Applying Traffic Merging to 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

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

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

ABSTRACT

The problem of reducing energy usage in datacenter networks is an important one. However, we would like to achieve this goal without compromising throughput and loss characteristics of these networks. 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. To this end we examine the problem of reducing the energy consumption of datacenter networks by merging traffic. The key idea is that low traffic from N links is merged together to create $K \leq N$ streams of high traffic. These streams are fed to K switch interfaces which run at maximum rate while the remaining interfaces are switched to the lowest possible rate. We show that this merging can be accomplished with minimal latency and energy costs (less than 0.1W total) while simultaneously allowing us a *deterministic* way of switching link rates between maximum and minimum. We examine the idea of traffic merging using three different datacenter networks – flattened butterfly, mesh and hypercube networks. In addition to analysis, we simulate these networks and utilizing previously developed traffic models we show that 49% energy savings are obtained for 5% per-link load while we get 20% savings for a 50% load for the flattened butterfly and somewhat lower savings are obtained for the other two networks. The packet losses are statistically insignificant and the maximum latency increase is less than 3μs. The results show that energy-proportional datacenter networks are indeed possible.

1. INTRODUCTION

The electricity consumption of datacenters is a significant contributor to the total cost of operation over the lifetime of these centers and as a result, there have been several studies that aim to reduce this cost. Since the cooling costs scale as 1.3x the total energy consumption of the datacenter hardware, reducing the energy consumption of the hardware will simultaneously lead to a linear reduction 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 underway to scale the energy consumption of the servers with load. It is expected that in the near future, sophisticated algorithms will enable us to scale the energy consumption of the servers linearly 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.

Various authors [1, 2, 3] note that the average traffic per link in different datacenter networks tends to range between 5% and 25%. To save energy, the authors in [1] implement a link rate adaptation scheme, whereby each link sets its rate every $10 - 100\ \mu s$ based on traffic prediction. The energy savings are shown to be $30 - 45\%$ for different workloads for 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 increase in latency is between $30 - 70\ \mu s$ for different loading scenarios. Other general approaches attempt to reduce network-wide energy consumption by dynamically adapting the rate and speed of links, routers and switches as well as by selecting routes in a way that reduces total cost [4, 5, 6]. In this respect, these green networking approaches have been based on numerous energy-related criteria, applied to network equipment and component interfaces [5, 6]. These approaches 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 energy consumption to load for datacenter networks. The key idea is to *merge* traffic from multiple links prior 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 [7, 8, 9], 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 the most other approaches, it works in the *analog domain*, so it does not introduce delays for store-and-forward Layer 2 (L2) frames,

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

e-Energy 2012, May 9-11 2012, Madrid, Spain.

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 three different datacenter network topologies – Flattened Butterfly [1, 10], Mesh and Hypercube [1, 11]. Using extensive simulations we then show that up to $20\% - 49\%$ energy savings are possible for loads between 50% and 5% respectively for the flattened butterfly and somewhat lower savings for the mesh and hypercube. The rest of the paper is organized as follows. The next section discusses the concept of traffic merging. The subsequent section describes the different datacenter network topologies we study and in Section 4 we present a theoretical model of energy savings when the merge network is applied to these topologies. Section 5 then presents our simulation methodology and results. Finally, our conclusions are presented in Section 6.

2. MERGE NETWORK

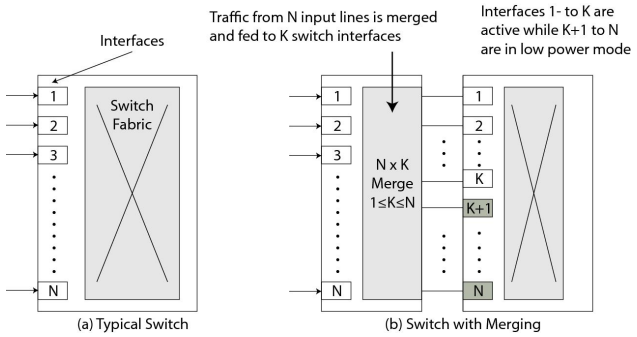

Figure 1: Switch without and with merge network.

The key idea we study is that of merging traffic arriving at a switch from multiple links and feeding that to few interfaces. The motivation for doing so is the observation made by various authors that per-link loading in datacenter networks tends to be well below 25% all the time and is frequently below 10% as well. Thus, by merging traffic we are allowing several of the switch interfaces to operate in low power modes. Indeed, as we discuss in [9] it is also possible to replace high port density switches with lower port density switches without affecting network performance in any way. Figure 1 illustrates the traffic to/from N links are merged and fed to K interfaces. Setting the parameter K according to the incoming traffic load allows us to reduce the number of active interfaces to K and enables $N - K$ interfaces to be in low power modes. As an example, if the average traffic load on 8 links coming in to a switch 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 power mode. This approach differs from the more traditional approaches as in IEEE 802.3az where each link makes decisions independently about when to enter low power states. Indeed, as we will show, our approach results in almost optimal energy savings with minimal increase in latency.

In order to understand how traffic merging can help in datacenter networks, we need to examine the details of the merge network itself. A generic $N \times K$ merge (with $K \leq N$) 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, consider a 4×4 merge network as in Figure 2. $a - d$ denote the incoming links (from hosts[2]) and $1 - 4$ denote the switch ports. The traffic coming in from these links is merged such that 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 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.

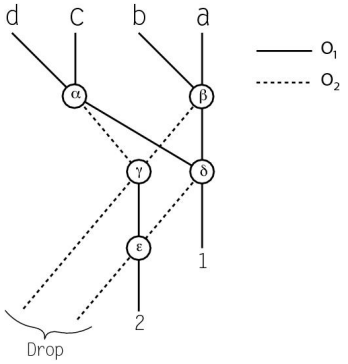

Figure 2: A 4×4 uplink merge network.

The key hardware component needed to implement this type of network is called *selector*, whose logical operation is described in Figure 3. 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 [7], 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* [3]. We have also shown previously [9] that 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 packets from any of the switch ports (connected to the K outputs of an $N \times K$ merge network) to any of the N downlinks and be able to forward up to N packets simultaneously. This network uses a simple implementation consisting of multiplexers since we

[2]We use host, node, end-host and endpoint interchangeably in this paper.

[3]In the the eyes of cost for manufacturing, the order is of ten dollars for large merge networks (if fabricated on chip). The primary cost comes from the multiplexers which may be in the thousands for a large merge network.

have to send packets from any of the K interfaces to any one of N links. 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 on [7]. In addition to this hardware, the merge network requires a software layer within the switch to ensure that the wide variety of LAN protocols continue working correctly (protocols such as VLANs IEEE 802.1P and 802.1H, access control IEEE 802.1X and many others). The needed software is essentially a port virtualization layer that maps K physical ports to N virtual ports in the switch. Thus, the protocol functionality is unaffected.

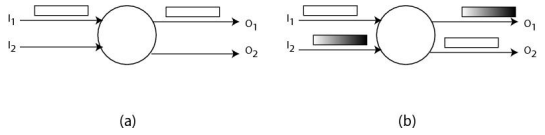

Figure 3: Operation of the selector.

3. DATACENTER NETWORK TOPOLOGIES

We study the application of our merge network to three datacenter network topologies in this paper – hypercube, mesh and flattened butterfly. Of these three topologies, the flattened butterfly has been shown to be inherently more power efficient than the mesh or hypercube as well as other commonly used topologies for high-performance datacenter networks. In particular, [1] shows why this topology is significantly more energy efficient than a comparable folded-Clos one (i.e. fat trees) [12, 13].

A flattened butterfly is a multi-dimensional direct network like a torus [11]. 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 flattened butterfly, each dimension is fully connected. Hence, within a flattened butterfly dimension, all switches connect to all others.

An example of interconnection is shown in Figure 4. It is a 2-dimensional flattened butterfly (8-ary 2-flat) with $8 \times 8 = 64$ nodes and eight $7 + 8 = 15$-port switches (7 ports to connect with the other switches and 8 ones to connect with the nodes). As shown in figure, the *concentration* c refers to the number of switch ports connected with the nodes. To increase the number of dimensions in a flattened butterfly we replace each of the 8 switches with a new full meshed group of 8 switches. Then we connect each switch with its peers 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, we get 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 flattened k-ary d-flat butterfly is given by:

$$s = k^{d-1} \tag{1}$$

where d is the dimension of the switch and k is the number of switches per dimension. Thus the number of ports for a switch is given by:

$$c + (k-1) \times (d-1) \tag{2}$$

where c is the number of end-hosts attached to each switch. Though a flattened butterfly scales exponentially with the number of dimensions, it is possible to scale by increasing the radix as well. Usually, it is advantageous to build the highest-radix, lowest dimension flattened butterfly 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.

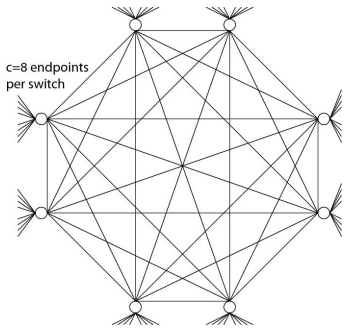

Figure 4: Logical diagram of an 8-ary 2-flat flattened butterfly topology.

Using the same notation as above for the hypercube and mesh we can write the total number of switches in a d-dimensional hypercube as $s = 2^d$ and the number of ports per switch as $c + d$. In the case of a d-dimensional mesh folded into a torus, the degree of each switch is $c + 2 \times d$. However, the total number of switches can be arbitrary. Let us assume that there are a total of n end-hosts that need to be connected via the network and as above assume that each switch is connected to c end-hosts. This gives us the total number of switches as $s = n/c$. In d dimensions, with an equal number of switches in each dimension, we obtain $\sqrt[d]{s}$ switches per dimension as a way to minimize diameter. Finally, from an energy computation stand-point the two parameters we are most interested in are the *number of switches that each switch is connected to* m and the *diameter* of the topology h_{max}[4]. These two parameters determine the throughput that a switch can support as well as the end-to-end time experienced by packets. Using the above discussion as a guide, we can summarize the values for these parameters in Table 1.

4. USING MERGE NETWORKS IN DATACENTERS

We propose adding the merge network to the datacenter networks in order to obtain energy savings. The manner in which we introduce the merge network into these networks is illustrated in Figure 5. We introduce a $c \times K_1$ merge between the c end-hosts connected to a switch and a separate $m \times K_2$ merge in the links connected to other switches. The reason for this separation is that the inter-switch links see more traffic and may well be higher bandwidth. Thus from a hardware standpoint we do need separate merge networks. The figure shows that switch ports from $K_1 + 1$ to c and from $K_2 + 1$ to m are in low power mode.

In order to save energy using the merge network, we need to run some number of switch interfaces at full rate while

[4]The worst case distance that a packet must traverse between any source and any destination is called *diameter* of the network and it is measured in hops. Hence, h_{max} refers to the max number of hops.

Parameters	Symbol	Datacenter Topology		
		Flattened Butterfly	Mesh	Hypercube
Number of switches connected to other switches	m	$(d-1) \times (\sqrt[d-1]{s} - 1)$	$2d$	d
Diameter	h_{max}	d	$\sqrt[d]{s}$	d

Table 1: Summary of key parameters. n − number of hosts, c − number of hosts/switch, $s = n/c$ − number of switches, d − dimension.

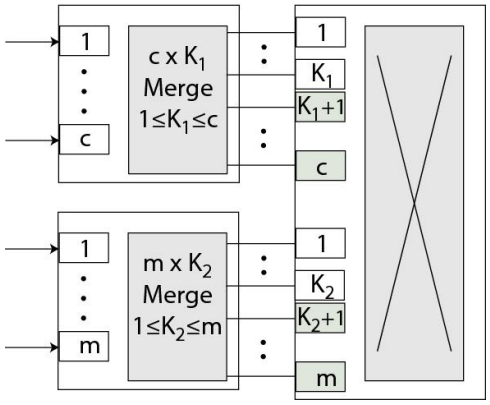

Figure 5: Merge networks in datacenter networks.

dropping the rate of the rest to the lowest possible. In other words, we need to *dynamically* determine the values of K_1 and K_2.

The merge network has the unique property that *links are loaded sequentially*. Thus, if link i is the highest numbered active link, then in the event of an increase in load (from any or all of the hosts) 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 i are active (at full rate), where i is K_1 or K_2, then we increase the rate of the $i+1$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 $i-2$ interfaces of the i ones operating at the full rate are active, then we reduce the rate of the ith 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 the time to change link rates is $1 - 10\,\mu s$ as in [1].

4.1 Estimate of Energy Savings

Let us assume that the combined average load per link to and from an end-host is μ. Then, the number of interfaces of the switch connected to the end-hosts that will be in active mode is well approximated as,

$$K_1 = \mu \times c + 1$$

Similarly, since the total load in and out of the interfaces connected to the other switches is $\mu \times c$, the number of active interfaces K_2 is,

$$K_2 = \frac{\mu \times c}{m} \times m + 1 = \mu \times c + 1$$

Thus, we have a total of $K_1 + K_2 = 2(\mu \times c + 1)$ active interfaces out of $c + m$.

When an interface is put into low power mode, it does continue to consume energy. For example, as noted in [1], a 40 Gbps Infiniband interface can operate at 16 different rates with the lowest rate being 1.25 Gbps. The lowest rate consumes 40% of the energy of the highest rate. Thus, in computing the energy savings, we need to consider this factor as well. Let us assume that a low power interface consumes a fraction α of a fully active interface. Then, we can write the energy savings in the interfaces when using the merge network as,

$$\begin{aligned} \rho_{es} &= E[\text{energy savings in interfaces}] \\ &= 1 - \frac{(K_1+K_2)+\alpha(m+c-(K_1+K_2))}{m+c} \\ &= (1-\alpha)\left[1 - \frac{K_1+K_2}{m+c}\right] \\ &= (1-\alpha)\left[1 - \frac{2(\mu \times c+1)}{m+c}\right] \end{aligned} \quad (3)$$

From Eq. 3 it is clear that to maximize energy savings we need the largest m possible. However, this observation is incorrect because we have ignored the cost of the switch chassis and other components. Indeed, as the number of interfaces of a switch increases beyond some value, we need to replace it with a different chassis that can support more interfaces. In other words, the true energy savings may be written as,

$$E[\text{overall energy savings}] = \frac{\mathcal{C}(c,m)+\mathcal{I}(\alpha(m+c)+(1-\alpha)(K_1+K_2))}{\mathcal{C}(c,m)+\mathcal{I}(m+c)} \quad (4)$$

where $\mathcal{C}(c,m)$ is the cost of the switch chassis supporting c end-hosts and m inter-switch links and $\mathcal{I}$ is the per-interface energy cost.

4.2 Selecting Topology Parameters

Based on Eq. 4 and Table 1 we have a simple algorithm for selecting the value of m and hence the dimension of the network topology. For a given concentration c and maximum anticipated switch throughput we select the smallest

switch and utilize its configuration that maximizes the number of interfaces. This is done because typically the marginal cost of interfaces is much smaller than the cost of the switch itself and thus it makes sense to maximize the interfaces supported. Having done this, we assign c interfaces to the end-hosts and the remaining interfaces are connected to other switches giving us m. Of course, the potential drawback of being too aggressive in switch selection is that m may be too small resulting in greater network diameter and hence latency. Therefore, the particular switches selected need to be determined also based on other network design considerations. *In this paper we only concern ourselves with interface energy savings given a already defined datacenter network topology. Thus, for the remainder of the paper we only concern ourselves with Eq. 3.*

Figures 6 and 7 show the trend of the average interface energy savings (Eq. 3) as a function of load for varying the values of m and α. It is interesting to note that with a load less than 10% the energy saving is never less than 40% for every possible configuration. Even with a load of 30% we are able to achieve an energy savings of more than 15%.

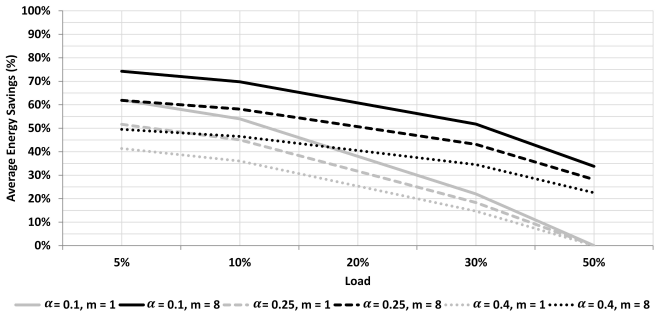

Figure 6: Average energy savings as function of load and different values of α and m.

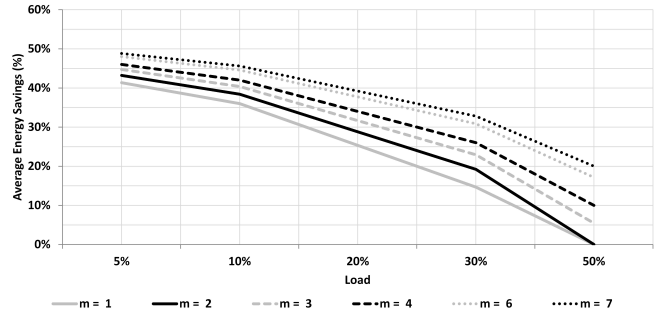

Figure 7: Average energy savings as function of load and different values of m with $\alpha = 0.4$.

5. EVALUATION

In order to demonstrate the usefulness and the effectiveness of 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, queueing networks, hardware architectures, and manufacturing and business processes [14].

For our simulation, we model different datacenter topologies: flattened butterfly, mesh and hypercube. For the flattened butterfly, we consider an 8-ary 2-flat one. For the mesh and hypercube topologies, we examine the (4,2)-ary 2 dimensional (2-D) and 8-ary 1 dimensional (1-D) cases. Hence, in each considered topology the concentration c is equal to 8 and the number of nodes is 64. We don't use 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 [7] for reference. 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 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 [3, 16, 17]. Typically, however, the different traffic patterns fit 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 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, aggregate throughput achieved and end-to-end (*e2et*) time. We note that the increased latency due to the merge network is 3 μs (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, please see [7]).

5.1 Results

Figure 8 plots the average number of asleep interfaces for the different datacenter topologies and configurations. The flattened butterfly topologies shows the best results with a 50% of asleep interfaces with 50% of load. However, the other topologies show poor performance when the load approaches 50%. In particular the 1-D configuration for the mesh and hypercube topologies have a very small percentage of asleep interfaces compared to that of 2-D configurations. In the 2-D case, the hypercube topology has slightly better results than the mesh one. In summary, the flattened butterfly topology has the best features to achieve significant energy savings due to a greater number of links connected to other switches m, that can be put in a low power mode.

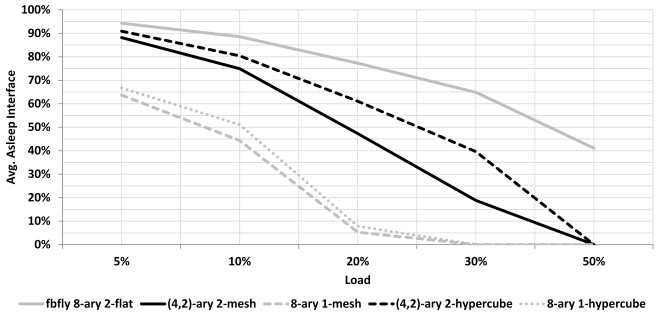

Figure 8: Average number of asleep interfaces as function of load for the 8-ary 2-flat flattened butterfly, (4,2)-ary 2-D, 8-ary 1-D mesh and (4,2)-ary 2-D, 8-ary 1-D hypercube topologies.

Figure 9 plots the average number of active interfaces as function of the average load. In our simulation the number of active interfaces is the same for the 8-ary 2-flat flattened butterfly (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube topologies. In addition, it is interesting to note that, even for a load of 50%, we see that, on average, only 4 interfaces are active. We note that the losses seen are very small (statistically insignificant) and only occur during the time that an interface is being woken up.

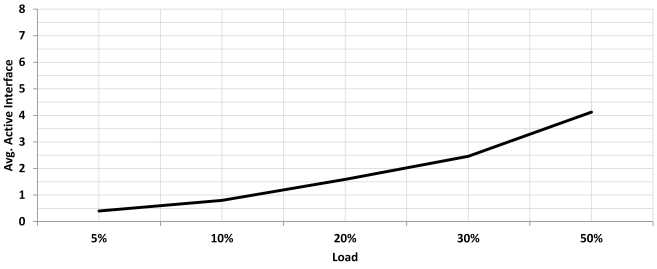

Figure 9: Average number of active interfaces as function of load for the 8-ary 2-flat flattened butterfly, (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube topologies (the trend is very similar for all the three cases)

Figures 10 and 11 show the number of active interfaces for the mesh and hypercube topologies. For the mesh topology, Figure 10 shows that the average number of active interfaces is less in (4,2)-ary 2-D configuration compared to the 1-D case. Similar observations can be made for the hypercube topology. Figure 11 shows that the (4,2)-ary 2-hypercube has the lowest value of the average number of active interfaces.

Figure 12 shows the throughput of the switch in terms of processed packets per second. As we can see, the throughput scales with load without influence of the merge network. Also in this case, the trend is identical for the 8-ary 2-flat flattened butterfly (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube topologies. For clarity, Figure 12 shows only one curve for the throughput.

The throughput for the mesh and hypercube topologies have slightly different values depending on the configuration. Figures 13 and 14 show that the (4,2)-ary 2-D configuration

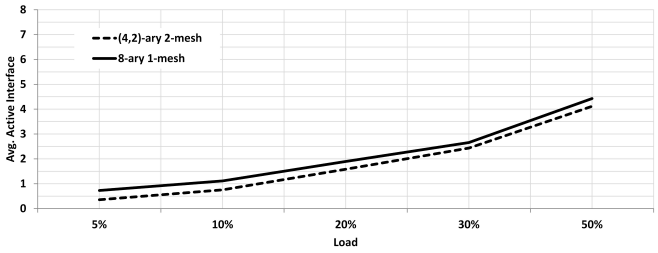

Figure 10: Average number of active interfaces as function of load for the (4,2)-ary 2-mesh and 8-ary 1-mesh topology.

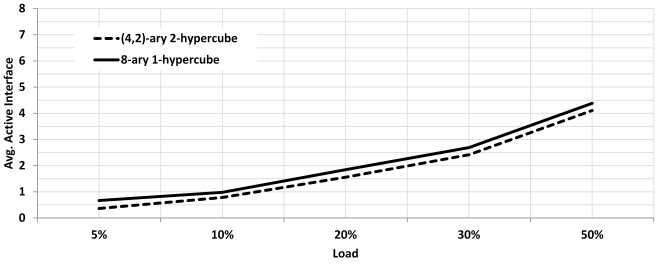

Figure 11: Average number of active interfaces as function of load for the (4,2)-ary 2-hypercube and 8-ary 1-hypercube topology.

has a lower throughput than the 8-ary 1-D one. The value of higher throughput for the 1-D case depends on the greater average number of hops required to reach the destination node.

The last metric we evaluated is the end-to-end time (*e2et*). Figure 15 shows the *e2et* for the 8-ary 2-flat flattened butterfly topology. It is very easily seen that most of the delay is due to the latency introduced by the merge network. However, this latency is additional and it occurs only at the beginning of the path when the packet arrives at the first switch (i.e. the first hop). Hence, the latency introduced by the merge network is not affected by the load increase or by the throughput of the nodes and switches.

The flattened butterfly topology has a smaller value of *e2et* than the mesh and hypercube ones. It is interesting to see that in the mesh topology (4,2)-ary 2-D configuration it becomes necessary to distinguish two different types of nodes: inner and outer. Figure 16 shows the trend of *e2et* for these two categories of nodes. The greatest delay is for the inner nodes that are also characterized by a higher throughput. Instead, for the 8-ary 1-D configuration, as shown in Figure 17, we have 4 different types of nodes: a, b, c and d. The nodes of the a type are the outer nodes, while the ones of the d type are the inner ones and b and c are the middle ones. Also in this case, the *e2et* has higher values in the nodes at the center of datacenter. Finally, the 2-D configuration of the mesh topology has values of *e2et* smaller than that of the 1-D case as shown in Figures 16 and 17. Also the 2-D configuration of the hypercube topology presents values of *e2et* lower than than the 1-D case (see Figure 18).

Let us now consider the energy savings obtained by using the merge network. As noted above, the maximum latency

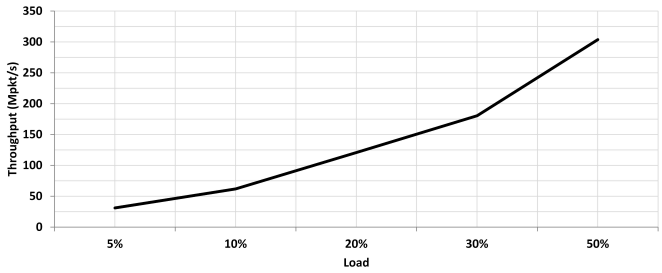

Figure 12: Throughput for switch as function of load for the 8-ary 2-flat flattened butterfly, (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube topologies (the trend is very similar for all the three cases).

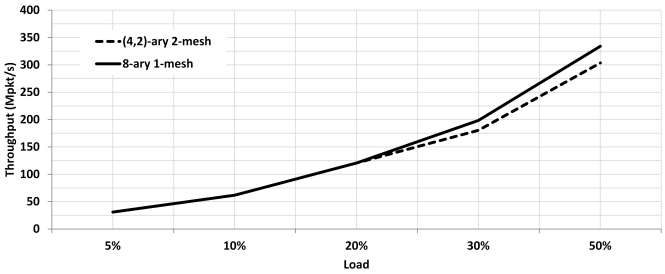

Figure 13: Throughput for switch as function of load for the (4,2)-ary 2-mesh and 8-ary 1-mesh topology.

introduced by the merge network is 3 μs, which is far below that one reported in [1]. As described in [7], 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 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 network it is below 0.1W. Therefore, we can effectively ignore the cost of the merge network in the overall energy calculation.

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, 18]. 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 (those ones that the packets are being forwarded by the merge network) 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 Figure 9 and Eq. 3, we obtain the energy savings for different loading patterns and the different datacenter topologies. Figure 19 plots the energy savings as function of the average load for the the 8-ary 2-flat flattened butterfly, (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube

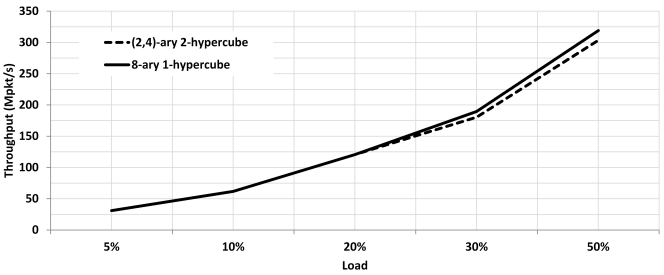

Figure 14: Throughput for switch as function of load for the (4,2)-ary 2-hypercube and 8-ary 1-hypercube topology.

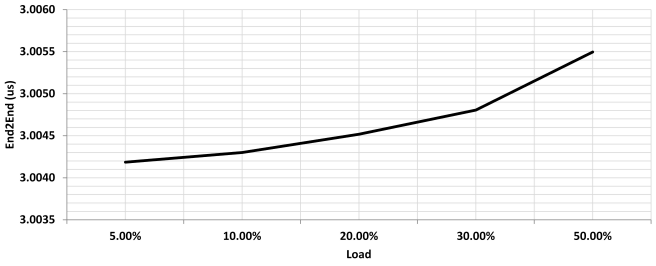

Figure 15: End-to-end time as function of load for the 8-ary 2-flat flattened butterfly topology.

topologies. As discussed in the previous sections the flattened butterfly topologies has the best results with an energy savings nearly 20% with a load of 50%. Table 2 summarizes this results in numeric format.

6. CONCLUSIONS

The paper studies the idea of traffic merging in datacenter networks. Earlier work by the author developed the notion of using an analog merge network to aggregate traffic from multiple links in an enterprise network and feed it to a much smaller port-density switch. The current paper extends the idea to datacenter networks where traffic merging is shown to enable large number of switch interfaces to operate in low power modes while having no impact on traffic. The paper explores the application of traffic merging to the flattened butterfly, mesh and hypercube networks. It is shown that the flattened butterfly yields almost 20% energy savings even under 50% loading while at 5% load it shows an almost

Table 2: Average energy savings (ρ_{es} – %) using a merge network.

Load	5%	10%	20%	30%	50%
8-ary 2-flat flattened butterfly	49%	46%	39%	33%	20%
(4,2)-ary 2-mesh	46%	42%	34%	26%	10%
(4,2)-ary 2-hypercube	45%	40%	32%	23%	5%

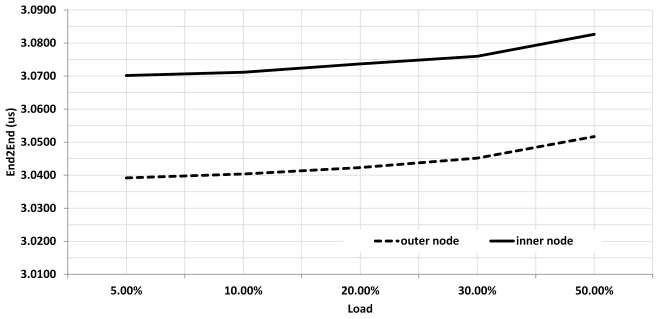

Figure 16: End-to-end time as function of load for the (4,2)-ary 2-mesh topology.

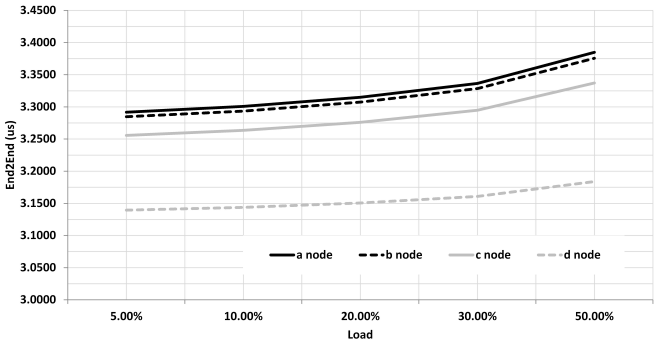

Figure 17: End-to-end time as function of load for the 8-ary 1-mesh topology.

50% energy savings. The mesh and hypercube networks also show energy savings at all loads but are not as energy efficient as the flattened butterfly. The paper also develops a theoretical model for energy savings for these networks.

7. ACKNOWLEDGEMENTS

This work has been supported by the ECONET (low Energy Consumption NETworks) project, funded by the European Commission under the 7th Framework Programme (FP7) [19] and by the EFFICIENT (Energy eFFIcient teChnologIEs for the Networks of Tomorrow) PRIN project, funded by MIUR (the Italian Ministry of University and Research) [20].

8. 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] 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.

[4] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh, "The Potential Impact of Green Technologies in Next-Generation Wireline Networks: Is There Room for Energy Saving Optimization? ," *IEEE Communications Magazine*, vol. 49, no. 8, pp. 80–86, August 2011.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

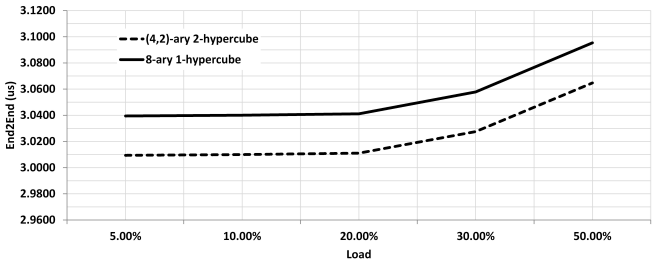

Figure 18: End-to-end time as function of load for the (4,2)-ary 2-hypercube and 8-ary 1-hypercube topology.

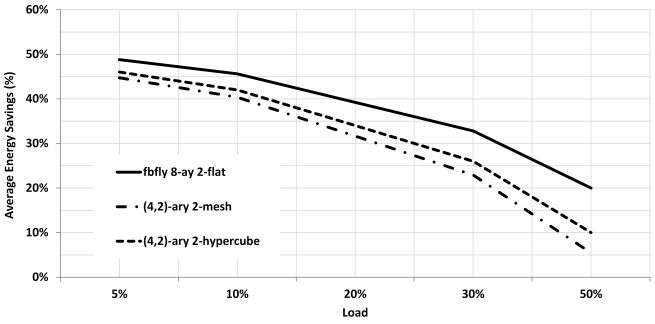

Figure 19: Average energy savings ρ_{es} for the 8-ary 2-flat flattened butterfly, (4,2)-ary 2-mesh and (4,2)-ary 2-hypercube topologies.

[9] 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.

[10] 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.

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

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

[13] R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," in *Proceedings of the Annual ACM Conference of Special Interest Group on Data Communication (SIGCOMM)*. Barcelona, Spain: ACM, August 2009, pp. 39–50.

[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] 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.

[17] 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 2011, submitted.

[18] 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.

[19] low Energy Consumption NETworks (ECONET) Project. [Online]. Available: http://www.econet-project.eu

[20] Energy eFFIcient teChnologIEs for the Networks of Tomorrow (EFFICIENT) Project. [Online]. Available: http://www.tnt.dist.unige.it/efficient/