through the same leftmost link. If flow *h* gets busy at the same time, and if the leftmost link is busy, the packets are sent out via the second link to ToR (link 2). *It is possible that more than one flow will be sent through the same link. This happens if flows are highly bursty.*

The key aspect of this design is that the consolidator needs to be able to detect the start of a transmission *in real-time* and dynamically send the flow via an idle link to the ToR. Furthermore, to enable systematic sleeping of ToR ports, flows are always sent out via the *leftmost free ports* from the consolidator. This ensures that the ToR ports to the right will be idle almost always and can be put into standby mode to save energy.

A. Design of the Consolidator

In order to explain the workings of the consolidator, it is convenient to first focus only on *uplink* traffic (i.e., from servers to the ToR). The consolidator needs to perform three tasks to combine traffic as described: (1) it needs to detect when traffic begins arriving from a link, (2) it needs to send traffic out via a free link to the ToR, and (3) it needs to maintain state so that it knows which links are currently free and to ensure that *downlink* traffic from ToR to the servers follows the exact same path.

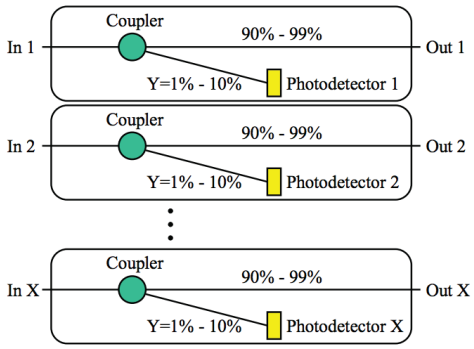

Fig. 3. DiCon Tap Detector Module.

In order to detect the start of a transmission from a server, we use a Dicon tap detector [1] as illustrated in Figure 3. This detector provides in-line power monitoring by using fused couplers on every input, which taps off 1% – 10% of the signal and delivers it to a photodetector. By reading the photodetector output, it is relatively easy to determine when a link has a transmission starting. Of course, it is necessary to introduce

an optical delay to the output of each link in order to give sufficient time for the control logic in the consolidator to send the traffic to an appropriate output link.

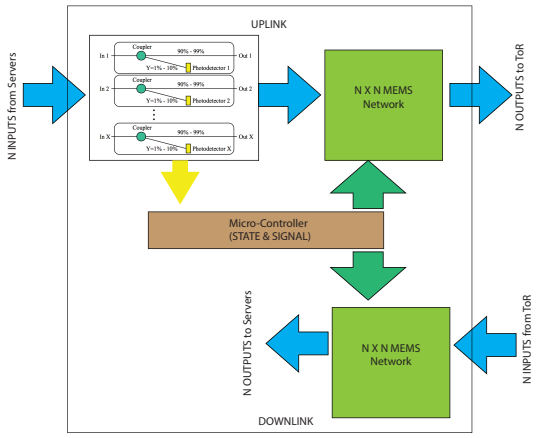

Fig. 4. Architecture of the *consolidator*.

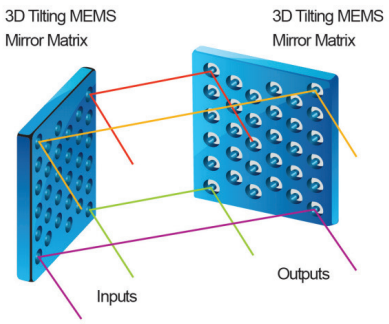

Fig. 5. Example of a MEMS optical switch.

The overall system consisting of the detectors, the MEMS switch, and the controller is illustrated in Figure 4. The output of the detectors is fed to a micro-controller which provides control signals to a fully-connected MEMS switch capable of supporting N simultaneous flows. This switch is completely optical and uses appropriately tilted mirrors to send flows to designated outputs as shown in Figure 5 (these devices are available from various manufacturers including DiCon). The MEMS switch is controlled via a RS232 interface or an I^2C interface and has a switching time of $\leq$ 40ms. We use a second MEMS for downlink flows.

B. Control Algorithm

The control algorithm for the consolidator uses two arrays called STATE and SIGNAL. The ith element of the STATE array stores the name of the server which is connected to the ith output port. For illustration purposes, assume we only have four servers A, B, C, and D. If the STATE array is CADB, this means host C is connected to port 1 (leftmost), host A is connected to port 2, and so on. When there is no traffic from a host, the value of corresponding item of STATE is set to 0. For example, if at some time the packet flow from Host A is completed, the STATE array will be changed from CADB to CDB0 (note that we shift the flows to the left). Following that, if Host C is done, the STATE changes to DB00. The STATE array records which ports are available for the following traffic flows, and the algorithm can use the value of STATE giving the leftmost available port highest priority. For example, if the next traffic flow is from Host A again, the STATE will change from DB00 to DBA0.

An important point to note is that the input to the MEMS switch which specifies the mapping of input to output links needs to be *complete*. That is, every input should be assigned a distinct output, even if the input is not transmitting any packets. Therefore, say only A is sending packets. We send A's packets to link 1 but the other three links also have to be assigned to some input. To handle this issue, we maintain a second array called SIGNAL which provides a complete mapping of inputs to outputs. Let us continue using the above example to illustrate the functioning of the algorithm.

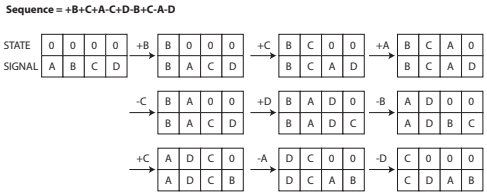

Fig. 6. Operation of STATE and SIGNAL for a 4×4 switch.

Initially, the STATE array is initialized to 0000 since all servers are idle. The SIGNAL array is set to ABCD which is the bypass state. For the traffic flows, we use '+A' to represent that Host A starts sending data, and '-C' to describe that Host C's data transmission is completed. And we use a string to represent a sequence of starts and ends of data transmission from specified hosts. For example, "+A+B-B-A" means a sequence of events - "Host A starts data transmission; then Host B starts data transmission; Host B data-transmission ends; and then Host A data-transmission ends". We illustrate the changing values of STATE and SIGNAL arrays when we have the sequence of data flows shown in Figure 6.

For this 4×4 case, the STATE array starts from 0000, and changes every time a new data flow starts and an old data flow completes. When a new data flow starts, the algorithm finds the first available port (first zero) from STATE array and updates that item to the name of the host that sends the data flow. When a host completes its data transmission, the algorithm traverses the STATE array again, finds the item with the host name and changes it to 0 and shifts assignments to the left. The algorithm needs to traverse the STATE array twice for each data flow, so the time complexity is O(2n). The SIGNAL array starts from ABCD and updates simultaneously with the STATE array. For example, when a new data flow starts, the algorithm changes the ith item of STATE array to Host x. Also, it checks the value of ith item of SIGNAL. If it is not equal to Host x, the algorithm finds Host x at jth position of SIGNAL, and switches the ith item and jth item. The time

complexity of updating SIGNAL array is O(n). Therefore, the overall complexity of this algorithm is O(3n).

C. Downlink

To understand the *downlink behavior* of our consolidator, let us consider two cases. First, let us assume that traffic is flowing from nodes in both uplink and downlink directions from nodes A and C. Assume the two arrays have values STATE = CA00 and SIGNAL = CABD. Since these arrays are shared between the virtualization layer in the ToR switch and the consolidator, when there are packets being sent on the downlink to the servers, they will be correctly sent by the port virtualization layer to either link 1 or link 2. For instance packets for C will be sent down via link 1 while those for A will be sent to link 2. Given that the downlink MEMS configuration mimics the uplink MEMS, these packets will continue through the MEMS switch to the correct server. Consider the second case where the STATE and SIGNAL arrays are as above but there is downlink traffic for D. In this case, the virtualization layer will send the packets to link 3 and will update STATE to CAD0 and the SIGNAL array to CADB.

III. Implementation and Evaluation

We provide a proof-of-concept of out system by building a 2×2 consolidator and implementing the port virtualization layer. The servers and the ToR switch are all Linux machines and use optical NIC cards, with the ToR PC having two cards. All the optical cards operate at 1 Gigabit/sec and use multimode SC fiber. We use an Arduino Uno board to control the MEMS switch. The NIC on the ToR is set to operate in promiscuous mode and passes all packets (regardless of MAC address) to the port virtualization layer. This layer examines the IP address of the source and forwards the packet to the appropriate state machine maintained in this layer. We have one virtual machine for each of the two ports with a static assignment of a virtual machine to one of the two hosts. So for instance, VM1 is linked to host A and VM2 is linked to host B. As packets arrive on the uplink into the ToR, they are sent to either of the two virtual machines. The VMs buffer packets and then call `ip_rcv()` (in `ip_input.c`) to process the packets. Once control is passed to this function, everything proceeds normally within the layer 3 stack.

On the downlink, `ip_finish_output2()` (from `ip_output.c`) sends the packet to the virtual machine that corresponds to the destination IP address in the packet. Thus, packets for B will be sent to VM2. In the virtualization layer, the packets are sent out along the link that is associated with that host in the SIGNAL array. For instance, say host A is idle and only host B is sending and receiving packets. The STATE array will be B0 while the SIGNAL array will be BA. The signal array is always updated at the virtualization layer in the ToR switch. Therefore, when there is a downlink packet for host B, VM2 receives that packet from `ip_output()` and sends it out via link 1 to the consolidator which results in the packet being switched to host B.

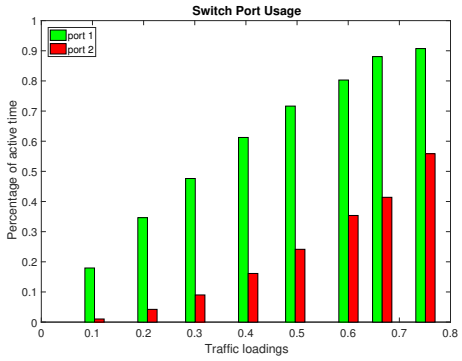

Fig. 7. Port utilization.

A. Measurement Results

We test the utilization of the two ports in a switch (implemented in a PC). We use Iperf to send packets from host A and host B and customize the traffic flows with log-normal distribution of flow length and inter-arrival time to generate traffic with different loadings. We measure the active time periods of the two ports and compare the throughputs with that of a switch without traffic consolidation.

We use the Iperf application to send traffic packet flows to the switch, with overall loadings from about 10% to 75%. The two ports of the switch are named Port 1 (left) and Port 2 (right) for simplicity. When there is consolidation used, Port 1 and Port 2 are connected to Host A and Host B respectively, and they are always in an active mode. With consolidation, the available left-most port is chosen first. That means, in our test case with the switch having two ports, the left port (Port 1) always has higher priority than Port 2 (on the right).

In Figure 7 we see that Port 1 is used much more than Port 2, especially when the loading is smaller. When the loading is about 75%, Port 1 is used close to the line capacity and Port 2 is active half of the time. This proves that our hardware perfectly consolidates the traffic to the left port, which is Port 1.

Figure 8 plots the total number of times flows from A and B switched the output links to the ToR as a function of per-link load. At light loads, there are few switches because both flows can occupy the leftmost link most of the time. Only when both flows have packets simultaneously will one flow have to switch to another link. As loading increases, the number of times flows have to switch links increases since there is more contention for the leftmost link. As load increases further, the number of such switches decreases because both the outgoing links are used most of the time and there is little opportunity to consolidate flows to the leftmost link.

B. Impact on Energy Savings

The overall energy cost of a switch can be roughly partitioned into the cost of the chassis and the cost of the interfaces.

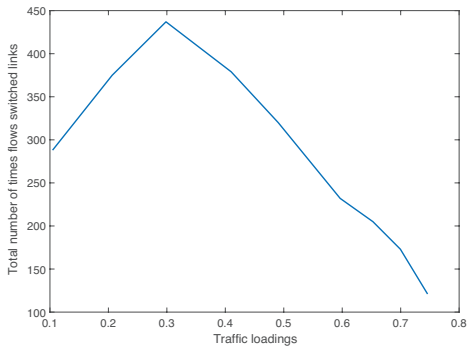

Fig. 8. Number of times port-to-link assignments changed.

As described in [9], [19], a reasonable approximation to the cost of a switch is

$$\text{Switch Cost } = C + m \log m + m$$

where m is the number of active switch ports. The constant C accounts for static costs of a switch such as processing engine, power supply, fan, etc. The second term corresponds to the cost of the interconnection fabric within the switch, which is a significant contributor to energy consumption (typically $30\% \sim 40\%$). This cost scales as $m \log m$ for a switch with m active ports. The last term is the cost contribution from the active interfaces. This term folds into itself the cost of the line cards that the interfaces are on. Note that in today's DCNs, m is essentially the number of links coming into a switch.

If we consolidate traffic, the *average cost* can be written as,

$$\text{Switch Cost with Consolidation } = C + \sum_{k=1}^{m} p_k[k \log k + k] + c$$

where k is the number of leftmost active ports and p_k is the probability that exactly k ports on the ToR are active, where $\sum_k p_k = 1$. c is the cost of the consolidator which is typically negligible. For instance, if using a 16×16 MEMS switch controlled by an Arduino Uno, the cost is:

$$c = 1.8 \text{ W for DiCon MEMS} + 0.5 \text{ W for Arduino Uno} + 1 \text{ W for DiConTapdetector} = 3.3 \text{ W}$$

In comparison the energy draw of a switch is in the hundreds of watts. Therefore, c can be ignored in the above equation.

We simulated a 16×16 system with 16 servers connected to a 16-port ToR through a consolidator network. We assume that C accounts for 50% of the switch cost. In the simulation, each server generates traffic as a Poisson process with mean ranging from 0.1 to 0.7. In Figure 9 we plot the savings as a fraction of total cost of a ToR switch when using consolidation. As the figure shows, at low loads, savings of almost 70% are obtained while at high loads savings drop but are still above 10%. Thus, as our results show, using traffic consolidation can save significant amounts of energy in DCNs at a low cost.

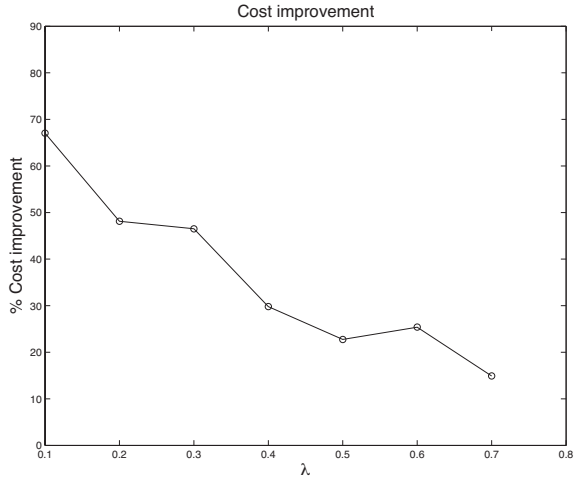

Fig. 9. Reduction in total cost after using traffic consolidation.

IV. Related Work

Most of the related work deals with saving energy by consolidating traffic over fewer links in the Internet. For instance, Chiaraviglio *et al.* [7] formulates an optimization model to find the set of routers and links that must be powered on so that the total power consumption is minimized, subject to flow conservation and maximum link utilization constraints. This network design problem falls into the class of capacitated multi-commodity flow problems, which is NP-complete. Similar work includes [11], which evaluates heuristics algorithms on topology and traffic data from the Abilene backbone network. They prove that the simplest heuristic algorithm can reduce energy consumption by 79% under realistic traffic loads. Other work further explores the practical application of energy-aware routing. Coudert [8] formulates a model combining redundancy elimination and energy-aware routing to increase energy efficiency for backbone networks. [15] quantifies the effects of five recently proposed power-aware routing approaches and shows that switching off redundant links affects terminal reliability (TR) and route reliability (RR) significantly. Accordingly, they propose a practical algorithm, called "reliable Green-Routing" to maximally switch-off network cables subject to link utilization as well as TR/RR requirements.

An alternative approach adopted by some authors is to redesign the network architecture in order to meet energy and QoS needs. Several authors propose new architecture design for energy savings [3] [5] [17]. For instance, [3] considers synchronizing the operation of routers and scheduling traffic in advance since traffic comes from predictable services (such as video). [5] proposes power awareness in the design, configuration and management of networks, and in protocol implementations. They conducted a measurement study about the power consumption of various configurations of widely used core and edge routers and created a generic power model for routers. Nevertheless, [17] proposes a planning model that

clearly shows the trade-off between energy consumption and network performance and emphasizes the importance of accounting for reliability in energy-efficient network design and analyzed robustness issues in some of the designs. To leverage the high efficiency of optical switching, some research on hybrid network architecture combines optical transport and electronic packet processing. For instance, Baldi *et al.* [3] propose to use a complemental Dense Wavelength Division Multiplexing (DWDM) optical cable for deterministic traffic.

In the context of energy efficiency for data centers, Heller *et al.* designed an ElasticTree topology [13] that changes network topology dynamically to adapt to varying traffic load. In the *ElasticTree* approach, Heller developed a variety of optimizers to compute a minimal-power subset of network elements according to different traffic patterns. The power control turns off unnecessary switches and links and the routing assigns routes accordingly. Tradeoffs between power, fault tolerance and performance are considered as well in their approaches. Similar work on dynamic topology is called *CARPO* [18], a correlation-aware power optimization algorithm. Different from *ElasticTree*, *CARPO* first consolidates traffic flows by putting negatively-correlated flows onto the same path and positively correlated flows onto different paths, and then feeds the consolidated flows into a smaller set of links before shutting off idle links.

More recently, Adnan and Gupta proposed an online path consolidation algorithm to dynamically right-size the networks [2]. From multiple equal cost paths between each pair of nodes, their algorithm selects the path, which has most overlap with the paths between other pairs of nodes and meets the total flow bandwidth requirement. By combining all the best overlapping paths together, the minimum energy-proportional topology is formed. When there are large amount of flows in the network, their method outperforms the *ElasticTree* approach. More aggressive approaches include consolidating traffic flows [18] or merging transmission paths [2] to find minimal energy proportional topology. Finally, we note that [6] presents an adaptive optical network to connect ToR switches in a way that can adapt to loads. They achieve this by utilizing a WDM network and selectively powering off links during times of low loads.

Our contribution is different from all of the above work because it is applicable primarily to a single data center rack and because it adapts very rapidly to changing traffic loads by continuously shifting flows to a small set of switch ports.

V. Conclusions

This paper presents a novel method to architect the network between servers and the ToR switches in datacenters to enable significant energy savings. The technique does not affect the loss or delay experienced by packets. Indeed, the overhead of consolidating flows is minimal in terms of cost and in terms of hardware/software complexity. Given that datacenters contain a large number of server racks, the energy savings overall can be dramatic. We implemented a complete prototype of our system for two servers connected to a switch. We are currently scaling this to a 16×16 system in a cluster. We are also metering the actual power usage of the consolidator as well as the ToR to obtain real-world energy saving estimates.

References

[1] https://www.diconfiberoptics.com/products/scd0308/scd0308A.pdf.

[2] 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.

[3] M. Baldi and Y. Ofek. Time for a "greener" internet. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6, June 2009.

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

[5] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[6] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In *NSDI*, 2012.

[7] L. Chiaraviglio, M. Mellia, and F. Neri. Reducing power consumption in backbone networks. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, June 2009.

[8] David Coudert and Alvinice Kodjo and Truong Khoa Phan. Robust energy-aware routing with redundancy elimination. 64:71 – 85. http://www.sciencedirect.com/science/article/pii/S0305054815001252.

[9] V. Eramo, A. Germoni, A. Cianfrani, E. Miucci, and M. Listanti. Comparison in Power Consumption of MVMC and BENES Optical Packet Switches. In *Proceedings IEEE NOC (Network on Chip)*, pages 125–128, 2011.

[10] K. Barket et al. On the feasibility of optical circuit switching for high performance computing systems. In *Super Computing*, 2005.

[11] Will Fisher, Martin Suchara, and Jennifer Rexford. Greening backbone networks: Reducing energy consumption by shutting off cables in bundled links. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, Green Networking '10, pages 29–34, New York, NY, USA, 2010. ACM.

[12] Albert Greenberg, James R. Hamilton, and Navendu Jain. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM*, pages 51–62, 2009.

[13] 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.

[14] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *ACM HotNets*, 2009.

[15] Gongqi Lin, Sieteng Soh, and Kwan-Wu Chin. Energy-aware traffic engineering with reliability constraint. *Computer Communications*, 57:115–128, 2015.

[16] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramnya, and Amin Vahdat. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In *SIGCOMM*, pages 39–50, 2009.

[17] Brunilde Sanso and Hakim Mellah. On Reliability, Performance and Internet Power Consumptiong. In *Proceedings of 7th International Workshop on the Design of Reliable Communication Networks*, Oct 2009.

[18] 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.

[19] Indra Widjaja, Anwar Walid, Yanbin Luo, Yang Xu, and H. Jonathan Chao. Switch Sizing for Energy Efficient Datacenter Networks. In *Proceedings GreenMetrics 2013 Workshop (in conjunction with ACM Sigmetrics 2013)*, Pittsburgh, PA, June 2013.

[20] Minlan Yu, Albert Greenberg, Dave Maltz, Jennifer Rexford, Lihua Yuan, Srikanth Kandula, and Changhoon Kim. Profiling Network Performance for Multi-Tier Data Center Applications. In *NSDI*, 2011.