Competitive Fault-Tolerance in Area-Universal Networks

SIVAN TOLEDO*
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

In this paper we study fault tolerance in area-universal networks and provide both positive and negative results. We present a new network layout, the *mesh of ladders*, which is area-universal and fault-tolerant. This network, laid out in an $n \times n$ area, can simulate any other network layout laid out in the same area with $O(\log^4 n)$ slowdown, even if $cn/2\log n$ blocks of size $2\log n \times 2\log n$ become faulty (for some constant $c < 1$). Furthermore, it can simulate any other network layout which has at most $f(n)$ bends in each wire with slowdown $O(f(n)\log^2 n)$ even after *any number* of such blocks become faulty in both networks. Our results are tight, in the sense that if one of our assumptions is removed, it is no longer possible to obtain similar simulation results. We show for example that the width of a layout for a network with at least n nodes and diameter at most $n^{1-\epsilon}$ for any fixed $1/2 < \epsilon < 1$ must be at least $\Omega(\log n)$. Therefore, if faults can happen in any pattern the remaining non faulty area may be too narrow to be useful. We also show that it is impossible to have a network that can simulate any other network (without respect to the number of bends) in the presence of many faults. This result implies, for example, that any network layout in an $n \times n$ area that can simulate any other network laid out in the same area in the presence of $\log n \times \log n$ block faults, must have an $\Omega(n)$ slowdown.

*This research was supported in part by the Defense Advanced Research Projects Agency under Grant N00014–91–J–1698. Author's e-mail address: sivan@theory.lcs.mit.edu.

1 Introduction

Fault-tolerant and efficient networks are a major component in the design of large scale parallel computers. In this paper, we propose such a network and investigate its properties. We also show that the limitations of our proposed network cannot be overcome by any other network.

First let us outline our goals. We seek a network N, laid out in Thompson's VLSI layout model [14], that can efficiently simulate any other network R laid out in the same area. This property is called *area universality*. Furthermore, we would like N to be *fault-tolerant*, that is, to be able to simulate R efficiently even after the two networks are similarly damaged. In other words, the remaining undamaged parts of N should be able to efficiently simulate the remaining parts of R. The fat-tree network, proposed by Leiserson [10], is area-universal. Laid out in an $n \times n$ square, a fat-tree can simulate any other network laid out in the same area with an $O(\log n)$ slowdown.

We propose a network, the *mesh of ladders*, that has the second property as well. This network is modeled after the mesh of trees network, [4], with the trees replaced by ladders. It is also related to the fat-pyramid network [3]. A *ladder* is the graph shown in Figure 1. It is very similar to the X-tree graph [13], and the reason we introduce it is to simplify some of our proofs. A ladder graph with n "leaves" can be laid out in an $n \times \log n$ area. Consider a grid of $n \times n$ nodes. Now let us add a ladder on each of the n rows of the grid such that the original nodes are now the leaves of the ladders. Similarly, we add ladders to the columns of the grid. The $2n$ ladders are connected only through their leaves. Having $\Theta(n^2)$ nodes and laid out in a $\Theta(n^2\log^2 n)$ area, it can simulate any other network laid out in the same area with $O(\log^4 n)$ slowdown (we will show below how to raise the number of nodes in the network to $\Theta(n^2 \log^2 n)$). The network's ability to sustain cn faulty blocks of size $2 \log n \times 2 \log n$ (for some constant $c < 1$) follows from its regular structure. Indeed, if only that many blocks are faulty, then at least $(1-c)$ of the row ladders and $(1-c)$ of the column ladders are intact. The network therefore contains a similar but smaller network, so its performance is degraded only by a constant factor. It is not even necessary to assume that the network being simulated is also damaged.

There are two drawbacks to this approach of tolerating faults. One of them is the necessity to know where the faults are and to reconfigure the network accordingly. The second and most important, is that since some rows and column ladders are not used, the inputs and outputs terminals of the network will have to be moved. We are able however to show a much stronger result. Namely, after disabling any number of blocks of size $2 \log n \times 2 \log n$, the network is still able to simulate most other network layouts which have undergone a similar damage, without moving the input and output terminals.

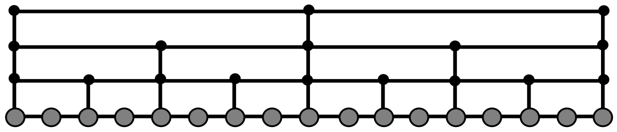


Figure 1: A layout of a ladder graph. The gray nodes are called leaves, and the smaller black nodes are routing nodes.

There are two discrepancies between the statement of the problem and the statement of our result:

- The damage is assumed to be in blocks of $2 \log n \times 2 \log n$.
- Once damage happens, our network can only efficiently simulate most network layouts that have undergone a similar damage, but not all. Specifically, it may not be able to efficiently simulate network layouts which have many bends in some wires.

Section 3 is devoted to the investigation of why these restrictions cannot be overcome. Regarding the first remark, we shall show that there are shapes in which no area-universal network can be embedded. Therefore, if the area remaining after damage has such a shape, there is no hope that the remaining parts of the network will be able to efficiently simulate any other network laid out in the same area. The restriction of the damage patterns to occur in blocks eliminates the possibility of being left with such a "useless" shape. Another viewpoint on this issue is that we do not expect the universal to do as well as every other network in any situation. But it must be able to perform *almost* as well. Therefore, if a fault occurs, the universal network will behave as if a whole block around that fault is damaged. We will also justify the second restriction. We will show that even if faults occur in blocks, there is no network that can efficiently simulate *any* other network. If we only consider network layouts which have only a small number of bends in each wire, however, the mesh of ladders can simulate any of them even after a similar block pattern of faults happened to both.

Therefore, trying to find a network layout without one of the above restrictions is provably futile. It might be the case, of course, that replacing one of them (or both) with other restricting conditions can still lead to a fault-tolerance area-universal network. But dropping even one of them altogether is impossible.

There have been several successful attempts to study the problem of fault tolerance in parallel computing. One direction of research is probabilistic analysis. Leighton and Leiserson [5] for example, study the problem of analyzing randomly placed faults on a chip. There are several drawbacks to this approach. The most important one is that it is assumed that faults in different elements of the system are independent. In practice, faults cluster, and the fact that one element of the system is faulty gives a strong hint that nearby elements may be faulty as well. In addition, the actual probability of faults for a particular element of the system is usually very small, once the system is initially set up and tested.

Aiello and Leighton [1] study embeddings of hypercubes in faulty hypercubes with constant load, congestion, and dilation. They show that an n node hypercube can be embedded in another, even if the host hypercube has up to $\log^{O(1)} n$ worst case faults, or if each of the nodes and edges fails independently with some small constant probablility. References to previous work in this area can be found there.

Leighton and Maggs studied both deterministic and random faults in multibutterflies [7]. Although their results are encouraging and can be applied to area-universal networks, they are not concerned with the competitive behavior of the remaining resources. In other words, they do not analyze what is possible to do with the undamaged areas of a network (i.e., the remaining undamaged resources), but rather analyze the behavior of the *given* network after the damage.

2 A Fault-Tolerant Area-Universal Network

In this section we show that the mesh of ladders network, introduced in the Introduction, is indeed area-universal and fault-tolerant. We first outline our scheme:

- Let R be a given network layout which occupies an $n \times n$ area and has at most β bends in each wire.

- Let R' be a modification of R, in which a node is inserted at every bend on every wire. All the wires in R' are straight, and are either horizontal or vertical. Clearly, R' can simulate a step of R in β steps, by routing messages in R' along paths corresponding (geometrically) to the wires in R.

- Now suppose that we superimpose a mesh of ladders N on R', where every leaf of a ladder is connected to a $\log n \times \log n$ grid. We show below how every ladder can simulate the straight wires it "covers" (horizontal for row ladders and vertical for column ladders) in $O(\log^2 n)$ steps. Therefore N can simulate R' in $O(\log^2 n)$, and hence simulate R in $O(\beta \log^2 n)$ steps.

The fault tolerance of the proposed scheme follows from an interesting property of our routing algorithm on a ladder, namely that a massage being sent from a leaf to another to its right, never moves left along its path. Consider a specific message that R is able to deliver after some damage has occurred both to R and N. The wire carrying the message is therefore intact, and its segments in R' are also intact. But this means that the contiguous pieces of ladders that cover such segments are also intact, because we assume that faults happen in blocks. Therefore the ladder can route the message, since the path traveled by the message does not contain any damaged nodes or wires. We conclude that every message that can be delivered by R can also delivered by N, within the same slowdown bound as without damage.

2.1 Routing on a Ladder

Consider a ladder graph with n leaves laid out in an $n \log n \times \log n$ area, where each leaf is connected to a $\log n \times \log n$ grid (as in [8]). The number of processors in this network is $n \log^2 n$. We show that this network can simulate any network composed of pairs of nodes connected by horizontal wires laid out in the same area with $O(\log n)$ slowdown. (Actually it is not hard to see that it can simulate any network laid out in the same area).

As a preliminary phase, every message that must be routed within a grid is routed using a greedy scheme. This preliminary phase takes $\log n$ steps, since the paths are all horizontal The rest of simulation is performed in $\log n$ phases. There can be at most $\log n$ wires crossing the vertical line that separates the strip into two equal-size strips of $(n \log n)/2 \times \log n$. In the first phase we route messages that use these wires. The routing can be pipelined, so this phase takes $O(\log n)$ steps. Now, there are no wires the routing of which uses the root of the tree, so the routing problem is decomposed into two independent routing problems of half the size, that can be solved in parallel. Therefore the total running time is $O(\log^2 n)$.

We now show how to deliver a single message in the ladder (we assume that the message originates at the leaf; delivering it from an internal node of the grid to the leaf poses no problems). The path is easy to construct, and has two parts, one moving up the ladder and the other moving down. The rules for the path construction are simple. The "up" part rules are:

1. If, by going right, we pass the destination column, begin the "down" part.

2. Otherwise, if going up is possible, go up.

3. Otherwise, go right.

And the "down" part rules are:

1. If, by going right, we pass the destination column, go down.

2. Otherwise, go right.

It is easy to see that the number of steps is at most $4 \log n$ because horizontal moves are interleaved with vertical moves, and there are at most $2 \log n$ vertical moves, at most $\log n$ up moves followed by the same number of down moves. As there is always a pair of consecutive up-down moves at the top of the path, they can be eliminated. The path constructed is monotone, as promised. That is, for a message delivered to a node to the right of its originating node, the path goes always from left to right.

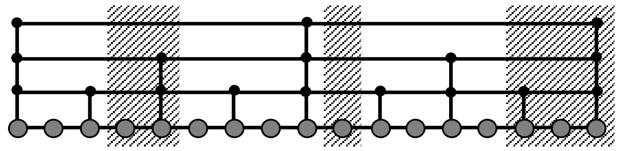


Figure 2: A damaged ladder graph. The shaded areas represent dead parts of the network.

We now show how to pipeline $\log n$ messages, so all of them are delivered in $O(\log n)$ time. The way to make pipelining work is to make sure that all messages in a single step reach exactly the same height in the ladder. Consider the first phase for example. A message might not need to go up any further in the ladder in a certain step, but sending it down might interfere with other messages going down. What we can do is to send it up in any case, and then down in the same path back. from some internal nodes however, there is no path up. So we add it artificially, so that from each leaf there is a linear array of $\log n$ nodes going up, only in some point the linear array nodes are no longer a part of the ladder (in the sense that they are not part of the horizontal rows). Now that all routing paths are of same length (assuming that all messages traverse up-down edges on even steps, and left-right or none at all on odd steps), we can pipeline the messages.

As a final point, we note that by slightly altering the path construction algorithm it is possible to eliminate about half the edges and switching nodes, and to remain with a structure very close to an X-tree.

This has concluded the proof that the mesh of ladders can simulate any other network with at most b bends in each wire with slowdown $O(b \log^2 n)$, even after any number of blocks became faulty in both.

Definition 1 *A family of network layouts is said to be an* $f(n)$-bend *family if each of its members that occupies an* $n \times n$ *area has at most* $f(n)$ *bends in each of its wires.*

Theorem 2 *The mesh of ladders network, laid out in an* $n \times n$ *area, can simulate any* $f(n)$*-bend network laid out in the same area, with* $O(f(n) \log^2 n)$ *slowdown, even after any number of* $2 \log n \times 2 \log n$ *blocks are damaged.*

2.2 Area Universality of the Mesh of Ladders

Although up to now we restricted our attention to networks with few bends, the mesh of ladders (and the mesh of trees) is actually area universal over all networks.

Definition 3 *A network layout* N_1 *occupying an* $n \times n$ *area is said to be* equivalent with expansion α *to another network layout* N_2 *if both are layouts of the same graph, and if* N_2 *occupies an area of at most* $\alpha n \times \alpha n$.

Definition 4 *A layout for a network with at most* n^2 *nodes is said to be* regular *if the nodes occupy a subset of an* $n \times n$ *grid whose rows and columns are spaced apart by uniformly-sized channels.*

We now show that every network can be converted into an equivalent network with only a few bends in each wire.

Theorem 5 *Every network laid out in an* $n \times n$ *area is equivalent with expansion* $4 \log n$ *to a 4-bend network layout.*

Proof: This result follows immediately from [9].

We decompose the area into four quadrants of $\frac{n}{2} \times \frac{n}{2}$. We reroute the wires that connect nodes in the same quadrant recursively, expanding the area as little as possible, and then combine the four sub-layouts, by connecting the wires that connect wires that connect nodes in different quadrants, expanding the layout as necessary.

The base case is easy. If $n = 1$, the whole layout is a single node and therefore already 4-bend.

Suppose that we have already laid out the four quadrants. The number of wires that can connect nodes in the upper half of the original layout to its lower half is at most n (because each of them has to cross the line separating the two halves). We connect these pairs of nodes by *slicing*, as in Leiserson's algorithm. Every wire we must connect requires us to add 2 units to the width and height of the layout (actually less is necessary). Overall, this adds $2n$ to the width and height of the layout. Repeating this for the right and left halves requires $2n$ units more in each dimension, so the overall additional width and height is $4n$.

Denoting the maximum length of the side of the square needed for embedding the expanded layout by $S(n)$, we solve the recurrence

$$\begin{cases} S(1) = 1 \\ S(n) = 2S(\frac{n}{2}) + 4n \end{cases}$$

whose solution is $S(n) = 4n \log n$. ■

We now prove a slightly different form of the same result. Its proof however, will reveal the close relation between area-universal networks on one hand, and the existence of an equivalent network with small expansion which has few bends to any given network layout on the other hand.

Theorem 6 *Every network laid out in an $n \times n$ area is equivalent with expansion $O(\log n)$ to an $O(\log n)$-bend network layout.*

Proof: We use the following facts:

- Using an off-line routing algorithm, it is possible to simulate any network laid out in an $n \times n$ area on a fat-tree (see [10]) with $O(\log n)$ slowdown. This result is due to Leighton et al. [8].

- The layout of such a fat-tree can be performed such that every wire has only a small constant number of bends. Layouts based on the H-tree layout [12] or on the "fold and squash" method [2] have this property.

To construct the equivalent network, simulate on the fat-tree a step of the given network, in which a message is sent on every wire. The fat-tree will simulate it in at most $O(\log n)$ steps, which means that each message passes through at most $O(\log n)$ wires and nodes.

The basic idea is very simple — the new wire replacing an old one will trace geometrically the path a node traversed in the fat-tree. This path has only $O(\log n)$ bends, since each of the wires in the fat-tree has only a constant number of bends. The problem of course is that a wire in the fat-tree could be used by $O(\log n)$ messages, and we cannot pass in that path so many wires.

So we trade time for area; instead of having $m = O(\log n)$ messages going through a wire in $O(\log n)$ time steps, we replace the wire by a bundle of m wires, on which we can send m messages in parallel. So every wire in the fat-tree is replaced by $O(\log n)$ parallel wires, one for each time step. Each node in the network is replaced by an $O(\log n) \times O(\log n)$ grid, on which we can jump wires going in different directions and different times (corresponding to different wires in a "bundle"). See Figure 3 for an example. ■

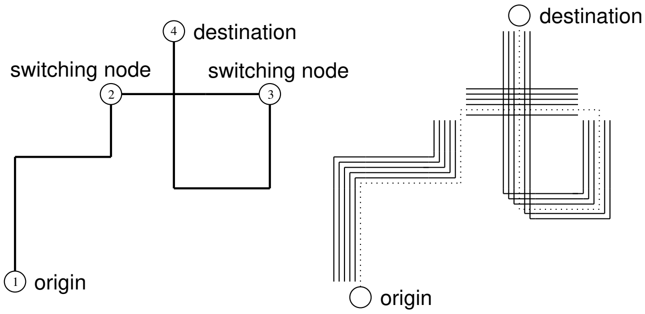


Figure 3: Routing a wire according to a path in a fat-tree. The path on the fat-tree is shown on the left, with numbers inside the processors signifying the step in which the message passed through the processor. On the right, the wire routed is dotted, with its position in the bundle determined by the time at which the message traversed the corresponding wire in the fat-tree.

The regular layout can also be found using an algorithm similar to the one in the first proof, without relying on fat-trees.

Using the result of Theorem 6, we adopt the following strategy for simulating an arbitrary network R laid out in an $n \times n$ area by a mesh of ladders occupying the same area:

- Compute an equivalent $O(\log n)$-bend regular layout R'.

- Simulate R' on the mesh of ladders.

The only difficulty with this strategy is that the R' will require $n \log n \times n \log n$ area. But since the layout is regular, we can consider R' to be an $n \times n$ layout, where the channels have $\log n$ capacity. Simulating such a network by a mesh of ladders require another slowdown of $\log n$, to deal with the increased volume of messages. Taking this last slowdown into account, and the fact that R' might have $O(\log n)$ bends in each wire, we conclude that the mesh of ladders can simulate R with $O(\log^4 n)$ slowdown.

The fact that for a given network layout there is also an equivalent layout with only 4 bends in each wire, can be used to obtain faster simulations, with only $O(\log^2 n)$ slowdown, but only on an $O(n \log n) \times O(n \log n)$ area mesh of ladders. The reason for this is that these layouts with few bends may be non regular, and therefore $\log^2 n$ processors might be packed into one if the layout is compressed.

Note that our simulation algorithm requires some off-line computation. To construct a routing path we need to know the routing of wires in an equivalent network with few bends, the computation of which is done off-line (the path followed by a message depend however only on the source and the destination of the message, not on the other messages sent). There is however a simulation algorithm for the mesh of trees due to Leighton and Leiserson [6], which is on-line and proves the area universality of the mesh of trees. Their algorithm works by simulating a fat-tree on a mesh of trees. Then, simulating an algorithm on the fat-tree that simulates on-line a given network gives the desired result.

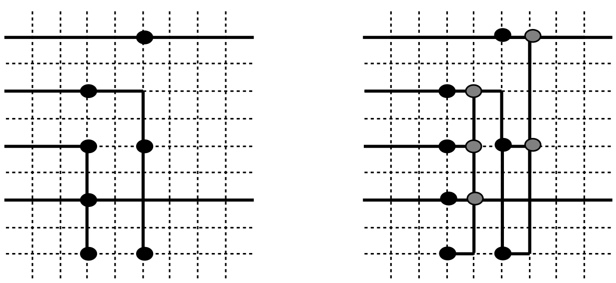
3 Impossibility results

The mesh of ladders is area-universal only over networks with few bends and only for faults which occur in whole blocks of size $2 \log n \times 2 \log n$. In this section we show we explore the limits of fault-tolerant area-universal networks. We will show that if faults are not limited to blocks of size $\Omega(\log n) \times \Omega(\log n)$, the remaining shape can be such that no area-universal network can be laid out in it. In addition, even if the faults occur in blocks, we show that no network can be fault-tolerant and area-universal over all network layouts. So the deficiencies of our fault-tolerant area-universal network cannot be overcome by any other network.

3.1 A Shape in Which No Area-Universal Network Can Be Laid Out

We begin by showing that there is a shape for which there is no area-universal network. In other words, we cannot lay out a network in this shape which can simulate with a small slowdown all other networks that can be laid out in the shape. Therefore, if such a shape remains after some damage has occurred to the network, there is no hope that what remains is area-universal. Even if we knew exactly what damage is about to happen, we could not have designed a good network, because there isn't any.

Before we proceed, we comment that the following theorem holds under the assumption that the processor in an area-universal network simulating a specific processor along the boundary of the shape (we call these nodes *terminals*) in another network, is located geometrically in the same place. This assumption is can be satisfied by area-universal networks; see [3, 11]. This assumption is be also satisfied by our area-universal networks in Section 2. (Our subsequent impossibility result do not need this assumption.) An area-universal network must be able to route quickly a message between any two terminals, because for any two terminals, there is a network in which there is a wire connecting them. Furthermore, if the network is laid out in a narrow rectangle of $n \times k$ where $k << n$, the area universal network must be able to route quickly a message between any two nodes, because every network can be slightly modified such that every node is within distance k from a terminal (see Figure 4). Therefore, the *diameter* of an area-universal network in a narrow rectangle must be small. As the next theorem shows, this is impossible if the rectangle is too narrow. Before presenting the result, we would like to comment that although assuming geometric mapping of terminals reduces the generality of the following theorem, the theorem still shows that with that particular shape one cannot do as well as one can with a square, say.



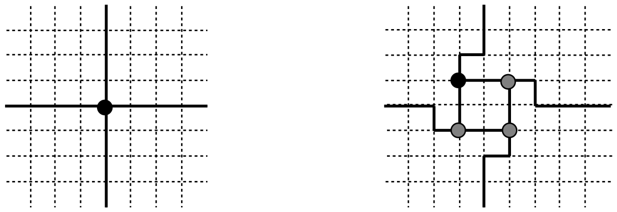
(a) **(b)**

Figure 4: Adding a short path between every node and a terminal node. The original network is shown in (a), and the modified one in (b). New nodes added in the process are marked in gray.

Theorem 7 *Let N be any network laid out in an $n \times k$ area with a diameter of at most $n^{1-\epsilon}$ for any fixed $1/2 < \epsilon < 1$. Then $k = \Omega(\log n)$.*

Proof: The network N has under Thompson's model of VLSI layout [14] a maximum degree of 4. By expanding the dimensions of the shape by a constant we may assume that the maximum degree of a node is only

3 (See Figure 5). This only multiplies the length of any path by a constant.



(a) **(b)**

Figure 5: Reducing the degree of a node from 4 (a) to 3 (b). New nodes added in the process are marked in gray.

Because the diameter of N is only $n^{1-\epsilon}$, the shortest path between any pair of nodes must be at most $n^{1-\epsilon}$. Now find some node r of degree 2. If there is none, remove an edge adjacent to a node of degree 3 to create one (and this last modification does not much affect the path lengths). From this node, create a rooted breadth-first-search tree of the network.

Because the diameter of the network is $n^{1-\epsilon}$, the height of the BFS tree is at most $n^{1-\epsilon}$. Note that this is a binary tree; the root has degree at most 2 by construction, and once we reach a non-visited node in the network and adjoin in to the tree, the edge on which we arrived to the node is used, so the node can have at most 2 children, since its degree is at most 3.

As the following combinatorial lemma will show, every such binary tree contains a subgraph which is homeomorphic to a complete binary of height of at least $\left(\epsilon - \frac{1}{2}\right) \log n$ (that is isomorphic to it up to contraction of nodes of degree 2). Using a result due to Leiserson [9], we find out that any layout for such a complete binary tree has a minimum dimension $\Omega(\log n)$. The same certainly holds for any network that contain such a tree, so we conclude that our area-universal network cannot be laid out in a strip of constant width for any constant. ■

Lemma 8 *Let T be a rooted binary tree with n nodes and height at most $n^{1-\epsilon}$ for some fixed $1/2 < \epsilon < 1$, and assume $n \geq n_0$ for some constant n_0. Then T contains a subgraph which is homeomorphic to a complete binary tree of height $\left(\epsilon - \frac{1}{2}\right) \log n$.*

Proof: First we eliminate all the nodes of degree 2, to obtain a full binary tree. But in the process of elimination, we must bound the number of eliminated nodes from above, to ensure that we end up with a big enough tree.

In the tree, we define a *chain* to be a maximal simple path of nodes of degree 2. We contract each chain to the node just below the lowest element of the chain (see Figure 6).

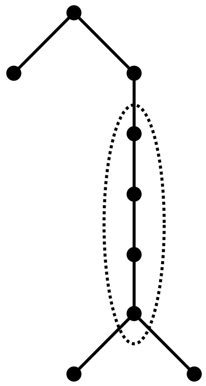


Figure 6: Contracting a chain into the node below it.

Suppose for contradiction that in T there are more than $n - \sqrt{n}$ nodes of degree 2 (and therefore there are less then $\sqrt{n}$ nodes of degrees 1 and 3) arranged in m chains. By the definition of a chain as a maximal path, there is at most one chain contracted into each node of degree 1 or 3, so the number of chains $m \leq \sqrt{n}$. Therefore there is a chain of length of at least

$$\frac{n - \sqrt{n}}{m} \geq \frac{n - \sqrt{n}}{\sqrt{n}} = \sqrt{n} - 1.$$

For sufficiently large n, we have

$$\sqrt{n} - 1 > n^{1-\epsilon},$$

since $1 - \epsilon < 1/2$, and this is a contradiction, since the height of T is at least the length of the longest chain in it. Therefore the number of nodes of degrees 1 and 3 that are left after the contraction is at least $\sqrt{n}$. We denote the full binary tree left after the contraction T'. It has at least $\sqrt{n}$ nodes, and height at most $n^{1-\epsilon}$. Therefore, at least one level of T' contains at least

$$\frac{\sqrt{n}}{n^{1-\epsilon}} = n^{\epsilon - \frac{1}{2}}$$

nodes. Since all of them have parents at a level one higher, they can be matched into pairs with a common parent and the number of the parents is exactly half the number of nodes, because there are no nodes with degree 2 in T'. The same argument holds for the parents, so the nodes in the chosen level are leaves in a complete binary tree which is a subgraph of T', and a subgraph up to nodes of degree 2 (which were contracted) of T.

The height of this complete binary tree is the logarithm to the base 2 of the number of its leaves, that is at least

$$\log\left(n^{\epsilon-\frac{1}{2}}\right) = \left(\epsilon - \frac{1}{2}\right)\log n,$$

which concludes the proof. ∎

3.2 There is No Fault-Tolerant and Area-Universal Network in a Square

We now prove more general results concerning the impossibility of general fault-tolerant area-universal networks. In our proofs, we will mostly use destruction patterns that still allow an area universal network to be embedded in the undamaged area, but still there is no network which is good for all shapes that might remain. We begin by considering again a geometric mapping of terminals: We will show later that this requirement is not necessary, but it makes it easier to introduce the proof technique.

Theorem 9 *Let the functions p, t, b, δ satisfy*

1. $p(n)\log\delta(n) < t(n)\log t(n)$ *and*
2. $t(n) \leq \frac{n}{4b(n)}$

for all $n > n_0$ for some constant n_0. For any network layout N in an area of $n \times n$ with maximum degree $\delta(n)$, there is a way of killing at most $16t^2(n)$ blocks of size $b(n) \times b(n)$ each and a network R laid out in the same $n\times n$ area, such that N takes more than $p(n)$ time to simulate a step of R, under a geometric mapping of terminals.

Corollary 10 *There is no area-universal network that can be embedded in a square of $n \times n$ that can simulate any other network laid out in the same area with a slowdown of $\log^c n$, after both networks have undergone a destruction of the same pattern of at most $16\log^{2c} n$ blocks of size $\log n \times \log n$ each.*

Corollary 11 *There is no area-universal network that can be embedded in a square of $n \times n$ that can simulate any other network laid out in the same area with a slowdown less than $n/16$ (for sufficiently large n) after both networks have undergone a destruction of the same pattern of blocks of size $\log n \times \log n$ each.*

Proof of Theorem 7: Consider a scheme for killing blocks, in which the live blocks in the lower left corner of the area form a *snake* of $t(n)$ vertical "towers", each of which starts at the bottom of the square, and can have any height between 1 and $t(n)$, measured in blocks (see Figure 7). The remaining area is alive, but isolated from the snake. Note that it is possible to embed such a pattern in an $n \times n$ area, because $t(n) \leq \frac{n}{4b(n)}$.

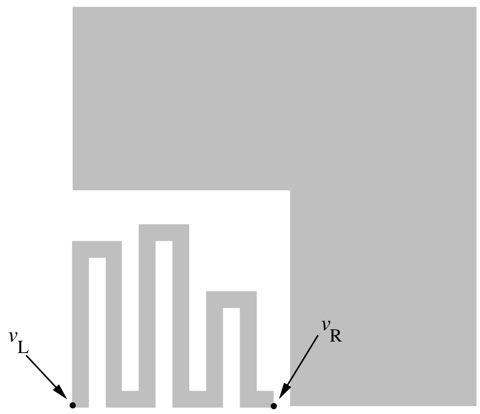


Figure 7: A snake of live blocks is isolated from the remaining live area.

There are $(t(n))^{t(n)}$ possible different snake patterns. For each of them, there is a network R that has a wire run from the terminal in the lower left corner to the terminal in the lower right corner of the snake, so it can route a message between them in unit time. (Note than in all snakes these terminals are fixed.) We shall show however that there is a snake in which N cannot route a message between these two terminals in $p(n)$ steps or less, simply by showing that the shortest path between the terminals is longer than $p(n)$.

Consider a given short path in the (undamaged) network N, originating from the lower left terminal v_L and terminating at the lower right terminal of the snake v_R. We denote the path by

$$\mathcal{P} = v_0 w_0 v_1 w_1 \cdots v_{l-1} w_{l-1} v_l$$

where the v_i's are nodes and the w_i's are wires, $v_0 = v_L$, $v_l = v_R$, and $l \leq p(n)$.

Let S be a given snake. If all the wires in $\mathcal{P}$ pass only through the live blocks of S, the path is usable for routing, and we say that *the path $\mathcal{P}$ passes through the snake S*. Otherwise, let w_i be the first wire of $\mathcal{P}$ that is not completely contained in S. Then w_i pass through a dead block, and is therefore disconnected, and $\mathcal{P}$ is not usable for routing.

The crucial point of the proof is that if $\mathcal{P}$ passes through S, it does not pass through any other snake $S' \neq S$. This is true because S and S' differ in at least the height of one tower, which means that there is a block in S which is not in S' (the block in the top of that tower in S), and $\mathcal{P}$ must pass through all the blocks of S.

Since the degree of a node in the network is bounded by $\delta(n)$, the total number of short paths (of length $\leq p(n)$) originating in v_L is at most

$$(\delta(n))^{p(n)}.$$

Obviously the number of short paths originating in v_L and terminating in v_R is not larger than this bound, and each of them passes through at most one snake.

But it is easy to show that the number of snakes is larger than the number of short paths. Indeed, using the fact that

$$p(n) \log \delta(n) < t(n) \log t(n)$$

and exponentiating both sides we obtain

$$(\delta(n))^{p(n)} < (t(n))^{t(n)}.$$

Therefore there are snakes that contain no short path. If such a snake remains after the network is damaged, there is no short path between v_L and v_R, so routing a message between the two terminals must take more than $p(n)$ steps, while there is a network R that can route such a message in unit time. ■

The impossibility result holds even if we drop the requirement that terminals are mapped in a geometric manner. Instead, we require that each processor in the area-universal network simulate no more than $c(n)$ processors in the simulated network for some function c. We also assume that a specific processor in the simulated network is simulated by a specific processor in the area-universal network. Note however, that the number of faulty nodes must be large for the result to hold. Indeed, as explained in the Introduction, if the number of faulty blocks is small, and geometric mapping of terminals is not required, the mesh of ladders is area-universal and fault-tolerant.

Theorem 12 *Let the functions c, p, t, b, δ satisfy*

1. $p(n) \log \delta(n) < \left(\frac{t(n)}{8c(n)} - 1\right) \log t(n)$,
2. $t(n) = \frac{n}{4b(n)}$ *and*
3. $b(n) \geq 2 \log n$

for all $n > n_0$ for some constant n_0. For any network layout N in an area of $n \times n$ with maximum degree $\delta(n)$, there is a way of killing blocks of size $b(n) \times b(n)$ each and a network R laid out in the same $n \times n$ area, such that N takes more than $p(n)/5 \log n$ time to simulate a step of R, under the constraint that no node in N simulates more than $c(n)$ nodes of R.

Proof: The essence of the proof is the same as in Theorem 9. The problem is that now we cannot assume that there are some two fixed nodes that must always be connected by a short path, because N might not use a specific node after some damage patterns have occurred. Therefore the main effort in this proof will be to show that we can always find two such nodes, that are used in many damage patterns, and should always be connected by a short path.

By our restriction of b, R can always be a layout of a complete binary tree in a given snake, where each leave is attached to an $2 \log n \times 2 \log n$ mesh. The diameter of such a network is $5 \log n$ ($2 \log n$ to route within each mesh, one at the source and one at the destination, and a path of at most length $\log n$ in the binary tree). If we restrict the towers in snakes to be at least $n/2$ high (in terms of nodes, not blocks), the number of processors that remain live in R is at least $n^2/8$. So to be able to simulate R, N must be able to route messages between any two nodes it is using in $p(n)$ steps.

For any snake, N may choose to use only some nodes for simulating other networks, in only some snakes. But since there are $(t(n))^{t(n)}$ different snakes, there must be one tower T_L which is the leftmost used in at least $(t(n))^{t(n)-1}$ of them (because there are only $t(n)$ choices of the leftmost one). Let us denote this collection of snakes Λ. We can assume that there is a node v_L in the base of the tower to the right of T_L, which is connected to every wire leaving this tower (see Figure 8).

Because one tower contains less then $n^2/t(n)$ nodes, N must use at least $t(n)/8c(n)$ towers in each snake. Therefore for all the snakes in Λ, N uses a tower which is at least $t(n)/8c(n)$ towers to the right of the leftmost one use. Again we can assume that there is a node v_R

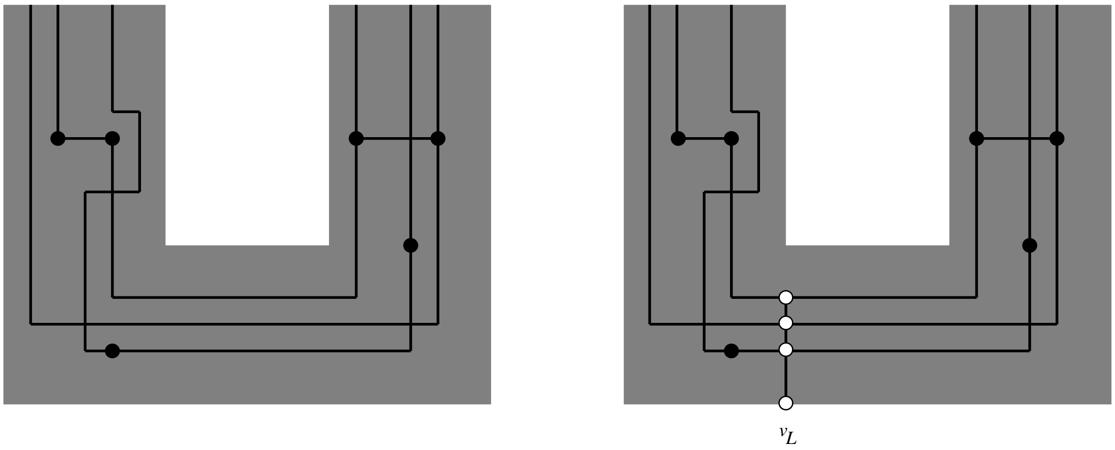


Figure 8: Ensuring that there is a vertex connected to all the wires crossing between two specific towers.

which is connected to every wire leaving a tower T_R which is $t(n)/8c(n)$ towers to the right of T_L. How many snakes there are in Λ which are different in one of the towers between T_L and T_R?

Since there are only $(t(n))^{t(n)-t(n)/8c(n)}$ different patterns for snakes outside this segment of towers, there must be at least

$$\frac{(t(n))^{t(n)-1}}{(t(n))^{t(n)-t(n)/8c(n)}} = (t(n))^{t(n)/8c(n)-1}$$

snakes in Λ that differ between T_L and T_R.

But now we obtain a contradiction, because from the inequality

$$p(n)\log\delta(n) < \left(\frac{t(n)}{8c(n)} - 1\right)\log t(n)$$

we conclude that the number of short paths between v_L and v_R is smaller than the number of different snakes in Λ between the two nodes, whereas in each snake in Λ these two nodes are used and therefore should have a short path between them. ■

4 Conclusions and Open Problems

This paper investigates the problem of fault tolerance in area-universal networks. Most of our results can be extended to 3-dimensional layouts and volume-universal networks. The main results obtained are the construction of a fault-tolerant network which is area-universal over all network layouts with few bends and the impossibility of constructing a fault-tolerant area-universal network. Other results which are interesting on their own are showing that it is impossible to have an area-universal network laid out in a narrow strip, and a new proof that the mesh of trees is area-universal. The proof of the last result together with the proof of Theorem 6, show the relation between the existence of area-universal networks and the ability to re-route a layout such that every wire has a small number of bends.

A few issues deserve some further discussion. One of them is the relation between shape and area universality. While in a very narrow rectangle it is impossible to lay out an area-universal network, it is possible to do so if the aspect ratio (the length divided by the width) is at most $n/\log n$. In such a rectangle it is even possible to have a fault tolerant area-universal network; if somebody chopped the strip into some shorter ones (along its short dimension), each one of them will still perform well. But exactly this is was shown to be impossible for a square.

Another issue is the relevance of universality over the class of networks with few bends. We believe it is quite relevant from several reasons. An obvious reason is that this is close to what one can hope for, because the general case is impossible. In other words, if a damage occurred, and our once universal network is outperformed by another network, it does not mean that the other one is fault tolerant; it is just suited very well to that particular damage pattern, but it too has a damage pattern that will severely degrade it. Another reason is that most or all layouts used in practice have few bends. and layout algorithms generate layouts with a small number of bends. Leiserson's layout algorithm [9] for example, generates 4-bend layouts. The same is true for Bhatt and Leighton's [2] layout framework, which generates $O(\log n)$-bend layouts.

Modeling faults in terms of blocks of size $\Theta(\log n) \times \Theta(\log n)$, and the reason for this, given in Theorem 7 imply that large systems should be dealt with in terms of subsystems, and not individual components. In fact, this is the way faults are usually handled in engineering environments — when a component fails, the module containing it is removed completely for repair. Another practical implication of our work is that when faults degrade the performance of a network too severely, it is a good idea to reconfigure the network. This is so because there are many cases in which a network that behaves poorly because of faults, can be reconfigured to give excellent performance.

We conclude the section with some open problems raised by this work.

- Is the expansion associated with the conversion of an arbitrary layout to one with few bends necessary? Removing the expansion in the area necessary for the layout will improve the running time of our simulation algorithms. Besides, it will indicate that there is no reason for having many bends layout other than to beat area-universal networks for some types of damage.

- This last conjecture bring forth the general question of the utility of networks with many bends. While the last problem was phrased in terms of rerouting the wires, the question can also be asked without regard to the present layout: Are there graphs all the compact layouts of which have many bends ? And is there another advantage for such layouts?

- Our routing fault tolerant algorithm for routing on the mesh of ladders is an off-line algorithm, because we use the routing of wires in the simulated network as a basis for the routing decisions (and therefore routing decisions are not based only on the origin and destination of messages). Can one obtain a comparable online algorithm?

Acknowledgments

Thanks to Charles E. Leiserson for suggesting this problem to me, and for numerous helpful discussions. Thanks to Mike Klugerman, Tom Leighton, Bruce Maggs and Boaz Patt-Shamir for many helpful discussions.

References

[1] B. Aiello and T. Leighton, Coding theory, hypercube embeddings, and fault tolerance, *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 125–136, 1991.

[2] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems, *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

[3] R. L. Greenberg. *Efficient Interconnection Schemes for VLSI and Parallel Computation*. PhD thesis, Department of Electrical Engineering and computer Science, Massachusetts Institute of Technology, September 1989. MIT/LCS/TR-456.

[4] T. Leighton. *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks*. MIT Press, Cambridge, Massachusetts, 1983.

[5] T. Leighton and C. E. Leiserson. Wafer-scale integration of systolic arrays. *IEEE Transactions on Computers*, C-34(5):448–461, 1985.

[6] T. Leighton and C. E. Leiserson. Personal communication.

[7] T. Leighton and B. Maggs. Expanders might be practical: fast algorithms for routing around faults on multibutterflies. *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 384–389, 1989.

[8] T. Leighton, B. Maggs and S. Rao. Universal packet routing algorithms. *Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science*, pages 256–269, 1988.

[9] C. E. Leiserson. *Area-Efficient VLSI Computation. ACM Doctoral Dissertation Award Series*, MIT Press, Cambridge, Massachusetts, 1983.

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

[11] C. E. Leiserson. VLSI theory and parallel supercomputing. In C. L. Seitz, ed, *Advanced Research in VLSI: Proceedings of the decennial Caltech Conference on VLSI*, pages 5–16, MIT Press, Cambridge, Massachusetts, 1989.

[12] C. Mead and M. Rem. Cost and performance of VLSI computing structures, *IEEE Journal of Solid State Circuits*, SC-14(2):455–462, 1979.

[13] C. H. Séquin, A. M. Despain and D. A. Patterson. Communication in X-trees, a modular multiprocessor system, *ACM 78 Proceedings*, ACM, 1978.

[14] C. D. Thompson. Area-time complexity for VLSI. In *Caltech Conference on Very Large Scale Integration*, pages 495–508, 1979.

[15] L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140, 1981.