Filtering Random Networks to Synthesize Interconnection Networks with Multiple Objectives

Vijay Lakamraju, Israel Koren and C.M. Krishna

Abstract— **Synthesizing networks that satisfy multiple requirements, such as high reliability, low diameter, good embeddability etc., is a difficult problem to which there has been no completely satisfactory solution. In this paper, we present a simple, yet very effective, approach to this problem. The crux of our approach is a filtration process that takes as input a large set of randomly generated graphs and filters out those that do not meet the specified requirements. Our experimental results show that this approach is both practical and powerful. The use of random regular networks as the raw material for the filtration process was motivated by their surprisingly good performance with regard to almost all properties that characterize a good interconnection network. This paper provides results related to the generation of networks that have low diameter, high fault tolerance and good embeddability. Through this, we show that the generated networks are serious competitors to several traditional well-known networks. We also explore how random networks can be used in a packaging hierarchy and comment on the scope of application of these networks.**

Keywords— **Interconnection networks, synthesis, random regular graphs, filters, diameter, fault tolerance, embedding, packaging.**

I. Introduction

THE design of the interconnection network is an important and integral part of the design of any multiprocessor system. Interconnection networks are as much a determinant of the performance and reliability of the system as the processors themselves. The network impacts the cost of the architecture and the cost of communicating between processors, as well as system reliability and the extent to which the system can degrade gracefully under processor or link failures. Considerable attention must be paid to ensure the network provides the services required of it, without greatly increasing the cost and complexity of the system.

A plethora of interconnection network topologies have been described in the research literature. They range from very simple structures such as bus, ring or tree, to more sophisticated ones such as the shuffle-exchange, hypercube, chordal ring, or the Banyan network [1], [2], [3]. Many of these networks were designed to target a particular interconnection requirement or to efficiently run specific applications. There are however many application areas where multiple requirements need to be satisfied, e.g., a distributed spacecraft computer. Here, the computer must not only be designed to have small internodal distances and good embeddability so as to minimize the energy consumed but must also be designed to have sufficient reliability so that it can last the entire mission. Most traditional networks are not apt for such applications because they do not perform well when it comes to multiple performance measures. Moreover, very little has been reported in the literature on the problem of synthesizing networks that satisfy multiple performance and reliability criteria [4].

This paper describes a simple, yet effective, approach to the synthesis of interconnection networks that satisfy multiple requirements. The requirements are specified in terms of appropriate measures and the available resources are specified in terms of the number of nodes and their degrees. First, a large number of random regular graphs of the desired size and degree are generated. They are then passed through a bank of *filters*. These filters identify a subset of networks which have the desired performance with respect to the specified measures. The crux of our approach lies in this simple filtration process. The distinguishing features of our technique are that it can be easily tailored to the specific performance and fault-tolerance measures of interest to the designer, and that it can be used even by those who are not experts in interconnection network design. Although the input to the filtration process is not limited to random networks, the selection of random networks was motivated by their surprisingly good properties and their flexibility and ease of generation.

Random graphs have been investigated for a long time: one of the earliest key papers being that of Erdös and Rényi [5]. From then on, interesting results have been reported on random graphs [6], [7], and the past few decades have seen the use of random graphs in many applications from computer science [8], [9], engineering [10], sociology, and ecology [11]. However, most of the results from random graph theory are of an asymptotic nature, in that a random graph is said to have a certain property Q if the proportion of graphs with this property tends to 1 as the size of the graph (i.e., the number of nodes in the network) tends to infinity. Therefore, these results are less useful for our problem in which the size of the network we are considering can range from tens of nodes to hundreds, or at the most, thousands of nodes. Moreover, the usefulness of our filtering approach rests on its efficiency, which is a measure of the number of random networks one has to generate before obtaining a useful short-list of "good" networks. This problem does not readily lend itself to theoretical analysis, and hence must be studied by simulation experiments.

The authors are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst MA 01003. Email: {vlakamra,koren,krishna}@ecs.umass.edu

This research was supported in part by DARPA and the Air Force Research Laboratory under Grant F30602-96-1-0341. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Projects Agency, the Air Force Research Laboratory, or the US Government.

This paper provides extensive experimental results on the performance of random regular networks with regard to various performance measures and the efficiency of the filtering process.

A. Performance Measures

Appropriate measures are required to convey the network-specific needs of the application in a form that is intelligible to the designer. A number of measures have been proposed for interconnection networks and there has been a lot of argument about which of these measures are most informative [12]. In this paper, we consider a subset of those measures in order to illustrate the advantages of our approach, rather than to provide a survey of the performance of random networks. We describe these measures in a graph-theoretic setting and use the terminology of [13]. The size of the network, which is the number of its nodes, is denoted by n and its degree by d. The measures we consider relate to the following desirable properties of an interconnection network: small internodal distances, good embeddability and good fault tolerance.

Processors in a distributed system communicate with each other through messages, and an important factor in determining the communication delay are the node-pair distances. The greater the node-pair distances, the greater the communication delay, the greater the time a message will spend in the network, the greater the energy consumed in delivering it, and the greater the chances of network congestion. Two measures used to describe the node-pair distances are the diameter (Δ) and the average node-pair distance ($\overline{D}$). The diameter is the maximum of the node-pair distances and provides an upper bound on the inter-task communication time, in terms of the number of hops. It can have a dominant impact on application runtime.

The efficient execution of parallel programs on a multiprocessor machine requires that the communicating processes of the program be assigned to the processors in such a way that the overall execution time is minimized. This is the mapping problem [14]. What we are more interested in, from the point of view of network design, is how to design a network that lends itself well to mapping different parallel programs. Some networks can embed different communication patterns more efficiently than others. This feature of a network is called *embeddability* and is important when applications with different communication patterns need to run on the same network. An interconnection network ill-suited to the prevailing communication pattern can result in message congestion and excessive communication delays which can greatly impact the execution time and the power consumption. A good network should be able to embed a wide range of topologies with low *dilation* [15], thus ensuring that a large number of algorithms will run efficiently on the system. We provide a precise definition for embeddability later in Section III.

Systems used in applications requiring high levels of reliability need to take into account the vulnerability of the interconnection network to link failures. Traditional measures, such as connectivity, are worst-case measures and can have limited expressiveness. Recently, more useful network measures have been proposed [16]. The probability of disconnection, $\pi_d(p_f)$, refers to how easily the network gets disconnected in the presence of link failures occurring independently with probability p_f. The diameter stability, $\Delta(p_f)$ and the average node-pair distance stability, $\overline{D}(p_f)$, measure the impact of link failures on the node-pair distances, given that the network is still connected. The size of the maximum connected component size, $\chi_{max}(p_f)$, captures how the network splinters after it gets disconnected; is it more likely to splinter into one large component which is still useful, and several small and useless components, or will all the components be too small to be useful? We will use these measures to assess the fault tolerance capabilities of the network.

B. The Problem Statement

The designer is provided with a set of resources and a set of design requirements. Cost is one of the main factors that limits the resources that the designer has to work with. In most cases, this manifests itself as a fixed number of nodes and a fixed number of communication links. A network should have good construction flexibility so that the network can be constructed for any desired size and degree. It is often advantageous to consider networks with small degrees since a small degree translates to reduced wiring and fewer I/O interfaces. Furthermore, if the degree is constant over all nodes, only one basic node design may be necessary. It is therefore preferable for the designer to consider fixed n and d.

The design requirements, on the other hand, are typically specified in the form: $m_1 \leq M \leq m_2$ where M is the performance measure and m_1 and m_2 are thresholds between which the performance of the network must fall. Sometimes, the designer does not have definite values for specifying the requirements but has some idea of the relative importance of the requirements. In such cases, thresholds for the design requirements can be specified in terms of *relative* rather than absolute terms (e.g., the best 5%). In summary, the designer is faced with the following problem:

Given a fixed set of nodes and a set of design requirements, how should the nodes be interconnected such that the given requirements are met under the specified degree constraints?

C. Organization of the paper

Our approach to the above problem consists of a two-step process: first, the generation of a large set of random regular networks, and then the isolation of just the right ones in the filtration step. We describe both these steps in some detail in the next section. Section III illustrates the effectiveness of our filtering approach by considering the synthesis of networks with required diameter, fault-tolerance characteristics and embeddability. En route, it also provides extensive experimental evidence to the good properties of random regular networks. In this section, we also explore the use of the filtration process to study tradeoffs between various performance measures. Such a tradeoff can help

us obtain better insight into the measures and could also help us reduce the time required to synthesize the required networks. One of the main impediments to the practical applicability of random networks is their lack of symmetry. This aspect can render them difficult to manufacture. In Section IV, we point out areas where random networks can be used in spite of their lack of symmetry and what performance advantages they have over traditional networks when used in a hierarchical packaging environment.

II. Approach

A. The Graph Generation Step

Random regular networks can be constructed for any value of n and d as long as nd is even. This flexibility of generation has been another reason that motivated their use. Two factors need to be considered while generating these networks for our purpose; speed and selection strategy. It is practically impossible to explore the entire space, $G(n,d)$, of all d-regular graphs with a fixed set of n vertices, because this space is huge even for small values of n; for example, the number of connected regular graphs for $n = 20$ and $d = 4$ is 985870522 [17]. Since we typically need to work with a limited time budget, only a subset of those graphs can be evaluated. Clearly, how we select these graphs from $G(n,d)$ can have an impact on the final output. In this paper, we explore schemes that attempt to select the graphs uniformly and randomly.

While a number of algorithms that generate d-regular graphs uniformly at random have been proposed, only few of them are of practical significance [18]. Bollobás' algorithm [19] takes an expected time of the order $nde^{(d^2-1)/4}$ and hence is not practical. In [20] a polynomial time algorithm of the order $O(nd^3)$ is given, however, it is prohibitively difficult to implement, and only applies to $d = O(n^{1/3})$. Simpler algorithms have been proposed in [21], [22], however, the graphs there have not been proven to be generated uniformly at random. Recently, Steger and Wormald [18] presented an algorithm that is both easy to implement as well as fast in practice, with an expected runtime of $O(nd^2)$. Although it does not generate the graphs uniformly at random, they prove that the algorithm generates d-regular graphs approximately uniformly, in the sense that all d-regular graphs on n nodes have in the limit the same probability to be selected as $n \to \infty$. We use this algorithm for our random graph generation. To generate a graph, we start with a set $U = \{1, 2, \ldots, nd\}$ of unpaired points. These nd points are grouped into n groups, each corresponding to a node in the graph. A pair of points is chosen at random from U and an edge is placed between nodes corresponding to their groups if they are *suitable*, i.e., they lie in different groups and no edge currently exists between nodes corresponding to the same two groups. The selected points are then deleted from U and another pair chosen. This process continues till no suitable pair exists or until the graph is d-regular. Finally, the graph is checked for connectedness; after all, at the end, each node in a network should be reachable from all other nodes.

In order to ascertain its practical applicability, the algorithm was implemented using simple data structures and run on a 500MHz machine with 256MB of memory. Figure 1 shows the runtime averaged over 1000 graphs for various sizes and degrees. Note that this time also includes the runtime of the trials in which the graphs had to be discarded, i.e., when no *suitable* pairs were available and when the graph was not connected. In spite of these cases, it was observed that networks of even 2048 nodes could be generated in about one second using this algorithm. This shows that the algorithm is quite fast in practice, which is important if we wish to generate a large number of graphs for filtering. The fraction of times a graph had to be discarded was well under 5% for graph sizes greater than 32. It is also important that the generated networks not be isomorphic to each other, since otherwise there will be multiple copies of the same network. We tested isomorphism between all the pairs of graphs using the NAUTY tool [23], [24] and observed that more than 95% of the networks generated were non-isomorphic to each other.

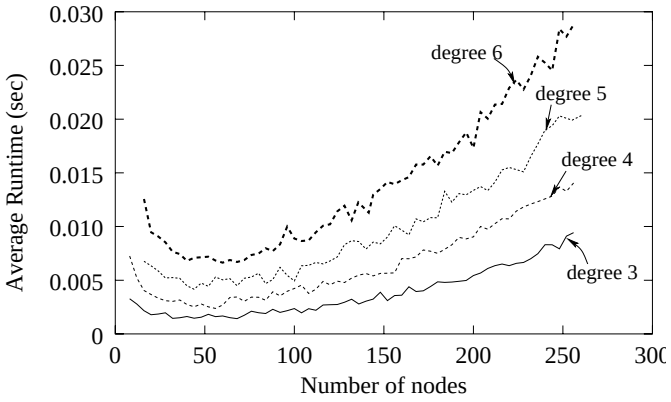


Fig. 1. Average runtime of the generation algorithm

B. The Filtering Step

This step takes as input the generated networks and tries to identify those that have the properties desired by the designer. It consists of a set of filters, one filter for each requirement to be satisfied. A filter consists of two parts: the evaluation part calculates the value of the measure associated with the requirement, and the checking part compares the value of the measure with the *threshold* as specified by the requirement. Each filter takes as input a set of random networks and outputs only those that pass the checking part. By using the output of one filter as input to the next, the output at the end of the series of filters, if there is one, is a set of networks that satisfy all the requirements. If the filtering process produces no output, the designer could either start all over again with a new batch of random networks or refine the thresholds of the filters. It helps to sequence the filters in decreasing priority order of the measures they represent if the designer has the option of relaxing the thresholds of some filters. The threshold of a higher-priority filter should not be relaxed before that of every lower-priority filter has been relaxed to

the maximum allowed extent. As the results in Section III will show, this simple technique produces networks that compare favorably to many traditional networks. The key feature of this filtering approach is its versatility, as the set of selected filters and their order is easily determined by the specific application requirements. For a different set of application requirements, we simply use the appropriate new set of filters and arrange them accordingly.

The evaluation part of the filter is typically much more time-consuming than the checking part and is mainly influenced by the computational complexity of the corresponding measure. For each filter, simulation or some algorithm is used to compute the measure. While the checking part is quite simple when thresholds are given in absolute terms, when relative thresholds are used, it requires that the networks be sorted according to their values obtained from the evaluation part. The time spent in the filtration process depends on the order in which the filters are arranged. The filters can be arranged in a serial or parallel fashion as shown in Figure 2. In sequential filtering, the threshold determines the number of networks that pass through at each stage. If a stringent threshold is used, a smaller number of networks pass through, and this greatly impacts the time spent in the remaining filters. The most time-efficient ordering would depend on both the thresholds and the evaluation times of each filter. In parallel filtering, the evaluation corresponding to each of the filters is carried out in parallel, and a single checking part that combines the checking parts from all the filters sifts out networks that comply with all the requirements. Here, the time spent in the filtration process is bounded by the maximum evaluation time among the filters. Note that, in either case, the final output is not impacted as the output of the filters is set-associative.

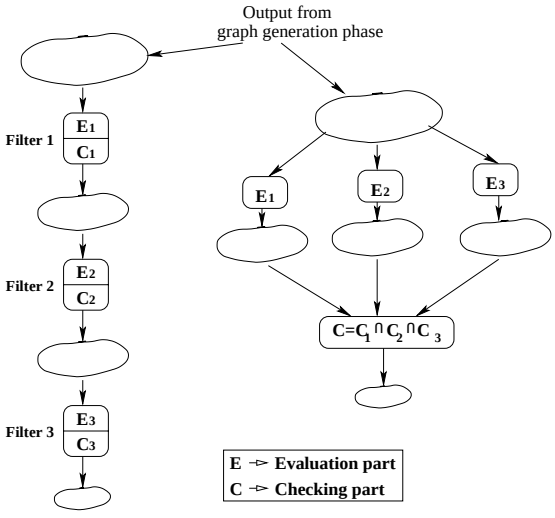


Fig. 2. Sequential and parallel filtering

III. Experimental Results

Networks synthesized using our approach are compared with various well-known and promising networks to illustrate the effectiveness of the approach. Though most of our examples are networks of degree 3 and 4, our approach is not restricted to these degrees. We use the following standard topologies for comparison against our degree-3 networks: shuffle exchange networks [25], cube connected cycles(CCC) [26], chordal rings of degree 3 [27], Moebius trivalent graphs [28] and multi tree structures (MTS) of degree 3 [29]. In the degree-4 category, we use meshes, torii, chordal rings of degree 4 [30], and wrapped butterfly networks.

A. The Diameter Filter

The problem of constructing a network of a given size and degree with the smallest possible diameter has been the focus of much research [31]. While the diameter of random graphs has also been studied [32], the asymptotic results provide little guidance for network sizes of interest to us. In order to evaluate the performance of random regular networks in conjunction with a diameter filter, we generated graphs sized between 8 and 256 nodes, with degrees ranging from 3 to 6, and calculated their diameters. For each size and degree, 1000 random networks were generated and the ones with the smallest diameter were selected. Figure 3 shows how the diameter varies as the size and degree is increased. These results give an idea of the lower bound that the threshold can be set to, for the diameter filter.

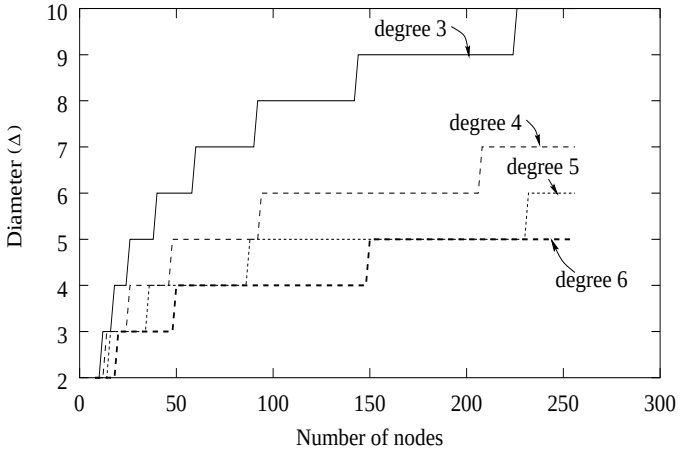


Fig. 3. Diameter of random regular networks

Figure 4 compares the diameter of random networks of degree 3 with other networks of the same degree. The diameters of the networks plotted are the ones with the least diameter as specified in the respective references. From Figure 4 it is clear that random networks have lower diameter than almost all the other networks considered, but they fail to win over some well-crafted interconnection networks such as the MTS network for some network sizes. However, graphs such as MTS do not have sufficiently good construction flexibility. The MTS network is defined only for certain sizes given by $m(d-1)^{t-1}$ where m and t are integer parameters[1]. Among networks of degree 4 that we have considered, random regular networks performed the

[1]Diameters of incomplete MTS networks have not been analyzed

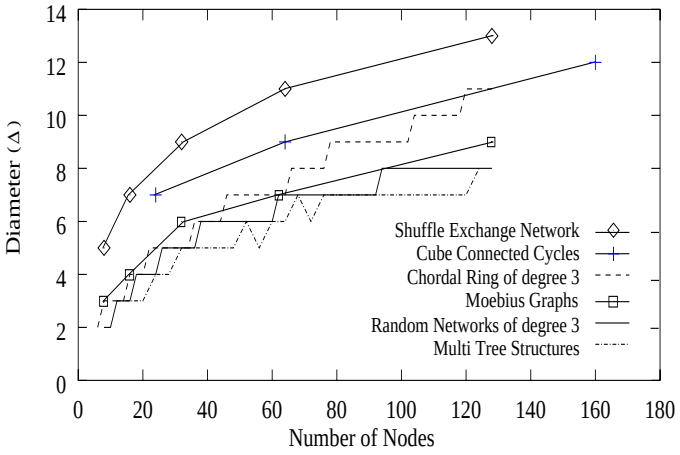


Fig. 4. Diameters of different networks of degree 3

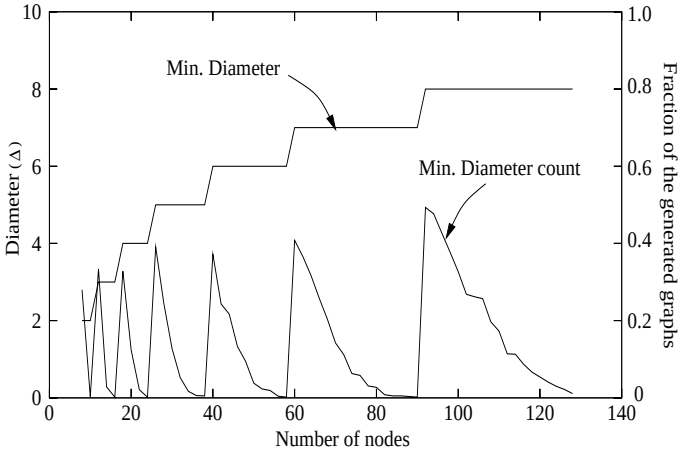


Fig. 5. Frequency of minimum diameter random networks of degree 3

best. Experiments have shown that random networks have better diameter than networks such as hypercubes, whose degree is a function of the size of the network. In fact, the higher the degree, the better is the performance that one can expect from random networks. All these results show that if we generate a sizeable number of random networks and select one with the smallest diameter, we will (with a high probability) get a network that is diameter-competitive with most of the interconnection networks described in the literature. In fact, a comparison between diameters obtained using this method and the entries in the (d, Δ) table[2] show that the diameters of the random graphs are greater by at most 1 than the corresponding best known diameters [34]. Note however that while the networks from the (d, Δ) table are constructed by different methods for different degrees and diameters, the random networks follow the same simple construction algorithm.

It is also worthwhile to find out the number of graphs that pass through the diameter filter when its threshold is set to different values. Figure 5 shows the fraction of the generated networks that passed through filters whose thresholds have been set at the minimum diameter (as shown in Figure 3). It gives some idea of the "yield" one can expect from a diameter filter.

B. The Embeddability Filter

Many parallel applications are designed to run efficiently on a certain network topology, e.g., parallel implementations of the Fast Fourier Transform run very efficiently on hypercubes. On the other hand, there are many other applications that do not make use of just one regular communication pattern but involve a mixture of several computational tasks and many patterns. One such suite of applications is that being considered by NASA's Remote Exploration and Experimentation (REE) project [35]. These applications not only exemplify the need for substantial computing power in future space missions, but also indicate the diversity of communication patterns that could be used in a single application. Each of these applications consists of multiple modules which use different algorithms and hence have different communication patterns. Table I shows the various modules, and the associated communication patterns for NGST (Next Generation Space Telescope), one of the applications in the REE application suite.

Running such an application efficiently on a distributed system requires that the interconnection network embed all the communication patterns without much performance degradation [36]. Similarly, while designing networks to support general-purpose parallel programming, the design cannot be unduly influenced by the characteristics of a single program. Moreover, a network that lends itself easily to embedding various communication patterns can greatly decrease the power requirement, which is an important issue in space systems where power is limited. The problem of designing such networks is NP-hard [37] and hence does not readily yield itself to theoretical solutions. Our filtering approach proves again to be an effective method here. Each filter in our filtration step now corresponds to one of the application modules. Putting these filters in tandem and relaxing their *relative* thresholds until some output is obtained, we can obtain networks that have good overall embeddability. Before we continue, we will first define embeddability.

A parallel program can be modeled as a graph, in the same way we model interconnection networks. This graph is termed the *guest graph* $G = \{V(G), E(G)\}$ whereas the interconnection network is termed the *host graph* $H = \{V(H), E(H)\}$. For our purposes, a mapping is a function $\tau : V(G) \rightarrow V(H)$ and each edge in $E(G)$ is mapped to a shortest distance path. The goal of the mapping is to balance the computational weight of the processes among the processors of the machine (i.e., the load), while reducing the cost of the communication by keeping intensively intercommunicating processes on nearby processors. A measure of how well the mapping algorithm has achieved this goal can be obtained from the communication cost. The com-

[2]The (d, Δ) table gives the *state of the art* with respect to a largest known graphs with degree d and diameter Δ. The table can be obtained from [33]. Elsewhere, this table has also been referred to as the (Δ, D) table.

TABLE I
MODULES AND ASSOCIATED COMMUNICATION PATTERNS IN NGST

NGST Module	Procedures	Communication pattern
Cosmic Ray Detection	Matrix inversions, summations	Mesh
Adaptive Optics Optimization	Derivatives, logarithms	Adaptive meshes
Spacecraft Attitude Control	Expert systems, logic trees	Trees
Data Compression	Searching, sorting, FFTs	Hypercube

munication cost that we have chosen is

$$C_{G,H} = \frac{\sum_{e \in E(G)} w(e) \cdot |\rho(e)|}{\sum_{e \in E(G)} w(e)}$$

where $|\rho(e)|$ is the length of the path in H used to route edge e and $w(e)$ is a weight function indicative of the amount of communication on edge e. A strong positive correlation between values of this function and effective execution times have been experimentally verified by some researchers [36], [38]. Since every hop that a message takes counts towards energy consumed, this function also encapsulates the amount of power spent on communication. Embeddability, as we define it, is a measure of how well a network is able to embed several guest graphs and is given by

$$C_H = \omega_1 \cdot C_{G_1,H} + \omega_2 \cdot C_{G_2,H} + \omega_3 \cdot C_{G_3,H} + \ldots + \omega_n \cdot C_{G_n,H}$$

where the graphs $G_1, G_2, G_3, \ldots, G_n$ represent the various communication patterns of programs to be run on the host graph and the weights $\omega_1, \omega_2, \omega_3, \ldots, \omega_n$ represent the relative frequency with which those communication patterns manifest themselves. A network is said to have better embeddability than another network for a certain certain set of guest graphs if it has a lower C_H value.

We used the following guest graphs for our experimental results: a binary tree, a mesh, a hypercube and a star. They represent the filters we would have to use to design a network to run the NGST application. The intuition behind choosing a star graph was to account for collective communication primitives such as broadcast, reduction and parallel prefix sums which are common in data parallel applications. We made a few assumptions while obtaining the results. First, we took the weights ω_1, ω_2, ω_3 and ω_4 to be equal. We also assumed that all the edge weights are equal to 1. Both the guest graph and host graphs used were of the same size and only one guest node was allowed to be mapped to a host node. We believe that these assumptions are valid in many cases and do not greatly skew the results obtained. Moreover, they help us to show the effectiveness of the filtering process without getting too specific about any particular application. Table II compares the performance of the "best" random network that we obtained with our filtering technique with some traditional well-known networks. The thresholds on each filter were adjusted to obtain at least one network at the output. This network is evidently the one with the least C_H value. 1000 networks were used in the filtering process.

For a fair comparison, we divided the candidate networks according to their degrees. Results from the table show that for the set of four guest graphs chosen, random graphs seem to perform better than any of the traditional networks considered in both the degree-3 and the degree-4 category. Since C_H is a normalized function, a small difference in the values could translate to big differences in execution time and power dissipation depending on the total communication volume. It is important to note that the results obtained are dependent on the mapping algorithm used. Since the mapping problem is NP-complete, nearly all practical implementations use heuristic methods to achieve a near-optimal solution. An important criteria for choosing the mapping algorithm is that it should not make any assumptions about the type of graphs used as inputs. Though a number of packages for solving mapping problems are publicly available [39], their main focus is to produce results quickly rather than reaching optimality. We have found, through numerous experiments, that a simple implementation based on the simulated annealing heuristic works better for the input graphs that we have considered and so, we used that to provide the mapping. In instances such as the embedding of the 64-node hypercube into the 8×8 mesh where theoretical results are available in the literature [40], the reported mapping was used. For a fuller treatment of the embeddability of random networks, see [41].

C. The Fault Tolerance Filter

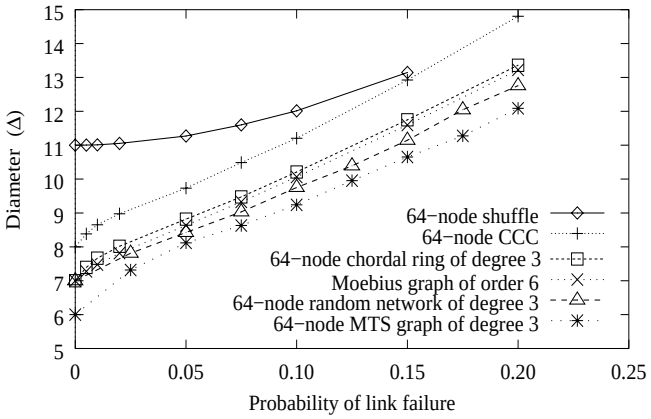
Four fault tolerance measures introduced earlier, namely diameter stability, $\Delta(p_f)$, average node-pair distance stability, $\overline{D}(p_f)$, probability of disconnection, $\pi_d(p_f)$, and maximum connected component size, $\chi_{max}(p_f)$, adequately capture network qualities such as graceful degradation and robustness. Here, we evaluate the vulnerability of regular random networks of size 64 and degree 3 and 4 and compare their performance with other networks of similar size and degree. The random networks were first passed through a diameter filter with threshold set to the minimum value and a network was selected at random from the output. Figure 6 shows the comparison of diameter stability among degree 3 and degree 4 networks. Here again, the random network of degree 4 outperforms all the networks in its category whereas in the degree 3 category, it is second-best. Figure 7 shows the average node-pair distance stability for degree-3 networks. The node-pair distance stability of degree-4 random networks behaves similarly to their diameter stability. The performance of random graphs in

TABLE II
COMPARING THE EMBEDDABILITY OF NETWORKS OF SIZE 64 AND DEGREE-3 AND DEGREE-4

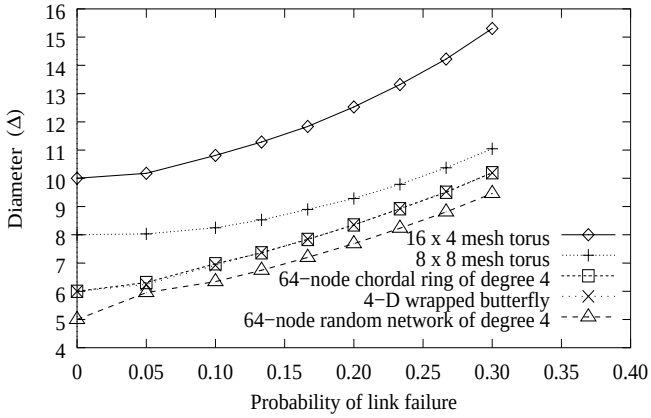
Guest → / Host ↓	degree	6D hypercube	8 × 8 mesh	64-node binary tree	64-node star	Composite application
64-node CCC	3	2.333	1.705	1.380	4.698	10.116
64-node chordal ring	3	2.448	1.784	1.349	4.412	9.993
64-node random	3	2.442	1.714	1.317	3.792	9.268
8 × 8 mesh	4	2.333	1.00	1.222	4.063	8.618
64-node butterfly	4	1.667	1.50	1.190	3.428	7.785
8 × 8 mesh-torus	4	1.667	1.00	1.190	4.063	7.920
64-node random	4	1.979	1.553	1.111	2.936	7.580

comparison with other networks with respect to the probability of disconnection and the maximum connected component size followed the same trend [21]. All these results show that random networks perform better than most of their counterparts with respect to fault tolerance as well. Careful examination of the results reveals that networks that are not regular are more vulnerable compared to those that are regular, as can be see in the case of shuffle exchange networks and meshes. This is another reason that motivates the use of *regular* random graphs rather than irregular ones.

Each of the fault tolerance measures is a function of the link failure probability, p_f. If the designer knows the exact link failure probability, then the value of the measure corresponding to that failure probability can be used for comparison with the threshold of the filter. However, in most cases, the designer would have to deal with a range of link failure probabilities. In this case, the designer would have to convert the values obtained for different link failure probabilities to a single number for use in filtering. This can be done by introducing a weight function, $w(p_f)$, and using a weighted sum of the values corresponding to different link failure probabilities. Figures 6 and 7 give an idea of the threshold values that can be applied for the $\Delta(p_f)$ and the $\overline{D}(p_f)$ filters.



(a) Comparison among degree 3 networks



(b) Comparison among degree 4 networks

Fig. 6. Diameter vs. probability of link failure

D. Putting the Filters Together

We now analyze the filtration process further by using it to study tradeoffs between measures. In other words, *how much do we have to lose in terms of one performance measure in order to achieve a desired level of performance with respect to another measure?* Filters corresponding to two measures are taken and their thresholds set such that only those networks, say within the top x%, pass through the filter. If the sets of networks output by both filters are identical (or nearly so), we can conclude that the measures associated with these filters are closely related. In such a case, we might even make do with just one of the two measures, which translates to one less filter. In this way, the filtering process can be used to determine how orthogonal two measures are with respect to each other. The greater the orthogonality between two measures, the higher should be the relative threshold set in each filter to obtain at least some non-zero output when the two filters are used in tandem.

As an example, we use the filters corresponding to the four fault tolerance measures. Experiments were performed using networks of size 64 and degree 3. For each filter, the networks were ranked according to their performance over link failure probabilities in the range $[0.0, 0.2]$, with each link failure probability having the same weight. The filters were then taken in pairs and the best x% at the output of the each filter were set aside. Let y denote how

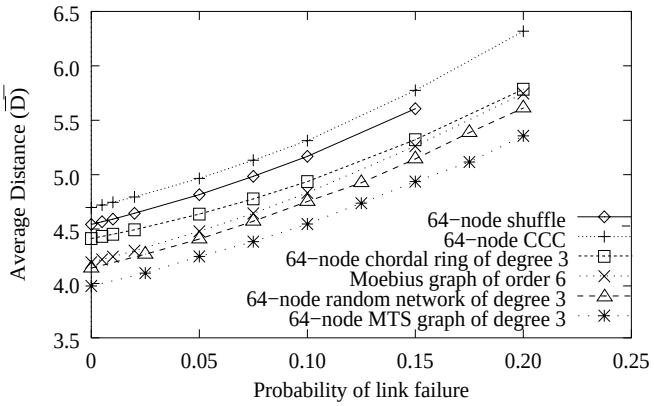


Fig. 7. Average node-pair distance vs. probability of link failure

many of the $x\%$ at the output of the first filter were present among the $x\%$ at the output of the second filter. Figure 8 shows y as a function of x for different pairs of measures. Pairs that are closer to the straight dotted line are less orthogonal compared to other pairs. It is clear from the figure that ($\Delta(p_f)$ and $\pi_d(p_f)$) are quite orthogonal whereas $\Delta(p_f)$ and $\overline{D}(p_f)$) are not. Looking at the closeness of the $(\Delta(p_f), \overline{D}(p_f)$ curve to the (ideal) dotted line, we may be able to make do with one of the two and so, only three of the four measures may be sufficient to describe the fault tolerance characteristics of the generated interconnection networks.

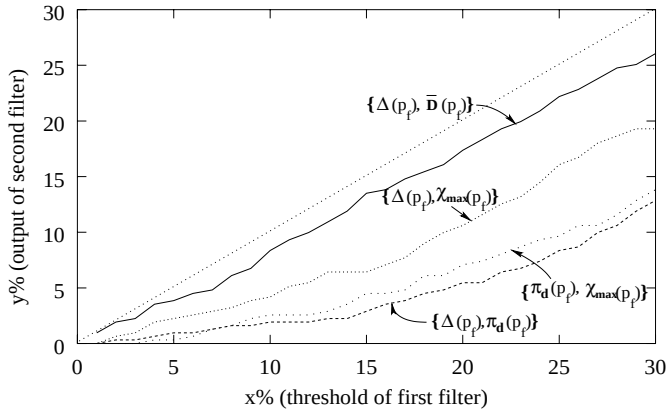


Fig. 8. Orthogonality between different pairs of measures

Let us now consider diameter, embeddability and fault tolerance together and see how random networks perform with regard to these measures simultaneously, keeping in mind the example of the spacecraft computer. For embeddability, consider the same set of guest graphs as considered earlier and for fault tolerance, consider average node-pair distance stability, $\overline{D}(p_f)$, for link failure probabilities between 0.0 to 0.2. Table III compares the performance of random networks with other networks of size 64 and degree 3. Results of the random network are based on an input set of 1000 graphs. It is clear that random networks perform consistently better than the other networks for each of the filters.

TABLE III
COMPARING THE NETWORKS WITH RESPECT TO THREE MEASURES

Graph	Diameter	Embeddability	$\overline{D}(p_f)$
8×8 mesh	8	8.618	1.116
8×8 Mesh-torus	6	7.920	0.845
Chordal Ring	6	7.785	0.748
Random	5	7.580	0.690

IV. Discussion

In this section, we comment on the practical applicability of random networks. Whether a random network can be used in a certain system or not is influenced by a number of factors. In the case of a network of workstations (NOWs) [42], there is usually a great deal of freedom with regard to the way the workstations can be interconnected. This provides the flexibility of employing a random network which can be synthesized using our approach, with the requirements of the application(s) that would run on the system. Cluster computing [43] provides another related application area for random networks. In reconfigurable computers [44] where the interconnection network can be programmed through a series of reprogrammable elements, random networks find immediate applicability. In commercial multiprocessor systems however, random networks have long been considered "misfits" for purposes of interconnection due to their lack of symmetry and hence, are not favored by designers in spite of their good overall properties. However, one must also realize that only a miniscule fraction of the multitude of interconnection networks that have been described in the literature have ever been employed in commercial systems; the mesh and the hypercube being the most popular [45]. The main reason for the popularity of these networks is their amenability to packaging.

A. Using Random Networks in a Hierarchical Structure

For large-scale multiprocessor systems[3], *multi-level hierarchies* become inevitable due to technological constraints [46]. It is not practical to have a 256-node network of processors inside a single VLSI chip using the current packaging technology. The current packaging hierarchy typically consists of chips, boards, racks and cabinets. This model allows increasing flexibility with regard to the interconnection between the elements of a level as we move up the hierarchy[4]. At the higher levels of the hierarchy, interconnections are typically done manually using wires or special cables. This increasing flexibility allows a designer to employ random networks higher up in the hierarchy. Here, we report on the performance advantages obtained by using random networks in such a hierarchical structure.

[3]By "large-scale", we mean network sizes greater than a couple of hundreds.

[4]Level 0 corresponds to processing elements packaged into chips

The packaging model that we use is similar to the one proposed in [46]. It reflects a generic packaging technology consisting of two levels, in which the processors are organized into a number of *modules* at the lower level. What a module physically corresponds to depends on the scale of the machine; it could be a board containing an interconnection of processor chips, a rack consisting of multiple boards or even a cabinet consisting of multiple racks. The interconnection within a module forms the intra-module network, whereas the interconnection between modules forms the inter-module network. A schematic of the hierarchy is shown in Figure 9. Each module has a fixed number of pins through which they communicate with each other. This pin count can also be viewed as the degree of the inter-module network. We consider the intra-module network to be some regular network and the pins at the boundary of the module as the outcome of some edges in the intra-module network being deleted and replaced by lines that connect to the pins. Such a procedure was adopted to enforce the notion of regularity which we have been considering throughout this paper. For every edge that is deleted, we get two pins at the boundary of the module. We then design the inter-module network using our filtering approach with the number of pins as the degree and the number of modules as the size of the network. We are interested in the performance advantage provided by such a design.

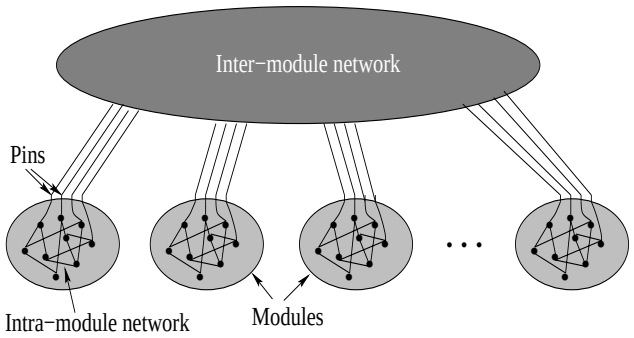


Fig. 9. Schematic of a 2-level packaging hierarchy

For our first set of experiments, we chose a 4×4 mesh-torus for the intra-module network. We considered 64 such modules for a machine size of 1024 processors. Our filtering technique was used to generate the inter-module network that would give the least overall diameter. Note that, as always, we could use any other measure or set of measures besides diameter in the filtration process but we feel that considerable insight can be obtained using a simple measure like diameter. The 4×4 mesh-torus was then replaced by a random network of size 16 and degree 4 and the experiments rerun. Figure 10 summarizes the results obtained for different values of the pin-count. For a pin-count of 2, there is no other option of interconnecting the modules other than as a ring. Increasing the pin-count to 4 provides a substantial improvement in the diameter. A diameter of 21 is obtained when the inter-modular network is implemented as a random network. This should be compared against a diameter of 36 (not shown in Figure 10) which would have been obtained if the inter-modular network used a 8×8 mesh-torus interconnection. The figure shows that further improvements can be obtained if both the intra-modular as well as the inter-modular networks are implemented using random networks. This improvement is not significant as we further increase the number of pins from 6 to 12. Note that a greater pin count translates to a greater cost and the cost per wire is higher at higher levels of the hierarchy. A simple cost analysis favors the use of 4 as the pin-count. It is important to note that for each pin count, selecting a different set of edges to be deleted can give a different diameter. For this reason, we tried a number of edge combinations for each pin-count and used the one that gave the minimum diameter.

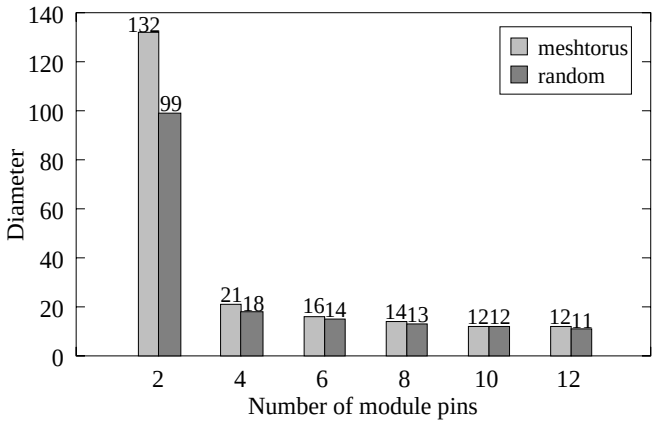


Fig. 10. Diameter as a function of the number of pins in the module

Our next set of results examines the effect of the number of modules and the topology of the intra-modular network on the diameter of the entire network. The size of each module is reduced as the number of modules is increased in order to keep the size of the network constant. For example, for the number of modules equal to 128, the size of each module is reduced to 8. As before, results were generated for both the mesh-torus and the random network as the intra-modular network. Figure 11 shows a plot of the results obtained with a pin-count of 4. Advantages obtained from using random networks are apparent from the figure. If the entire network were implemented as a 32×32 mesh-torus, the best diameter we can get is 32 which is much larger than the 22 we get if it is implemented as a 2-layer network, with a 32-node mesh-torus forming the layer-1 network and a 32-node random network, obtained through our filtering approach, forming the layer-2 network. A further improvement is obtained if the layer-1 network is also generated using our filtering technique. Such a design can be justified for applications requiring customized solutions [47] where the system is designed from scratch and the designer has complete freedom. Further analysis of packaging using a detailed cost model is presented in [34].

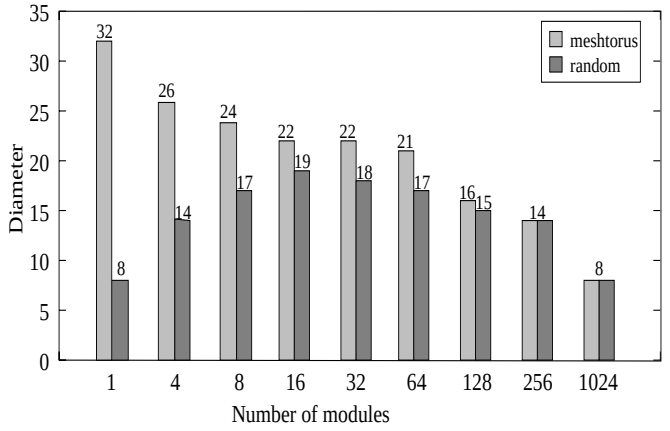


Fig. 11. Diameter as a function of the number of modules

B. Closing Remarks

The contribution of this paper is two-fold: First, we demonstrate that random regular graphs are serious competitors to several traditional networks that are used in practice. Second, we show that filtering these networks provides a more flexible and practical method to design networks that satisfy multiple performance requirements than the traditional approach of choosing one of the standard networks or constructing a network by hand.

The strength of this approach lies in the versatility and extendability of the filtering step, in that a different set of filters can be used for a different set of requirements and new filters can be added as and when newer measures are developed. This paper demonstrates the effectiveness of this approach through extensive experimental results. Superior interconnection networks were synthesized using a moderate number of graphs; around 1000 graphs were sufficient in most of the cases that we studied. We have also used the filtering approach to test orthogonality and study tradeoffs between various measures. Lastly, we have tried to motivate the future use of random networks by discussing where and how they can be used.

Acknowledgement

The authors wish to thank Zahava Koren for stimulating discussions and suggestions.

References

[1] T. Feng, *Editorial Introduction, Tutorial Interconnection networks for Parallel and Distributed Systems*, IEEE Press, Piscataway, NJ, 1984.

[2] R.A. Finkel and M.H. Solomon, "Processor interconnection strategies," *IEEE Trans. Computers*, vol. 29, pp. 360–370, May 1980.

[3] N. Srinivasan J. Opatrny, D. Sotteau and K. Thulasiraman, "DCC linear congruential graphs: A new class of interconnection networks," *IEEE Trans. Computers*, vol. C-45, pp. 156–164, Feb 1996.

[4] S.B. Akers and B. Krishnamurthy, "A graph theoretic model for symmmetric interconnection networks," *IEEE Trans. Computers*, vol. C-38, pp. 555–565, Apr 1989.

[5] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[6] B. Bollobs, *Random Graphs*, Academic Press, London, 1985.

[7] S.Janson, T. Luczak, and A.Rucinski, *Random Graphs*, John Wiley and Sons, 2000.

[8] S.E. Fahlman, "The hashnet interconnection scheme," Technical Report CMU-CS-80-125, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1980.

[9] B. Hayes, "Graph theory in practice: Part I and II," *American Scientist*, vol. 88, no. 1–2, pp. 9–13,104–109, January,April 2000.

[10] A.K.C. Wong and M.L. You, "Entropy and distance of random graphs with application to structural pattern recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 7, pp. 599–609, 1985.

[11] D. Stoyan and J. Ohser, "Correlations between planar random structures with an ecological application," *Biometrical Journal*, vol. 24, pp. 631–647, 1982.

[12] F.S. Annexstein, *Interconnection Networks: Regularity and Richness*, Ph.D. thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, 1991.

[13] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, American Elsevier, New York, 1976.

[14] S.H. Bokhari, "On the mapping problem," *IEEE Trans. Computers*, vol. C-30, no. 3, pp. 207–214, Mar 1981.

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

[16] V. Lakamraju, Z. Koren, I. Koren, and C.M. Krishna, "Measuring the vulnerability of interconnection networks in embedded systems," in *Proceedings of the First Merged Symposium IPPS/SPDP, EHPC Workshop*, April 1998, pp. 919–924.

[17] R.C. Read and R.J. Wilson, *An Atlas of Graphs*, Oxford University Press, Oxford, UK, 1998.

[18] A. Steger and N. C. Wormald, "Generating random regular graphs quickly," *Combinatorics, Prob. and Comput.*, vol. 8, pp. 377–396, 1999.

[19] B. Bollobas, "A probabilistic proof of an asymptotic formula for the number of labelled regular graphs," *European Journal of Combinatorics*, vol. 1, pp. 311–316, 1980.

[20] B.D. McKay and N. C. Wormald, "Uniform generation of random regular graphs of moderate degree," *Journal of Algorithms*, vol. 11, no. 1, pp. 52–67, Mar 1990.

[21] V. Lakamraju, I. Koren, and C.M. Krishna, "Synthesis of interconnection networks: A novel approach," in *Proceedings of the 20th International Conference on Dependable Systems and Networks*, June 2000, pp. 56–64.

[22] G. Tinhofer, "On the generation of random graphs with given properties and known distribution," *Appl. Comput. Sci. Ber. Prakt. Inf.*, vol. 13, pp. 265–296, 1979.

[23] B.D. McKay, "Practical graph isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, Feb 1981.

[24] B.D. McKay, "The nauty page," http://cs.anu.edu.au/∼bdm/nauty/.

[25] H. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Computers*, vol. C-20, pp. 153–161, Feb 1971.

[26] F.P. Preparata and J. Vuillemin, "The cube connected cycles: A versatile network for parallel computation," *Communications of the ACM*, pp. 300–309, May 1981.

[27] B. Arden and H. Lee, "Analysis of chordal ring networks," *IEEE Trans. Computers*, vol. C-30, pp. 291–295, Apr 1981.

[28] W.E. Leland and M.H. Solomon, "Dense trivalent graphs for processor interconnection," *IEEE Trans. Computers*, vol. vol C-31, pp. 219–222, March 1982.

[29] B. Arden and H. Lee, "A regular network for multicomputer systems," *IEEE Trans. Computers*, vol. C-31, pp. 60–69, Jan 1982.

[30] K.W. Doty, "New designs for dense processor interconnection networks," *IEEE Trans. Computers*, vol. C-33, pp. 447–450, May 1984.

[31] J.C. Bermond and B.Bollobas, "The diameter of graphs - a survey," *Congressus Numerantium*, vol. 32, pp. 3–27, 1981.

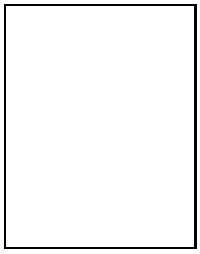
[32] B. Bollobas and W.F. La Vega, "The diameter of random regular graphs," *Combinatorica*, vol. 2, pp. 125–134, Feb 1982.

[33] A World Combinatorics Exchange resource, "The (d-k) table," http://www-mat.upc.es/grup_de_grafs/grafs/taula_delta_d.html.

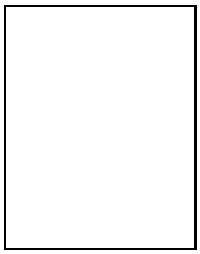
[34] V. Lakamraju, *Networking Issues in Distributed Real-Time Systems*, Ph.D. thesis, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, 2002.

[35] R. Ferraro, "NASA Remote Exploration and Experimentation Project," http://www-ree.jpl.nasa.gov/.

[36] B. Hendrickson and R. Leland, "An empirical study of static load balancing algorithms," in *Proceedings of the Scalable High Performance Computer Conference*, 1994, pp. 682–685.
[37] J.B. Saxe, "Embeddability of graphs in k-space is strongly NP-hard," in *Proceedings of the 17th Allerton Conference in Communication, Control, and Computing*, 1979, pp. 480–489.
[38] S.W. Hammond, *Mapping unstructured grid computations to massively parallel computers*, Ph.D. thesis, Ransellear Polytechnique Institute, Troy, NY, 1992.
[39] F. Pellegrini and J. Roman, "SCOTCH: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs," *Lecture Notes in Computer Science*, vol. 1067, pp. 493–499, 1996.
[40] B. Monien and H. Sudborough, "Embedding one interconnection network in another," in *Computational graph theory*. 1990, vol. 7 of *Computing. Supplementum*, pp. 257–282, Springer.
[41] V. Lakamraju, I. Koren, and C.M. Krishna, "Embeddability in random regular graphs," Tech. Rep. ECE-CSE-00-6, Dept. of Electrical and Computer Engineering, University of Massachusetts Amherst, 2000.
[42] T. E. Anderson, D. E. Culler, and D. A. Patterson, "A case for NOW (Networks of Workstations)," *IEEE Micro*, vol. 15, no. 1, pp. 54–64, Feb 1995.
[43] M. Baker, "Cluster computing white paper," http://www.top500.org/.
[44] S. Casselman, "Virtual computing and the virtual computer," in *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, D. A. Buell and K. L. Pocek, Eds., Napa, CA, Apr 1993, pp. 43–48.
[45] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Press, 1997.
[46] M.T. Raghunath, *Interconnection network design based on packaging considerations*, Ph.D. thesis, Computer Science Division, University of California, Berkeley, 1993.
[47] K.B. Clark, "From the sun to pluto," in *Proceedings of IEEE Digital Avionics System Conference*, Nov 1998, pp. 3–17.

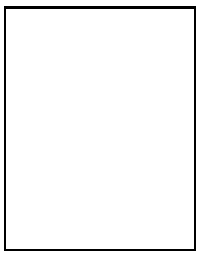


Vijay Lakamraju received the BE degree in Electonics and Communication Engineering from Bhopal University, Bhopal, India in 1994 and the M.S.E.C.E from the University of Massachusetts, Amherst in 1999. He received his Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts, Amherst in May 2002. His research interests include distributed real-time systems, cluster computing, fault tolerance, high-speed networking and performance analysis. He is a member of the Honor Society of Phi Kappa Phi and IEEE.



Israel Koren (S'72 - M'76 - SM'87 - F'91) received the B.Sc., M.Sc. and D.Sc. degrees from the Technion - Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in Electrical Engineering. He is currently a Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. Previously he was with the Technion - Israel Institute of Technology. He also held visiting positions with the University of California at Berkeley, University of Southern California, Los Angeles and University of California, Santa Barbara. He has been a consultant to several companies including IBM, Intel, Analog Devices, AMD, Digital Equipment Corp., National Semiconductor and Tolerant Systems.

Dr. Koren's current research interests include Techniques for Yield and Reliability Enhancement, Fault-Tolerant Architectures, Real-time systems and Computer Arithmetic. He published extensively in several IEEE Transactions and has over 160 publications in refereed journals and conferences. He currently serves on the Editorial Board of the IEEE Transactions on VLSI Systems. He was a Co-Guest Editor for the IEEE Transactions on Computers, special issue on High Yield VLSI Systems, April 1989 and the special issue on Computer Arithmetic, July 2000, and served on the Editorial Board of these Transactions between 1992 and 1997. He also served as General Chair, Program Chair and Program Committee member for numerous conferences. He has edited and co-authored the book, *Defect and Fault-Tolerance in VLSI Systems,* Vol. 1, Plenum, 1989. He is the author of the textbook *Computer Arithmetic Algorithms,* A.K. Peters, Ltd., 2002.



C.M. Krishna received his Ph.D. from the University of Michigan in 1984, and has been on the faculty of the University of Massachusetts ever since. His research interests include real-time systems, computer networks and power-aware computing.