

The fat-stack and universal routing in interconnection networks

Kevin F. Chen, Edwin H.-M. Sha

*Department of Computer Science, University of Texas at Dallas, Richardson, TX
75083, USA*

Abstract

This paper shows that a novel network called the *fat-stack* is universally efficient and is suitable for use as an interconnection network in parallel computers. A requirement for the fat-stack to be universal is that link capacities double up the levels of the network. The fat-stack resembles the fat-tree and the fat-pyramid in hardware structure, but it has unique strengths. It is a construct of an atomic subnetwork unit consisting of one ring and one or more upward links to an upper subnetwork. This simple structure entails easy wirability. The network also uses fewer wires. More importantly, it has the capability to scale up to represent a large-scale distributed network. We developed efficient routing algorithms specific to the fat-stack. Our universality proof shows that a fat-stack variant with increased links and of area $\Theta(A)$ can simulate any competing network of area A with $O(\log A)$ overhead independently of wire delay. The universality result implies that the augmented fat-stack of a given size is nearly the best routing network of that size. The augmented fat-stack is the minimal universal network for an $O(\log A)$ overhead in terms of hardware usage. Actual simulations show that the performance of the augmented fat-stack approaches that of the fat-pyramid and is far higher than that of the fat-tree.

Key words: fat-stack, interconnection networks, universality, VLSI

1 Introduction

A parallel computer can be comprised of thousands of processors with local memory that are connected by an interconnection network. The network must be efficient to ensure computing speed and cost-effectiveness. An efficient network should move traffic fast for computing tasks and require as little hardware

Email address: edsha@utdallas.edu (Edwin H.-M. Sha).

as possible. The network's speed is determined largely by how fast it can route packets among its nodes. In this paper we show that the *fat-stack* is such an efficient network by showing that it is *universal*. Being universal means that the network can simulate, in terms of routing, any other network with an overhead of no more than (some power of) the logarithm of the area of the hardware containing the network. Furthermore, we show that the fat-stack is the *minimal* universal network for a given simulation overhead, that is, it can simulate any other network with the least amount of hardware at given speed. This universality result implies that the fat-stack performs much better than or as well as most, if not all, of known networks. We also show that the fat-stack has good scalability. We formally define the fat-stack network in Section 2.

Universal networks have been studied in the past. Work on the fat-tree and the fat-pyramid has initiated some principles and methodologies for designing universal routing networks [1–3]. Our initial motivation of this research is to determine whether the theory so far acquired in the interconnection network domain applies to large-scale distributed networks. 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 this paper, we report the results of the fat-stack as an interconnection network for parallel computers. This work also provides a solid foundation for studying the fat-stack's application in distributed settings. We prove the universality of an augmented fat-stack using similar VLSI layout and wire delay conditions as those in [1–5].

A typical fat-tree layout assumes a complete 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 are described in Section 4. The fat-tree is the first proved universal network. The proof was done by Leiserson under unit wire delay assumption [1]. Later work has reaffirmed Leiserson's result [2, 3, 6, 7]. But the fat-tree is universal only under unit wire delay condition. According to Greenberg [3], its universality does not hold under nonunit wire delays. Early research could not determine universality under nonunit wire delays, but Greenberg figured out a way to do it. Greenberg proved that the fat-pyramid is universal under both unit and nonunit wire delay conditions [3]. Compared to the fat-tree, the fat-pyramid also has better absolute efficiency due to its hierarchical meshes. But these same meshes of the fat-pyramid reduce its scalability, increase its wire usage considerably, and make it not scalable to represent a distributed network. Therefore, the fat-stack has advantages over both the fat-tree and the fat-pyramid. The fat-stack, unlike the fat-tree, is universal under both unit and nonunit wire delay conditions, and it is much more scalable than the fat-pyramid. (To date, the fat-tree has been used in the CM-5 parallel computer,

whereas the fat-pyramid has not been adopted for any machine.)

Universality, however, does not directly depend on scalability. Scalability is determined by connectivity and hardware usage. Universality is a property typical of a given scalability. As a network structure implies a scalability, we need consider only the network hardware dimensions to prove its universality. Universality proofs for the fat-tree and 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 [1, 3]. This method has its basis in the theories and constructions of VLSI graph layouts [8–10]. We shall adopt this method in our proofs. In applying the bisecting method, we analyze the networks in terms of their areas. Extension of computation analysis from two dimensions to three dimensions has been shown to be straightforward [11, 12].

Network computation consists of routing and nodal support. In this work we consider routing as the only measure of network efficiency. In doing so, we assume that computational power of nodes is arbitrarily extensible to keep up with any demand by routing. Throughout the paper, we say that network A can simulate network B with overhead μ if, for any t , the *routing* performed by B in time t can be performed by A in time μt .

Routing schemes have direct impact on the universality of a network. We develop efficient routing algorithms specific to the fat-stack. These algorithms are of practical value. We did not include them in a preliminary version of this work [13]. To prove the fat-stack’s universality, we use a much acclaimed theorem for packet routing in general networks proved by Leighton et al. [7, 14–16]. The theorem provides a measure of network efficiency convenient to use in proofs. The universality of the fat-stack thereby relies on routing capability in terms of a linear combination of congestion and distance that a packet travels.

Besides efficiency, we are to show that the fat-stack also has good scalability. Scalability is determined by wire usage and connectivity. The fat-stack uses fewer wires and its link connections have higher locality than the fat-pyramid. The fat-stack is relatively simple in structure. We construct it by stacking atomic subnetwork units following a fat-tree framework. The common subnetwork unit is made of a ring of 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 recursively.

Depending on the number of up links from a subnetwork, the fat-stack structure can vary. We consider two variants of the fat-stack in this paper. The *general fat-stack* (GFS) has 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 links. The *augmented fat-stack* (AFS), the main focus of this paper, has as many upward links as the number of nodes in the subnetwork. These two variants allow us to construct networks useful for parallel computers and as distributed networks. Both variants are composed of few wires, have local connectivity, and therefore are highly scalable.

In addition to scalability, another advantage of the fat-stack over the fat-pyramid is that the AFS is the minimal universal network for the same asymptotic overhead of $O(\log A)$ under both unit and nonunit wire delay conditions, where A is the area of the hardware containing the network. This lower bound network notion is in terms of hardware usage.

While much of our work focused on algorithm design and analytical proof, we also simulated the AFS together with the fat-tree and the fat-pyramid to visualize and compare their performances in realistic terms. The simulations were carried out using the *ns-2* network simulator. The simulations demonstrate that the AFS has a high performance improvement over the fat-tree and approaches the fat-pyramid in efficiency. The AFS can be adopted as a network structure when the computing tasks require no more than the capacity that the AFS can offer.

Our contributions are thus the following: (1) We added another universal network class into the repertory of known universal networks. (2) The AFS is the minimal universal network under nonunit wire delay condition with $O(\log A)$ overhead. (3) The fat-stack is easier to construct and easier to scale than the fat-pyramid. (4) The GFS can scale to the size of a distributed network, so analysis results would be valuable to evaluate the performance of large-scale distributed networks. (5) We designed efficient routing algorithms for the fat-stack which could be implemented for practical use.

The remainder of this paper is organized as follows. Section 2 describes the precise graph-theoretic topologies of the fat-stack, an addressing scheme, and the postulated computation and communication model on which routing analysis is based. In Section 3, we establish routing algorithms for a set of routing scenarios in the fat-stack. The routing capability implied by these algorithms forms the basis for universality proofs. Section 4 contains the universality proofs for an AFS. The AFS universality indicates its direct application as an interconnection network with the same processor packing and hardware layout as the fat-tree and the fat-pyramid but with differing mesh connection in subnetworks. Section 5 describes the experiments we did and the results obtained in evaluating the routing performance of the fat-stack against a fat-tree and a fat-pyramid. We analyze and discuss the algorithmic, analytical, and experimental results in Section 6. Section 7 contains concluding remarks.

2 Network model

We now describe the precise graphic-theoretic topology of the fat-stack, an addressing scheme, and a computation and communication framework on which routing is based.

The fat-stack¹, a hierarchical network, consists of tiers or levels (see Figure 1). Each level has one or more subnetworks. Each subnetwork is a ring. One link connects a subnetwork to an upper level from a lower node to an upper node. A fat-stack can have arbitrary levels of rings. In our analysis of the fat-stack application as an interconnection network, we use the AFS variant of the fat-stack which is shown in Figure 1(a). We refer to the GFS variant (Figure 1(b)) in developing routing algorithms. Although an AFS contains an n -ary tree, the GFS does not. In an AFS, each non-leaf node represents a ring of switches and a leaf node represents a processor.

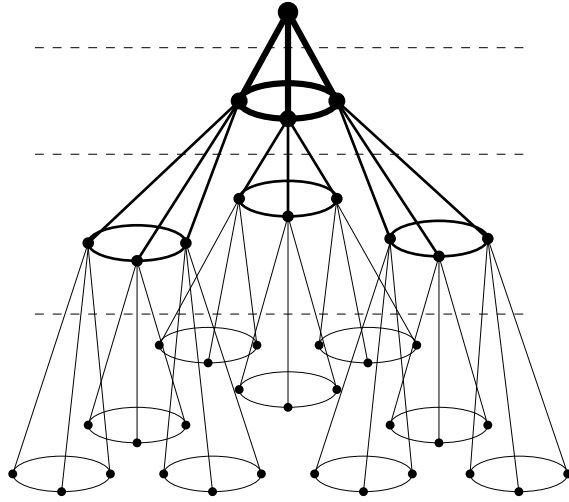
The structure of the fat-stack differs significantly from the structures of the fat-tree and the fat-pyramid in that the fat-stack can be viewed as a stack-up of basic subnetwork units of a common structure containing a ring and one or more upward links.

The simple structure of the AFS entails easier wiring and packaging than the fat-pyramid. The network also uses fewer wires than the fat-pyramid. For a 4-ary layout, the fat-tree uses $2N - \sqrt{N}$ wires, the AFS uses $N/2 - \sqrt{N}$ more wires than the fat-tree, and the fat-pyramid uses $N/2 - \sqrt{N} \log_4 N$ more wires than the AFS, where N is the number of processors. In the layout, each leaf node is a single processor and the link capacities double up the levels.

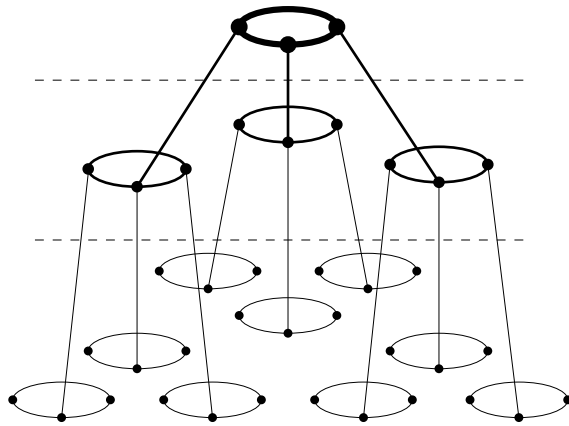
The connectivity of a GFS can be given in the following addressing scheme. Let n be the number of nodes in a subnetwork, and N the number of processors at the bottom level. Nodes in a subnetwork at a level is numbered in sequence of 0 through n , where node 0 (a joint node) always connects the subnetwork to the upper layer. The numbering is sequential (clockwise) from the joint node. Each node's number is the same as the number of the subnetwork subtending to this node. This sequential numbering scheme allows flexibility to cover varying number of nodes in the subnetwork no matter what topology the subnetwork assumes.

Nodes in the top tier is numbered 0 to n , and represented by a uni-tuple in the form of (0). A subnetwork of the next lower tier connecting to a node of the upper tier can be regarded as having been assigned a number the same as the upper node number. The address of a node u in a lower subnetwork at

¹ 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.



(a) AFS



(b) GFS

Fig. 1. Topologies of the fat-stack. Each subnetwork has three nodes. Dashed lines represent tier boundaries. In (a) every node has a link to a common node at its next upper level. In (b) each subnetwork connects to its upper level node by a single link.

level l will be a l -tuple composed of the numbers of u and the nodes at upper levels to the top level through each of which a lower subnetwork is connected that leads to u . For example, $(1, 2, 2)$ is the address of node 2 of a subnetwork at level 3 that connects to node 2 of a subnetwork at level 2 that connects to node 1 of the top level subnetwork. The cardinality of the address of an node ranges from 1 to the height of the tree.

By convention, an edge in the network graph corresponds to a channel with certain capacity. A channel is composed of a certain number of wires. Each wire has one unit of capacity. This notation bears well on hardware specification in interconnection networks.

The capacity specification common in the fat-tree and fat-pyramid analyses provides an alternative representation of capacity. In both cases, each edge represents a wire. A wire has one unit of capacity. To increase capacity, extra edges and nodes of the same capacity are created. Then each duplicate of a node and an edge can be regarded as a split of a node and a link of larger capacity of an integer times. If a reverse process of the split and duplicate is performed, the fat-tree and the fat-pyramid can all be reduced to a singular structure resembling that of the AFS shown in Figure 1(a). We will illustrate and use this representation in Section 4.

Whichever the representation, capacity distribution is fixed. The capacities of the channels and the nodes of a fat-tree are defined as doubling from one level to the next up the tree and growing a little less than doubling at levels close to the root [1]. The normal form of the fat-pyramid doubles the capacities up the tree [3]. The butterfly fat-tree which is what the fat-pyramid is based on has also a prefixed capacity distribution [2]. In proving the AFS' universality, we assume that link capacities double up the levels of the network.

As indicated above, an edge and a node of an fat-stack can have various capacities assigned. If the links and nodes all have the same capacities such as 1, the network will be homogeneous. As demonstrated for the fat-tree in [17], 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.

Not only are edges a critical consideration, routing model is also crucial. We employ an abstract model of packet routing. In this model, routing is done completely in parallel threads, each tantamount to a single atomic switch unit at a node. The atomic switch has one input-port and one output-port with the minimal capacity necessary for routing. Similarly, each link on an upper tier is modeled as a composite bundle of minimal-capacity wires.

Packet routing is part of the computation model. The basic mode of operation assumed of the fat-stack is the usual distributed random-access machine (DRAM) model [18]. 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.

3 Routing on the fat-stack

In this section, we establish some routing algorithms specific to the fat-stack under unit wire delay assumption and prove some theorems indicating their routing efficiency. The routing algorithms use schemes that work both online and offline. We consider queuing delays and distances that routed packets travel. We attempt to establish the most efficient algorithms possible.

We assume unit link capacity throughout the network. Extensions to doubled link capacities up the network should be straightforward.

Unit wire delay denotes that it takes unit time for a packet to move through a wire. It simplifies establishing the lower bounds of throughput, bisection, and connection availability. 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.

Delay at a switch is caused by how fast a packet can be processed and put onto the output wire, which in turn is a function of the packet queue length. The halting of a packet at a node is usually larger than wire delays provided that the wires are not too long.

In the GFS, the location of the source and destination nodes decides a locality property (Figure 1(b)). For nodes 1 and 2 indexed by i , let n_i represent the numbers of nodes in the subnetwork they are in and a_i their address tuples in the form of $(c_i^1, c_i^2, c_i^3, \dots)$. The distance a packet needs to travel to go from node 1 to node 2 is at least

$$d(1, 2) = \sum_{i=1}^2 [(|c_1^1 - c_2^1| \bmod n_i + \lfloor |c_1^1 - c_2^1| / n_i \rfloor) + (\sum_{j=1}^{|a_i|-1} c_i^j + |a_i| - 1)]$$

The diameter of the network is therefore $\max[\forall i \forall j d(i, j)]$, where i and j are any two leaf nodes of the entire network.

In view of the locality feature, we have the following theorem.

Theorem 1 *The number of link and subnetwork levels a packet travels from source node u to destination node v is at least $\log_n N - \min_{0 \leq i \leq \max(|a_u|, |a_v|)} (c_u^i \neq c_v^i)$, where N is the number of leaf nodes and c_u^i or c_v^i takes the value of null if it does not exist.*

By this theorem, when $\log_n N$ is one, packet movement is confined to a network

that has only leaf nodes. If all the packets are local, upper levels are not required. This applies to all the levels of the network.

We first demonstrate efficient routing within a ring subnetwork where at each node there is a number of packets queued up and all the packets are to be routed to the same node. In all of the following lemmas and theorems we use the same notations. Let n be the number of nodes in a subnetwork. Let q_i be the number of packets queued at node i , where $0 \leq i \leq n$, and $q = \max_{0 \leq i \leq n} (q_i)$.

Lemma 2 *In a ring subnetwork R of a fat-stack, there is a routing schedule of length $O(nq)$ requiring a maximum queue size of $\Theta(q)$.*

PROOF. Consider the situation in which all the packets are routed to the same node and sent out of R along only one outgoing link at that node. Without loss of generality, assume that the joint node (node 0) is the sink node. The scheduling uses the following algorithm. At each time step, allow one half of the n nodes in R to transmit, i.e. the nodes of the two halves of R takes turns to transmit and receive. Each node sends a packet along the shortest path towards node 0 if its queue is not empty. The passing of packets takes $O(nq)$ time steps and requires a maximum queue size of $\Theta(nq)$. To see that this algorithm works, consider cutting R next to node $n/2$ if n is even and between node $\lceil n/2 \rceil$ and node $\lfloor n/2 \rfloor$ if n is odd. If n is even, route the packets of node $n/2$ to either left or right. Nodes to the left of the cut point route packets clockwise, and nodes to the right route packets counterclockwise. At each time step, allow node 0 and one half of the nodes in R to transmit. Each node sends out a packet if its queue is not empty, and receives the packet from its neighbor that is upstream and has sent out a packet two steps back. There are $O(nq)$ packets to be sent out the ring, and the queue at each node does not grow.

The routing algorithm used in Lemma 2 entails certain flow control to prevent nodes from overwhelming a node. The following corollary can be easily verified.

Corollary 3 *There is a greedy scheme that takes the same amount of time ($O(nq)$) but requires a queue size of $O(nq)$ at the sink node.*

We now show how much latency is incurred and how latency can be mitigated when packets at each node are to be routed to some random nodes. The packets at a node can be from the lower or upper level subnetwork. We begin by assuming that there is a queue at each node which contains a certain number of packets, and packets routed to the destinations are “consumed” there – they are sent out of the subnetwork on outgoing links.

Lemma 4 *In a ring subnetwork of a fat-stack, using the greedy routing scheme, barring that the packets at a node go out of the ring from the downward link at the node, there is a worst case routing schedule of length $O(nq)$ requiring a maximum queue size of $O(nq)$.*

PROOF. This is equivalent to proving that Corollary 2 is the worst case. Observe that a link is duplex and that packets rotate continuously along the ring path. The subnetwork can be seen as a system with its outgoing links as sinks. Given nq packets existing on the ring, if the packets are to exit through a single outgoing link, the worst case schedule length is obviously $O(nq)$. For the queue size, for routing from a node to some arbitrary number of nodes, the packet passing can be envisioned as if the ring is composed of a few logical rings at each time step, which do not interfere with one another. Possible collision occurs and hence queuing is needed only when pairs of nodes at the same distance from the sink node on each side of the sink node transmit at the same time. Assume that there is one node in the logical ring that does not send packets to the sink node, then by our greedy routing scheme, the transmission queue size will be $O(nq)$. Thus, Corollary 3 has the worst case queue size as well.

Theorem 5 *For a ring subnetwork, without flow control, the corresponding joint node has an upper bound of $O(nq)$ on its queue size and an upper bound of $O(nq)$ on its transmission time.*

PROOF. The theorem follows from Corollary 3 and Lemma 4.

Lemma 2, Corollary 3, Lemma 4, and Theorem 5 apply to both the GFS and the AFS. Figure 2 illustrates how routing works in the flow-controlled and the greedy cases. The circles labeled 0 through 3 represent nodes (switches and processors). The squares represent packets which are numbered sequentially. All packets go to the same destination (joint node 0). Figure 2(a) shows the initial state (step 1).

Figure 2(b) and Figure 2(c) indicate how packets move around by the flow-controlled scheme. In step 2, packet 1 moves out, packet 4 moves to node 0, and packet 5 moves to node 1. Step 3 is the turn of the other half of the ring (node 0 and node 3), in which packet 2 gets out, packet 7 gets to node 0 and is queued there. In the process, no queue increases in length.

Figure 2(d) and Figure 2(e) show how packets move around and queue length increases at the joint node (node 0) for the greedy scheme, starting from the state depicted in Figure 2(a). In step 2, all nodes transmit. Node 0 sends out

packet 1. Packets 4 and 7 get to node 0. Packet 5 gets to node 1. In this manner, at step 3, the queue at node 0 has increased in length ($= O(nq)$).

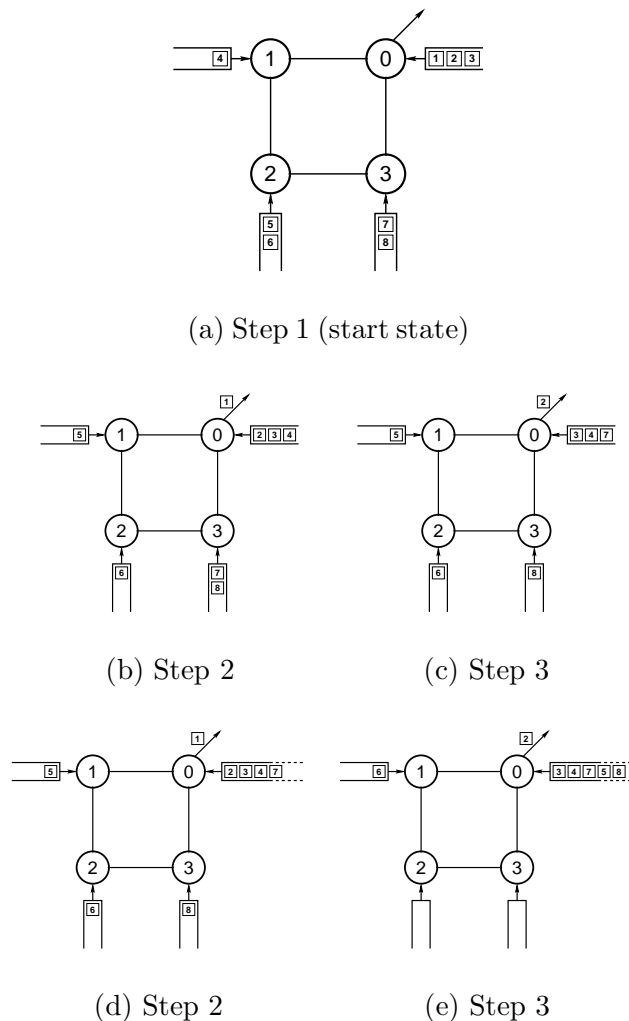


Fig. 2. Routing in a ring subnetwork of the fat-stack. Both schemes start from the same state (a). The flow-controlled scheme is shown in (b) and (c); the greedy scheme is shown in (d) and (e).

We now analyze how much delay can occur over an entire GFS when each processor sends packets to different destinations. Again we focus on the worst case scenario. Each of the nodes in the subnetworks of the bottom level of the GFS has a queue that is within the length of q .

Theorem 6 *By the worst case greedy routing scheme, the routing schedule length of packets traversing the entire network of a GFS is $O(N + n \log_n N)$ and the queue length required is $O(1)$.*

PROOF. The joint node in each subnetwork will need a queue of size $O(nq)$, according to Theorem 5. Although the queue length at each processor may be

q , at each time step there can be no more than one packet being sent out. In effect we are considering that each such node has only one packet.

At each upper level i including the top level, the queue length at the joint node and at all top level nodes is, in the worst case, $O(n^i)$, whereas it is $O(1)$ at other nodes. This follows from Theorem 5 now that the queue length is $O(1)$ at each non-joint node and $O(n^i)$ at the joint nodes and the top level nodes. A packet goes from a bottom node to another bottom node, traversing the entire height ($\log_n N$) of the underlying n -ary tree twice. In the subnetwork at each level i a packet may traverse $O(n)$ extra edges and may be delayed at most $O(n^i)$ steps due to existing queue. The delayed steps caused by the queues at all the levels of the network amount to N which is the sum of the first $\log_n N$ terms of a geometric series with a common denominator n . Note that packets travel through the tree in a completely parallel manner because the edges are bidirectional (duplex links). To make the queues to be $O(1)$, we can just view the schedule as one executed packet by packet, i.e. by spreading out the queued packets.

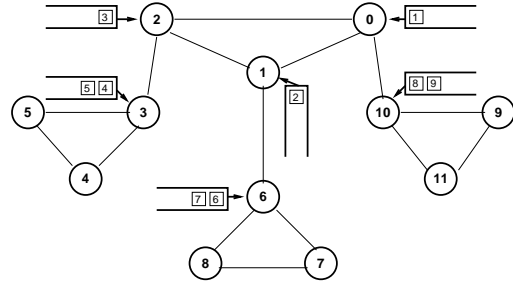
By the above analysis, there can be no more than $2n$ packets traversing on the duplex downward links of the top level subnetwork at one time. Packet passing is restricted by the fixed number of nodes on the top level. Lower levels before the bottom level impose similar constrictions.

We give an example to show how routing works out on a GFS network. Figure 3 uses the same symbols and notations as Figure 2. Figure 3(a) represents a starting state (step 1). Figure 3(b) and Figure 3(c) show packet movement in the next two time steps after step 1. Since the routing scheme is greedy, all nodes move a packet out. Packets that have been moved downward are no longer shown. Queues at the joint nodes would become longer. The queue at node 1 would eventually be as long as N .

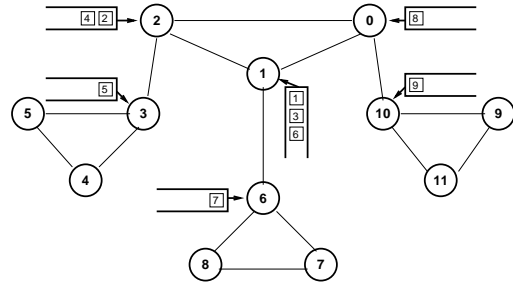
We now consider the effect of increasing the number of links between subnetworks at different levels of the GFS. We show that the AFS can be more efficient. In an AFS, every node in a subnetwork is linked to the same node on the immediate upper level (Figure 1(a)).

Theorem 7 *In a ring subnetwork of an AFS, using the greedy routing scheme, barring that the packets at a node go out of the ring from the downward link at the node, there is a worst case routing schedule of length $O(nq)$ requiring a maximum queue size of $O(nq)$.*

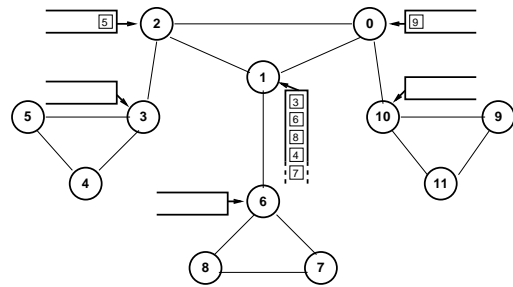
PROOF. The motion of packets incurs the most delay when all the nodes except the sink node transmit packets towards a single node on the same subnetwork, just as in the situation described in Lemma 4.



(a) Start state (step 1)



(b) Greedy scheme (step 2)



(c) Greedy scheme (step 3)

Fig. 3. Routing on a GFS. Routing starts from step 1 shown in (a). Steps 2 and 3 are shown in (b) and (c).

We are most interested in the delay of packets traversing the entire network from the bottom level nodes to some destination nodes.

Theorem 8 *By the worst case greedy routing scheme, the routing schedule length of packets traversing the entire network of an AFS is $O(N + \log_n N)$ and the queue length required is $O(1)$.*

PROOF. The proof is similar to that of Theorem 6. A packet at node u destined to node v takes the path that passes through the upper node of the

subnetwork R that connects u and v instead of R itself. It traverses the tree height twice to move from one leaf node to another. So the total distance a packet travels is at most $O(\log_n N)$. Also, the longest queue a packet encounters along its path is $n^{\log_n N}$ ($= N$).

For examples to show how routing works out on an AFS, please refer to Figure 2 and Figure 3 that illustrate routing on a ring subnetwork and on a GFS. Figure 2 applies to the AFS, whereas Figure 3 shows scenarios similar to an AFS.

4 Universality of the AFS

In this section, we prove the universality of the AFS with increased link capacities. The added capacities are necessary for the network to be universal. The only limitation to the universality is that the competing network has to have the same total area as the AFS. The AFS can be compared to the fat-tree and fat-pyramid networks and hence enables us to prove its universality following a similar approach. Its universality makes the AFS a good candidate as the interconnection network for a general-purpose parallel computer.

Our proofs will be based on the general and powerful routing results obtained by Leighton et al. [7, 14–16]. Their results pertain to offline algorithms that work under unit wire delay condition. 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 9 ([15]) *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 $O(1)$.*

To compare with the fat-tree and the fat-pyramid, the AFS will assume a similar hardware configuration as follows: (1) it is based upon a 4-ary tree in singular representation; (2) the capacities of nodes and links are modulated as doubling up the tiers; (3) we assume that the leaf nodes are not connected with each other and each leaf node is an H-tree layout of $\log_2 A$ processors each of area $\Theta(\log A)$; and (4) a procedure of “split-and-duplicate” of nodes and links is performed on the structure shown in Figure 1(a) to obtain an adequate interconnection layout.

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 [1–3, 6, 7], and that the fat-pyramid can achieve a logarithmic efficiency under a nonunit (linear) wire delay model [3]. There has been some variation in the exact number of links and the overall connectivity implemented in the models of the fat-tree since its universality was first proved by Leiserson [1]. We refer mainly to the results proved by Greenberg along with those for the fat-pyramid [3]. The fat-pyramid is based on superimposing hierarchical mesh connections on a butterfly fat-tree.

We first establish the similarity between a typical butterfly fat-tree and an AFS and prove the universality of the AFS under unit wire delay assumption. We then prove that the AFS is universal under nonunit wire delay condition due to its added ring connectivity. To facilitate our discussion, we illustrate the fat-tree, the fat-pyramid, and the AFS together in Figure 4, which is adapted from Figure 1 of [3].

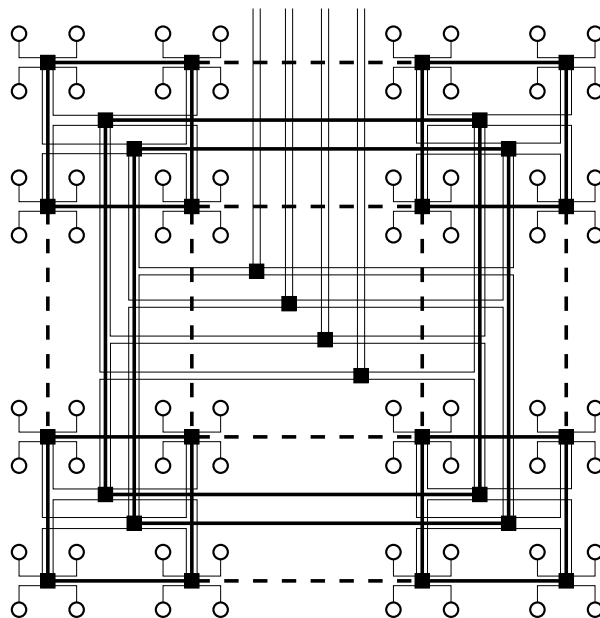


Fig. 4. A fat-tree, a fat-pyramid, and an AFS in one layout. Processors are represented by circles; the squares are switches. The fat-tree links are represented by thin lines, the fat-pyramid mesh links by thick solid and dashed lines, and the AFS ring links by the thick solid lines.

The channel capacities in the fat-tree in Figure 4 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 similar to the one shown in Figure 1(a).

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, and the AFS features the switch nodes of the same parent 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.

Universality proofs for the fat-tree and 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 [1, 3]. This method has its basis in the theories and constructions of VLSI graph layouts [8–10]. 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, 12]. In the following analysis, we shall use this decomposing method and parameterize the networks by their areas.

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 [1].

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. As such, there are $O(\log A)$ packets amortized on each wire of the fat-tree of area $\Theta(A)$, resulting in an overhead. With this postulate and based on the general routing theorem of Lemma 9, Greenberg [3] proved the universality of the fat-tree in terms of hardware area.

Lemma 10 ([3]) *A fat-tree of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

Adding a ring on each subnetwork of the fat-tree can cause only a constant factor increase in area. In addition, the connections and the framework of the fat-tree are intact when the rings are added. Thus, we have the following theorem.

Theorem 11 *An AFS of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

The benefit of the AFS rings is that they enable us to drop the unit wire delay assumption used in arriving at the above theorem. The fat-tree is not universal when simulating a mesh under nonunit wire delay assumption [3].

We now proceed to prove the universality of the AFS under nonunit wire delay condition. We again base our proof on the result of Greenberg pursuant to the fat-pyramid. We shall again adopt the condition that each bottom level node is a cluster of $\Theta(\log A)$ processors in an H-tree layout.

In the fat-pyramid proof, an appropriate 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 [3]. In this layout, the length of the edges of the tree of meshes becomes $O(\log A)$ whereas the length of the wires connected to a switch h levels up from the H-tree blocks are $O(2^h \log A)$. Now the routing path of a packet goes along fat-tree edges upwards until it reaches a switch that is adjacent in the same mesh to another switch that leads to the destination by going down fat-tree edges. Two processors separated by a distance δ in the competing network R is $\lceil \delta / \log_2 A \rceil$ H-tree blocks apart in the fat-pyramid F . A packet must traverse the height of the subtree enclosing the two processors ($\log_2(\delta / \log A)$ levels) and the H-tree blocks. It travels a distance of $O(\delta + \log A)$. It is thereof proved that any message following a path of length δ in a competing network travels $O(\delta + \log A)$ distance in the fat-pyramid.

The congestion in F to route the same message set as in R is $O(T \log A)$ where T is the time required to deliver the message set in R and, given a linear delay function w , $T \geq w(\delta)$. A message must traverse at most $2 \log_2 \delta$ fat-pyramid edges and each edge has a maximum delay of $w(\delta + \log_2 A)$ steps. Hence the following theorem is proved by computing the overhead based on the total delivery times in F and R and the definition of w [3].

Lemma 12 ([3]) *Using transmission lines, a fat-pyramid of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead under nonunit wire delay assumption.*

We can now show the universality of the AFS by following the same lines of the fat-pyramid proof. In the AFS layout a packet has to travel a path of the same length as for the fat-pyramid to reach a switch that is adjacent (at most

two edges apart) to another switch in the same mesh (ring) that leads to the destination node, and hence the same overall distance as in the fat-pyramid layout. Similarly, the congestion and the dilation are the same. Note that if the layout is n -ary where $n > 4$, the packet can get to the destination via the root of the underlying n -ary tree covering the source and destination processors by traveling two edges, the same as taking two mesh edges in a 4-ary layout. The below theorem follows.

Theorem 13 *Using transmission lines, an AFS of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead under nonunit wire delay assumption.*

Note that the path that a packet in the AFS travels could not be shorter for the given connectivity, so the AFS is the minimal universal network with $O(\log A)$ overhead in terms of wire usage.

5 Experimental results

We did several experiments using the *ns-2* network simulator to visualize and compare the performance of the networks discussed so far. 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. A fixed traffic scenario of three traffic types is used to run traffic through each of the networks in order to obtain data on how well each performs.

We simulated the fat-tree, the AFS, and the fat-pyramid. The general template is similar to the node layout illustrated in Figure 4. Now the switches of the second and top levels are combined into singular nodes. Wires are also combined into singular links with composite capacities. Each network has a distinct wire layout. In the template there are 64 processor nodes and 21 switches as required by a 4-node subnetwork configuration. Downward links from the second level switches have a capacity of 0.2 Mbps. Links from the top switch are 0.4 Mbps in capacity. Upward links from the bottom level have a capacity of 0.1 Mbps. Links within subnetworks on a level have the same capacities as their upward links.

The split wires and nodes used previously in discussing the networks 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. Leaf nodes consist of single processors that act as switches as well as processors, and they are connected within a mesh in the fat-stack and 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 in the first four networks, there are three tiers. In each tier, links in the subnetworks and its upward link(s) are assigned the same capacity. Upwards across the three tiers, link capacities are 0.1, 0.2, and 0.4 Mbps successively, i.e., they double up the underlying 4-ary tree.

Table 1
Simulation Results

Network	Traffic Type	Throughput (bps)	Delay (sec)
Fat-tree	CBR	928113.333333	0.763641
	Exponential	917320.000000	0.841578
	Pareto	875986.666667	0.780242
Aug. fat-stack	CBR	1126821.333333	0.194550
	Exponential	1130601.333333	0.230433
	Pareto	1097736.000000	0.215983
Fat-pyramid	CBR	1652286.666667	0.146700
	Exponential	1624204.000000	0.169914
	Pareto	1589376.000000	0.162286

A simple 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 for 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.

The two performance metrics effectively indicate a network’s efficiency. Overall throughput has unit of bits per second and measures how much traffic in terms of bits actually reached their destinations from their sources during the simulation interval. Overall delay measures how long, on average, each packet spent in the network from the time it was generated to the time it reached its destination.

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.

The data in Table 1 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 mesh links are added in that order. In other words, the AFS has a considerable improvement over the fat-tree and approaches the fat-pyramid in efficiency. Note that the fat-pyramid uses much more wires and thereby incurs more complexity in wiring and packaging. In addition, the fat-pyramid does not scale to represent a distributed network.

6 Analysis and discussion

In the previous three sections, we developed efficient routing algorithms specific to the fat-stack, proved the universality of the AFS, and described the experiments we did to compare the performance of the AFS, the fat-tree, and the fat-pyramid. In this section we analyze these obtained results and discuss their implications.

First, we discuss the efficiency of the routing algorithms and their possible application. The routing algorithms are some simple schemes such as the greedy mechanism which simply looks at the header of a packet and sends the packet to a prefixed node. These schemes are relatively easy to design since the fat-stack is a “leveled” network and each of its subnetwork is a ring. The routing capabilities we derived in Section 3 of the two variants of the fat-stack using these schemes are tantamount to the general routing results obtained by Leighton et al. [7,14–16] (Lemma 9). Our routing algorithms indicate a routing delay of congestion plus dilation which is exactly the routing efficiency of general networks implied by Lemma 9. In fact, we used Lemma 9 in proving the

universality of the AFS in Section 4. The main difference between our results described in Section 4 and Lemma 9 is the following: our results are specific algorithms that can be implemented for practical use, whereas Lemma 9 indicates that there are algorithms to be found that can be as efficient asymptotically as the sum of two summary measures, namely the congestion and dilation of the network.

In the routing algorithms, there is a tradeoff between congestion and schedule length. When congestion is reduced, schedule length in most cases is inevitably increased. This, however, does not affect the efficiency in asymptotic terms. To make the algorithms efficient in the absolute sense, we designed the algorithms so as to decrease congestion but at the same time not to lengthen schedule. In designing the algorithms, dilation was a variable that could not be minimized.

In addition, the algorithms work online as well as offline. Our universality proofs are based on offline routing capability. But online routing under unit wire delay condition should have the same efficiency due to analysis of packet routing on a “leveled” network [7, 14]. Under nonunit wire delay condition, online routing can incur larger overhead.

Second, we consider the effectiveness of the asymptotic overhead in describing the universality of the network. Our experiments showed that the fat-pyramid has a better absolute efficiency than the fat-stack despite the same asymptotic overhead for both networks. The $O(\log A)$ only indicates that the computing cost in the long run is a function of the logarithm of A ; it does not attest the precise absolute speed of a network. There are bound to be constant terms that affect the speeds, but their effect is oblivious in the asymptotic expression. A more accurate solution might be obtained by a radically different mathematical formulation of the problem. Such a formulation should take into account the average scenarios in choosing routing paths for the packets and not omit any intermediate terms. Nonetheless, asymptotic representation remains a robust indicator of computing efficiency.

Third, the setup for the simulations epitomizes the state-of-the-art technologies in networking. Some components of the setup may not have been implemented in any practical networks. But results should not be far off even if a component of another variant had been used. For example, as the data in Table 1 show, when traffic is of different types, the throughput and delay change only slightly.

Fourth, we comment on existing results relevant to this work. Since Lemma 9 affirms an optimal property of networks, subsequent works by researchers have been to find constructive algorithms that can approach the optimal bound for various networks (e.g. [16, 19, 20]). The standard routing paradigm requires buffers at the nodes, one buffer for each edge. Recently, Busch et al. worked

on universal packet routing on networks with no buffers [21, 22]. They found the routing time on bufferless networks to be much longer than on standard buffered networks. Bufferless routing is applicable in optical networks where buffering can be too costly or impossible.

Finally, we describe actual applications of the AFS and explain where it can be best suitable and where it would not be.

Many interconnection networks for parallel computers are based on the hypercube. A hypercube is not planar. It requires large number of wires and hence large physical volume to interconnect the processors [1]. However, many computing tasks are planar and would waste much of communication bandwidth of a hypercube-based network.

The fat-pyramid is non-planar as a mesh is present on every level of a fat-pyramid network. Still, since the fat-pyramid is a leveled network, it uses much fewer wires than the hypercube.

The AFS is quasi-planar; its full planarity is only limited by the rings at its subnetworks. The rings, however, provide a locality feature distinct of the AFS. This locality is crucial for processing many typical parallel tasks at high speed.

Typical parallel computing applications include weather prediction, ocean currents simulation, and ray tracing in computer graphics [23]. These and many other applications can be characterized as finite-element problems in which the object is dissected into cross sections which are further discretized into elements. The algorithms to solve these problems involve intensive iterative updates among *neighboring* elements. This is where the locality of the AFS is advantageous. On the other hand, since there is no or little interaction among elements that are far apart, connections provided by the fat-pyramid would be wasted at the expense of volume, wirability, and packaging.

In such applications as galaxy evolution simulation and data mining for associations, hierarchical algorithms are often used that may require more inter-area communication. But even in these algorithms, locality still predominates. In these cases, as inter-area communication is relatively frequent, using the fat-pyramid may be more meaningful. But since inter-area communication is still much less than local communication, the AFS can substitute for the fat-pyramid and approach it in performance quite effectively.

7 Conclusion

In this paper, we have shown that the AFS is universally efficient and is suitable for use as an interconnection network. The fat-stack is superior in efficiency, scalability, and minimality. Analytically the fat-stack has the same logarithmic overhead as the fat-tree ($O(\log A)$ in the unit wire delay case, Lemma 10 and Theorem 11) and the fat-pyramid ($O(\log A)$ in the nonunit wire delay case, Lemma 12 and Theorem 13) in simulating any other network. In simulations, the AFS performs almost as well as the fat-pyramid (off by $\sim 30\%$ in both throughput and delay) and much better than the fat-tree (by $\sim 23\%$ in throughput and $\sim 73\%$ in delay). The fat-stack is universal under both unit and nonunit wire delay conditions, while requiring the minimal amount of hardware. Note that the fat-tree is not universal under nonunit wire delay condition and that the fat-pyramid requires far more wires than the AFS although both are universal under both unit and nonunit wire delay conditions.

The fat-stack is more scalable than the fat-pyramid because it uses fewer wires and has simpler connectivity. The fat-stack's scalability is implied by its simple structure. Each of the two fat-stack variants can be viewed as a construct of a common atomic unit consisting of a ring and one or more upward links.

Those strengths of the AFS make it a valuable candidate for certain classes of supercomputing tasks that would require an interconnection network more efficient than the fat-tree. In some cases, the AFS can be a substitute for the fat-pyramid as we have discussed in Section 6.

As our routing theorems indicate, the GFS variant incurs some unsurmountable delay as each of its subnetworks has only one upward link. However, the GFS can be scaled up to represent a distributed network. It would be desirable to study the scalability of the fat-stack and its application to distributed networking.

Acknowledgements

The authors thank the three anonymous reviewers for their useful comments that helped improve the paper.

This work was supported in part by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001 and NSF CCR-0309461.

References

- [1] C. E. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, *IEEE Trans. on Comput.* C-34 (10) (1985) 892–901.
- [2] R. I. Greenberg, C. E. Leiserson, Randomized routing on fat-trees, in: S. Micali (Ed.), *Randomness and Computation*, Vol. 5 of *Advances in Computing Research*, JAI Press, 1989, pp. 345–374.
- [3] R. I. Greenberg, The fat-pyramid and universal parallel computation independent of wire delay, *IEEE Trans. on Comput.* 43 (12) (1994) 1358–1364.
- [4] V. Strumpen, A. Krishnamurthy, A collision model for randomized routing in fat-tree networks, Tech. Memo MIT-LCS-TM-629, Laboratory for Computer Science, Massachusetts Institute of Technology, 15 July 2002.
- [5] V. Strumpen, A. Krishnamurthy, A collision model for randomized routing in fat-tree networks, *J. Parallel Distrib. Comput.* 65 (9) (2005) 1007–1021, journal version of [4].
- [6] P. Bay, G. Bilardi, Deterministic on-line routing on area-universal networks, *J. ACM* 42 (3) (1995) 614–640.
- [7] F. T. Leighton, B. M. Maggs, A. G. Ranade, S. B. Rao, Randomized routing and sorting on fixed-connection networks, *J. Algorithms* 17 (1) (1994) 157–205.
- [8] F. T. Leighton, A layout strategy for VLSI which is provably good, in: *Proc. 14th Annual ACM Symp. on Theory of Computing*, 1982, pp. 85–98.
- [9] S. N. Bhatt, F. T. Leighton, A framework for solving VLSI graph layout problems, *J. Comput. and System Sci.* 28 (1984) 300–343.
- [10] S. N. Bhatt, C. E. Leiserson, How to assemble tree machines, in: F. P. Preparata (Ed.), *VLSI Theory*, Vol. 2 of *Advances in Computing Research*, JAI Press, 1984, pp. 95–114.
- [11] R. I. Greenberg, Efficient interconnection schemes for VLSI and parallel computation, Ph.D. thesis, Massachusetts Institute of Technology (Aug. 1989).
- [12] R. I. Greenberg, C. E. Leiserson, A compact layout for the three-dimensional tree of meshes, *Appl. Math. Lett.* 1 (2) (1988) 171–176, also see erratum in vol. 1, no. 3, p. 315.
- [13] K. F. Chen, E. H.-M. Sha, The fat-stack and universal routing in interconnection networks, in: *Proc. ISCA 17th Internat. Conference on Parallel and Distributed Computing Systems*, San Francisco, CA, USA, 2004, pp. 321–326.
- [14] T. Leighton, B. Maggs, S. Rao, Universal packet routing algorithms, in: *Proc. 29th Annual Symp. on Foundations of Computer Science*, IEEE Computer Society Press, 1988, pp. 256–269.
- [15] F. T. Leighton, B. M. Maggs, S. B. Rao, Packet routing and job-shop scheduling in $O(\text{congestion} + \text{dilation})$ steps, *Combinatorica* 14 (2) (1994) 167–186.

- [16] T. Leighton, B. Maggs, A. W. Richa, Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules, *Combinatorica* 19 (3) (1999) 375–401.
- [17] C. E. Leiserson, VLSI theory and parallel supercomputing, in: C. L. Seitz (Ed.), *Advanced Research in VLSI: Proc. Decennial Caltech Conference on VLSI*, MIT Press, 1989, pp. 5–16.
- [18] C. E. Leiserson, B. M. Maggs, Communication-efficient parallel algorithms for distributed random-access machines, *Algorithmica* 3 (1988) 53–77.
- [19] F. M. auf der Heide, B. Vöcking, Shortest-path routing in arbitrary networks, *J. Algorithms* 31 (1) (1999) 105–131.
- [20] Y. Rabani, É. Tardos, Distributed packet switching in arbitrary networks, in: *Proc. 28th ACM Symp. on Theory of Computing*, 1996, pp. 366–375.
- [21] C. Busch, M. Magdon-Ismail, M. Mavronicolas, Universal bufferless routing, in: *Proc. 2nd Workshop on Approximation and Online Algorithms*, LNCS 3351, 2004, pp. 239–252.
- [22] C. Busch, S. Kelkar, M. Magdon-Ismail, Efficient bufferless routing on leveled networks, in: *Proc. Euro-Par 2005*, LNCS 3648, 2005, pp. 931–940.
- [23] D. E. Culler, J. P. Singh, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, San Francisco, California, 1999.