A Universal Approach to Data Center Network Design

Aditya Akella
U of Wisconsin - Madison
akella@cs.wisc.edu

Theophilus Benson
Duke University
tbenson@cs.duke.edu

Bala Chandrasekaran
Duke University
balac@cs.duke.edu

Cheng Huang
Microsoft
cheng.huang@microsoft.com

Bruce Maggs
Duke University/Akamai
bmm@cs.duke.edu

David Maltz
Microsoft
dmaltz@microsoft.com

ABSTRACT

This paper proposes an approach to the design of large-scale general-purpose data center networks based on the notions of volume and area universality introduced by Leiserson in the 1980's in the context of VLSI design. In particular, we suggest that the principle goal of the network designer should be to build a single network that is provably competitive, for any application, with any network that can be built for the same amount of money. After describing our approach, we survey the technology choices available to network designers today, and examine several existing commercial data center networks. In the most recent of these networks resources are allocated roughly as we suggest in this paper.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology, Packet-switching networks

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

By the early 1980s dozens, if not hundreds, of designs of interconnection networks had been proposed for use in parallel computers. (See [22] for the seminal text on this subject.) These networks could be compared on the basis of diameter, bisection width, maximum degree, etc., but there was no consensus on which network was "best" or most general purpose, and parallel computers were built around a variety of designs.

In 1985, however, Leiserson suggested that, at least in the "network on a chip" context, where cost equates to VLSI layout area, a fair comparison between two different classes of networks can be made by comparing instances of the two classes that occupy the same amount of area. He then proposed a class of hierarchical networks called fat-trees that allows a network designer to choose how much bandwidth to allocate at each level of the hierarchy. Furthermore, he demonstrated that one specific allocation strategy leads to a fat-tree that is *area universal* in the sense that, for a given amount of layout area, n, an area-universal fat-tree of area n can emulate any other network that can be laid out in area n with slowdown at most logarithmic in n [5, 24, 25]. One interpretation of this theorem is that although it may be possible to build a special-purpose network to solve a specific problem quickly, the improvement in performance over that achievable by an area-universal network built at the same cost is limited.

Over the past few years, interest in data center network design has burgeoned in the networking research community, and we now find ourselves in a situation somewhat analgous to that facing the parallel computing community in the 1980's. Researchers have proposed disparate designs based on cliques of cliques [20], meshes of stars [19], expander graphs [32], fat-trees [2, 30], and Clos networks [14, 16]. As before, there is no consensus on which of these networks is best.

This paper does not advocate a specific network for use in data centers, but instead promotes a methodology for designing networks that are "cost universal". A network is cost universal if it can emulate any other network that could be built at the same cost with limited slowdown. The notion of cost universalilty is a generalization of area universality in the following sense. The crux of Leiserson's proof that certain fat-trees are area universal is to allot an equal amount of layout area to the links at each level of the hierarchy. (A short tutorial on area-universal fat-trees is given in Section 2). The analog in the data center context, where there is no convenient homogeneous medium such as layout area, is to expend the same amount of money on each level of the hierarchy, and this is the gist of our approach. For example, we recommend that the designer spend the same amount of money on the networking connectivity within a rack (summed over all racks), on connecting racks in a row, and on connecting rows of racks.

Our approach results in the constuction of hierarchical networks, which, when provisioned properly, should be competitive with any networks that could be built at the same cost. We do not rule out the possibility that a non-hierarchical network could also be cost universal, but we do not know of any such examples. In any case, the commercial networking technology available today is well suited to hierarchical design, and hierarchical networks have the advantages of simplicity and modularity.

After a short tutorial on fat-trees and Clos networks, we survey several influential recent proposals for data center networks, analyzing how bandwidth, and hence cost, is allocated to the various levels of these networks. Then we provide an overview of the theory of area and cost universality. We use fat-trees as our examples of uni-

ICDCN '15 Jan. 4-7, Goa, India

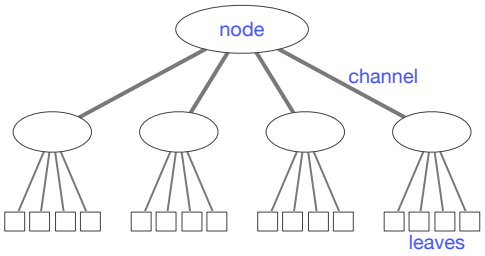

Figure 1: The coarse structure of a fat-tree could, for example, be a complete 4-ary tree.

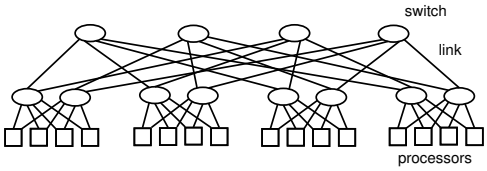

Figure 2: One possible fine structure, called a butterfly fat-tree, where the coarse structure is a complete 4-ary tree. In this example, the capacities of the channels have been to chosen to make the network area universal.

versal networks, but other hierarchical networks might suffice. The fat-trees, however, are proportioned very differently than those described in recent papers on data center network design, where the cost of the top level of the hierarchy dominates the costs of the other levels. We then examine several modern data center networks, and observe that, whether consciously or not, the designers roughly followed our recommendation of spending the same amount of money on each level of the hierarchy.

2. NETWORK STRUCTURES

This section begins with a brief primer on fat-tree networks. It also examines the relationships between fat-trees and other networks such as folded Clos networks, and explains how expanders can be incorporated into fat-trees.

Fat-trees were introduced by Leiserson [25] in 1985. A fat-tree has a *coarse structure* and a *fine structure*. The coarse structure is a rooted tree, consisting of nodes and bidirectional *channels* between parents and children. The degrees of the nodes can be chosen by the fat-tree designer. An example coarse structure for a fat-tree, a complete 4-ary tree, is shown in Figure 1.

The fine structure provides the details necessary to implement the fat-tree. For each leaf in the coarse structure there is a processor, server, or other device in the fine structure. For each internal node in the coarse structure, there is a corresponding set of switches in the fine structure. The number of switches per node, and the number of ports per switch, are chosen by the designer. For each channel in the coarse structure, there is a corresponding set of links connecting ports on switches at the parent to ports on switches at the child. How these ports are connected is again up to the designer. The capacity (bandwidth) allocated to each channel is determined by the number of links, which in turn depends on the number of switches and ports at the parent and child nodes. Leiserson's original paper [25] contemplated links between switches at the same node, but in practice such links have not been used. One possible fine structure for a complete 4-ary fat-tree is shown in Figure 2.

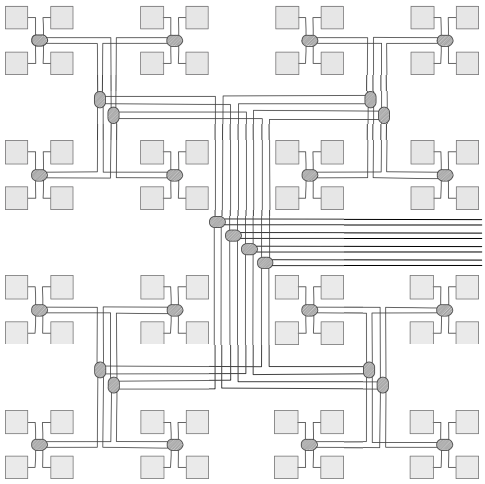

Figure 3: A VLSI layout of an area-universal fat-tree.

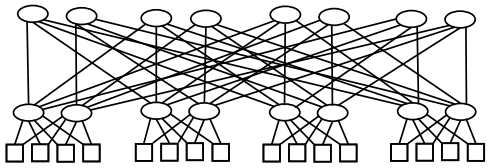

Figure 4: A full 4-ary butterfly fat-tree.

2.1 Butterfly fat-trees

In [18], Greenberg and Leiserson introduced a fine structure simpler than those in [25], which Greenberg calls a *butterfly fat-tree* in his thesis [17]. Today the term fat-tree almost always refers to a butterfly fat-tree. Figure 2 depicts a butterfly fat-tree whose channel capacities (doubling every level in a 4-ary fat-tree) have been chosen so that the network is area universal.

In a butterfly fat-tree, every link connects a port on a switch at a parent node to a port on a switch at a child node (i.e., there are no links between switches at the same tree node). Furthermore, there is at most one link between any two switches, and each switch at a parent node is connected to exactly one switch at each of the children. A butterfly fat-tree is not actually a tree (it contains cycles!), but its coarse structure is a tree.

Figure 3 shows a VLSI layout of the fine structure shown in Figure 2. This way laying out a tree is called an *H-tree* layout [29]. In the figure, the square leaf nodes denote processors, while the ovals denote switches. Note that although the number of links at the root (8) is only the square root of the number of processors (64), these links occupy a significant fraction of the layout area.

In general, switches with any number of children or parents can be used to construct a butterfly fat-tree. Typically only a single type of switch is used, but, for example, an area-universal binary fat-tree could be constructed in which each switch has two links to child switches, but on alternating levels, each switch either has a single link to a parent switch, or two links to parent switches. In general,

the number of parents per switch can be varied from level to level in order to control the rate at which bandwidth is allocated among the levels. At the top level of a fat-tree, the number of switches can be reduced because there are no parent switches to connect to. In particular, ports that would otherwise be used to connect to parent switches can instead be used to provide additional connections to children. It would be unusual to use a switch with more parents than children, because the children would not be able to send or receive enough traffic to saturate the connections to the parents. In a fat-tree built out of commodity Ethernet switches or routers, each switch would typically have a large number of children and parents.

The special case in which every switch in a butterfly fat-tree has as many parents as children, shown in Figure 4, is called a *full* fat-tree [11, 12]. Full fat-trees are not area or volume universal, and would likely be prohibitively expensive to build on a large scale due to their large bisection bandwidths. In particular, in a full fat-tree with N leaves, the amount of layout area required for the different levels of the tree vary greatly. For example, the wires at the root require $\Theta(N^2)$ area, whereas the wires at the leaves require only $\Theta(N)$ area. Confusingly, some authors depict fat-trees as full fat-trees without pointing out that the full fat-tree is a special case, and many experimental network simulations are carried out on full fat-trees without justifying the choice of this special case.

A defining feature of the butterfly fat-tree is the algorithm for finding a path from one leaf to another. The path first proceeds upwards in the tree, following parent links. Upon reaching a switch at the lowest common ancestor of the two leaves (in the coarse structure of the tree), the path then proceeds down child links to the destination leaf. On the upward part of the path, any parent switch can be chosen. A common strategy for load balancing is to select a parent at random, which is precisely the technique suggested by Valiant and Brebner [34] (and somewhat unfairly known as "Valiant load balancing" today). On the way down from the lowest common ancestor, there is a unique path to the destination leaf.

2.2 Redundant downard paths in fat-trees

One defininition of an *expander* is an N-node graph with the property that any set of $k \leq \alpha N$ nodes is connected to at least βk other nodes, for some fixed constants $0 < \alpha < 1$ and $\beta > 0$. An $N \times N$ bipartite graph is an expander if any set of $k \leq \alpha N$ nodes on the left side is connected to a set of at least βk nodes on the right side, for $\beta > 1$. With nonzero probability, a bipartite graph in which each node on the left has at least three neighbors, and those neighbors are chosen at random, is an expander. Explicit constructions are also known.

Unless α and β are very small, an expander will have high bisection bandwidth. Hence, building a large-scale network that exhibits a global expansion property is challenging. Adding to the challenge, a randomly-wired expander also exhibits an irregular wiring pattern. The use of expanders as components within structured networks, however, has proven useful in theory. For example, the AKS sorting network [1], which can sort any set of N keys using $O(\log N)$ levels of comparators, incorporates expanders, as does the N-input multibutterfly [33] network, which has a deterministic routing algorithm that can route any permutation of N packets in $O(\log N)$ steps, and is highly resilient to faults [23]. The nonblocking multi-Beneš network [3], which is based on the multibutterfly can establish a set of circuits among N inputs and N outputs in any permutation using $O(N \log N)$ crosspoints.

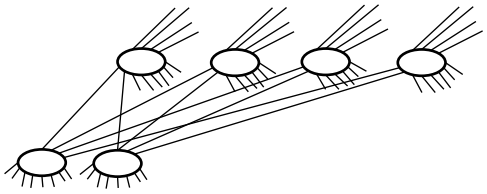

Figure 5: A channel in a 4-ary multi-fat-tree. Each switch has four links to parents and eight links to children, the proper ratio for an area universal 4-ary fat-tree. The four parent switches at the top of the channel each connect to both of the child switches at the bottom, providing redundant paths from the root to the leaves.

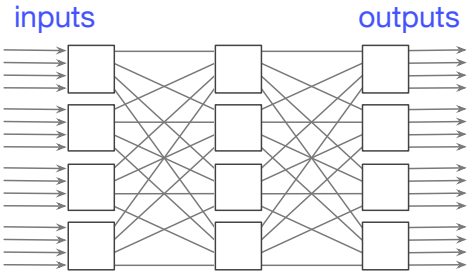

Figure 6: A 3-stage, 16-input Clos network composed of 4×4 crossbar switches.

Expanders can be incorporated into fat-trees in a straightforward way. In particular, each channel, which forms a bipartite graph between a child and a parent (and vice versa) in the coarse structure of the fat-tree can be wired to form a bipartite expander. Such a network is known as a multi-fat-tree [15]. (The first paper on fat-trees [25] incorporated a closely related graph structure called a concentrator.)

Multi-fat-trees have several advantages over fat-trees. In a butterfly fat-tree, the number of children of each switch is equal to the arity of the tree. I.e., in a binary fat-tree, each switch has two children, one in each subtree of the coarse structure. Hence, although there are multiple paths up in a fat-tree, there is only a single path down to any particular leaf. In contrast, in a multi-fat-tree, a switch may have multiple children in the same subtree, as shown in Figure 5. With multiple paths both up and down the tree, multi-fat-trees are more resilient to faults than fat-trees. The expansion in the channels of a multi-fat-tree can also be exploited by adaptive load balancing techniques. There is a cost for incorporating redundant links: the size of each switch may grow by a constant factor (e.g., two), as will the bandwidth requirements of each channel, and the VLSI layout area or volume. Nevertheless, it is possible to construct area-, volume-, and cost-universal mult-fat-trees.

2.3 Clos networks

Clos networks [13] were designed for use in telephone systems where the goal is to establish a set of disjoint circuits between a set of N input terminals (inputs) and a set of N output terminals (outputs) according to an arbitrary permutation. As shown in Figure 6, a Clos network is composed of crossbar switches, which are organized into multiple stages, the switches in each stage connecting to those of the next stage. The Beneš network [7], which has $O(N \log N)$ crosspoints and can implement any permutation and

hence is called *rearrangeable*, is a popular example of a Clos network. In a Clos network, the path from every input to every output passes through all stages, and hence all paths have the same length. Because the goal of Clos networks is to establish disjoint circuits in permutations, the number of links between successive stages is typically at least N. Hence, Clos networks are full-bandwidth networks, making them expensive and cumbersome to build on a large scale.

Sometimes the corresponding inputs and outputs of a Clos network are identified as one and the same. I.e., rather than having an ith input terminal and an ith output terminal, there is a single, bidirectional, ith terminal connected to switches on both the first and last stages. Such a network was originally called a folded Clos network [9], but today the term folded is used differently (e.g., in [31]), as explained below. In this configuration, it remains the case that the path between any two terminals passes through all stages of the Clos network.

A Clos network can also be "folded" around its middle stage so that corresponding switches on stages at the same distance from the middle are identified, as are the corresponding inputs and outputs. Such a network appears in [13] and was originally called a *truncated* Clos network [4], but is now called a *folded* Clos network. In a folded Clos network, the path between two terminals might travel "up" towards the middle stage only as far as necessary to make a connection.

A folded Clos network might belong to the class of butterfly fat-trees. For example, the folded Beneš network becomes a full binary butterfly fat-tree with 2×2 switches. Indeed, some authors use the terms folded Clos network and fat-tree interchangeably (see e.g., [31]), but this practice is problematic as Clos networks are full-bandwidth networks whereas fat-trees were specifically designed for the flexible allocation of bandwidth.

3. RELATED WORK

This section describes important related work. Rather then attempting to survey the large and growing body of research in this area, we focus on four influential papers, and attempt to explain how the networks proposed in these papers are related to the network structures and terminology established in Section 2. We also evaluate to what extent the approaches taken in these papers are in congruence with our cost-universal approach.

Several recent developments have allayed certain concerns addressed by earlier work. First, low-end routers are now available at costs close to those of switches, so there is less impetus to implement a network entirely at layer two. Indeed vendors such as HP sell devices that can be configured to operate as either routers or switches. As a consequence concerns about loops in layer two networks and avoiding the Ethernet spanning tree algorithm have lessened. More excitingly, software defined networking (SDN) is growing in acceptance and multiple vendors now provide support for OpenFlow in their devices. SDN is ideally suited to data center applications, where all switching and routing devices are under the administration of a single entity. Hence, throughout this paper we assume support for effective network load balancing stratgies.

3.1 Al-Fares et al.

In an early and inspirational paper [2], Al-Fares et al. advocate building data-center-scale networks using (almost) commodity Ethernet switches. The paper is similar in spirit to our work in that it focuses on the price-performance ratio of available hardware, and takes many practical considerations into account. The paper asks for minor modifications in the forwarding algorithm employed by Ethernet switches, which seems reasonable today in light of SDN and other developments. We also note in Section 5 that the designer of a cost-effective data center network may not be strictly limited to commodity hardware, as vendors are willing to offer some customization when fullfilling an order on the scale of one or more data centers. The Al-Fares et al. paper advocates building a full butterfly fat-tree, which it terms a network with an *oversubscription ratio* (ratio of bandwidth connecting end hosts to bisection bandwidth) of 1:1. The paper also suggests allocating addresses systematically to match the structure of the network, a technique that is compatible with all of the structures described in Section 2, and which has been applied whenever these networks have been incorporated in parallel computers.

The terminology in Al-Fares et al. is not entirely consistent with the historical terminology. The paper states that the fat-tree is a special case of a Clos network which is unfortunate because it obscures the distinction between Clos networks, which are full bandwidth networks, and fat-trees, whose bandwidths can be adjusted to match available packaging technology. We would prefer to say that the Clos network of [2] is a special case fat-tree called a full butterfly fat-tree, and might add that when fat-trees were introduced, full fat-trees were not viewed as likely candidates for implementation on a large scale due to the difficulty of providing such a high degree of connectivity at the root.

At first glance, it appears that the cost of each level of the hierarchy in the Clos network is equal because each level has the same total number of links and the same total number of ports. But the long cables connecting racks of servers on opposite sides of the data center are more expensive than the short cables connecting servers to top-of-rack (TOR) switches. Furthermore, there are limits on the lengths of copper cables driven at 1 Gbps or higher, so high grade copper cables (e.g., cat 6a or 7), or more expensive optical fiber, may be needed to span greater distances. Al-Fares et al. acknowledge that they do not take cabling costs into consideration. In their defense, however, the cost per port of the proposed switches ($145) is fairly high (these costs have since come down), so the relative impact of cabling costs is limited. The paper does provide a careful plan for cabling, but it doesn't account for the actual weights and volumes of the cables. We would caution that such physical considerations must be taken into account.

3.2 VL2

VL2 [16] is a Clos network, similar in structure to a full fat-tree. One of the goals of VL2 is to provide an oversubscription ratio of 1:1. To achieve this, however, servers are connected to TOR switches at 1 Gbps, whereas switches are connected to other switches at 10Gbps. Hence the number of wires at each level above the racks is one tenth the number within racks. One disadvantage of this approach of "throttling" access to the network is that for the relatively small cost of upgrading to 10Gbps connections to the TOR switches, applications that mainly require communications with other servers in the same rack could perform much better. In a cost universal network, technology providing the best price-performance ratio would be employed at each level. The VL2 paper analyzes the cost of various types of switches, but does not analyze cable cost, volume, or weight.

VL2 uses Valiant-Brebner-style load balancing at the granularity

of flows, with paths from servers behind different TOR switches traveling all the way to the root. Note, though, that there is provably little load-balancing advantage gained from sending all packets to the root whether they need to travel that far up in the tree or not.

The VL2 prototype consists of 80 servers in four racks, with the TOR switches connected to three 24-port "aggregation" switches. The aggregation switches are in turn connected to three 24-port "intermediate" switches. (Not all ports are used in the prototype.)

3.3 DCell

The paper by Guo et al. [20] introduces a network called DCell. The stated goals of the DCell design are fault-tolerance and scalability. Fault-tolerance is achieved through redundant paths between source-destination pairs. DCell also aims to provide high bisection bandwidth. The authors argue that tree-based networks contain single points of failure and cannot provide a high degree of connectivity. The multi-fat-trees described in Section 2.2, however, provide multiple paths between pairs of endpoints, and as we have seen in Section 2, the amount of bisection bandwidth provided by a fat-tree can be chosen by the network designer.

DCell is constructed as follows. At the lowest level of the hierarchy, the N servers in DCell_0 are connected to a single switch. In general, suppose that a DCell_i contains t_i servers. Then a DCell_{i+1} is built from $t_i + 1$ instances of DCell_i. In each of the DCell_i instances, each of the t_i servers has a link to a server in a different instance. Hence, each DCell_i instance is connected by one link to every other DCell_i instance. For this reason, we might call DCell a clique of cliques. DCell is hierarchical in the sense that if any two servers belong to the same DCell_i, there is a path between them that is contained entirely with that DCell_i. On the other hand, there may be redundant paths between the servers that are not entirely contained in the DCell_i.

A major difference between the fat-trees described in Section 2 and DCell is that the DCell network structure is inflexible. The total number of servers and the topology of DCell is determined entirely by two parameters: the number of servers in a DCell_0, n, and the number of levels in the hierarchy. For example, in a DCell_2, the number of servers is given by the ugly formula $t_2 = ((N+1)N+1)(N+1)N$. The designer cannot adjust the structure to match the available packaging and wiring technologies.

Apart from the use of switches in DCell_0, all other connections are made directly between servers. This type of network is called a *direct* network in the literature. Clos networks and fat-trees, on the other hand, are *indirect* networks. In a direct network, switching and routing are implemented at the servers themselves. One advantage to this approach is that it allows for flexible routing algorithms, and may be used to avoid the pitfalls of cycles in layer-2 networks. Without augmenting the servers with specialized hardware, however, performance is likely to suffer. With specialized hardware, the price-performance ratio (today) is unlikely to be competitive with an indirect network.

The DCell paper does not contain any analysis of the lengths or weights or costs of the cabling, although it acknowledges that the design uses more and longer cables than competing designs.

The authors implemented a 20-server testbed, but did not face the cabling challenges that would arise were the network to be implemented at the scale of a data center with tens of thousands of servers.

The DCell paper states that in a fat-tree the aggregate bandwidth is the same at all levels. As we have discussed, this is true only of full fat-trees.

3.4 BCube

The BCube [19] is another network in which servers participate in the routing. Its topology could be called a "mesh of stars," i.e., a variant of the mesh of trees network [22], in which each tree consists of a root (a switch) connected directly to leaves (servers).

To the best of our knowledge, the exact oversubscription ratio of the BCube is not known. Calculating the bisection width of the mesh of stars, i.e., the number of links that must be cut to separate the N leaves into two equal-sized groups, is somewhat complicated. A BCube_k is formed by connecting d BCube_{k-1} networks using d^k degree-d roots. Hence the bisection width is at most $d^{k+1}/2$. There are $N = d^{k+1}$ leaves, so the bisection width is at most $N/2$. The standard technique for proving a lower bound on the bisection width of a network is to find an efficient and symmetric embedding of the complete graph K_N in the network. This technique provides a lower bound of $N/4$ for the BCube. Thus, the BCube is a full-bandwidth network in the sense that the bisection bandwidth is proportional to the number of servers in the network. For those curious about whether the true bisection width is closer to $N/2$ or $N/4$: the simple bisection described above is not optimal. Instead, it is more efficient to cut out a $(1/2)^{1/(k+1)}d \times \ldots \times (1/2)^{1/(k+1)}d$ submesh leading to a bisection width of $(k+1)((1/2)^{(k/(k+1))} - (1/2))d^{k+1}$, which approaches $((\ln 2)/2)d^{k+1}$, about $.35N$, as k grows large. We don't know if this bound is tight.

Wiring volume is estimated in [19], but only for a 2048-server BCube. Weight and cable length are not estimated. The paper explains that an Ethernet cable has diameter 0.54cm, that the number of level-2 cables along a column of racks is 256, and that the number of level-3 cables is 512 (for a total of 768). These cables need 176cm^2, which is less than $(20\text{cm})^2$. The available height is 42cm, so there is enough room for the cables. Note however, the exponential growth in the number of cables needed at level 2 versus level 3. Adding another dimension would require routing another 1024 cables along the top of the racks, so from a packaging perspective, this design is not scaling well.

4. UNIVERSALITY

This section reviews the theory of area-univeral networks and then describes the approach that we take in constructing cost-universal data center networks.

The main result in Leiserson's paper is a construction of a fat-tree network that can be laid out with VLSI area n and that can emulate any other network that can be laid out with area n with only $O(\log^3 n)$ slowdown. Such a fat-tree is called *area universal*, and there is an analogous *volume universal* fat-tree for three-dimensional VLSI layouts. Greenberg's thesis also shows how to reduce the slowdown of the emulation to $O(\log^2 n)$. The emulation results described thus far assume the *bit model* of computation in which a wire in a circuit can transmit a single bit of information in one time step. In the *word model*, in which a wire can transmit an $O(\log n)$-bit address in a single step, Leighton, Maggs, Ranade, and Rao [24] showed how to reduce the slowdown to $O(\log n)$. Bay and Bilardi then reduced the slowdown to $O(\log n)$ in the bit

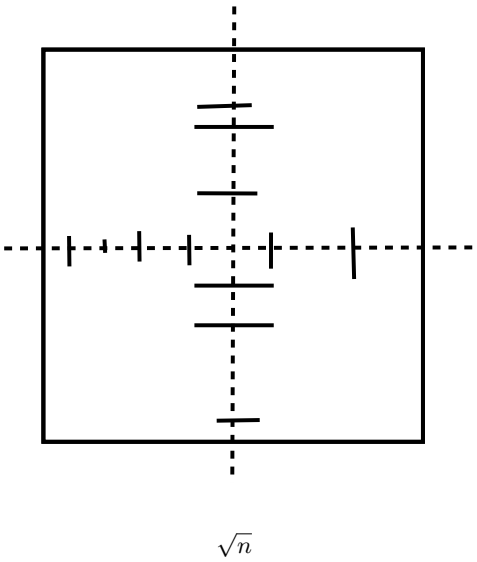

Figure 7: The VLSI layout of network $\mathcal{N}$, which has area n, is divided into four quadrants, each with a boundary of $\sqrt{n}$. The diagram shows wires severed by the vertical and horizontal cuts.

model [5]. All of these emulation results apply in the scenario in which the entire network is integrated onto a single chip.

It is not too difficult to sketch a proof that a certain 4-ary butterfly fat-tree is area universal. The goal of the fat-tree is to emulate any other network that could be laid out in area n. The fat-tree has n leaves, and the capacities of the channels are chosen so that the capacity at the leaves is one, and the capacity doubles at each of the $\log_4 n$ levels on any path up the tree from the leaves to the root, so that the capacity at the root is $\sqrt{n}$. Doubling the channel capacity in a 4-ary fat-tree is achieved using switches with twice as many children as parents. There are two ways to see why doubling the capacity at each level is the right growth rate. First (as we shall see below), the boundary of the area to be emulated by each subtree of the 4-ary tree doubles at each level. Second, as seen in Figure 3, starting at the leaves and moving up the tree, at each level the length of the wires in each channel roughly doubles and the capacity (number of wires) also doubles, so that the area required to lay out a channel increases by a factor of about four. But the number of channels is reduced by a factor of four, so the total area allocated for each level is roughly equal, which is the right allocation for area universality. (Technically the area of the H-tree layout for this fat-tree is $\Theta(n \log^2 n)$, rather than n, a complication we ignore here, but which can be addressed by constructing a fat-tree with root capacity $\sqrt{n}/\log n$ and $n/\log^2 n$ leaves, which has area $O(n)$, and then augmented the tree by attaching a $\log n \times \log n$ mesh of nodes to each leaf with a single wire [24], which also requires a total area of $O(n)$.)

Suppose that the fat-tree is to emulate a network $\mathcal{N}$ that has been laid out with area n using unit-width wires. Network $\mathcal{N}$ is mapped to the fat-tree as follows. Without loss of generality (but without proof here), we can assume that the layout of $\mathcal{N}$ is square. We begin by cutting the layout of $\mathcal{N}$ into four equal-sized square quadrants, as shown in Figure 7. Because wires have unit width, the number of wires leaving a quadrant is at most equal to the perimeter of the quadrant. Since the layout of $\mathcal{N}$ is square and has side-length $\sqrt{n}$ the perimeter of each quadrant is $\sqrt{n}$. The root of the fat-tree has four children, and the subtree rooted at each of these children emulates one of the quadrants. The capacity of the channel between the root and each child, $\sqrt{n}$, has been chosen to match the maximum number of wires connecting the quadrant to the other quadrants. Hence, at the root the fat-tree has sufficient bandwidth to carry any interquadrant traffic that network $\mathcal{N}$ could carry. The mapping then proceeds recursively, with the processors of $\mathcal{N}$ ultimately mapped to the leaves of the fat-tree. A similar approach can be used to construct a volume-universal fat-tree. Instead of cutting a two-dimensional layout into quadrants with lines, a three-dimensional layout is cut into octants with planes. In three dimensions, the number of wires that cross a cut is bound by the surface area of the cut.

The fat-tree emulates network $\mathcal{N}$ in a step-by-step fashion. For each message that $\mathcal{N}$ sends across a wire, the fat-tree sends a message between two of its leaves. A message may have to travel a distance (called the *dilation*) of up to $2 \log_4 n$. In each step, the number of messages sent by $\mathcal{N}$ across any fat-tree channel matches the capacity of that channel. It is not difficult to show that if messages follow random paths on the way up towards lowest common ancestors, the maximum number of messages that cross any link of the fat-tree (i.e., the congestion) is at most $O(\log n/\log\log n)$, with high probability. The combination of logarithmic congestion and logarithmic dilation implies logarithmic slowdown for the emulation [5, 24].

4.1 Building-scale networks

Thus far in the discussion of universality we have considered only the scenario in which the entire networks fits on a single chip. This is a convenient but limited scenario. In this context we can derive a simple mathematical formula for the amount of bandwidth to allocate at each level of the tree in order to achieve universality because cost equates to a single, homogeneous, quantity: layout area (or volume). But a data-center-scale network will not fit on a single chip, so our notion of universality must be generalized, and our approach to achieving universality may be more complex. Leiserson [26, 27] contemplates the more general case, observing that a network may be packaged in a hierarchical fashion using a variety of technologies, from chips to circuit boards to backplanes to racks. The following passage from [27] explains that the structure of a fat-tree should be adjusted to match the available technology.

> In practice, of course, no mathematical rule governs interconnect technology. Most networks that have been proposed for parallel processing, such as meshes and hypercubes, are inflexible when it comes to adapting their topologies to the arbitrary bandwidths provided by packaging technology. The growth in channel bandwidth of a fat-tree, however, is not constrained to follow a prescribed mathematical formula. The channels of a fat-tree can be adapted to effectivelty utilize whatever bandwidths the technology can provide and which make engineering sense in terms of cost and performance.

Fat-trees were first implemented in parallel computers such as the Connection Machine CM-5 by Thinking Machines [28], the CS-2 by Meiko [6] and more recently the BlackWidow by Cray [31]. In its KSR-1 and KSR-2 machines, Kendall Square Research used a

hierarchy of rings, which has at times been called a fat-tree (see, e.g., [10]).

The coarse structure of the CM-5 was a complete 4-ary tree, constructed using two types of switches. Every switch had four children, but the number of parents could be either two or four. Thinking Machines exploited the flexible structure of fat-trees to match engineering constraints, as explained in the following passage from [28]. The passage uses the term "router chip" synonymously with the term "switch" in this paper.

> Based on technology, packaging, and cost considerations, the CM-5 bandwidths are as follows. Each processor has 2 connections to the data network, corresponding to a raw bandwidth of 40 megabytes/second in and out of each processing node. In the first two levels, each router chip uses only 2 parent connections to the next higher level, yielding an aggregate bandwidth of 160 megabytes/second out of a subtree with 16 processing nodes. All router chips higher than the second level use all 4 parent connections, which, for example, yields an aggregate bandwidth of 10 gigabytes/second, in each direction, from one half of a 2K-node system to the other.

4.2 Provisioning cost-universal networks

The proof that a data-center-scale fat-tree is cost universal cannot strictly follow the proof that a certain fat-tree is area or volume universal. While it is amusing to contemplate cutting a data center building into octants and allocating as many cables to the root channels of the fat-tree as can possibly cross the surface area of the corresponding cut, the analogy with the VLSI model doesn't entirely hold up. The biggest difference is that in the VLSI model the cost of a wire, i.e., the area or volume that it occupies, is exactly equal to the length of the wire. Furthermore, the area or volume devoted to a wire often dominates that devoted to the circuitry that drives the wire. In a data center, different cabling technologies, e.g., copper versus optical fiber, impose different limits on the lengths of cables, and vary in transmission speed, cost, weight, and volume. While weight and volume are proportional to length, cost can be better modeled as the sum of a fixed-cost term and a term proportional to length [21]. The fixed cost, which includes the cost of ports and line cards, is higher for optical cables than for electrical cables, but the cost per meter is lower. The operational cost of wiring the cables, which depends on the complexity of the network design, may also be significant. In Section 5, we discuss current cabling and switch port costs.

On the other hand, whereas VLSI layout technology offers a network designer the freedom to create arbitrary structures within the area bounds of a chip, there are many practical constraints that limit the options of a data center network designer, including, most importantly, the technologies available at low cost at the time of the design. At the endpoints of the network, the best price-performance ratio is achieved by installing commodity servers, or in any case servers composed of commodity processors on commodity motherboards. In a denser installation, server blades might be plugged into a blackplane, which could implement a high-speed network that connects the blades and also interfaces with the rest of the network. Servers are typically housed in racks, and the best price-performance ratio approach to connecting the servers within a rack is often achieved by using a switch or router. Racks are organized in rows (for ventilation and wiring purposes), and the same sorts of switches and routers can be used to connect the racks within a row, or to connect different rows.

The cost-universal approach to data center design leverages the fact that a data center network designer makes investments in only a small number of connectivity categories. For any network that is implemented in practice, the designer must decide how much money to spend connecting processors or servers within racks, how much to spend connecting racks within a row, and finally, how much to spend connecting the rows. To build a cost-universal network, the designer allocates roughly the same amount of money to each of these categories, heeding physical constraints such as limits on the weight of cables that can be carried from one row to another. When physical constraints prevent the designer from spending an equal amount on all levels (due to weight, volume, or wiring complexity), the designer's goal is then to spend as close as possible to an equal amount.

For the purposes of our approach, the differences in the interconnection patterns of fat-trees, multi-fat-trees, folded Clos networks, etc., are not especially significant. What is required is that the chosen class of networks can be configured in a hierarchical fashion with a small number of levels in the hierarchy. A second requirement is that it must be possible to allocate bandwidth at each level of the hierarchy as a function of the costs and engineering constraints of the networking technology, e.g. maximum wiring length and wiring volume, that is most appropriate for implementing that level. Finally, the path in the network from one leaf to another should travel no higher than their least common ancestor, and the network and its routing algorithm must provide an effective load balancing mechansim so that any bandwidth allocated at a particular level can be fully utilized.

We are now in a position to argue that a network such as a fat-tree in which an equal amount has been invested symmetrically in each level of the hierarchy (e.g., within the racks, between the racks, between the rows) is cost universal. As in the proof that certain fat-trees are area universal, we describe an emulation. Suppose that the cost-universal network is to emulate a network $\mathcal{N}$ that has been implemented at the same cost. We assume that $\mathcal{N}$ is a roughly symmetric network, i.e., there is no rack that is much better connected internally than any other rack, there is no row that is much better connected internally than any other row, and each row has roughly the same amount of connectivity to the rest of the network. This assumption is limiting, but virtually all proposed data center networks are symmetric. Consider an application $\mathcal{A}$ that is executing on $\mathcal{N}$, and now imagine instead that the same application $\mathcal{A}$ is executed on the cost-universal network. How much slower will it run? Suppose that the performance of $\mathcal{A}$ is limited by a network bandwidth bottleneck in some category, e.g., the links between racks in a row are saturated. (If instead, the performance of $\mathcal{A}$ is limited by the latency of $\mathcal{N}$, then it should not be much more limited by the latency of the cost-universal network, which is a shallow hierarchical network.) The cost-universal network has a level in its hierarchy for each category of connectivity in $\mathcal{N}$, and can use all of its bandwidth efficiently. Because there are only a small number of levels in the hierarchy (e.g., four), even if the designer of $\mathcal{N}$ invested everything in this single bottleneck category, the designer of the cost-universal network has made at least one fourth of the same investment in the corresponding level of the hierarchy. We assume that this one-fourth investment will provide application $\mathcal{A}$ with at least one fourth the bandwidth, and hence at

most a factor of four slowdown. (In practice, it is not strictly true that the optimal price/performance ratio is constant for all levels of investment. The price/performance ratio might change at inflection points where different technologies requiring different minimum investments are employed. If a small increase in investment would allow the designer to switch to a technology with a better price/performance ratio, it might make sense to invest a little more.) While a factor of four (in this example) certainly matters in practice, this worst-case slowdown would apply only if the designer knew the application in advance of building the network, and the application had a communication pattern that exclusively exercised one category of connectivity. The fact that such an upper bound applies to all application running on all possible networks that could be built at the same cost is an indication that the cost-universal network is a sound general-purpose design.

Of course if the application or applications to be executed in a data center are all known in advance, it may make sense to design a custom network with the aim of optimizing these applications. In this case, a cost-universal network can serve as a design starting point, and the capacities of the channels at different levels of the hiearchy can be adjusted to match the demands of the applications.

Another important cost to consider when designing a network is the cost of developing the software for the applications that will run on the network. Blelloch et al. [8] argue that achieving good performance on networks with low bisection bandwidth requires carefully engineering algorithms that are specialized to the network architecture, which is an expensive and time-consuming process. Hence, a larger investment in the wiring of the upper levels of the hierarchy may be offset by savings in software development costs. This argument does not contradict the universality argument presented in this paper. Instead, it suggests that in determining the total budget for building a network, the impact on future software development costs should be taken into consideration.

5. PRACTICAL ISSUES

The art of designing a realizable and cost-effective data center network requires a careful blending of theory and practical concerns. Previous sections have reviewed some of the applicable theory, and we now turn to some of the important of the practical issues.

As previously pointed out, the overall network serves as one large "switch" made of ASICs (switching elements) connected together by a variety of motherboard traces, backplanes, or cables (links) and knit together by software (the control plane) [26]. Due to space constraints, we examine the ASICs, backplanes, and cable, leaving the software for future work. We examine several large data centers built by a major online services company, analyzing the spent at each of the layers.

5.1 Trends in ASIC Design

"Commodity" switching has been revolutionized by single-chip switch ASICs with astonishing performance. Each switch ASIC generation is usually opened by a new fabrication node, just like CPUs. This sets how many transistors the switch designer has to work with on a cost effective die-area of silicon.

The trend is to spend these transistors on more links, faster links, and integrating more elements of a physical switch onto the chip (e.g., the SERDES). As a result, designers have spent fewer transistors on buffer space or forwarding features. This trend is illustrated by Table 1 that shows how the number of ports, port speeds, total buffer, and buffer per port have changed over multiple generations of switch ASICs.

Year	Ports	Throughput	Total Buffer	μs of Buffer
2007	24 x 1Gbps, 4 x 10 Gbps	64Gbps	2MB	250
2009	48 1Gbps, 4 10Gbps	88Gbps	4MB	363
2009	24 10Gbps	240 Gbps	4MB	133
2011	64 10Gbps	640 Gbps	9MB	112
2013	32 40Gbps or 128 10Gbps	1.2Tbps	9MB	56

Table 1: Timeline of ASIC port counts, speeds, buffer size, and how many microseconds of buffer are available per port if each port is outputting at maximum utilization.

Continuing the similarities with CPUs, since the die area of each generation of switch ASIC is roughly the same, the per-unit manufacturing cost of each generation is also roughly the same. Like with CPUs, prices vary greatly depending on a buyer's volume and market relationship, but are probably in the range of $500 - $1,500 each.

5.2 ASIC Interconnection Strategies

Consider the challenge faced by a designer wishing to create a network that can interconnect 2,048 ports of 10Gbps Ethernet with no oversubscription, and the designer has a switch ASIC with 64 ports of 10Gbps available as the basic building block. The designer could create a two layer Clos network of switches, each switch having 64x10Gbps ports and the two layers of switches connected by fiber links. This would consume 64 switches in one layer and 32 switches in the other for a total of 96 switches and 2,048 cables inside the Clos. Alternatively, the designer could first assemble a set of ASICs into a 128x10Gbps switch, creating a Clos network among the ASICs using traces on a PCB motherboard, and then build a network using these 128x10Gbps switches.

The challenge faced by commodity network device vendors is to assemble a collection of switch ASICs into a switch that they can sell profitably to a large market. Since the complexity is higher and the market smaller for a multi-chip switch, the per-port cost of such a multi-chip switch is usually higher than the costs for a network device housing a single switch ASIC. This can be counter-intuitive, given that the cost of traces between multiple switch ASICs on a motherboard are much cheaper than wires in the backplane of a multi-motherboard switch, which are much cheaper than the cost of fiber or copper connection between two physical separate network devices.

To give some feel for the costs: motherboard traces are so cheap as to be essentially free. The cost of a switch chassis containing a roughly 4Tbps backplane is on the order of $50K, or $125 per 10Gbps. The cost of a 10Gbps fiber cable, with optical modules on each side, is roughly $200. A copper-based Direct Attach Copper (DAC) 10Gbps cable, which can used up for distances up to about 5m, costs about $50-$100 (one problem with DAC cables is their volume, which is significantly larger per cable than fiber).

This means that today, any network that can be built using single ASIC switches within about 5m of each other connected by copper cables is the cheapest way to build such a network. Above that size, the use of multi-ASIC switches starts to become cost effective.

Active Optical Cables (AOCs) are changing the game, however. At

a cost around $125, AOCs provide 10 or even 40Gbps over 3m to 20m of fiber. They achieve this feat by hermetically sealing the fiber to the optical modules, which enables them to use non-standard lasers or even LEDs. The major drawback is a deployment issue — the cables have to be pulled or otherwise installed with the optical modules attached.

Planning for the cabling between switches is a first order concern in any network design, but especially in designs like those we describe. Physically running the cables needed to realize these designs is quite possible, but does require carefully computing the weight and volume of all the cable paths so that properly sized horizontal and vertical cables trays can be installed. Failure to plan ahead has led fibers to overflow and spill out of cable managers, resulting in such a mess that the network layout had to be redesigned and recabled.

Similarly, the cost of cables and cabling needs to be accounted for in the design process. For example, consider building a network using three layers of single ASIC switches or two layers of multi-ASIC switches (each switch then having a higher radix than a single ASIC switch). The network with three layers of single ASIC switches will use fewer ASICs than a similarly sized network with multi-ASIC switches. However, the three-layer network uses twice as many physical cables, and is much harder to construct.

5.3 Costs

While we have some data on the costs of data center components, we do not have enough information to fully flesh out the design of a cost-universal network. However, we were able to obtain, from a large on-line services provider, the relative capital expenses on each tier (top-of-rack, aggregation switch, spine switch) of two generations of production networks. Interestingly, the data suggests that, in practice, network designers are now roughly following our "equal cost per tier of the hierarchy" suggestion. The cost of the cables to connect two layers is included with the cost of the layer above (e.g., server to ToR cables are included in the cost of the ToR layer, and ToR to Aggregation cables are included in the cost of the Aggregation layer). In a network over 4 years old with a relatively high oversubscription ratio, the ratio of the cost of the network layers was 6:2:1 ($6 spent on top-of-rack switches and $2 on aggregation switches for every dollar spent on spine switches). For newer networks with lower oversubscription ratios, we found the ratio to be close to 2:2:1. As predicted by the model, the factor driving the equalization of cost is the increased expense for cables in the networks with less oversubscription.

6. CONCLUSION

As we enter an era of ubiquitous cloud computing and online services, investment in data centers is going to be enormous. It is incumbent on the networking community, therefore, to develop principled methods for designing data center networks. In this vein, we have suggested a rule for allocating resources to the tiers of hierarchical networks: roughly equal investment at each tier. Our focus is on network bandwidth and the cost, volume, and wiring complexity of the network. There are numerous other important considerations, however. For example, the network designer must also address the cooling, energy consumption, latency, fault tolerance, and management of the network. At the moment we do not see any reasons why addressing these issues will conflict with our cost-universal approach. Further confirmation, however, will require a complete and holistic design of a cost-universal network.

7. REFERENCES

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proceedngs of SIGCOMM 2008*.

[3] S. Arora, F. T. Leighton, and B. M. Maggs. On-line algorithms for path selection in a non-blocking network. *SIAM Journal on Computing*, 25(3):600–625, June 1996.

[4] L. A. Bassalygo, I. I Grushko, and V. I. Neiman. The structures of one-sided connecting networks. In *Proceedings of the 6th International Teletraffic Congress*, pages 245/1–245/9, September 1970.

[5] P. Bay and G. Bilardi. An area-universal circuit. In *Proceedings of the 1993 Symposium on Research on Integrated Systems*.

[6] J. Beecroft, M. Homewood, and M. McLaren. Meiko CS-2 interconnect Elan-Elite design. *Parallel Computing*, (10–11):1627–1638, November 1994.

[7] V. E. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, 1965.

[8] G. E. Blelloch, B. M. Maggs, and G. L. Miller. The hidden cost of low bandwidth communication. In U. Vishkin, editor, *Developing a Computer Science Agenda for High-Performance Computing*, pages 22–25. ACM Press, 1994.

[9] T. L. Bowers. Blocking in 3-stage "folded" switching arrays. *IEEE Transactions on Communication Technology*, 13(1):14–37, March 1965.

[10] E. L. Boyd and E. S. Davidson. Communication in the ksr1 mpp: Performance evaluation using synthetic workload experiments. In *Proceedings of the 8th ACM International Conference on Supercomputing*, pages 166–176, July 1994.

[11] T. Callahan and S. C. Goldstein. NIFDY: A low overhead, high throughput network interface. In *Proceedings of ISCA 1995*.

[12] D. Chiou, B. S. Ang, R. Greiner, Arvind, M. J. Beckerle, James E. Hicks, and G. A. Boughton. StarT-NG: Delivering seamless parallel computing. In *Proceedings of EURO-PAR '95*.

[13] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32:406–424, 1953.

[14] A. Curtis, S. Keshav, and A. Lopez-Ortiz. LEGUP: Using heterogeneity to reduce the cost of data center network upgrades. In *Proceedings of CoNEXT 2010*.

[15] A. V. Goldberg, B. M. Maggs, and S. A. Plotkin. A parallel algorithm for reconfiguring a multibutterfly network with faulty switches. *IEEE Transactions on Computers*, 43(3):321–326, March 1994.

[16] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A scalable and flexible data center network. In *Proceedings of SIGCOMM 2009*.

[17] R. I. Greenberg. *Efficient Interconnection Schemes for VLSI and Parallel Computation*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, August 1989.

[18] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In Silvio Micali, editor, *Randomness and Computation.Volume 5 of Advances in Computing Research*, pages 345–374. JAI Press, Greenwich, CT, 1989.

[19] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of SIGCOMM 2009*.

[20] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: A scalable and fault-tolerant network structure for data centers. In *Proceedings of SIGCOMM 2008*.

[21] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. In *Proceedings of ISCA 2008*.

[22] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays • Trees • Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

[23] F. T. Leighton and B. M. Maggs. Fast algorithms for routing around faults in multibutterflies and randomly-wired splitter networks. *IEEE Transactions on Computers*, 41(5):578–587, May 1992.

[24] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao.

Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, July 1994.

[25] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C–34(10):892–901, October 1985.

[26] C. E. Leiserson. VLSI theory and parallel supercomputing. In *Advanced Research in VLSI 1989: Proceedings of the Decennial Caltech Conference on VLSI*, March 1989.

[27] C. E. Leiserson. VLSI theory and parallel supercomputing. Technical Memo MIT/LCS/TM-402, MIT Laboratory for Computer Science, Cambridge, MA, May 1989.

[28] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Jill, W. D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells, M. C. Wong-Chan, S.-W. Yang, and R. Zak. The network architecture of the connection machine CM-5. *Journal of Parallel and Distributed Computing*, 33(2):145–158, March 1996.

[29] Carver A. Mead and Martin Rem. Cost and performance of VLSI computing structures. Technical report, California Institute of Technology, Pasadena, CA, April 1978.

[30] R. N. 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 SIGCOMM 2009*.

[31] S. Scott, D. Abts, J. Kim, and W. J. Dally. The BlackWidow high-radix Clos network. In *Proceedings of ISCA 2006*.

[32] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *Proceedings of NSDI 2012*.

[33] E. Upfal. An $O(\log N)$ deterministic packet routing scheme. *Journal of the ACM*, 39(1):55–70, January 1992.

[34] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, May 1981.