Traffic Merging for Energy-Efficient Datacenter Networks

Alessandro Carrega
University of Genoa
Genoa, Italy
alessandro.carrega@unige.it

Suresh Singh
Portland State University
Portland, OR 97207
singh@cs.pdx.edu

Roberto Bruschi
National Inter-University Consortium for Telecommunications (CNIT)
Genoa, Italy
roberto.bruschi@cnit.it

Raffaele Bolla
University of Genoa
Genoa, Italy
raffaele.bolla@unige.it

***Abstract*—Numerous studies have shown that datacenter networks typically see loads of between 5% – 25% but the energy draw of these networks is equal to operating them at maximum load. In this paper, we propose a novel way to make these networks more *energy proportional* – that is, the energy draw scales with the network load.**

We propose the idea of *traffic aggregation*, in which low traffic from N links is combined together to create $H < N$ streams of high traffic. These streams are fed to H switch interfaces which run at maximum rate while the remaining interfaces are switched to the lowest possible one. We show that this merging can be accomplished with minimal latency and energy costs (less than 0.1W) while simultaneously allowing us a *deterministic* way of switching link rates between maximum and minimum. Using simulations based on previously developed traffic models, we show that 49% energy savings are obtained for 5% of the load while we get an energy savings of 22% for a 50% load. Hence, forasmuch as the packet losses are statistically insignificant, the results show that energy-proportional datacenter networks are indeed possible.

I. Introduction

The electricity consumption of datacenters has become a significant factor in the overall cost of these centers. As a result, there have been several recent studies that aim to reduce the energy consumption profile of servers and datacenter networks. Since the cooling costs scale as 1.3x of the total energy consumption of the datacenter hardware, reducing the energy consumption of the hardware will simultaneously lead to a linear decrease in cooling costs, as well. Today, the servers account for around 90% of the total energy costs, regardless of loading. However, since typical CPU utilization of server clusters is around $10 - 50\%$ [1], there are several efforts to scale the energy consumption of the servers with load. Indeed, it is expected that in the near future sophisticated algorithms will enable us to scale the energy consumption of the servers with load. When this happens, as noted in [1], the energy cost of the network will become a dominant factor. Hence, there is significant interest in reducing the energy consumption of the datacenter networks, as well.

Abts *et al.* [1], and Benson *et al.* [2] note that the average traffic per link in different datacenter networks tends to range between 5% and 25%. The authors in [1] implement a link rate adaptation scheme to save energy. Each link sets its rate every $10-100\,\mu s$ based on the traffic prediction. The energy savings are shown to be in the range $30-45\%$ for different workloads and loads less than 25%. However, the scheme suffers from the problem of packet losses due to inaccurate traffic prediction as well as significantly increased *latency*. Indeed, the mean increment in latency is between $30-70\mu s$ for different loading scenarios.

Previous studies attempt to reduce network-wide energy consumption by dynamically adapting the rate and the speed of links, routers and switches as well as by selecting routes in a way that reduces the total cost [3], [4]. In this respect, these green networking approaches have been based on numerous energy-related criteria applied to network equipment and component interfaces [3], [4]. These ideas tackle the minimization of the network power consumption by setting the link capacity to the actual traffic load.

In this paper, we present an innovative approach to adapt the energy consumption to load for datacenter networks. The key idea is to *merge* traffic from multiple links prior to feeding it to the switch. This simple strategy allows more switch interfaces to remain in a low power mode[1] while having a minimal impact on latency. We have explored the idea of traffic merging in depth in the context of enterprise networks in [5], [6], [7]; where we show that savings in excess of $60-70\%$ are obtained with no affect on traffic. Indeed, the big advantage of the merge network is that, unlike most other approaches, it works in the *analog domain*, so it does not introduce delays for the store-and-forward Layer 2 (L2) frames, rather it redirects such frames on-the-fly at Layer 1 (L1) between external and internal links of the merge network itself. In addition, the merge network allows reducing frequent link speed transitions due to the use of the low power mode. In our approach, such transitions happen only infrequently thus allowing us to minimize the delay due to the negotiation of the new link rate and the additional energy required for the rate transition.

In this paper, we apply the merge network concept to a datacenter network topology called Flattened Butterfly [1] [8]. Using extensive simulations we show that up to $22\% - 49\%$ energy savings are possible for loads between 50% and 5%, respectively. The choice to use the FBFLY topology is moti-

[1]The low power mode is realized by setting the link speed at the minimum rate.

vated by the fact that it is the best type of datacenter topology in terms of the energy consumption and we use it in our simulations. However, it is possible to consider other types of datacenter topologies, such as hypercube, torus, folded-Clos, etc. [9].

The rest of the paper is organized as follows. The next section describes the flattened butterfly network and in Section III we describe the traffic aggregation introducing the merge network . In Section IV we explain how the merge network is combined with the flattened butterfly topology. The subsequent section discusses our simulation methodology and results. Our conclusions are presented in Section VI.

II. Flattened Butterfly Topology

As outlined in [1], the flattened butterfly (FBFLY) topology is inherently more power efficient than other commonly proposed topologies for high-performance datacenter networks. It is proposed as a cornerstone for energy-proportional communication in large-scale clusters with 10,000 servers or more. In [1], the authors show why the topology, by itself, is lower in power than a comparable folded-Clos one (i.e. fat trees) [10].

The FBFLY k-ary n-flat topology [8] exploits high port count switches [11] to create a scalable low-diameter network, in order to reduce latency and network costs [12]. This is achieved using fewer physical links compared to a folded-Clos at the expense of increased routing complexity to load balance the available links. Indeed, unlike the simple routing in folded-Clos, the FBFLY topology requires a global-adaptive routing to load balance arbitrary traffic patterns. However, a folded-Clos has a cost that is nearly double that of a FBFLY with equal capacity. The reason for this higher cost is due to the presence of a double number of long cables in the network, which approximately doubles the cost. The FBFLY topology exploits high-radix switches to realize lower cost than a Clos on load balanced traffic, and provide better performance than a conventional butterfly.

A FBFLY topology is a multi-dimensional direct network like a torus (k-ary n-cube) [9]. Every switch in the network is connected to hosts as well as other switches. Unlike the torus, where each dimension is connected as a ring, in the FBFLY each dimension is fully connected. Hence, within a FBFLY dimension, all switches connect to all others.

An example of interconnection is shown in Fig. 1. It is a 2-dimensional FBFLY (8-ary 2-flat) with $8 \times 8 = 64$ nodes[2] and eight $7 + 8 = 15$-port switches (7 ports connected with the other switches and 8 ones connected with the nodes). The *concentration* c refers to the number of switch ports connected with the nodes.

Scaling the number of dimensions in a FBFLY consists, essentially, of taking this single 8-switch group, replicating it 8 times, and interconnecting each switch with its peer in the other 7 groups (i.e. the upper-left switch connects to the 7 upper-left switches in the other 7 groups). In this way, you have an 8-ary 3-flat with $8^2 = 64$ switches and $8 \times 8^2 = 512$ nodes each with $8 + 7 \times 2 = 22$ ports. So, the total number of switches for the FBFLY is given by:

$$S(n, k) = k^{n-1} \tag{1}$$

Instead, the number of ports for switch can be written as[3]:

$$P(n, k) = c + (k - 1) \times (n - 1) \tag{2}$$

and the total number of nodes can be expressed as follows:

$$N(n, k) = c \times k^{n-1} \tag{3}$$

Though a FBFLY scales exponentially with the number of dimensions, it is possible to scale by increasing the radix, too. Usually, it is advantageous to build the highest-radix, lowest dimension FBFLY that scales high enough and does not exceed the number of available switch ports. This reduces the number of hops a packet takes as well the number of links and switches in the system.

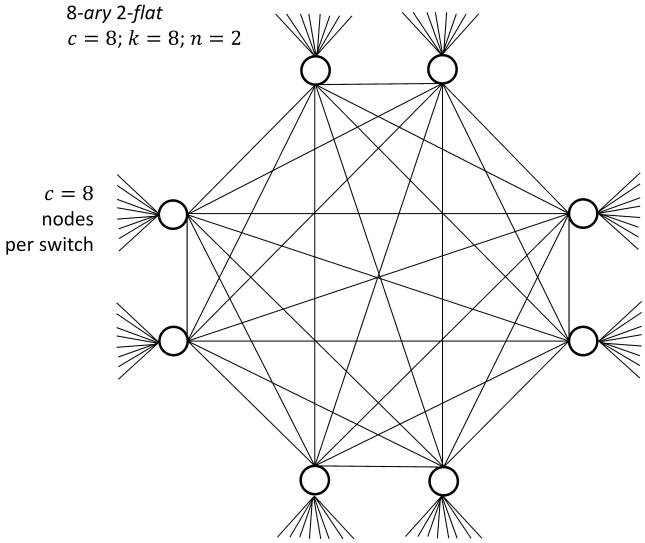


Fig. 1. Logical diagram of an 8-ary 2-flat flattened butterfly topology.

III. Traffic aggregation

In this paper, we propose the *merge network* [5] for energy-efficient datacenters. The merge network is based on the assumption of a low utilization of most ports in a switch. Hence, the approach is to merge traffic from multiple links and feeding the merged streams to a smaller set of active switch ports. As shown in Fig. 2, the traffic to/from N links are merged and fed to N interfaces. Setting the parameter H, according to the incoming traffic load, it is possible to reduce the number of active interfaces exactly to H.

For example, if the average traffic load on 8 links coming in to a switch (as in Fig. 1) is 10%, we could merge all the traffic onto one link and feed it to one switch port running at maximum rate, thus allowing the remaining ports to enter low

[2]We use node and host interchangeably in this paper.

[3]In this paper, we do not distinguish between the links that connect the switches and the ones that connect the hosts.

power mode. This solution is different from the approach in [1], where each link individually makes a decision of which rate to use every $10 - 100\ \mu s$ resulting in high latency and losses. Indeed, our approach results in almost optimal energy savings with minimal increase in latency (primarily due to the merge network).

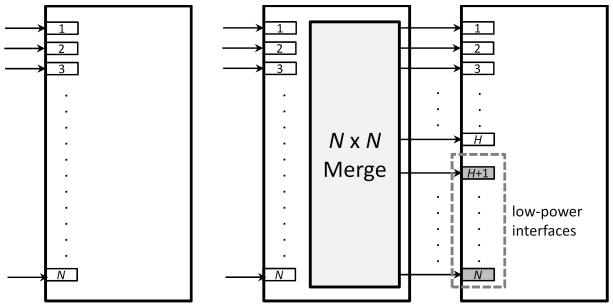
However, before evaluating the impact of merging on datacenter network traffic, we need to develop a better understanding of the merge network itself. A generic $N \times K$ (with $K \leq N$) merge is defined with the property that if at most K packets arrive on the N *uplinks* (i.e. from N links into the switch) then the K packets are sent on to K sequential ports (using some arbitrary numbering system). For example, we consider a 4×4 merge network as in Fig. 3. The incoming links from the hosts are identified with a – d and the switch ports with 1 – 4 . The traffic coming in from these links is merged such that the traffic is first sent to interface 1 but, if that is busy, it is sent to interface 2, and so on. In other words, we load the interfaces sequentially. This packing of packets ensures that many of the higher numbered interfaces will see no traffic at all, thus allowing them to go to the lowest rate all the time.

The key hardware component needed to implement this type of network is called a *selector*, whose logical operation is described in Fig. 4. There are 2 incoming links and 2 outgoing links. If a packet arrives only at one of the two incoming links, then it is always forwarded to the top outgoing link. However, if packets arrive along both incoming links, then the earlier arriving packet is sent out along the top outgoing link and the latter packet along the other one. The hardware implementation, described in [5], is done entirely in the *analog domain*. Thus, a packet is not received and transmitted in the digital sense; rather it is switched along different selectors in the network much as a train is switched on the railroad. *This ensures that the latency seen by a packet through the merge is minimal and the energy consumption is very small, as well.*

An important measure of the merge network is the complexity. We can define the complexity with two numbers: the depth of the network and the total number of selectors. The minimum depth of an $N \times K$ merge network is $\log_2 N + K - 1$ with the number of selectors needed equal to $\sum_{i=1}^{K} N - i$.

On the *downlink* (i.e. from the switch to the N links) the merge network has to be able to forward up to N packets simultaneously from any of the switch ports (connected to the K outputs of an $N \times K$ merge network) to any of the N downlinks. This network uses a simple implementation that consists of multiplexers. However, in order for this part to work correctly, we need to embed the control logic inside the switch because the packet header has to be parsed to determine which of the N links they must be send out [5].

Finally, the merge network requires a special software layer called port virtualization. The modern day switches support a large number of protocols. For instance, switches today support QoS (Quality of Service) and VLANs (IEEE 802.1P, 802.1H), port-based access control (IEEE 802.1X), and many other protocols. Hence, the solution is to build this software



(a) without merging (b) with merging

Fig. 2. Switch without (a) and with (b) the merge network.

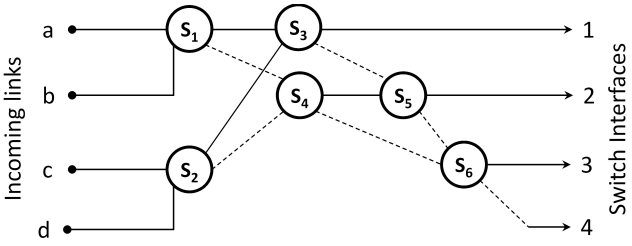


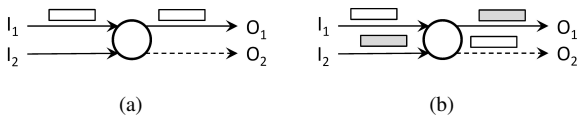
Fig. 3. A 4×4 uplink merge network.

layer within the switch. It maps packets coming on the uplink to one of the N virtual ports. Instead, on the downlink it schedules packets for transmission over one of the physical ports to appropriate downstream links. This mapping is needed to ensure that security protocols like 802.1X and VLANs work unchanged.

IV. Merge Network + Flattened Butterfly

We propose adding the merge network to the FBLY datacenter network of [1] in order to maximize energy savings. The manner in which we introduce the merge network into the FBLY is simple – we interpose the merge network between connections from the hosts to the switches. However, the connections between the switches are unchanged. In the context of the example from Fig. 1, we introduce eight 8×8 merge networks that are connected to the eight switches. Thus, the eight hosts that connect to a switch have their traffic routed through a merge network.

In order to save energy using the merge network, we need to run some number of switch interfaces at full rate while dropping the rate of the rest to the lowest possible. As noted in [1], a 40 Gbps interface can operate at 16 different rates with the lowest rate equal to 1.25 Gbps. The challenge is to run most of the links into the switch at the lowest rate (which consumes less than 40% of the power of the maximum rate



(a) (b)

Fig. 4. Operation of a selector.

link for InfiniBand switches [1], [13]), minimizing, at the same time, loss and latency. Since the merge network has the unique property of loading links sequentially, we know that, if link i is the highest numbered active link, then in the event of an increase in load (from a host) the next link that will need to run at full rate will be link $i+1$. This determinism in link loading gives us the key to maximizing energy savings. Specifically, the *algorithm* we use for changing link rates at switches is as follows:

1) if interfaces 1 to H are active (at full rate), then we increase the rate of the $H+1$th one to the full rate, as well. This is done to offset packet loss in the event of a burst of packets;
2) if at most $H-2$ interfaces of the H ones operating at the full rate are active, then we reduce the rate of the Hth interface to the lowest rate (after it goes idle).

This simple algorithm does not require any traffic prediction and ensures very low packet loss assuming that the time to change link rates is $1-10\,\mu s$ as in [1].

V. Evaluation

In order to demonstrate the usefulness and the effectiveness of the traffic aggregation inside a high-performance datacenter, we evaluate the merge network using the OMNeT++ discrete-event-driven network simulator. OMNeT++ is an open-source (and free for research and educational purposes) sophisticated system used for modeling communication networks, queuing networks, hardware architectures, and manufacturing and business processes [14].

We model an 8-ary 2-flat FBFLY (with a concentration $c=8$ and 64 nodes) with no over-subscription, so that every host can inject and receive at full line rate. Links have a maximum bandwidth of 40 Gbps. Switches are both input and output buffered. We model the merge traffic network and port virtualization in software using parameters from our prototype in [5]. For our simulations we use 8×8 merge networks.

In order to model the traffic in the network, we rely on several previous studies. The authors in [2] examine the characteristics of the packet-level communications inside different real datacenters including commercial cloud, private enterprise, and university campus datacenters. They note that the packet arrivals exhibit an ON/OFF pattern. The distribution of the packet inter-arrival time fits the Lognormal distribution during the OFF period. However, during the ON period, the distribution varies in different datacenters due to the various types of running applications. For example, MapReduce [15] will display different inter-switch traffic characteristics than typical university datacenters. Furthermore, traffic between nodes and switches displays patterns quite different from the inter-switch traffic [16] [17] [18]. The different traffic patterns fit typically one of Lognormal, Weibull and Exponential. We can consider the exponential distribution as the most restrictive one among the various identified distributions and we use it to represent the general distribution of the packet inter-arrival times. In order to obtain a comprehensive view of the benefits and challenges of using the merge network, we use

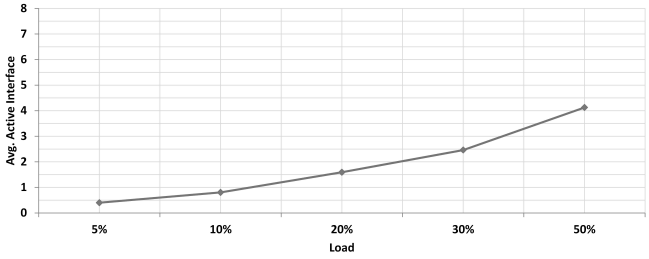


Fig. 5. Average number of active interfaces as function of the load.

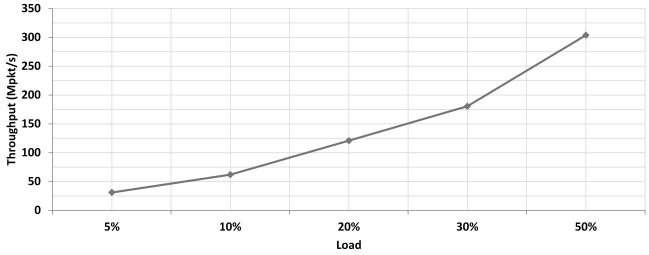


Fig. 6. Throughput for switch as function of the load.

different average traffic loads on each link. The values we use are: 5%, 10%, 20%, 30%, and 50% of the maximum link capacity of 40Gbps. The duration of each simulation is 24 hours. In addition, each run is repeated 10 times and the average performance values have been calculated and plotted.

The *metrics* of interest are: energy savings, packet loss due to merging traffic and aggregate throughput achieved. We note that the increased latency due to the merge network is $3\mu s$ (this is based on the time for the selectors in the merge network to sense the presence of packets and appropriately configure the network to switch the packet [5]).

A. Results

Fig. 5 plots the average number of active interfaces as a function of the average load. It is interesting to note that, even for a load of 50%, we see that an average of only 4 interfaces are active. We note that the packet losses are very small (statistically insignificant) and only occur during the time that an interface is being woken up. Fig. 6 shows the throughput of the switch in terms of processed packets per second. As we can see, the throughput scales with the load without influence of the merge network.

Let us now consider the energy savings obtained by using the merge network. As noted above, the maximum latency introduced by the merge network is $3\,\mu s$, which is far below that one reported in [1]. As described in [5], the energy consumption of the merge network is derived by simply extrapolating the energy cost of the selectors and multiplying that with the number of selectors needed plus a 10% increment to account for the cost of the control logic. Although, the number of selectors necessary to build a merge network grows linearly with increasing the number of output and input ports, its energy cost is very low and even for the largest merge

TABLE I
ENERGY SAVINGS USING A MERGE NETWORK.

Load	5%	10%	20%	30%	50%
Savings ($\rho_{es} - \%$)	49%	47%	41%	34%	22%

network it is below 0.1W. Therefore, we can effectively ignore the cost of the merge network in the overall energy calculation.

In order to compute the energy efficiency of our scheme, we rely on the energy analysis of [1]. As described there, a 40 Gbps InfiniBand link can operate at several lower rates as low as 1.25 Gbps. This is accomplished by exploiting the underlying hardware. Each link is composed of four lanes with its own chipset for transmitting and receiving. The chipset can be clocked at four different rates and thus we have 16 different possible rates on any link [1], [13]. The energy consumption of the lowest rate is 40% that of the maximum. In our system, the links are either operating at the maximum rate (the packets are being forwarded by the merge network to those ones) or at the minimum. Thus, we can very easily calculate the energy savings relative to the baseline, which is the case when all links operate at the maximum rate.

Using the data from Fig. 5, we obtain the energy savings for different loading patterns. Recall that when H interfaces are active, $H + 1$ interfaces are running at maximum rate while the remaining $N - H - 1$ are operating at the lowest rate. Table I provides the average energy savings calculated as:

$$\rho_{es} = 1 - \frac{(H+1) + 0.4(N-H-1)}{N} \quad (4)$$

These energy savings are greater than those obtained in [1] with only a minimal latency cost.

VI. CONCLUSIONS

This paper discusses the idea of merging traffic inside an energy-efficient datacenter. We consider the FBFLY topology because it is inherently more power efficient than other commonly proposed topologies for high-performance datacenter networks. Simulations with different configurations of traffic load are used to characterize and understand the effectiveness of the traffic aggregation. The results of these simulations show that it is possible to merge traffic inside a datacenter network in order to obtain $22 - 49\%$ energy savings. An important implication of this work is that the datacenters can be made very lean by using merge networks. Given the fact that the size of large-scale clusters is of the order of 10.000 nodes or more, this degree of energy savings has enormous global impact. In addition, in our current work, we are using the merge network architecture to replace high port density switches with lower port density switches, thus yielding even greater energy savings. Despite the positive results concerning energy saving, the proposed merge network solution is not proven to be optimal but we are studying that problem as part of future work. In addition, it would be interesting to test the merge network in other datacenters than the FBFLY and with real traffic traces.

REFERENCES

[1] D. Abts, M. Marty, P. Wells, P. Klausler, and H. Liu, "Energy Proportional Datacenter Networks," in *Proceedings of the 37th International Symposium on Computer Architecture (ISCA)*. Saint Malo, France: ACM, June 2010, pp. 338–347.

[2] T. Benson, A. Akella, and D. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proceedings of the 10th Conference on Internet Measurement (IMC)*. Melbourne, Australia: ACM, November 2010, pp. 267–280.

[3] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," *IEEE Communications Surveys & Tutorials (COMST)*, vol. 13, no. 2, pp. 223–244, Second Quarter 2011.

[4] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–14, 2011.

[5] C. Yiu and S. Singh, "Energy-Conserving Switch Architecture for LANs," in *Proceedings of the 47th IEEE International Conference on Communications (ICC)*. Kyoto, Japan: IEEE Press, June 2011, pp. 1–6.

[6] S. Singh and C. Yiu, "Putting the Cart Before the Horse: Merging Traffic for Energy Conservation," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 78–82, June 2011.

[7] C. Yiu and S. Singh, "Merging Traffic to Save Energy in the Enterprise," in *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking (e-Energy)*, New York, NY, USA, May–June 2011.

[8] J. Kim, W. Dally, and D. Abts, "Flattened Butterfly: a Cost-Efficient Topology for High-Radix Networks," in *Proceedings of the 34th International Symposium on Computer Architecture (ISCA)*. San Diego, CA, USA: ACM, June 2007, pp. 126–137.

[9] D. Abts and J. Kim, *High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities*. Morgan & Claypool Publishers, 2011.

[10] C. E. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transaction on Computers*, vol. 34, no. 10, pp. 892–901, October 1985.

[11] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow High-Radix Clos Network," in *Proceedings of the 33rd IEEE International Symposium on Computer Architecture (ISCA)*. Boston, MA, USA: IEEE Computer Society Press, May 2006, pp. 16–28.

[12] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[13] P. O.I., "An Introduction to the InfiniBand Architecture," in *Proceedings of the International Computer Measurement Group (CMG) Conference*, Reno, NV, USA, December 2002, pp. 425–432.

[14] OMNeT++ Network Simulation Framework. [Online]. Available: http://www.omnetpp.org/

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data processing on Large Clusters," *ACM Communications Magazine*, vol. 51, pp. 107–113, January 2008.

[16] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving Energy in Data Center Networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*. San Jose, CA, USA: USENIX Association, April 2010, p. 17.

[17] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power Awareness in Network Design and Routing," in *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM)*, Phoenix, AZ, USA, April 2008, pp. 457–465.

[18] C. Pan, T. amd Zhang, X. Guo, J. Jiang, R. Duan, C. Hu, and B. Liu, "Reducing the Traffic Manager Power in Routers via Dynamic On/Off-chip Packet Management," in *Proceedings of the 31st IEEE Conference on Computer Communications (INFOCOM)*, Orlando, FL, USA, March 2012.