Opportunistic Power Savings with Coordinated Control in Data Center Networks

Madhurima Ray, Sanjeev Sondur, Joyanta Biswas, Amitangshu Pal and Krishna Kant
Computer and Information Sciences, Temple University, Philadelphia, PA 19122

ABSTRACT

In this paper, we propose a holistic power management of the data center network (DCN) via coordination among distinct controllers with different functionality and scope. This includes - a local controller (LC) that has visibility only at individual switch/router level, a global controller (GC) that has a global view of the network, and a hint based topology aware user request assignment controller (RAC) that controls placement of the external requests on the endpoint hosts. The key function of these controllers from energy management perspective is to properly direct and consolidate network traffic to maximize low power (or "sleep") opportunities for the network interfaces. We show that the coordinated "hints" provided by the GC are vital to correct the myopic view of LCs and RAC for both avoiding congestion and maximizing network sleep opportunities. We show that these mechanisms can reduce power consumption by upto ~45% in the common fat-tree based DCNs using the low power idle (LPI) feature of the Ethernet.

CCS CONCEPTS

• Networks → Network simulations; **• Computing methodologies → Simulation tools**; **• Hardware → Enterprise level and data centers power issues**;

KEYWORDS

Power management; coordinated control; data center network; FAT tree; low power idle

This research was supported by NSF grant CNS-1422921.

ACM Reference Format:
Madhurima Ray, Sanjeev Sondur, Joyanta Biswas, Amitangshu Pal and Krishna Kant. 2018. Opportunistic Power Savings with Coordinated Control in Data Center Networks. In *ICDCN '18: 19th International Conference on Distributed Computing and Networking, January 4–7, 2018, Varanasi, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3154273.3154328

1 INTRODUCTION AND MOTIVATION

With increasing size and energy footprint of data centers, their effective energy management is crucial. The major power consumers of the IT infrastructure in a data center are compute/memory subsystem, storage subsystem, and data center network (DCN) [5] [9]. In this paper, our focus is the DCN, which consumes an increasing percentage of power as the network speeds and connectivity rise due to network upgrade. For example, the power consumption of a 10 Gb/sec Ethernet can be anywhere between 2-10 times the power consumption of 1 Gb/sec Ethernet, depending on the number of ports and the technology used [25]. Also, the power consumption goes up with the number of ports whether they are used or not. Furthermore, the increasing speeds generally result in much lower network utilization, since the deployment of higher speed links is mostly motivated by the technological availability, latency considerations, and ability to handle highly bursty workloads, and much less by sustained high bandwidth demands. This is particularly true in HPC data centers with dense interconnects (e.g., hypercube, toroid) [19] and HPC workloads that may go through multiple steps of collective communication (high network traffic) and parallel computation (very little network traffic). Such situations make network energy management techniques even more important.

Our underlying model of data center is one that serves *external* requests coming in through a user reuqest assignment controller (RAC). Each such request lands on a server, and in turn could generate *internal* flows across various servers using the DCN. For example, an HTTP request may initiate some application and database traffic across the servers. Since our focus is on DCN, we avoid the issues of application deployments and the complex multiserver interactions that may ensue. Instead, we only consider source and destination assignments for individual flows.

As in any energy management context, there are three potential ways in which energy consumption of DCNs can be reduced: (a) Shape the workload at the source by techniques like batching of requests or proper admission control policies, (b) Reduce the link speeds commensurate with the loading – a kind of dynamic voltage/frequency switching (DVFS) control, and (c) Assign the source, destination and network path for each request so that the traffic is largely concentrated on fewest links and thus maximizes opportunity for other links to sleep.

Workload shaping [7] is a well studied subject and we do not consider it here, although it could help in more effective use of sleep modes that we do explore here. Link level DVFS is also thoroughly explored, but it is practical only at a very large time-scale and coarse granularity level. This is because most networks provide only a few speeds (e.g., 40, 10 and 1 Gb/sec) and the switching is essentially amounts to PHY switching and can be very slow. Although IEEE introduced the Rapid PHY Selection (RPS) mechanism to allow for dynamic speed changes, such a mechanism is still too slow and may cause packet drops and other problems [4]. Therefore our focus is primarily on (c) and we use the link sleep mechanisms defined at PHY and higher layers, as detailed in section 4.

Flow path selection and consolidation to maximize sleep opportunities can be quite challenging in a large data center network because a workable scheme must simultaneously consider availability of endpoint (or server) resources (CPU, memory, etc.), local bandwidth demands at the switches, and end-to-end blocking/delay on the network paths. In a small data center, this can be accomplished via a central controller that has global network and endpoint visibility, however, such a scheme does not scale. In this paper instead, we design a low-overhead semi-distributed scheme that involves three cooperating mechanisms: (a) an energy aware distribution of requests to the endpoints by the user request assignment controller (RAC), (b) an intelligent routing of a new flow at each switch by a local controller (LC), and (c) a lightweight global controller (GC) that monitors network traffic and provides hints to LC's and RAC for better placement of flows at endpoints.

We explore these issues in the context of the almost universal DCN topology in traditional data centers – the fat tree. Although the underlying mechanisms are general and can work for any regular network topology, applying them well requires exploitation of the topology, and will not be discussed due to lack of space. Similarly, while similar approach can be used to non-Ethernet interconnects as well (e.g., infiniband), a proper investigation must consider energy management techniques applicable to the specific fabrics.

The rest of the paper is organized as follows. Section 2 provides a broad overview of our scheme along with the key contributions. Section 3 reviews the related works on network energy management. Section 4 discusses power management mechanisms for a network interface including the low power idle (LPI) mechanism for Ethernet. Section 5 discusses various controllers, their interaction, and how they are implemented. Section 6 presents details on our simulation based evaluation of the mechanisms as well as the simulation results. Finally, section 7 concludes the discussion and lays out the future work.

2 OVERVIEW AND CONTRIBUTIONS

An intelligent choice of flow paths through the DCN and their potential reorganization requires global visibility into the network that is not present in traditional networks. Although a software defined network (SDN) can easily provide such global visibility, we do not necessarily require SDN deployment in the data center. Instead, we assume that the management layer on each switch monitors the BW usage on all the interfaces and periodically provides this information to a global network controller or simply Global Controller (GC) running on some server. GC can use this information to make placement and/or reshuffling decisions; however, doing so will make it unscalable. Thus our approach is to simply provide some hints to the local controllers (LC) for choosing the outgoing link at each switch. The intent is to keep the GC both lightweight and non-critical – The LCs can continue to function uninterrupted but with degraded performance even if the GC fails and is restarted. Another advantage of this mechanism is that LC's and GC will not "fight" since LC is simply following the recommendation of GC, which is invoked infrequently. Because of the lightweight nature, GC can easily handle a rather large network (hundreds of racks). It is possible to define a federation scheme for even larger networks, but we do not consider that in this paper.

Properly directing the incoming request to the appropriate server and choice of destination server for the flow is crucial for ensuring that the hosts are not overloaded and the host level workload can be consolidated. From the perspective of DCN, an even more important reason to include request redirection is to enable consolidation of network traffic and avoidance of congestion. At the time of traffic forwarding as well, the network switches need to ensure that aggresive flow consolidation on specific links should never lead to network congestion. We accomplish this by having the GC provide hints to the RAC and LCs for sufficient adaptation and adjustments in a lightweight and non-critical manner.

It is important to note here that the energy management can be handled – simultaneously, if needed – at multiple time granularities. For example, built in hardware mechanisms often operate transparently at fine time scales (e.g., C1-C6 CPU states), whereas sophisticated traffic consolidation schemes based on solving optimization problems and reconfiguration can operate at long time granularities (10's more minutes or more). Most techniques explored in the literature fall in the latter category [15]. In contrast, our primary interest is in medium grain mechanisms that can be implemented in software but must be relatively lightweight and non-disruptive.

Given this, the goal of our mechanisms is primarily to incrementally handle the incoming flows at the switches, rather than dynamic reshuffling of the ongoing flows. By dynamic reshuffling we mean moving partial or complete path of an ongoing flow to a new and relatively more utilized path in order to improve network consolidation (without moving the source and destination of the flow). Our existing implemetation is able to reshuffle an ongoing flow. However, we found the mechanism to be very disruptive and often moving an ongoing flow results in packet drops, out of order packet arrivals, and even higher energy consumption. Hence we have not dicussed the shuffling mechanism in this paper. Thus, in this paper we largely obviate the need for dynamic actions via intelligent flow placement and routing actions. By extensive simulations, we show that our scheme can drastically *reduce the overall network power consumption (by upto ∼45% compared to the DCN without any power saving scheme) especially in case of low traffic hours, without degrading the overall network performance.*

3 RELATED WORKS

Mostowfi [17] studied EEE LPI operations that put the device into two states - deep sleep and fast wake. Thaenchaikun et al [27] proposed energy saving model using a control plane that utilizes an energy aware routing protocol. They showed that, a combined strategy for routing protocol and energy aware path augmenting solution can provide good energy savings. Gupta et al. [10] proposed idle power control techniques that slow down clock rates in the network links, put subcomponents in switches in sleep mode, and then reconfigures the network topology. Nedevschi et al. [20] proposed a buffer and burst scheme that shapes traffic to alternate active and idle periods, thereby providing increased opportunities to sleep. The authors also proposed a scheme where rate of operation of network links is dynamically adapted to the arrival rate of packets. Abts, Dennis and Marty [2] discussed link adaptation to dynamically reduce the link speed for less energy consumption using a central controller. They do so using adaptive link rate (ALR), which is a predecessor to LPI mechanism for EEE that attempts to switch PHY(s) at run time (as opposed to simply the initialization time). It turns out that even with Rapid PHY Selection (RPS) it takes time to change the speed of the link and there is a huge overhead of changing the link speed.

Besides LPI, there are many works on the minimization of the number of active links and switches by aggregating traffic over a small set of links [3][30]. These works have tried to implement different intelligent mechanisms to achieve the goal of traffic consolidation. All the solutions consist of a centralized optimizer, which tracks the traffic statistics and redirects the traffic according to the optimization solution. For example, Wang [30] discusses about the correlation of peak workloads among different flows while consolidating. In our work, centralized optimizer uses topology statistics for both the traffic merging and request aggregation in a coordinated manner to avoid the conflicts in agenda.

To save power at the endpoints, VM consolidation has also been considered as an efficient method in literature. Most of them [29],[14] have considered mainly VM placement and migration as optimization problems to reduce the server energy consumption without any non-local network side considerations. Thus, the server energy savings may be accompanied with increased packet drop and jitter. A coordination between VM consolidator and network is essential to address this problem, as we have explored in this paper. In our work, we perform the endpoint consolidation by aggregating the external request to the target servers. There are several previous works on server-network coordination like [31][11][16][13]. The primary concern of the [16] and [13] is to place VMs close to each others, those having more mutual communication. That reduces the path length in other word the delay and the bandwidth requirement at the higher layer of the data center. [31] has extended the work of [30], by considering VM utilization and correlation analysis for both the VMs and flows - in a coordinated way. In [11], the problem of VM placement and flow routing has been represented as a unified optimization problem and then solved.

In [8], the authors present an energy aware load balancer that distributes the tasks on VMs based on their speed and energy efficiency. In their model the migration decisions are based on the vCPU units demanded by an application and the available capacity of the host and of the other servers in the cluster. Authors in [21] schedule the tasks on VMs that are normalized based on system and application level resource management. Their scheduling and migration of VMs is based on the vCPU units demanded by an application and the available capacity of the host and of the other servers in the cluster.

Since VM migration can be expensive both in terms of latency and network traffic [29][28], we do not depend on VM migration in our work. Instead, our main goal is to provide hints to local controller and the request assignment controller so that they work in sync. Besides, our work is based on minimal assumptions: as per our assumption, there is no affinity among the VMs known to us in advance while placing the VMs, so that we can exploit that VM placement while redirecting the requests. In reality, these metrics would be very uncertain, so VM consolidation/migration and flow routing based on that statistics might not bring expected results.

4 ETHERNET POWER MANAGEMENT

4.1 Overview

Nearly all of the network links are currently based on the serial links that use differential signaling, because such a technology can easily handle noise and does not suffer from cross-talk and clock skew issues. The links are then built using one or more such serial interfaces called "lanes". This has led to the notion of repurposable Phy layer, i.e., the same basic Phy module that can support very different higher layer technologies including PCI-Express, Ethernet, Infiniband, Fiber Channel, etc. At a very low level, all the links possess three operating states:

L0: This is the normal operational state with highest power consumption, say, P_{L0}. P_{L0} does not have much dependence on the utilization since "filler" Idle messages are constantly exchanged whenever the link is idle even for very short periods. In other words, the link is always 100% busy.

L0s: This is a sleep state with entry and exit penalty typically in 10's to 100's ns range. In L0s, the power level P_{L0s} is around 40-50% of P_{L0} depending on the design [6]. There is usually a trade-off between transition latency and sleep power. L0s control applies independently to both sides of a bidirectional link. The link is kept "trained" during L0s and thus a part of the interface must stay awake.

L1: This is a much higher latency non-operational state with exit latencies in 10's of μs range, but generally with a very low power consumption (e.g., 10% of P_{L0}) [6]. This state requires a handshake between transmit and receive sides and link retraining upon exit from low power mode. If either side refuses to go into L1, L1 will not be entered. The training symbols are not exchanged (like L0s) during L1 and thus link retraining is required on wake up, which makes the exit latency quite high.

The L0s and L1 states are defined at PHY level but may not be exposed at higher level. In case of Ethernet, IEEE has defined an enhanced version of L1 called Low Power Idle (LPI) in the 802.3az-2010 [6] standard. LPI does not continuously transmit idle signal and instead sends periodic refresh to maintain the synchronization. Wake up from LPI involves a significant exit latency, since the transmitter needs to wake up the receiver before transmitting anything. There are also new emerging energy management standards for 40 & 100 Gb/s Ethernet links for example, 802.3bj [18] (deep sleep mode) and 802.3bm (shallow sleep with fast wakeup). The deep sleep mode is essentially same as LPI described here, but the shallow sleep allows faster wakeup. In our experiments (not reported here), the shallow mode did not offer any advantage, and thus is not considered further.

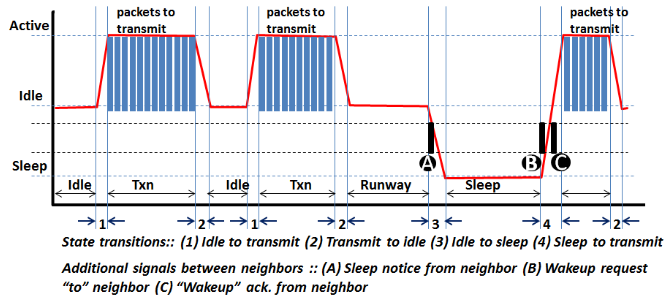

Active
Idle
Sleep
packets to transmit
packets to transmit
packets to transmit
Idle
Txn
Idle
Txn
Runway
Sleep
A
B
C
1
2
1
2
3
4
2
State transitions:: (1) Idle to transmit (2) Transmit to idle (3) Idle to sleep (4) Sleep to transmit
Additional signals between neighbors :: (A) Sleep notice from neighbor (B) Wakeup request "to" neighbor (C) "Wakeup" ack. from neighbor

Figure 1: TRS packet transmit model over-laid with inter-device communication

4.2 Simulation of Power Management

For this study, we enhanced the popular ns3 network simulator [24] with two energy management features: Transmitter Only Sleep (TOS) model and (ii) Transmitter Receiver Sleep (TRS), which roughly correspond to L0s and L1 states.

The TOS model is hardware driven and allows the transmitter to sleep independently when the gap between the traffic is small. The basic approach is to transition to L0s if the idle period exceeds some specified amount called "runway" [12]. The exit happens on arrival of the next packet. For simplicity, a fixed runway duration is used, although utilization dependent runway could be easily implemented [12]. The TRS model is software driven, birectional and it is an implemetation of LPI. In TRS, the transmitter sends sleep packet to the neighboring receiver before going to sleep. Both the transmiter and receiver have to come to an agreement before the transmitter can move to sleep mode. So it has extra overhead. As the transmitter wakes up, it has to send wake up packet to neighboring receiver before sending any traffic intended for it, as shown in Fig. 1. So, it has a higher exit latency. Thus TRS uses larger runway to ensure the transmitter and receiver can only move into the sleep mode when the gap between traffic is large.

5 NETWORK MONITORING & CONTROL

As stated earlier, network control in our model is effected by three entities: Global Controller (GC), Local Controller (LC) and the topology aware Request Assignment Controller (RAC). The functionality of GC is actually separated into two entities, namely, Global Traffic Monitor (GTM) and the Global Traffic Consolidator (GTC). The former does the monitoring and hinting, whereas the latter does consolidation of ongoing flows by reshuffling, as explained before. Since we have not used the GTC in this paper, so henceforth we are going to use the term GC and GTM interchangeably.

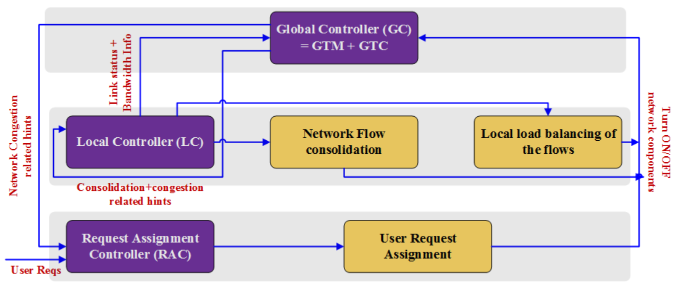


Figure 2: Interaction between various controllers

When multiple controllers are involved, it is crucial to build a consistent strategy that needs to be followed by the controllers to avoid any contradicting actions [23]. In terms of control, GTM merely provides hints to LCs and RAC for traffic consolidation and external request placements respectively. It is reasonably active in collecting up to date information, so that it can provide timely hints to RAC and LCs. The main challenge for the GTM is to build *consistent strategies* for the LCs and RAC in order to maximize sleep opportunities and to avoid network congestion.

Each LC periodically collects low-level network statistics (For example, link utilization, number of active flows etc.) and sends it to GTM. GTM builds a global view of the network based on those accumulated intelligence. Depending on the global state of the network, the GTM sends back very little hints that can help LCs to make routing decisions. By adopting this simple information exchange mechanism, it is possible to keep the GTM scalable even with reasonably large networks. However, a directed study of scalability is beyond the scope of this paper.

The LCs are responsible for routing of a new incoming flow and can only do local consolidation of the flows. For example, in the fat-tree network shown in Fig. 3, an edge switch has two links towards the aggregation switch. From a power management perspective, it is better to concentrate traffic on one of them (if possible), so that the other link can have longer idle durations and hence better chance to save power by going into the sleep mode. It is clear that if each LC greedily focuses traffic on one of the links, this may result in congestion or suboptimal traffic distribution at higher levels. The purpose of GTM is to provide hints to LCs to avoid this situation.

Besides providing hints to the LC, at the same time GTM provides these topology aware hints to the RAC. By doing so, the LCs and RAC can work collaboratively. The hints help the RAC to place an external requests in a way that LCs get the opportunities to consolidate the traffic without much degradation in the quality of services. The overall architecture has been shown in Fig. 2. In the following section, we discuss the advantages of the coordination between these various controllers.

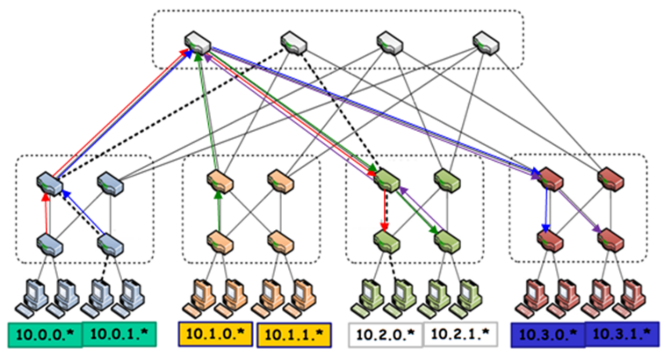


Figure 3: An illustration of fat-tree network. The flows are mostly concentrated at the leftmost links for better energy savings.

5.1 Local Controller

As mentioned before, the LCs work at the switching node level. They operate based on the probability factor assigned to the outgoing links by the GTM, and use a *customized routing table* which records all currently ongoing flows. When a packet comes to a switch, the LC at that switch tries to match the flow id and the destination of the packet with the existing entries. If it finds a match for that flow id and destination, it simply forwards the packet on the path associated with that entry. If not, the packet is forwarded to a path selected by the LC, based on the probability factor assigned by the GTM.

In case the LC at a switch is not able to place a flow in its first attempt (due to congestion at some links), we record it as a *"flow blocking"* event. We use flow blocking as an indicator to balance our twin objectives of flow consolidation and congestion avoidance through load balancing. Notice that in most cases, it is appropriate to retry the flow placement using a different path, and unless the paths are overly subscribed (rare in data centers), the attempt would succeed. However, we do not address this aspect here for simplicity.

5.2 Global Controller

As stated earlier GTM gives hints to LCs to consolidate the traffic over a small number of links. GTM does this by assigning a *probability factor*, which the LCs refer to while forwarding the flows. Notice that in a k-ary fat tree each edge and aggregate switch has $\frac{k}{2}$ candidate links for forwarding the traffic over other Pods. The candidate links are given ranks or priorities that are consistent throughout the network. Without any loss of generality, we assume that the candidate links can be ranked from left-to-right, i.e. the flows are mostly consolidated to the leftmost links of the switches. For example in Fig. 3, the LCs assign the flows at the leftmost links which ensures better flow consolidation and thus energy savings. The probability factor determines what fraction of the incoming traffic is forwarded to each one of the $\frac{k}{2}$ candidate links.

We now explain calculation of the probability factors for the candidate links for $k = 4$, which can be generalized for any other k. There are two candidate links at the i-th switch, which are denoted from left-to-right as $\{l_{i1}, l_{i2}\}$ and their probability factors are $\{P_1, P_2\}$ respectively. The key idea is to ensure that the flows are mostly consolidated at the leftmost links at the time of low-traffic hours, whereas are gradually spread across the network at the high-traffic hours to avoid unnecessary flow blocking. We define the overall network *utilization factor* as the ratio of cumulative bandwidth of all the inter-rack flows, and the total capacity of the edge-level links. This utilization factor is used as a knob to tune the values of P_1 and P_2 to consolidate or spread the traffic over the network. The overall scheme can be described by 4 states, as described below.

State1 (S1): The system stays in this state at the lull hours. This happens when there is no flow blocking in the last N windows (each window consists of n number of flows) and the utilization is less than a threshold τ_1. In this case, P_1 is incremented by Δ_i (whereas P_2 is decreased by the same amount) and N becomes 1. At this state P_1 is quickly incremented, so that the flows are more consolidated at the leftmost links.

State2 (S2): The network switches from S1 to S2 if the utilization grows from τ_1 to τ_2 ($\tau_2 > \tau_1$), without experiencing any flow blocking in the last N windows. In this case, P_1 is incremented by Δ_i and N is incremented by a factor of α ($\alpha > 1$). Thus, in this state the probability of forwarding the flows to the leftmost links increases, but at a slower rate.

State3 (S3): The system transitions from S2 to S3, when the utilization grows beyond τ_2. In this state the probability factors remain unchanged to avoid overburdening the leftmost links further.

State4 (S4): The system goes to S4 whenever it experiences any flow blocking. In this case, P_1 is decremented by Δ_d, whereas N is decremented by a factor of β ($\beta < 1$). Thus if there are more flow blocking in the successive windows, P_1 is decremented quickly to shift more flows to the rightmost links. The entire state diagram is depicted in Fig. 4, where $N_{\max}$ is assumed to be the maximum value of N beyond which N is not increased.

We adjust the τ_1 value by adopting a simple learning approach. If there is no flow blocking for $N_{\max}$ windows with an utilization level upto U_{NB}, then the value of τ_1 is set to $\min\left\{\tau_1^{\max}, U_{NB}\left(1 - k\Delta_d\right)\right\}$, where $k = 0, 1, 2, ..$ is a predefined constant, $\tau_1^{\max}$ is the maximum allowed value for τ_1 and is less than τ_2. In this fashion the value of τ_1 is set slightly

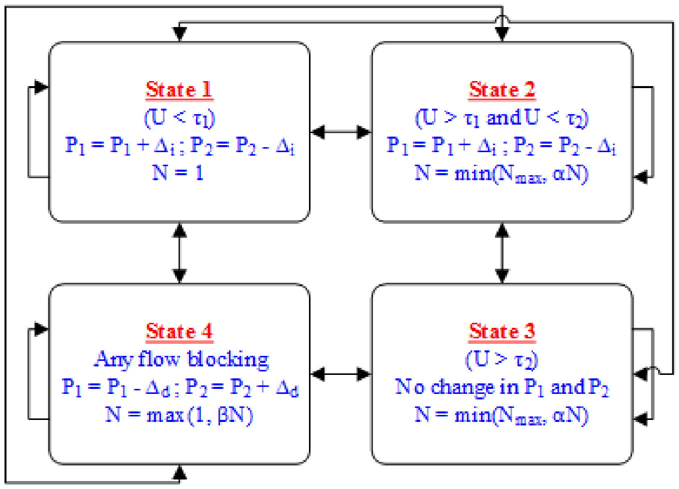


Figure 4: Proposed state diagram for assigning P_1 and P_2 to the links. U denotes the utilization factor.

lower than the maximum utilization level with no flow blocking. The objective of this is to keep τ_1 low enough to avoid flow blocking, and at the same time keep it sufficiently high for effective consolidation.

For simplicity we assume *identical probability factors* for all the LCs throughout the paper. Even then, by using this simple adaptation we ensure that the LCs forward most of the flows through the left-most links at low network load and gradually use the right-most links as the load grows over time. This simple adaptation strategy can be further extended to include more complicated and heterogeneous probability factors per LC.

In addition to that the GTM also provides hints to the RAC to assign the user requests for better energy efficiency as mentioned in section 5.3.

5.3 Request Assignment Controller

Each host (server rack) has limited resources in terms of number of user requests one application can serve simultaneously, in addition to the limit on how much traffic its network interface can carry. The *Request Assignment Controller (RAC)* assigns the user requests among appropriate server racks, depending on the policies adopted by the network manager. Some of the polices can be taken independently by the RAC, whereas other need network information provided by the GTM. RAC can be considered as a enhanced version of the traditional Data Center Load Balancer, with similar scalability but different optimization goal.

Policy1 (P1): User requests are assigned to the highly loaded rack of the highly loaded Pod as far as their load is within certain limit.

Policy2 (P2): The user requests are assigned to the least loaded server-rack of the least loaded Pod. This policy is generally adopted for balancing the user requests across the hosts and the network.

Policy3/4 (P3, P4): In these policies the RAC consults with the GTM to avoid the racks where downlinks experience high network load. In these strategies the policies are identical to P1, P2 except that in these policies the decisions are taken after referring to the GTM.

6 PERFORMANCE EVALUATIONS

6.1 Experimental Setup

In order to comprehensively evaluate the energy management for data center networks, we considered several detailed network simulation packages and ultimately decided to use the popular ns3 package, as it turned out to be much more comprehensive than others. Unfortunately, the ns3 libraries does not provide any power management capabilities (with the exception of very superficial and artificial power calculations). Thus, a considerable effort was spent in implementing the basic energy management capabilities at the level network transmit and receive "device" functionality. Many other enhancements to ns3 were required in order to perform the local consolidation of packets and the controller functionalities. In the following, we briefly discuss several aspects of the simulation model developed in the ns3 context. The details about the implementations in ns3 is mentioned here in this paper [26].

6.1.1 Traffic Model. A data center network will be expected to have many traffic flows with different characteristics, and a comprehensive exercising of various controllers requires a flexible synthetic traffic generation capability. Although we carried out the experiments based on the characteristics of the packet traces collected from the Internet routers [1] and used them in our packet generation process.

The ability to generate traffic flows with varying levels of burstiness is key to comprehensive evaluation. To support this in a simple way, we implemented a two state Markov Modulated Renewal Process (MMRP) [22] for a flow. In this model, the two states have different renewal processes for flow generation based on the given burstiness and average rate parameters. The residence time distribution in each state is exponential and is calibrated using the average residence time in each state. For example, a long average residence time would result in flows being generated at a rather rapid rate for quite some time (which may congest the network) followed by another long period where the flow rate is low and thus the network utilization may go down significantly (after the backlog built during the other MMRP state is cleared).

The mean flow rate in each state is determined by the desired long-term link utilization. For most of our experiments, we assume the arrival process in each state to be Poisson; however, the experiments calibrated using real traces use a hyperexponential interarrival time process with a specific coefficient of variation.

Flow duration in the network follows Pareto distribution with shape=3.5 and mean as 3.6 ms, ranging from 2.2ms to 375 ms. The bandwidth of each flow follows uniform distribution with mean 1 Gbps and a range from 500 Mbps to 1.5 Gbps. By varying all these parameters accordingly we can create traffic that maps to different applications that usually run on a real data center network.

6.1.2 Network Configuration. We choose the most commonly used data center network topology, namely fat tree for our experiment shown in Fig 3 with k=4. However, the mechanisms that we have developed are not dependent on any specific topology and can be used for any structured data center network. In fat tree architecture, the network bandwidth requirements increase significantly as we go up the hierarchy; therefore, the best infrastructure that one can have is link speed ratios of 1:2:4 at the 3 levels i.e. in the same ratio as the network infrastructure expands. However, unless otherwise mentioned, in our experiment we have used the link bandwidth in a ratio of 1:2:2 i.e. 10GB at edge routers and 20GB at the aggregate and core respectively. The motive here is to assess the performance of our algorithm under this stressed condition. All these links are synchronous links i.e. sends FISU(s) when they have no real traffic to send and consume nearly equal power as active links. The device or interfaces behave as TOS or TRS models, with the runway durations and state transitions as explained above. Furthermore, we collected metrics about device utilization factor, state transition, device idle time, sleep time and active transmit time. We studied these device metrics along side the traffic flow metrics and energy metrics. We summarize the results in the next section.

Table 1: Simulation Environment

Parameter	Values	Parameter	Values
Active Power	14.85 W/port	Idle Power	14.85 W/port
Shallow Sleep Power	5.94 W/port	Δ_i	0.03
Deep Sleep Power	1.485 W/port	Δ_d	0.05
Shallow Sleep wake-up latency	10 ns	$N_{\max}$	16
Deep sleep wake-up latency	15 μs	τ_2	50% utilization

6.1.3 Energy Model. Energy models were introduced to capture the device power and energy utilization. These models reflect device behavior as they start moving packets during active transmission and manage to steal sleep intervals during inter-packet time or idle time (based on runway times). We captured the device activity at three states: transmit time, idle time (while it is in runway duration) and sleep time (expiry of runway and till next packet transmission). These timings along with respective power consumption give us a good insight on the device energy consumption. The device activity time was used to compute the device-utilization factor. We summarize these metrics along side the traffic flow metrics to compare the power saved and observed effects on traffic flows such as delay, packets dropped etc. We discuss the observed metrics and summarize the results.

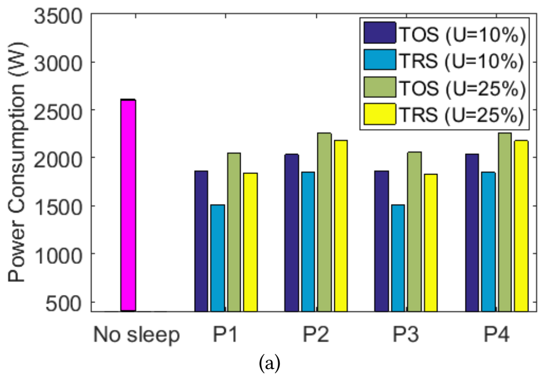
6.2 Experimental Results

For our performance evaluations, we assume α and β to be 2 and 1/2 respectively. While placing the user requests, we ensure the maximum Pod level utilization to be 70%. The performance is measured in terms of the average power consumption, and end-to-end delay. Relevant parameters used in the simulations are listed in Table 1.

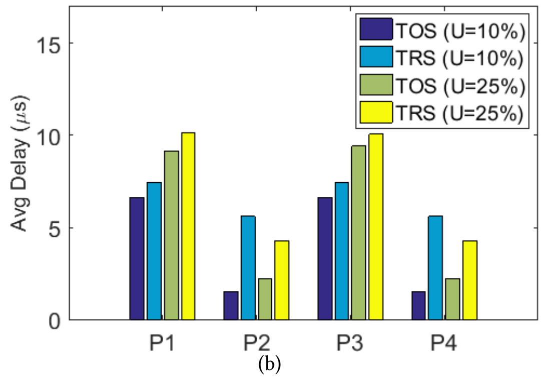
We tested the network under four different network utilizations of 5%, 10% and 25% respectively. We did not go beyond 25% utilization because at higher utilizations, it would be very difficult for a device to find opportunity to sleep. Also, any sleep at that high utilization can create performance issues in a data center environment. The use of maximum 25% utilization is also consistent with the typical usage patterns seen in data centers, where the *average* utilization is usually in teens, and it would be unusual to see sustained high utilization levels. Unless explicitly mentioned, the values of probability factors (P_1, P_2) kept below a maximum threshold of $P_{\max}$ to avoid network congestion through some links.

One other point to note is that our study only considers the device or port level energy; we have not considered the backplane of a switch. We have actually implemented and tested backplane power management [26] where the backplane is opportunistically put in low power mode if all switch ports are inactive. However, for simplicity of explanation we do not include this aspect in the results reported here.

6.2.1 Experiments with Real Traces . All of the experiments are done using synthetically generated traffic with parameters derived from real traces from [1]. A detailed analysis of these traces revealed that using them directly will not be meaningful for a variety of reasons including many different types of packets and many very short flows. Instead we extracted a number of relatively long flows from these traces and studied their statistical properties. We then used these statistical properties to generate the traffic. Such an approach allows us to generate long traces as needed for the experiments. Incidentally, the traces did not show any significant autocorrelation, which made the traffic generation easy. Thus, for all our experiments, we have studied the packet interarrival time from the traces and used the statistical properties i.e. coefficient of variation (in the range 5-6) to generate the packets following a Gamma distribution.



(a)



(b)

Figure 5: Comparison of (a) power consumption, and (b) average delay for different policies.

For our simulation we assume that each user request generates an inter/intra -pod flow in between a pair of host-racks, which are chosen depending on the polices P1-P4 as described in section 5.3.

6.2.2 Comparison of different policies. Figs. 5(a) and 5(b) show the effects of different policies on the metrics at 10% and 25% utilizations. Policy P1 ensures better flow consolidation, which results in more power savings. From Fig. 5 we can observe that P1 consumes ∼8% less power than P2 in case of TOS and ∼18% less power in case of TRS. Also we do not observe any flow blocking here with any of the policies. However the extra power saving in case of P1 and P3 comes at a cost of higher delay since these policies try to place a new flow on highly utilized racks.

We can also observe that with our proposed power saving schemes, the overall power consumption is reduced up to ∼45% under low traffic loads and ∼15% even in case of high network load, compared to the baseline with no power saving scheme. Thus the scheme is extremely useful especially for network with low traffic load.

6.2.3 Comparison of TRS and TOS. Figs. 6(a) and 6(b) show the performance of P3 at utilization levels 5%, 10% and 25%. From this figure we can observe that with flow consolidation, TRS always outperforms the TOS at all utilization

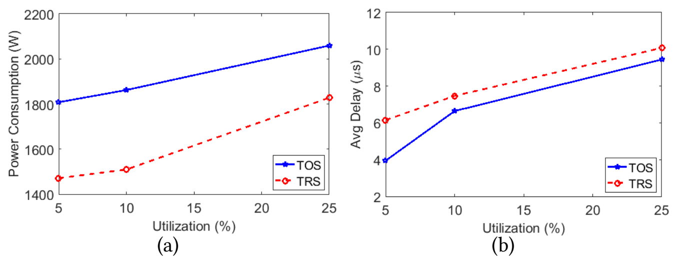


Figure 6: Comparison of (a) power consumption, and (b) average delay for P3 under different utilization levels.

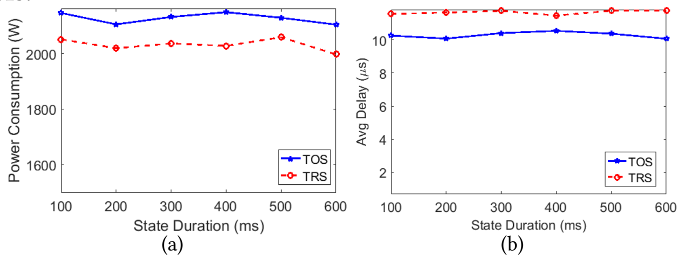


Figure 7: Comparison of (a) power consumption, and (b) average delay for bursty traffic.

levels. In particular TRS consumes ~11-20% less power than TOS. Notice that the improvement of TRS is more mainly in case of low utilization level. This is because in low utilization levels, the network wide consolidation provides more chances to put the devices in deep-sleep and it is less likely that soon they will receive a packet and wake up. However, TRS has a higher wake up penalty in the range of ~15 μsec, as compared TOS which is ~10 *nsec*. With higher utilization levels, the improvements of TRS starts reducing as the devices try to sleep opportunistically. In such a scenario, if a device moves to the deep sleep state, it may be forced to exit shortly and incur significant penalty. Thus TRS is more effective at low utilization scenarios, as *the flow consolidation creates adequate opportunity for the devices to sleep and thereby counters the inherent overhead of TRS.* It is not surprising that the extra power savings especially at low utilization comes at the cost of increased delay. As we can see from the Fig. 6(b), the power-performance trade-off is seen consistently – as the average power consumption decreases, the average delay increases. This is mainly because of its extra wake-up overhead. It is worth noting, however, that the power saving achieved by consolidation is still more than the percentage of delay increased due to this consolidation.

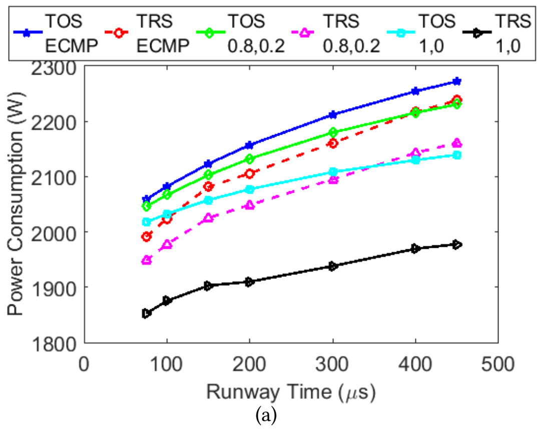
6.2.4 Effect of Bursty Traffic . We next evaluate the performance of our scheme under various levels of traffic burstiness. We assume 25% network utilization for this set of simulations. To generate bursts of traffic we have varied the residence time and the flow injection rate of state1 and state2 of the MMRP model. We have selected the rates as 0.5 and 1.5 for state1 and state2, respectively, and varied the average residence time from 10 to 60 ms. Under 25% utilization, the expected inter arrival time of the flow is 0.09 milliseconds and the expected life of a flow is 3.6 ms. In state1 a flow is generated with probability 0.5, whereas in state2 one flow is generated always and then the second one with probability 0.5. The purpose of using different burst durations is to assess the adaptability of our system to changing traffic conditions. In state1 we are likely to have fewer flows, which decreases the utilization and opens up the opportunity for traffic consolidation and power saving. In contrast, the traffic rate in state2 triples as compared to state1, which would trigger the switch to slowly move towards a more balanced load.

Figs. 7(a) and 7(b) shows the power consumption and end-to-end delay as a function of average burst duration. From this figure we can observe that the performance factors do not change significantly with different durations. This is because of the adaptive nature of our proposed scheme to maintain a balance between flow consolidation and load balancing depending on the network traffic. We can also observe that TRS saves more power than TOS irrespective of the state durations. However, with bursty traffic the network experiences flow blocking (less than 1%) because P3 places the requests to the highly loaded racks. In general P3 reduces the flow blocking compared to the other policies without hurting the energy consumption characteristics at high utilization levels or in a network handling bursty traffic . This shows the effect of using network hints provided by the GTM, instead of independently assigning the user requests to the racks.

6.2.5 Comparison with ECMP. Figs. 8(a) and 8(b) show the comparison of our proposed scheme with ECMP. From this figure we can observe that with $P_{\max} = 0.8$, the proposed consolidation scheme reduces the power consumption up to ~5% and average delay up to ~4-5%. By making the $P_{\max}$ equal to 1, the power consumption and delay can be improved up to ~14% and ~10% respectively. The improvement is more visible in case of TRS because of its ability to go into the deep sleep mode. However, this power savings come at the cost of higher delay as seen from Fig. 8(b). We can also observe that in case of TRS, the average delay decreases with the increase in runway time because it reduces state transitions for shorter gaps in traffic.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have studied the data center energy management that involves coordination among 3 types of controllers: local controller at each switch, a global network controller, and an energy aware user request assignment controller. We studied simple mechanisms to coordinate the actions of these controllers and showed that such coordination can result in



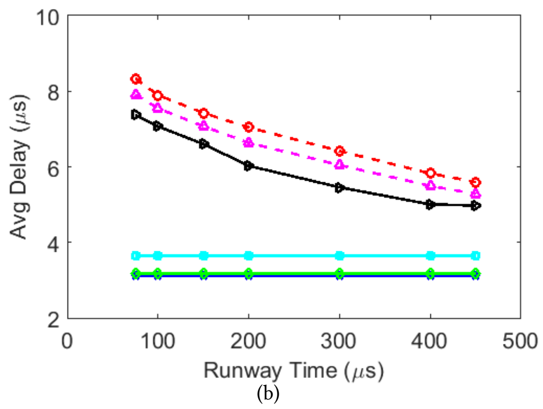


Figure 8: Comparison of (a) power consumption, and (b) average delay with ECMP and the proposed consolidation scheme. x and y in TOS(x, y) and TRS(x, y) denote $P_{\max}$ and $(1 - P_{\max})$ respectively.

significant energy savings without needing VM migrations or shuffling of established flows. In particular, while our global controller consolidates established flows if necessary, we found that it almost never gets invoked.

In the future we plan to study other types of networks, such HPC DC networks which can consume a significant portion of the overall energy consumption [19]. We will also consider more sophisticated hinting mechanisms and explore their use in more intelligent consolidation.

REFERENCES

[1] Caida data server. http://data.caida.org/datasets/passive/passive-oc48/20020814-160000.UTC/pcap/.

[2] Abts et al. Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, June 2010.

[3] Al-Fares et al. Hedera: Dynamic flow scheduling for data center networks. In *USENIX NSDI*, pages 19–19, 2010.

[4] Blanquicet et al. An initial performance evaluation of rapid phy selection (rps) for energy efficient ethernet. In *IEEE LCN*, pages 223–225, 2007.

[5] R. E. Brown, R. Brown, E. Masanet, B. Nordman, B. Tschudi, A. Shehabi, J. Stanley, J. Koomey, D. Sartor, P. Chan, and E. I. U. E. P. A. et al. with Alliance to Save Energy, ICF Incorporated. *Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431.* 2007.

[6] Christensen et al. Ieee 802.3 az: the road to energy efficient ethernet. *IEEE Communications Magazine*, 48(11), 2010.

[7] Erickson et al. *Using Network Knowledge to Improve Workload Performance in Virtualized Data Centers.* PhD thesis, Stanford University, 2013.

[8] Gao et al. Energy-aware load balancing in heterogeneous cloud data centers. In *ACM ICMSS*, pages 80–84, 2017.

[9] Greenberg et al. The cost of a cloud: Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, 2008.

[10] M. Gupta et al. Using low-power modes for energy conservation in ethernet lans. In *IEEE INFOCOM*, pages 2451–2455, 2007.

[11] H. Jin et al. Joint host-network optimization for energy-efficient data center networking. In *IEEE IPDPS*, pages 623–634, 2013.

[12] K. Kant. Multistate power management of communications links. In *Proc. of COMSNET*, 01 2011.

[13] D. Li et al. Joint power optimization through vm placement and flow scheduling in data centers. In *IEEE IPCCC*, pages 1–8, 2014.

[14] Liu et al. Greencloud: A new architecture for green data center. In *ACM ICAC-INDST*, pages 29–38, 2009.

[15] P. Mahadevan et al. On energy efficiency for enterprise and data center networks. *IEEE Communications Magazine*, 49(8):94–100, 2011.

[16] Meng et al. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *IEEE INFOCOM*, pages 1154–1162, 2010.

[17] Mostowfi et al. A simulation study of energy-efficient ethernet with two modes of low-power operation. *IEEE Communications Letters*, 19(10):1702–1705, 2015.

[18] Mostowfi et al. An analytical model for the power consumption of dual-mode eee. *Electronics Letters*, 52(15):1308–1310, 2016.

[19] M. Murugan et al. On the interconnect energy efficiency of high end computing systems. *Sustainable Computing: Informatics and Systems*, April 2012.

[20] Nedevschi et al. Reducing network energy consumption via sleeping and rate-adaptation. In *USENIX NSDI*, pages 323–336, 2008.

[21] Paya et al. Energy-aware load balancing and application scaling for the cloud ecosystem. *IEEE Transactions on Cloud Computing*, 2015.

[22] L. R. Rabiner et al. An introduction to hidden markov models. *IEEE ASSp Magazine*, 1986.

[23] Raghavendra et al. No power struggles: Coordinated multi-level power management for the data center. In *ACM ASPLOS*, pages 48–59, 2008.

[24] Riley et al. The ns-3 network simulator. *Modeling and tools for network simulation*, pages 15–34, 2010.

[25] Sohan et al. Characterizing 10 gbps network interface energy consumption. In *IEEE LCN*, pages 268–271, 2010.

[26] Sondur et al. Implementing data center energy management capabilities in ns3. In publication.

[27] C. Thaenchaikun et al. Augmenting the energy-saving impact of ieee 802.3az via the control plane. In *IEEE ICCW*, pages 2843–2849, 2015.

[28] Travostino et al. Seamless live migration of virtual machines over the man/wan. *Future Gener. Comput. Syst.*, 22(8):901–907, 2006.

[29] Verma et al. pmapper: power and migration cost aware application placement in virtualized systems. In *ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264, 2008.

[30] X. Wang et al. Carpo: Correlation-aware power optimization in data center networks. In *IEEE INFOCOM*, pages 1125–1133, 2012.

[31] K. Zheng et al. Joint power optimization of data center network and servers with correlation analysis. In *IEEE INFOCOM*, pages 2598–2606, 2014.