

# The Fat-Stack and Universal Routing in Distributed Networks\*

Kevin F. Chen, Edwin H.-M. Sha  
Department of Computer Science  
University of Texas at Dallas  
Richardson, TX 75083, USA  
{fxc015200, edsha}@utdallas.edu

## Abstract

This paper shows that a novel network called the *fat-stack* is efficient and is suitable for use as a baseline distributed network and as a crucial benchmark architecture for evaluating the performance of specific distributed networks. The fat-stack is a construct of an atomic subnetwork unit consisting of one ring and one upward link to its upper subnetwork. This simple structure makes the network scalable to closely represent a distributed network. We show that the fat-stack is efficient by proving it is universal. A requirement for the fat-stack to be universal is that link capacities double up the levels of the network. We also show how to scale the network from a VLSI graph layout to a large-scale distributed topology. Our universality proof shows that a fat-stack of area  $\Theta(A)$  can simulate any competing network of area  $A$  with  $O(\log^{\frac{3}{2}} A)$  overhead independent of wire delay. The universality result implies that the fat-stack of a given size is nearly the best routing network of that size. The fat-stack is also the minimal universal network for an  $O(\log^{\frac{3}{2}} A)$  overhead in terms of number of links. Actual simulations show that the fat-stack outperforms a mesh-based distributed network of comparable hardware usage.

## Keywords

System design, simulations, fat-stack, universality, network architectures, VLSI.

## 1 Introduction

The topology of a network determines its efficiency on the first order. A network architecture can be considered as consisting of a distinct topology, varied link capacities, and a specific routing scheme. An architecture re-

sembling practical networks provides not only a working model but also a foundation for studying different networks and contriving novel network services. In this paper we prove analytically that a certain architecture is the best suitable for distributed networking and can be used as a benchmark to evaluate the performance of specific network topologies. The proofs are based on routing results and hardware layouts developed in the context of interconnection networks for parallel computers. It is both theoretically significant and desirable to ensure the proving premises and results to be valid across scales. We show how to scale a VLSI network up to represent a distributed network such that routing properties are retained.

An efficient network should move traffic speedily for the computing task and require no excessive hardware to build. We show that the *fat-stack* is such an efficient network by showing that it is *universal*, i.e. it can simulate any other network with an overhead of no more than (some power of) the logarithm of the area  $A$  of the hardware containing the network. This universality result implies that the fat-stack performs much better than or as well as most, if not all, of known networks. The choice of the term “fat-stack” stems from the observation that the network is a construct of identical atomic subnetwork units stacked up and tapering upwards fast. We will formally define the fat-stack network in Section 3.

Universal routing networks have been studied in the past in the context of interconnection networks. Work on the fat-tree and the fat-pyramid has initiated some principles and methodologies for studying and designing universal routing networks [12, 14, 23]. To apply these tenets to large-scale distributed networks, it is crucial for the network to be scalable. We propose the fat-stack because its structure is amenable to scaling. It turns out that the fat-stack is versatile as well. In a previous paper [10], we have reported the results of the fat-stack as an efficient interconnection network.

---

\*Work supported in part by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001 and NSF CCR-0309461.

A typical fat-tree assumes a 4-ary tree structure with link capacities doubling up the levels of the tree. The fat-pyramid inherits the 4-ary tree framework of the fat-tree and adds a mesh on each level of the nodes up the tree. Specific hardware layouts of the two networks and a fat-stack will be described in Section 3. The fat-tree is the first proved universal network [23]. But it is universal only under unit wire delay condition; its universality does not hold under nonunit wire delays [12]. The fat-pyramid has been proven to be universal under both unit and nonunit wire delay conditions [12]. The fat-tree has been used in the CM-5 parallel computer whereas the fat-pyramid has not been adopted for any machine. Another clear advantage of the fat-pyramid over the fat-tree is its better absolute efficiency due to its hierarchical meshes. But these same meshes of the fat-pyramid increase its wire usage considerably and make it not scalable to represent a distributed network.

The fat-stack is relatively simplistic in structure, which makes it scalable to closely represent a distributed network. It can be constructed by stacking up atomic subnetwork units following a fat-tree framework. A subnetwork unit is made of a ring of certain nodes and one or more upward links each from one node of the unit. These links connect to the same node of a subnetwork right above the unit. The network is built up recursively. We consider two variants of the fat-stack in this paper. One has only one upward link from a subnetwork and the top level node is omitted. This variant is not strictly based on a tree due to the omission of some links. We refer to this variant as the *general fat-stack* (GFS) which is the main focus of this paper. The second variant has as many upward links as the number of nodes in the subnetwork. We refer to this variant as the *augmented fat-stack* (AFS).

In addition to scalability, another advantage of the fat-stack is that the GFS is the minimal universal network for the same asymptotic overhead. This notion of lower bound network is in terms of hardware usage. The GFS, AFS, and fat-pyramid all incur an  $O(\log^{\frac{3}{2}} A)$  overhead under nonunit wire delay assumption despite expectant variation in their absolute efficiencies. Also, the AFS represents a performance upper bound achievable when additional links were to be added up from the subnetworks in the GFS. In all, these three networks plus the fat-tree provide a range of performance attainments.

Routing schemes have direct impact on the universality of a network. The universality of the fat-stack relies on routing capability in terms of a linear combination of congestion and distance that a packet travels. This capa-

bility applies to offline routing. But online routing should have the same efficiency due to analysis of packet routing on a “leveled network” [19, 21]. We say that network  $A$  can simulate network  $B$  with overhead  $\mu$  if, for any  $t$ , the routing performed by  $B$  in time  $t$  can be performed by  $A$  in time  $\mu t$ .

The fat-tree and fat-pyramid have been studied under the conditions of unit and nonunit wire delays and compact VLSI hardware texture. Under the same conditions, the fat-stack can be considered as resembling the fat-tree and fat-pyramid networks. The established results for the fat-tree and fat-pyramid networks [12, 14, 23, 30] bear some useful implications on the universal properties of the fat-stack. We will prove the universality of the fat-stack using a similar physical layout and under both unit and nonunit wire delay conditions.

While much of our work is analytical proof, we also simulated the two variants of the fat-stack together with the fat-tree, the fat-pyramid, and a widely used mesh-based distributed network, in order to visualize and compare their performances in realistic terms. The simulations were carried out using the *ns-2* network simulator. The nodes of the networks are thought as fitted on a template of reference, and the wires have nonunit delay. This type of simplification enables us to relate the experimental data to the analytical results. The simulations demonstrate that the AFS has a high performance improvement over the fat-tree. The GFS is shown to have much better performance than the mesh-based distributed network. It stands to reason that the fat-stack as a tired network with upward increasing link capacities performs better than any flat mesh-like network.

Our contributions implicated in this paper are the following: (1) We initiate a theoretical and methodological framework for defining and scaling a universal distributed network architecture. (2) The GFS can scale up to closely represent a distributed network. (3) The GFS is the minimal universal network under nonunit wire delay condition with  $O(\log^{\frac{3}{2}} A)$  overhead. (4) We provide a crucial benchmark to evaluate the performance of practical distributed networks.

## 2 Related Work

The fat-tree architecture has found widespread application in parallel computers and switch fabrics since its advent [12, 14, 23, 25]. There has also been continued research on fat-tree networks. The bulk of the work either

proposes new routing models in a VLSI fat-tree layout (e.g. [30]), or casts the fat-tree onto a distributed setting to overcome certain impediments in some routing scenarios (e.g. [17]). Work by Greenberg and Oh shows that a fat-tree network of area  $\Theta(A)$  can simulate any network of comparable area with  $O(\log^3 A)$  slowdown when the networks run a wormhole routing algorithm [15]. Recent work by Hung and Robertazzi studies scheduling issues in cluster and grid computing systems based on the fat-tree [16].

The fat-stack can be usable for the aforementioned application scenarios of the fat-tree. The simple structure of the fat-stack ensures good scalability but offers higher performance. Furthermore, the fat-tree does not constitute a prevalent distributed architecture. Routing schemes specific to the fat-tree can be too restrictive for use to be the basis to compare and study the universality of networks. The fat-stack, however, mimics current deployed networks at the switched LAN, WAN, regional, and continental scales.

On another front, important results have been obtained in work of devising routing schemes applicable to general networks and traffic loads. For example, Andrews et al. proposed a randomized greedy scheduling protocol with polynomial queue size and delay [2]. Other results include those published in [1, 9, 27]. In proving the universality of the fat-stack, we use a routing theorem that is general and indicates a better efficiency than even routing protocols designed for some specific networks. We did not correlate this theorem to the others.

The Internet as a distributed network would defy a unified routing scheme. But many aspects of its architecture have been treated in a unified framework. Many services and solutions have been so obtained as exemplified by the works of [28, 29]. We seek to provide the fat-stack as a general exemplary framework that is applicable within variable scales with the Internet being an instance.

One attribute of the fat-stack is efficient packet routing. This is an easy choice in defining the architecture as data communication is by packets as in IP or by cells as in ATM. Transport, quality of service, and network services all relegate their tasks to the routing layer. There is important recent work done in the packet routing domain. For example, Borodin et al. and Andrews et al. studied the stability of packet routing networks [2, 6]. As an abstraction of IP-layer solutions and overlay solutions, *i3* can perhaps be regarded as a routing level infrastructure as well [29]. In our proofs, we use a general routing theorem of Leighton et al. The theorem is in terms

of two summary measures and does not directly indicate network capacity. In our simulations, we used a simplistic routing construct that represents a workable paradigm on par with today's actual technology advance.

In scaling the fat-stack from a VLSI layout to large-scale distributed network, there is an uncertainty. We resort to using an approach of extension by multiples. How much varying wire length and capacity affects routing efficiency is unresolved. Recent work by Borodin et al. addresses the effect on routing and stability when network links have capacities and slowdowns [7, 8]. They left as an open problem whether stability is preserved for either static capacities or slowdowns in a network running a certain scheduling protocol. But they did show that such a system remains stable with dynamic capacities and slowdowns.

### 3 Network Model

We now describe the precise topology of the fat-stack, a hardware layout, and a computation and communication framework on which routing is based.

The fat-stack is a hierarchical network, consisting of tiers or levels. Each level has one or more subnetworks. Each subnetwork is a ring. A graphical representation of a fat-stack is shown in Figure 1. A fat-stack can have arbitrary levels of rings.

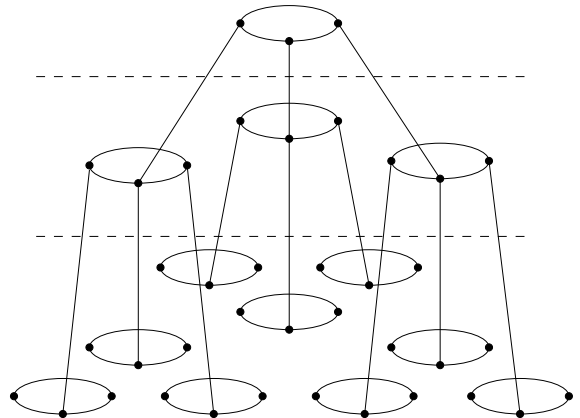


Figure 1: A fat-stack topology that has three nodes in a subnetwork. Each subnetwork connects to its upper level via a node by a single link. Dashed lines represent tier boundaries.

Let  $n$  be the number of nodes in a subnetwork. The base structure of an AFS is one of an  $n$ -ary tree in which each non-leaf node represents a switch and a leaf node represents a processor. The GFS does not contain an  $n$ -

ary tree, but we will nonetheless use an  $n$ -ary framework to prove its universality. A bottom level node in a GFS can be a workstation or a hub of workstations.

Figure 2 shows the hardware layout of a fat-tree, a fat-pyramid, and an AFS. The figure is adapted from Figure 1 of [12]. In these structures, each edge represents a wire of one unit of capacity. To increase capacity, extra edges and nodes of the same capacity are created. In analyzing distributed networks, it is often necessary to combine these separate edges and nodes to form a singular structure. In a singular network graph such as Figure 1, an edge will represent multiple units of capacities and corresponds to a channel consisting of a certain number of wires in the context of interconnection networks.

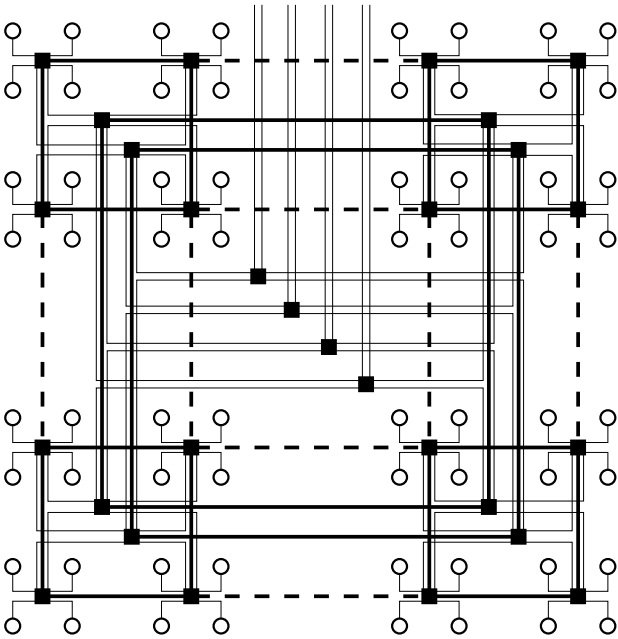


Figure 2: A fat-tree, a fat-pyramid, and an AFS in one layout. Processors are represented by circles; the squares are switches. The fat-tree links are represented by thin lines, the fat-pyramid mesh links by thick solid and dashed lines, and the AFS ring links by the thick solid lines.

The channel capacities in the fat-tree in Figure 2 double at increasing levels of the underlying 4-ary tree inasmuch as all 4 child nodes connect to one parent node with a double redundancy at each level. This capacity increment can be viewed as splitting the upper node into two and duplicating the four downward links once. We can do a reverse of the “split-and-duplicate” procedure to transform the fat-tree into a singular 4-ary tree structure by overlaying individual nodes and links and hence their capacities. We can perform the same overlay procedure on

the fat-pyramid to get a 4-ary tree in which all the switch nodes at each level of the underlying tree are connected within a mesh. The procedure can also collapse the AFS layout into a singular structure.

The difference of the three networks lies in if and how the nodes at a level of the tree are interconnected. Unlike the fat-pyramid, the fat-tree has no connecting links among switches on the same levels. In the AFS, the switch nodes with the same parent are connected by a ring. The degree of connectivity decreases in the order of the fat-pyramid, the AFS, and the fat-tree.

In overlaying the nodes and links of the fat-pyramid, the capacities of the mesh links increase at the same rate as those of the other nodes and links. The overall capacity increase is  $2^h$  times up the underlying fat-tree where  $h$  is the level number from the leaf nodes designated as level 0. We modulate the AFS with the same capacity distribution pattern as for the fat-tree and the fat-pyramid. Intuitively, compared to the fat-tree, the AFS merely possesses more connectivity due to the connected subnetwork nodes and thus can not be worse off in computing and routing efficiency.

An arbitrary number of capacity distributions can be specified for a fat-stack by combining two types of switches with a constant number of inputs and outputs as demonstrated in [24]. Our study concerns routing improvement from a uniform network to a network with link capacities doubling upwards. Capacity doubling makes the fat-stack universal.

In proving the universality of the fat-pyramid, each leaf node is treated as a cluster of  $\log_2 A$  processors each of area  $\Theta(\log A)$  in an H-tree layout (of size  $\Theta(\log A) \times \Theta(\log A)$ ) [12]. This condition will be modified to single-processor packing to prove the universality of the fat-stack.

We employ an abstract model of packet routing and operation. The basic mode of operation assumed of the fat-stack is the usual distributed random-access machine (DRAM) model [26]. In a DRAM, all memory is located at the processors and its access is made by messages routed through the routing interconnection network. Indivisible packets will be used for routing analysis and they can be perceived as the basic constituents of data traffic and as a convenient scalar quantity for mathematical analysis. Large messages can be fragmented into packets.

## 4 Universality of the Fat-Stack

In this section, we prove the universality of the GFS. We first set forth the necessary postulates for the proofs and affirm the universal properties of the fat-tree, the fat-pyramid, and the AFS under single-processor configuration. We then prove the universality of the fat-stack under both unit and nonunit wire delay conditions.

Universality proofs for the fat-tree and the fat-pyramid are parameterized by hardware area or volume. A geometric bisecting method is used to decompose the competing network to match with the structure of the base network [12, 23]. We shall adopt this method in our proofs. This method has its basis in the theories and constructions of VLSI graph layouts [4, 5, 18]. The competing network can be in terms of a cube or an area in a two dimensional design space. Extension of computation analysis from two dimensions to three dimensions has been shown to be straightforward [11, 13]. The decomposition is to recursively cut the cube or area into two pieces in the direction of the shorter edges until each piece contains either zero or one processors. It has been proven that a balanced decomposition tree can always be obtained, in which the number of processors on either side of a given node (cut) is equal to within one [23]. In view of the decomposition tree, a valid assumption is that the number of packets that can enter or leave an area (equivalently a subtree) in unit time is proportional to the perimeter of the area.

In addition, our proofs will be based on the general but powerful routing results obtained by Leighton, Maggs, and Rao [19–22]. The results pertain to offline algorithms that work under unit wire delay condition. Unit wire delay denotes that it takes unit time for a packet to move through a wire. This assumption implies that a packet traverses a distance of at most one during a single routing step or one unit time, and that at most one packet can pass through a wire during one routing step. In later analysis, wires will be considered as *transmission lines* that pipeline bits (packets) and have delay (speed) variations. The following lemma will suffice to prove the universality of the fat-stack. In the lemma, the term *congestion* refers to the maximum number of packets that must traverse a single edge in one direction, and *dilation* refers to the maximum number of edges that must be traversed by a single packet.

**Lemma 4.1 (Leighton et al.).** *For any set of packets with edge-simple paths having congestion  $c$  and dilation  $d$ , there is a schedule of length  $O(c + d)$  requiring a maximum queue size of  $O(1)$ .*

It has been shown that a fat-tree can *efficiently* (i.e. in no more than polylogarithmic slowdown) simulate any network of comparable volume or area under the unit wire delay assumption [3, 14, 19, 23]. With the provisions of the bisection method, the routing results as described above, and an H-tree processor packing, it is proved in [12] that the fat-tree can simulate any network with an  $O(\log A)$  overhead under unit wire delay condition, and that the fat-pyramid can simulate any network with an  $O(\log A)$  overhead regardless of wire delays, provided that the base network and the competing network are of the same area  $A$ . With similar postulates, we showed in [10] that the AFS is universal with an  $O(\log A)$  overhead under both unit and nonunit wire delay conditions and is in fact the minimal network for that efficiency.

We also want to make a modification of the processor packing used in [12] that each bottom level node is an H-tree layout of  $\log_2 A$  processors each of area  $\Theta(\log A)$ . In the subsequent analysis, each leaf node is thought to be of a single processor of area  $\Theta(\log A)$ . Also, there is a local ring for nodes in a subnetwork at the bottom level.

Using single processor leaf nodes introduces a complication on the number of processors ( $N$ ) that can be packed in area  $\Theta(A)$ . Referring to Figure 2, we can determine  $N$  by solving the following recursion for the side length  $S(N)$ :

$$S(N) = 2S(N/4) + O(\sqrt{N}), \text{ and } S(1) = \Theta(\sqrt{\log A}).$$

The solution is

$$S(N) = \Theta(\sqrt{N}(\sqrt{\log A} + \log N)).$$

Since  $\log N \leq \log A$  (because  $N \leq A/\log A$  necessarily), taking approximation gives

$$\begin{aligned} A &= \Theta(N(\sqrt{\log A} + \log N)^2) \\ &\leq N(\sqrt{\log A} + \log A)^2 \\ &\leq N(2\log A)^2 \\ &= 4N\log^2 A. \end{aligned}$$

Therefore,  $N \geq A/(4\log^2 A)$ , or  $N = \Omega(A/\log^2 A)$ .

Having obtained the above relation, we prove the universality of the fat-tree and its implications on the AFS and the fat-pyramid, before we discuss the GFS. In fact, processor packing has no effect on the universality overhead for the fat-tree as shown in the following theorem.

**Theorem 4.1.** *A fat-tree  $F$  of area  $\Theta(A)$  can simulate any network of area  $A$  with  $O(\log A)$  overhead under unit wire delay assumption.*

*Proof.* Use the decomposition method discussed above. The perimeter of a piece of the competing network corresponding to a subtree of  $N/2^l$  processors in  $F$  is  $O(\sqrt{A}/2^{l/2})$ . The capacity of a channel on top of a subtree of  $N/2^l$  processors in  $F$  is at least  $\sqrt{N/2^l}$  and  $N = \Omega(A/\log^2 A)$ . Thus, the congestion on a wire in  $F$  at any unit time is the number of packets allocated to a subtree divided by the capacity of the channel which is  $O(\log A)$ . Also, a packet travels a distance of  $O(\log A)$  in the tree. By Lemma 4.1, the simulation overhead is  $O(\log A)$ .  $\square$

Since the AFS and the fat-pyramid are all based on the fat-tree framework but only with more ring/mesh connections, we have the following corollary.

**Corollary 4.2.** *An AFS and a fat-pyramid of area  $\Theta(A)$  each can simulate any network of area  $A$  with  $O(\log A)$  overhead under unit wire delay assumption.*

One stark difference between the GFS and the AFS is that the AFS has a top level node and all subtending nodes have a link to an upper level node. To show that the GFS is universal, we retain the same framework as of the AFS and we examine the impact of the absence of a top node and extra links on the area and the number of packets on a wire in the GFS which is now directly the base network. For a fixed  $N$ , we will retain a  $\Theta(A)$  area for the GFS by imagining the existence of a top node and links but treating them as only “stuffing” dummy constituents in that they do not route packets. We can apply the same equal-area concept to address that there is now only one link (the joint link) connecting a subnetwork to its upper level.

We can now focus on the routing overloading on the GFS from a competing network to arrive at the following theorem.

**Theorem 4.3.** *A GFS  $F$  of area  $\Theta(A)$  with capacity doubling up the basis network tree can simulate any network of area  $A$  with  $O(\log A)$  overhead under unit wire delay assumption.*

*Proof.* The framework of the GFS is kept the same as the fat-tree except it includes a dummy node and some dummy links that do not route. Matching the packets out of a perimeter of  $O(\sqrt{A}/2^{l/2})$  of a piece of the competing network  $R$  is done in the same fashion, i.e. to a subtree of  $N/2^l$  processors in  $F$ . The link capacity (i.e. the number of wires) up a subtree  $s_0$  that has an upward link is still at least  $\sqrt{N/2^l}$  as it is fixed as doubling upwards

and  $N = \Omega(A/\log^2 A)$  as derived in the preceding recursion solution. Thus for that subtree the link load is  $O(\log A)$ . Now each other subtree in the same subnetwork as  $s_0$  routes its load  $O(\log A)$  to the  $s_0$  link through the mesh (ring) links. The  $s_0$  link capacity, i.e. the network congestion  $c$ , is now  $O(n \log A)$ . (Three notes are in order. First, we could exert flow control on the subnetwork, but that would be equivalent to extending the schedule length over the unit time constraint now being considered. Second, we could increase the capacity gradient to get a smaller  $c$ , but then  $d$  would still limit the overhead to be only as good as  $O(\log A)$ . Third, if any second link was added from a subtree node to the upper common node, then the load of  $O(n \log A)$  could be amortized, i.e. increased connectivity will reduce congestion.) Since a packet has to traverse  $O(n)$  extra (ring) links, the network dilation  $d$  is now  $O(n \log A)$ . This factor of  $n$  also occurs in overloading at the top level subnetwork of  $F$ . It is certainly reasonable to assume that  $n$  is fixed in all three cases. Then, by Lemma 4.1, the simulation overhead of  $F$  is  $O(\log A)$ .  $\square$

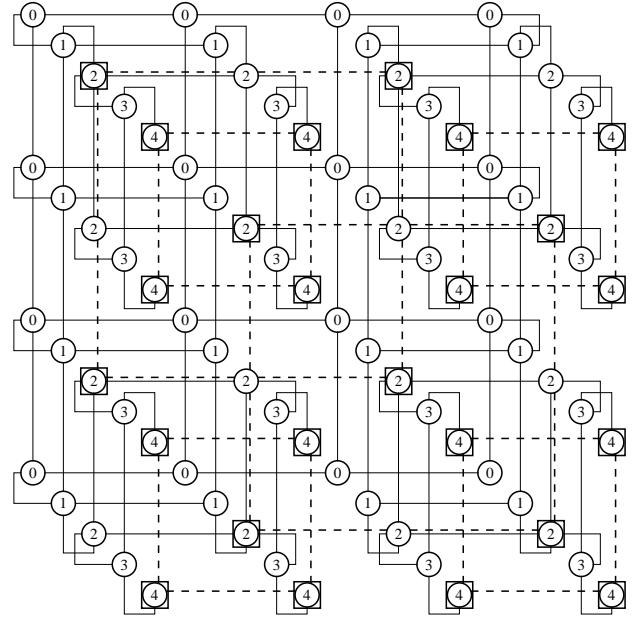


Figure 3: Regular layout of the GFS. Nodes corresponding to the nodes of the tree of meshes are represented by circles; the squares are switches. The fat-tree edges are routed through the edges of the tree of meshes shown as solid lines. The ring links are shown as dashed lines.

Note that Theorem 4.3 also implies Corollary 4.2. For both the AFS and the fat-pyramid, the  $n$  in the above proof does not occur, which also indicates that the two networks are more efficient than the GFS.

We now prove the universality of the GFS under nonunit wire delay assumption. We follow a similar procedure as in the fat-pyramid proof to create a regular layout which is shown in Figure 3 and is adapted from Figure 3 of [12]. The regular layout of the fat-pyramid is produced by embedding the base butterfly fat-tree into the graph of the tree of meshes, performing a “fold-and-squash” transformation, and adding the mesh connections of the fat-pyramid [12].

The layout now obtained differs from the fat-pyramid layout in the following ways: (1) level 0 will have no switch nodes; (2) level 4 switch nodes are locally connected by a ring; (3) the four processors subtending each of the level 4 switch form a ring subnetwork and have only one node (i.e. the joint node) connected to the level 4 node via a link; and (4) there is only one node in a local level 4 ring that is connected to a level 2 switch. Note that to accommodate capacity splitting, each level 4 ring must have two nodes connecting to two level 2 switches. Also note that we retained the framework of the tree of meshes and added ring links. In the succeeding analysis, we should imagine some of the tree links are “dummy links” which affect only the  $n$  factor.

We shall need to define a linear wire delay condition in order to prove our universality theorem. Let  $w(\delta)$  denote the wire delay function of wire length  $\delta$ . Function  $w$  should be nondecreasing and satisfy the following condition:

**Definition 4.1.** A function  $w$  is said to satisfy the linear delay condition if there exists a constant  $c$  such that

$$\frac{w(\delta x)}{w(\delta)} \leq x^\gamma$$

for all  $x \geq 0$ ,  $\delta \geq c$ , and  $0 < \gamma \leq 1$ .

Similar to the result of [12], the above condition can be met by most functions  $w(\delta)$  likely to be of interest in the context of wire delay such as those in the form of  $c\delta^q \log_2^k \delta$  for constants  $c$ ,  $q (\leq 1)$ , and  $k (\leq 0)$ .

**Theorem 4.4.** *Using transmission lines, a GFS  $F$  of area  $\Theta(A)$  with capacity doubling up the basis network tree can simulate any network of area  $A$  with  $O(\log^{\frac{3}{2}} A)$  overhead under nonunit wire delay assumption.*

*Proof.* Let  $\delta$  be the maximum physical distance that a message of a message set  $S$  travels in the competing network. The number of fat-stack edges which a message traverses is at most  $2 \log_2 \delta$ , plus  $O(n) \cdot 2 \log_2 \delta$

ring edges at each of the  $\log_2 \delta$  levels since on a subnetwork a packet traverses  $O(n)$  edges to get to the joint node. Since any link connected to a switch  $h$  levels up is of length  $O(2^h \sqrt{\log A})$  and each ring edge is of length  $O(\sqrt{\log A})$ ,<sup>1</sup> the routing path connecting processors at (horizontal) distance  $\delta$  in the competing network is of length  $O(\delta \sqrt{\log A})$ . Therefore, each of the  $2 \log_2 \delta$  fat-stack edges should contain at most  $w(\delta \sqrt{\log A})$  real and imaginary switches. (Imaginary switches are auxiliary switches thought placed on each wire in number equal to the delay for that wire. Their inclusion enables us to use the unit wire delay result.) The total distance that a packet travels (hence the dilation) in  $F$  is  $O(w(\delta \sqrt{\log A}) \log \delta)$ .

Now let  $T$  be the time required to deliver  $S$ . We have  $T \geq w(\delta)$ . Also, the congestion caused by  $S$  in  $F$  is  $O(T \log A)$  by the congestion argument in the proof of Theorem 4.3. The routing overhead  $\mu$  can be obtained based on Lemma 4.1 as follows:

$$\begin{aligned} \mu &\leq O\left(\frac{T \log A + w(\delta \sqrt{\log A}) \log \delta}{T}\right) \\ &\leq O\left(\frac{T \log A}{T}\right) + O\left(\frac{w(\delta \sqrt{\log A}) \log \delta}{w(\delta)}\right) \\ &\leq O(\log A) + O((\sqrt{\log A})^\gamma \log \delta) \\ &\leq O(\log A) + O(\sqrt{\log A} \log \delta) \\ &\leq O(\log^{\frac{3}{2}} A), \end{aligned}$$

where the third line follows from the linear delay condition (Definition 4.1). Note that when  $\sqrt{\log A} < 1$ ,  $(\sqrt{\log A})^\gamma < 1$ , and then we can obtain  $\mu \leq O(\log A)$ .  $\square$

**Corollary 4.5.** *Using transmission lines, an AFS and a fat-pyramid  $F$  of area  $\Theta(A)$  with capacity doubling up the basis network tree each can simulate any network of area  $A$  with  $O(\log^{\frac{3}{2}} A)$  overhead under nonunit wire delay assumption.*

*Proof.* For the same  $\delta$  as in the proof of Theorem 4.4, the number of fat-tree edges which a message traverses is still at most  $2 \log_2 \delta$ , but the number of ring edges it traverses is now just 2 for either network. (The absence of  $n$  in the number of ring edges shows that these two networks can be far more efficient than the GFS.) Henceforth, the proof is the same as that for Theorem 4.4. No improvement on the asymptotic overhead is possible.  $\square$

<sup>1</sup>Now each leaf node is a single processor of area  $\Theta(\log A)$  instead of an H-tree block of area  $\Theta(\log A) \times \Theta(\log A)$ .

## 5 Evaluating Distributed Networks

We now discuss how to apply the universality results obtained in the last section to evaluate the performance of some specific distributed networks from a routing perspective.

As already indicated in arriving at the theorems, the asymptotic overhead values are not good representation of absolute efficiency because some intermediate terms are omitted. Since the area of a network is always rather limited in real applications, these less powered terms can represent significant inefficiency of the network. Specifically, the  $n$  coefficient for the GFS can be nontrivial when a subnetwork has a large number of nodes. Secondly, in the derivations for the fat-pyramid, the dilation accounts for the longest possible path a packet has to travel, which turns out to be the same as that for the AFS. But in usual network operation, packets in the fat-pyramid can take the mesh edges between the subnetworks as short cuts which the AFS does not have. It is thus imperative to consider the precision of the overhead data and use them as a qualitative measure.

By using an equal-hardware approach, we showed that the GFS is also universal. Although it is less efficient than the AFS and the fat-pyramid, the GFS is a network that can possibly represent a distributed network. A distributed network is typically of much larger scale than an interconnection network implemented in VLSI. The links of a distributed network dominate its domain while in VLSI the processors and switches determine the densest packing. Thus this disparity has to be addressed in order to scale up the fat-stack. One approach is expanding the network by multiples of the lengths (delays) of the links, which effects an expansion of the area  $A$  of the base and competing networks. As such, the congestion and dilation in the fat-stack will be represented in the same way as in the proofs of Theorem 4.3 and Theorem 4.4, only now  $\delta$  and  $A$  are much larger. In essence this technique leaves out the VLSI densest processor packing condition but obeys the area constraint.

The GFS is likely the most efficient among practical distributed network architectures as to be shown in Section 6. It is not yet clear how much wire lengths (delays) affect the universality and efficiency of a network. But it seems reasonable to assume that varying wire delays can have only marginal effect on network efficiency. With these two premises, we can designate the fat-stack as a benchmark to evaluate other network topologies. The tiered topology implied in IPv6 resembles a GFS. This

means that the assumption of the Internet as a tiered network structure either logically or physically is fundamentally sound. Enterprise networks should also be deployed with fat-stack configurations. An overlay network that deviates from the fat-stack structure is susceptible to performance degradation. A peer-to-peer network can not exploit the fat-stack results because its topology is dynamic and ephemeral. In all these applications, it is either the physical network itself or the paths for the aggregate traffic within a network domain that assume a fat-stack structure.

## 6 Experimental Results

We did several experiments using the *ns-2* network simulator to visualize and compare the performance of the networks discussed so far and a fifth network in realistic terms. There are two fixtures in the experiments. The nodes are considered as fixed on a template, and wiring the various links for each network does not bloat the template. This makes it easy to stipulate that the networks are of the same area. Secondly, a fixed traffic scenario of three traffic types is used to run through all the networks in order to obtain data on how well each performs.

We simulated the fat-tree, the AFS, the fat-pyramid, the GFS, and the fifth network. The fifth network is composed of linked small meshes and is referred to as the *composite-mesh* in the following discussions. This network is illustrated in Figure 4. In the general template as shown in Figure 4, there are 64 processor nodes (nodes 0 – 63) and 21 switches as required by a 4-node subnetwork configuration.

The split wires and nodes used previously in discussing the fat-tree, the AFS, the fat-pyramid, and the GFS are collapsed into singular links and nodes with composite capacities. Link connections differ among the networks. However, their topologies can all be easily fitted into the template without violating the area constraint. Nodes 0 – 63 act as processors only in the fat-tree but as switches as well as processors in the other networks. They are connected within a ring in the fat-stack and a mesh in the fat-pyramid. The links are modeled as duplex links having a delay of 10 ms. This implies that the wires have varying (nonunit) delays and varying length as the materials of the wires could vary.

From the processors to the switch at the top (node 84) in the first three networks, there are four tiers. The GFS has three tiers less the top node. In each tier, links in the subnetworks and the upward links are assigned the

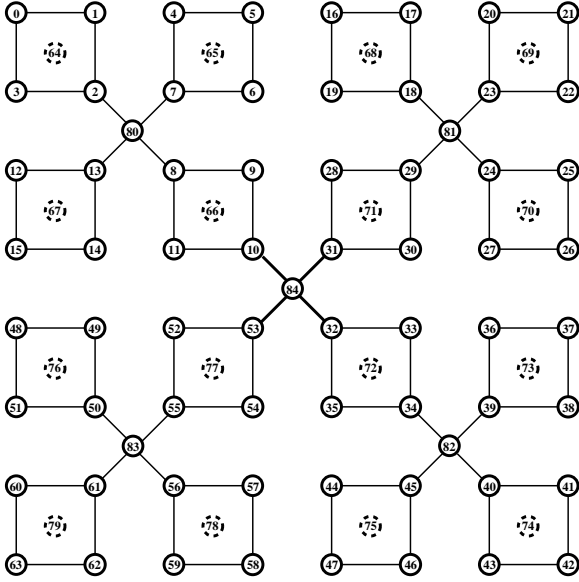


Figure 4: Topology of the composite-mesh distributed network. The switches encompassed by the processor subnetworks are not used and are shown as dashed circles.

same capacity. Upwards across the tiers, link capacities are 0.1, 0.2, and 0.4 Mbps successively, i.e. they double up the underlying 4-ary tree. For the composite-mesh, there are no tiers. Link capacity for the links connecting four bottom meshes is set as 0.1 Mbps, whereas the links from the center node (node 84) to the subtending meshes are set to be 0.25 Mbps in bandwidth.

The composite-mesh does not use the switches on the bottom layer subnetworks (nodes 64 – 79). Their effect on routing is immaterial since each of them would otherwise be confined to a local subnetwork and have one or more links to the subnetwork. Also note that the GFS does not include node 84 unlike the composite-mesh.

A simplistic communication model was used in all of the simulations. We used UDP as the transport protocol, so in effect the simulations merely emulate IP routing. We used the adaptive link-state routing as the routing protocol so that routing responds to traffic loads on the links. Each network was simulated under all three traffic types. The duration of the simulations is all 120 seconds. During this period, 40 processor nodes generate traffic at start and end times generated at random. Traffic sinks are also randomly selected among the 64 processor nodes. There are a total of 60 connections or flows created during each simulation. The flows start and end at random times within the simulation duration.

The types of the traffic generated at the sources are of

constant bit rate (CBR), exponential on/off, and Pareto on/off. The common attributes of the traffic types are that the packet size is 1,000 bytes and the average sending rate is 80 Kbps. Other attributes vary with the latter two types having bursty sending patterns. The queuing rule applied for all links is Stochastic Fair Queuing with default configuration parameters (maximum queue limit set at 40 packets and using 16 buckets for hashing each of the flows).

We extracted from the results of experiments the overall throughput and delay as the specific traffic moves through each network. They are calculated by obtaining the start time and the duration that each packet stayed in the network before arriving at its final destination. We use them as the metrics for comparison. The data are listed in Table 1.

Table 1: Simulation Results

Network	Traffic Type	Throughput (bps)	Delay (sec)
Fat-tree	CBR	928113.333333	0.763641
	Expon.	917320.000000	0.841578
	Pareto	875986.666667	0.780242
AFS	CBR	1126821.333333	0.194550
	Expon.	1130601.333333	0.230433
	Pareto	1097736.000000	0.215983
Fat-pyramid	CBR	1652286.666667	0.146700
	Expon.	1624204.000000	0.169914
	Pareto	1589376.000000	0.162286
GFS	CBR	642316.000000	0.869150
	Expon.	555161.333333	0.536892
	Pareto	623461.333333	0.834266
Comp.-mesh	CBR	519420.000000	1.258450
	Expon.	452820.000000	0.938238
	Pareto	505226.666667	1.238498

The asymptotic nature of the simulation overheads we derived in Section 4 does not warrant a direct comparison of these numbers in terms of  $O(\log A)$  although it is inferable that  $\log A$  corresponds to  $\log N$  ( $A$  is greater than  $N$ ). Therefore, without obscuring our analytical results, we can rate the performance of the networks relative to each other.

The data show that throughput increases and delay decreases in the order of the fat-tree, the AFS, and the fat-pyramid. This is because more and more intra-subnetwork links are added in that order. The AFS performs better than the fat-tree by  $\sim 23\%$  in throughput and  $\sim 73\%$  in overall delay. (It is less efficient than the fat-pyramid by  $\sim 31\%$  in throughput and  $\sim 34\%$  in overall delay. But note that the fat-pyramid uses much more wires and thereby incurs more complexity for wiring and packaging. In addition, the fat-pyramid does not scale

to represent a distributed network.) By its affinity to the AFS, the performance of the GFS can increase considerably if additional links are added up from each subnetwork. The AFS will be the upper bound of this increase. In fact, these networks provide a range of performance attainments.

The GFS is one that resembles a distributed network. Each subnetwork has only one link to an upper tier node. Each of the subnetworks in the fat-tree, the AFS, and the fat-pyramid has four upward links, which makes them impractical for distributed layout. A distributed network is apt to resemble the composite-mesh in actual deployment. The data show that the GFS performs much better than the composite-mesh ( $\sim 23\%$  in throughput and  $\sim 35\%$  in delay). So it is reasonable to infer that the performance of a composite-mesh can be much worse if the network is at a much larger scale, i.e. if it consists of much larger number of processor nodes.

## 7 Conclusion

In this paper we have shown that the fat-stack is universally efficient. The general fat-stack variant incurs some unsurmountable delay as each of its subnetworks has only one upward link, which results in a simulation overhead of  $O(\log^{\frac{3}{2}} A)$ . However, the general fat-stack can be scaled up and outperform a (or any) mesh-based distributed network. We showed how to apply the universality results to evaluate the performance of some specific distributed networks.

The regular layout used to prove the universality of the general fat-stack imposes a restriction on the wire lengths (Figure 3). In that layout, any link connected to a switch  $h$  levels up is of length  $O(2^h \sqrt{\log A})$  and each ring edge is of length  $O(\sqrt{\log A})$ . The scaling method we proposed in Section 4 still abides by the area limitation and uses extension by multiples. In the simulations, we used a template which in effect bypasses this restriction. How much varying wire length and capacity affects routing efficiency is unresolved. It would be desirable to study this impact. Results would be particularly useful for network scaling with good precision.

## References

- [1] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosn. Adaptive packet routing for bursty adversarial traffic. *Journal of Computer and System Sciences*, 60(3):482–509, 2000.
- [2] M. Andrews, B. Awerbuch, A. Fernandez, T. Leighton, Z. Liu, and J. Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, 2001.
- [3] P. Bay and G. Bilardi. Deterministic on-line routing on area-universal networks. *Journal of the ACM*, 42(3):614–640, May 1995.
- [4] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer System Sciences*, 28:300–343, Apr. 1984.
- [5] S. N. Bhatt and C. E. Leiserson. How to assemble tree machines. In F. P. Preparata, editor, *VLSI Theory*, volume 2 of *Advances in Computing Research*, pages 95–114. JAI Press, 1984.
- [6] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, 2001.
- [7] A. Borodin, R. Ostrovsky, and Y. Rabani. Stability preserving transformations: Packet routing networks with edge capacities and speeds. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 601–610, New York, Jan. 7–9 2001. ACM Press.
- [8] A. Borodin, R. Ostrovsky, and Y. Rabani. Stability preserving transformations: Packet routing networks with edge capacities and speeds. *Journal of Interconnection Networks*, 5(1):1–12, Mar. 2004. Journal version of [7].
- [9] A. Z. Broder, A. M. Frieze, and E. Upfal. A general approach to dynamic packet routing with bounded buffers. *Journal of the ACM*, 48(2):324–349, 2001.
- [10] K. F. Chen and E. H.-M. Sha. The fat-stack and universal routing in interconnection networks. In *Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems*, San Francisco, CA, Sept. 2004. To appear.
- [11] R. I. Greenberg. *Efficient Interconnection Schemes for VLSI and Parallel Computation*. PhD thesis, Massachusetts Institute of Technology, Aug. 1989. MIT/LCS/TR-456.

- [12] R. I. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Transactions on Computers*, 43(12):1358–1364, Dec. 1994.
- [13] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988. Also see erratum in vol. 1, no. 3, p. 315.
- [14] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 345–374. JAI Press, 1989.
- [15] R. I. Greenberg and H.-C. Oh. Universal wormhole routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(3):254–262, Mar. 1997.
- [16] J. T. Hung and T. G. Robertazzi. Scalable scheduling for clusters and grids using cut through switching. *International Journal of Computers and Their Applications*. In press.
- [17] S. Kumar and L. V. Kale. Scaling collective multicast on fat-tree networks. Technical Report 03-11, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2003.
- [18] F. T. Leighton. A layout strategy for VLSI which is provably good. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 85–98, May 1982.
- [19] 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, 1994.
- [20] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in  $O(\text{congestion} + \text{dilation})$  steps. *Combinatorica*, 14(2):167–186, 1994.
- [21] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.
- [22] T. Leighton, B. Maggs, and A. W. Richa. Fast algorithms for finding  $O(\text{congestion} + \text{dilation})$  packet routing schedules. *Combinatorica*, 19(3):375–401, 1999.
- [23] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, Oct. 1985.
- [24] C. E. Leiserson. VLSI theory and parallel supercomputing. In C. L. Seitz, editor, *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pages 5–16. MIT Press, 1989.
- [25] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak. The network architecture of the connection machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285. Association for Computing Machinery, 1992.
- [26] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.
- [27] Y. Rabani and É. Tardos. Distributed packet switching in arbitrary networks. In *the 28th ACM Symposium on Theory of Computing*, pages 366–375, Philadelphia, PA, USA, May 22–24 1996.
- [28] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, Sept. 1995.
- [29] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surama. Internet Indirection Infrastructure. *IEEE/ACM Transactions on Networking*, 12(2):205–218, Apr. 2004.
- [30] V. Strumpfen and A. Krishnamurthy. A collision model for randomized routing in fat-tree networks. Technical Memo MIT-LCS-TM-629, Laboratory for Computer Science, Massachusetts Institute of Technology, 15 July 2002.