Synthesis of Interconnection Networks: A Novel Approach

Vijay Lakamraju, Israel Koren and C.M. Krishna

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst 01003
E-mail: {vlakamra,koren,krishna}@ecs.umass.edu

Abstract

The interconnection network is a crucial element in parallel and distributed systems. Synthesizing networks that satisfy a set of desired properties, such as high reliability, low diameter and good scalability is a difficult problem to which there has been no completely satisfactory solution.

In this paper, we present a new approach to network synthesis. We start by generating a large number of random regular networks. These networks are then passed through filters, which filter out networks that do not satisfy specified network design requirements. By applying multiple filters in tandem, it is possible to synthesize networks which satisfy a multitude of properties. The filtered output thus constitutes a short-list of "good" networks that the designer can choose from. The use of random regular networks was motivated by their surprisingly good performance with regard to almost all properties that characterize a good interconnection network.

Experimental results have shown that this approach is practical and powerful. In this paper we focus on the generation of networks which have low diameter, good scalability and high fault tolerance. These generated networks are shown to compare favorably with several well-known networks.

[1]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.

1. Introduction

Interconnection networks (ICNs) are as much a determinant of performance and dependability in a parallel or distributed 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.

This paper describes a new approach to the synthesis of interconnection networks for parallel and distributed systems. The distinguishing features of our technique are that it can be 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 networks. It is especially useful when seeking to synthesize a network that performs well with respect to *multiple* performance measures. For example, a designer may place a high premium on both scalability and network resilience, while simultaneously needing to constrain the degree of the network. It can also be used to study tradeoffs among several performance or dependability parameters.

A vast literature on interconnection networks exists. Networks such as the hypercube, shuffle-exchange, Banyan, bus, chordal ring, tree and others, have been extensively studied [8, 9]. However, much less has been reported on the problem of synthesizing a network to meet specific performance and reliability criteria.

In our approach, the designer specifies the performance measures of interest. These may be commonplace measures such as bandwidth, diameter, connectivity, or more exotic measures like diameter stability in the face of failure, the extent to which the network splinters as node and link failures accumulate, or scalability. A large number of random regular networks of the desired size are then generated and passed through a bank of *filters*. Each filter is associated with a per-

formance requirement. The filters identify a subset of networks which have the desired performance with respect to the specified measures. This subset constitutes a short-list of networks from which the designer can choose.

The usefulness of this approach rests on its efficiency. That is, the number of random networks one has to generate before obtaining a useful short-list of "good" networks. This problem does not readily yield to theoretical analysis, and must be studied by simulation experiments. We have found, through extensive experimental work, that our technique is surprisingly efficient.

The rest of the paper is organized as follows. In the next section, we briefly review various desirable properties of ICNs. In Section 3, we describe our random network generation algorithm and the filtering process. Section 4 provides extensive experimental evidence to the good performance of random regular networks. It also shows the effectiveness of the filtering approach through examples. Section 5 summarizes our findings and discusses future work.

2. Preliminaries

A good interconnection network is characterized by a number of desirable properties. Some of these are listed below:

- *Small internodal distances.* One factor in the communication delay is the node-pair distances. The greater the average node-pair distance, 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.

- *Small, fixed degree.* Each physical connection costs money and a small degree corresponds to reduced wiring and fewer I/O interfaces. Furthermore, if the degree is constant over all nodes, then only one basic node design may be necessary.

- *Good fault tolerance.* Many parallel or distributed systems are used in applications requiring levels of reliability that can only be achieved by making the system fault-tolerant. There are many measures of fault-tolerance: we list below a partial list of the more useful network measures.

  - Probability of network disconnection.
  - Diameter stability, i.e., how the network diameter is expected to increase as nodes or links fail.
  - Stability of the average node-pair distance as nodes or links fail.
  - 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?

- *Easy construction and good scalability.* It should be possible to construct a network of any desired size. Further, adding a few nodes to the network should not cause drastic changes in such properties as diameter or average node-pair distance. A scalable ICN should be able to accommodate small increases in size rather than only large increases.

- *Embeddability.* Some algorithms are designed to run well on certain topologies, i.e., those that map well to the communication pattern of the application. A good network should be able to embed a wide range of topologies with low dilation, thus ensuring that a large number of algorithms will run efficiently on the selected ICN.

- *Easy routing algorithms.* It is advantageous to have a simple routing algorithm, for example, one that requires only the knowledge of the destination address. Routing algorithms can have a big impact on congestion and power requirements. Networks that facilitate the use of such simple algorithms are preferable.

These measures will vary in importance from one application to another. For example, space applications may require massive levels of fault-tolerance and low power consumption, while not placing a large premium on scalability.

Interconnection networks can be represented as graphs in which the vertices correspond to processors and the edges to communication links. In this paper we use the terms networks and graphs interchangeably. We consider only undirected graphs and the size of a network refers to the number of vertices in the graph. In this paper, we mainly concentrate on networks of degree 3 and 4, though our approach is not restricted to these degrees. For comparing the performance of different measures among degree-3 networks, we use the following topologies: shuffle exchange networks [16], cube connected cycles(CCC) [14], chordal rings of degree 3 [1], Moebius trivalent graphs [12] and multi tree structures(MTS) of degree 3 [2]. In the degree-4 catogory, we use meshes, torii, chordal rings of degree 4 [7] and the wrapped butterfly networks. In the next section, we describe our approach to synthesizing networks which meet the designer's requirements.

3. Approach

Our approach to network synthesis consists of a two-step process: first, the generation of a large set of random regular networks and second, the isolation of just the right ones through a process of filtering.

3.1. Generating Random Regular Networks

We use the following definition of random regular graphs:

Definition 3.1 *A random regular connected graph of size n and degree d is a d-regular connected graph in which node pairs connected by an edge are selected at random.*

Random regular graphs of n nodes and degree d are generated as follows. We start with a set of n isolated nodes. Edges are placed between node pairs selected at random. This process continues until all the nodes in the network satisfy the following two requirements: *(i)* the degree of all the nodes is the same and equal to the specified value, d and *(ii)* no pair of nodes is joined by more than one edge, and no self-loops exist. Finally, the generated network is tested for connectedness. Algorithm 1 contains the pseudocode used to generate random regular networks.

Algorithm 1
generate_regular_random_network(size,degree,seed)

```
1:  A ⇐ {1, ..., n}
2:  repeat
3:     Randomly pick two nodes, u and v, from set A
4:     if ((u ≠ v) and edge(u, v) not already present)
       then
5:        Add edge(u, v) to the adjacency matrix
6:        update A by removing nodes whose degree has
          been satisfied
7:     else if (size(A) = 1) or
             (nodes in A form a fully connected subgraph)
       then
8:        discard and start all over again
9:     end if
10: until size(A) = 0
11: check for connectedness
12: if graph not connected then
13:    discard and start all over again
14: else
15:    return adjacency matrix
16: end if
```

A is the set of all nodes whose degree has not been satisfied and is initialized on line 1 to the set of all n labelled nodes. Lines 4–6 ensure that the two conditions stated above are met and lines 7–8 ensure that the algorithm does not loop infinitely. If, during construction, the nodes in A form a fully connected subgraph, then no matter which two nodes are picked, the connecting edge will always be superfluous. No attempt is made to backtrack from this situation, and so the current adjacency matrix is discarded and a new one generated.

The above algorithm generates a random regular network each time it is called with a different seed value. Note that under some conditions, such as those on lines 7 and 12, the network that is being generated needs to be discarded and the generation restarted. To estimate the runtime required to generate a valid graph, we generated a large number of random graphs for various network sizes and calculated the average runtime for each network size and for different degree networks. Results shown in Figure 1 were obtained on a 500MHz Pentium having 256MB of memory. It was observed that networks of even 2048 nodes could be generated in less than a second using this algorithm. This shows that the generation algorithm can output a random regular graph in reasonable time. It was also observed that the check for graph connectedness (line 12 of the algorithm) was almost always satisfied. It is also important that the generated networks are non-isomorphic to each other; otherwise the filtering process will not make sense. To find out how many of the generated networks are non-isomorphic, we checked the isomorphism between all pairs of networks and observed that more than 99% of the networks were non-isomorphic to each other. All these results show that the generation algorithm provides a cheap and versatile method for producing the "raw" material for the filtration process. To get a better idea of the number of distinctive networks that can be generated with size n and degree d, see [15].

3.2. The Filtering Process

The raw material for the filtering process is the set of random graphs generated. Filtering consists of identifying those networks which have the properties desired by the designer.

We use one filter for each requirement to be satisfied. Typically each requirement is associated with a single performance measure or a set of measures. 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 a *threshold* specified by the requirement. For example, if the requirement was a diameter no greater

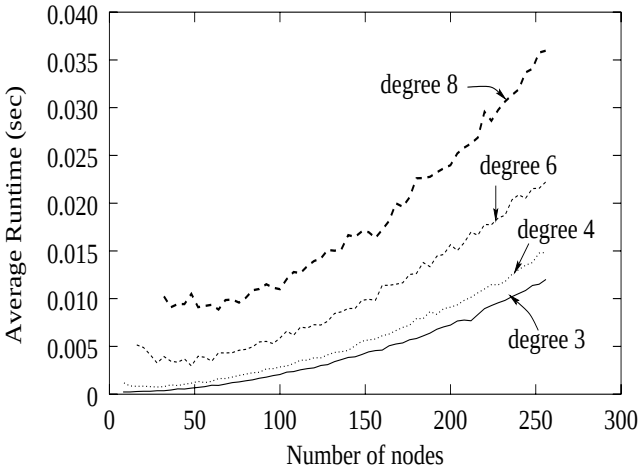


Figure 1. Average runtime of the generation algorithm

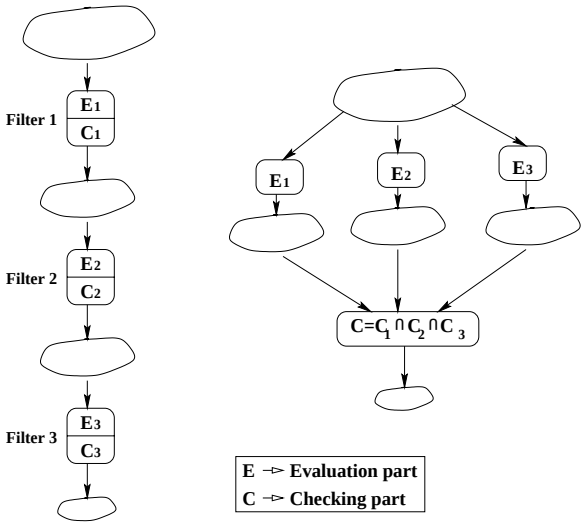


Figure 2. Sequential and Parallel Filtering

than k, then the evaluation part computes the diameter of the network and the checking part checks whether this requirement has been met. Each filter takes as input a set of random networks and outputs only those that pass the checking part. The output of one filter is used as input to the next. The filters are arranged sequentially one after the other in decreasing priority order of the measures they represent. The output at the end of the entire filtering process depends on the threshold values that have been set for each filter. If the filtering process produces no output, the designer will have to refine the threshold values. 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. The key feature of this filtering approach is its versatility, as the set of selected filters and their order is determined by the specific application requirements.

The evaluation part is typically much more time-consuming than the checking part. In order to speed up the filtering process, the evaluation corresponding to each of the filters can be carried out in parallel and a single checking part that combines the checking parts in all filters used to sift out networks that comply with all the requirements. This approach is called parallel filtering, compared to the sequential filtering that we described earlier (Figure 2). Note that the time taken in the case of parallel filtering is bounded by the maximum evaluation time among the filters, and evaluation is carried out for all the input networks. 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. Thus, the time taken in the case of sequential filtering is dependent on the threshold set in each filter.

Some implementation details are worth mentioning. One need not store the adjacency matrices of all the input networks because they can be regenerated easily and quickly using the seed value. So, only the seed values used to generate the random networks need to be stored. Also, thresholds can be specified in relative terms rather than using an absolute threshold value (for example, take the best 5% of the input networks), although this requires sorting the input networks according to the value obtained from the evaluation part.

4. Experimental Results

In this section, we demonstrate the efficiency of our filtering approach by considering the synthesis of ICNs with required diameter, scalability and fault-tolerance characteristics.

4.1. The Diameter Filter

The diameter, Δ, which is the maximum of the node-pair distances, provides an upper-bound on the inter-task communication time, in terms of hops, and can be a decisive factor in application runtime. The problem of constructing a network of a given size and degree with the smallest possible diameter has been the focus of much research [4, 13]. While the diameter of random graphs has also been studied, the published results tend to be of an asymptotic nature, valid as

the size of the graph approaches ∞. These asymptotic results provide little guidance for graph sizes that are of practical interest. In order to evaluate the diameter of random regular networks, we generated random networks sized between 8 and 256 nodes and 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 least diameter were selected. Figure 3 shows how the diameter slowly increases with size and how it reduces as the degree is increased. These results provide a lower bound to the threshold that can be set for the diameter filter. Figure 4 shows the comparison of 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 their respective references.

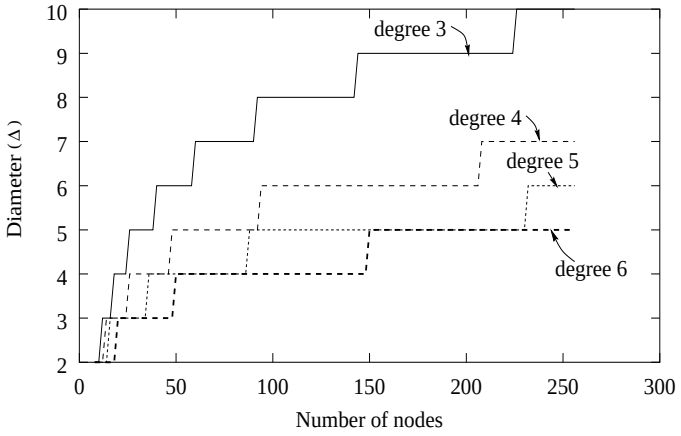


Figure 3. Diameter of random regular networks

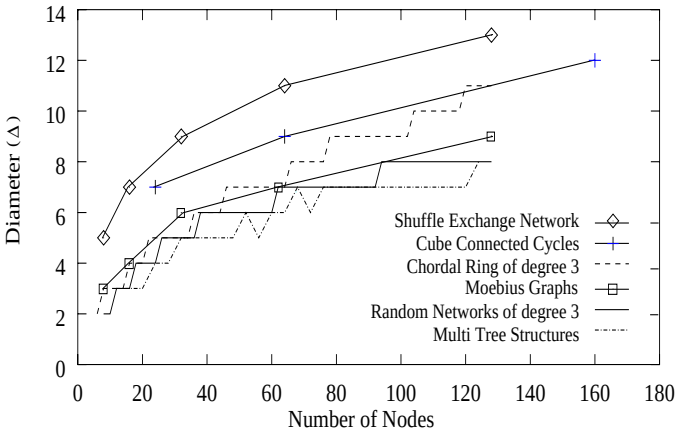


Figure 4. Diameters of different networks of degree 3

From Figure 4 it is clear that random networks perform better than such common ICNs as the mesh and the hypercube, but have greater diameter than some well-crafted ICNs such as the MTS network for some network sizes. However, graphs such as MTS are not as flexible. 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 best. It is worthwhile to find out the number of graphs that pass through when the threshold of the diameter filter is set to different values. Figure 5 shows the frequencies of networks of degree 3 that pass through diameter filters whose thresholds have been set at the minimum diameter (as shown in Figure 3).

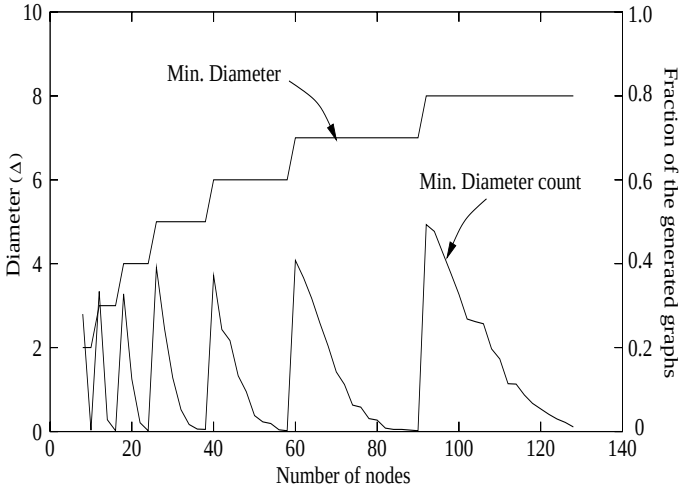


Figure 5. Frequency of minimum diameter random networks of degree 3

Figure 4 is very interesting because it shows that if we generate a sizeable number of random networks and then select the 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. It should be pointed out, however, that the size of the random graphs of a given degree and diameter tend to be greater than theoretical bounds, such as the Moore bound[3] or the bound obtained from theoretical studies of random graphs[5, 6].

Further comparisons can be carried out with the entries in the (d, Δ) table[2]. Table 1 shows some of the

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

[2] The (d, Δ) table gives the *state of the art* with respect to a largest known graphs with degree d and diameter Δ [10].

results. The diameter of the random graphs was at most larger by 1 than the corresponding best known diameters. It is worth pointing out that these known networks are constructed by different methods for different degrees and diameters whereas the random networks follow the same simple construction algorithm.

Size of Network	Degree	Diameter	
		Best Known	Random
10	3	2	2
15	4	2	3
20	3	3	4
70	3	5	6
364	4	5	6
532	5	5	6
740	4	6	7

Table 1. Comparison of diameter between best known networks and the best of the random networks generated in our experiments

4.2. The Scalability Filter

Some applications and situations require networks to be scalable. A network is said to have good scalability if the size of the network can be increased with minimum disruption and this does not cause a drastic change in its properties. For reasons of cost, it is better to have the option of small increments since this allows the network to be upgraded to the required size within a particular budget. The hypercube, for example, has poor scalability, in that its size cannot be increased by small increments while still maintaining its structural properties. Random graphs, on the other hand, have good scalability. They can be constructed for all sizes and degrees (as long as $n*d$ is even) and Figure 3 shows that the diameter remains constant for a considerable range of network sizes.

If regularity of the graph must be maintained even after scaling, some edges must be removed and some added to accomodate the new node. The minimum number of edges that must be removed to scale an even degree network by one node is $d/2$ whereas that required to scale an odd degree network by two nodes is $d-1$. Typically, one does not possess the flexibility of adding new nodes anywhere in the network. It may be required to attach the new node adjacent to a given set of nodes. This is typically the case in a fault-tolerant design when a spare processor must serve as a backup for a given set of processors. We define a measure for scalability in this context by the average increase in diameter caused by connecting a new node to all possible designated sets of d nodes. The network is said to have good scalability if its diameter does not increase considerably on average.

Not all randomly generated graphs scale well. To evaluate the performance of random graphs with regard to scalability, we generated 100 random graphs of size 64 and degree 4 and diameter 5. Note that 5 is the minimum diameter obtained for graphs of size 64 and degree 4 as shown in Figure 3. For each network, we then evaluated scalability by selecting sets of four nodes at random and adding a new node adjacent to the designated nodes. If the designated nodes are connected by an edge, then this edge is removed, otherwise edges incident on the designated nodes are selected at random and removed to create connections to the new nodes as shown in Figure 6.

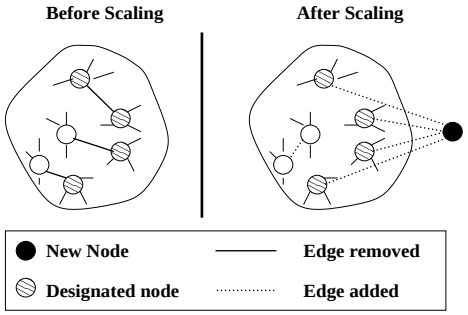


Figure 6. Example of edges that can be removed to accomodate a new node

The diameter is calculated for each set of four nodes and the increase in diameter is averaged over a large number of such runs. Figure 7 shows the cumulative distribution of the increase in diameter for the input graphs. It gives an idea of the threshold that can be set for a scalability filter, e.g., if the best 10% of the graphs are selected then we can expect that the average increase in diameter will be no more than 0.05.

4.3. The Fault-Tolerance Filter

Reliability is an important criterion in the selection of an interconnection network. Measures are required to adequately capture network qualities such as graceful degradation and robustness. Traditional measures, such as connectivity, are worst-case measures and have limited expressiveness. In this paper, we look at the following more expressive measures: the diameter stability, $\Delta(p_f)$, the average node-pair distance stability, $\overline{D}(p_f)$, the probability of disconnection, $\pi_d(p_f)$, and the size of the maximum connected component,

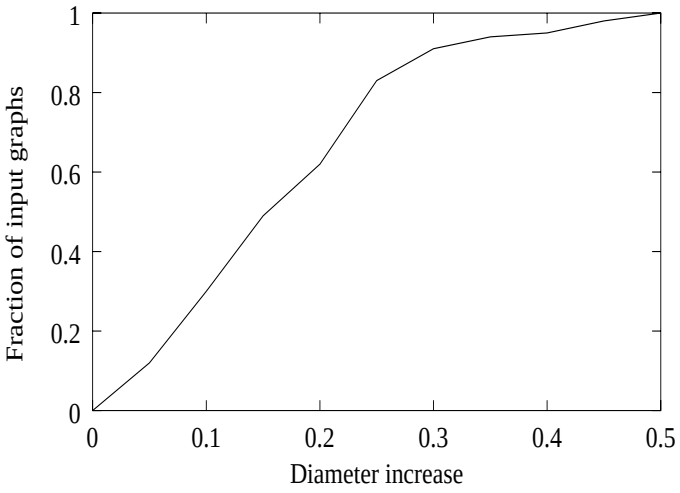
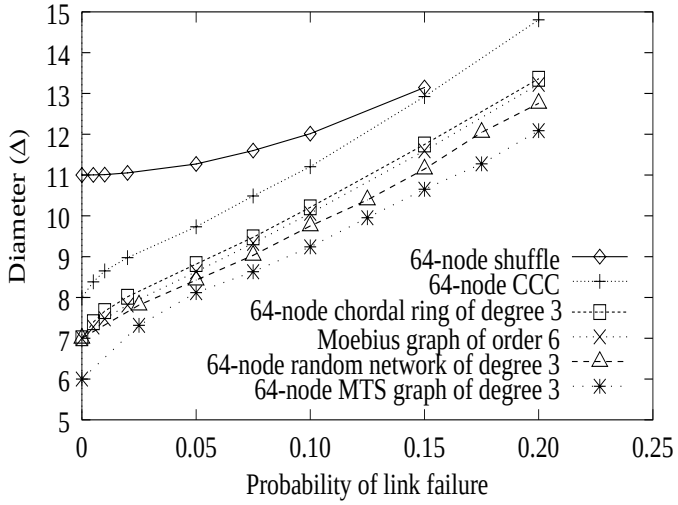
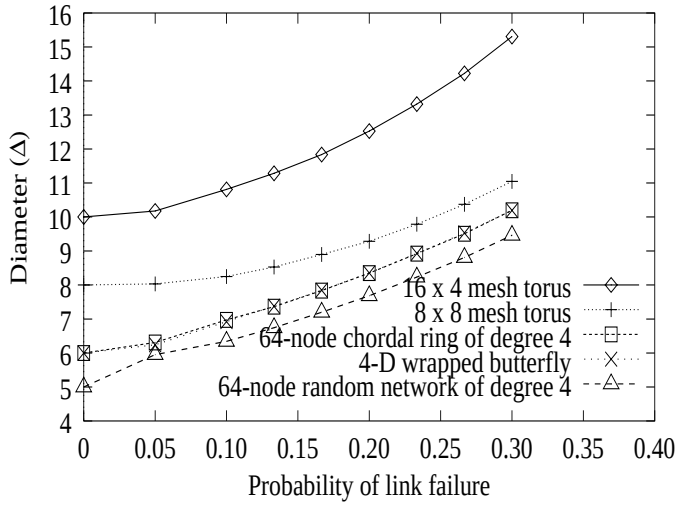


Figure 7. Cumulative frequency of the increase in diameter for random networks of size 64 and degree 3



(a) Comparison among degree 3 networks



(b) Comparison among degree 4 networks

Figure 8. Diameter vs. probability of link failure

$\chi_{max}(p_f)$, all in the presence of link failures occuring independently with probability p_f. These measures were introduced in [11] and some research has already been done in characterizing various networks with respect to these measures. We performed experiments to evaluate the vulnerability of regular random networks with respect to these four measures. We used random networks of size 64 and degree 3 and 4 and compared their performance with other networks of similar size and degree. The network used was chosen at random from the set of minimum diameter networks obtained at the output of the diameter filter.

Figure 8 shows the comparison of diameter stability among degree 3 and degree 4 networks. The random regular networks of degree 4 outperform all the networks in its category whereas in the degree 3 category, it is second-best. Though Figures 9 and 11 show average node-pair distance stability and probability of disconnection for degree 3 networks only, the performance of random networks in the degree 4 category was observed to be the same as in the case of diameter stability. 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 seen in the case of shuffle exchange networks and meshes.

The fault-tolerance filter that we use is a combination of four filters: one for each of the four measures. Thresholds are typically specified as a scalar value and since each of the fault tolerant measures is given by a vector of values (corresponding to different link(node) failure probabilities), some transformation must be used to convert the vector of values to a single value. If the designer knows the exact value of the link failure probability, then the value of the measure corresponding to that failure probability can be used to do the comparison. However, it may be difficult for the designer to decide on the exact value of the failure probability. One transformation that can be applied would be to use the area under the curve obtained when plotting the values. This transformation assumes that each failure probability in the range

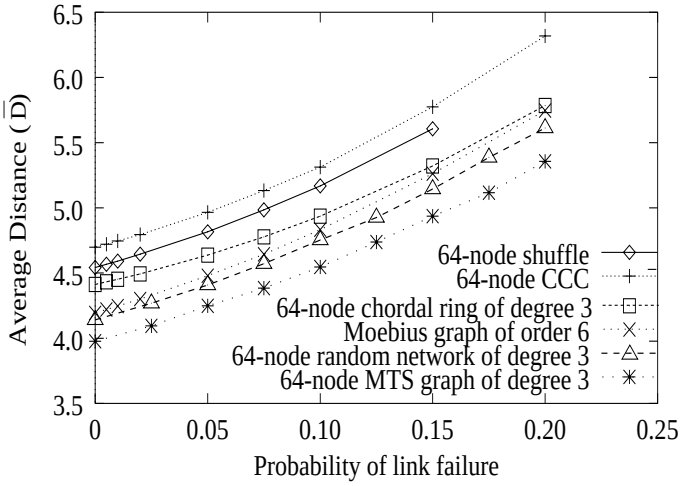


Figure 9. Average node-pair distance vs. probability of link failure

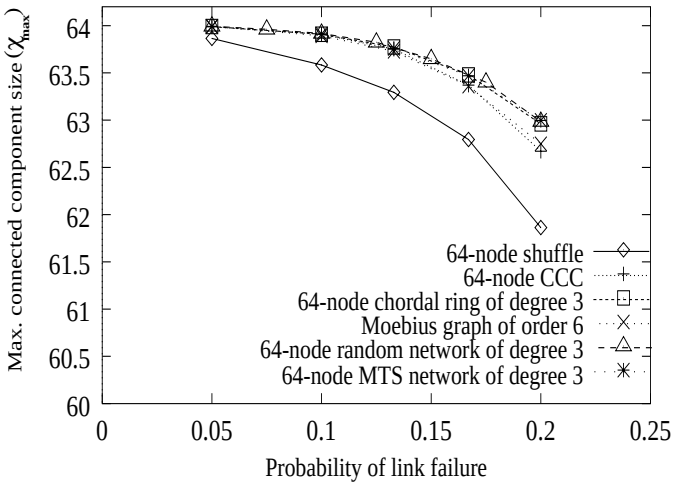


Figure 11. Maximum component size vs. probability of link failure

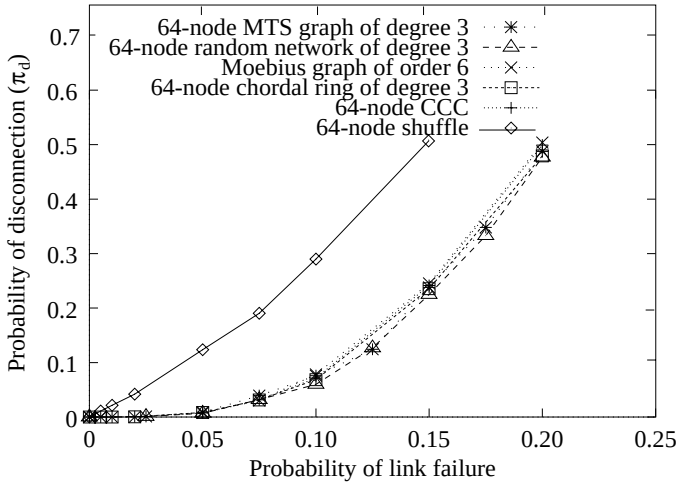


Figure 10. Probability of disconnection vs. probability of link failure

of interest is equally likely. Since this may not be the case in most situations, a more appropriate transformation would be to associate weights with each of the failure probabilities and use a weighted sum of the values corresponding to different failure probabilities.

The output at the end of the set of filters depends on the threshold values that have been set for each filter. A stringent threshold value passes a smaller number of networks through it. If the filter produces no output, the designer would have to refine the threshold values or increase the number of input graphs generated.

Experiments were performed to evaluate the efficiency of the approach by passing random graphs through a multitude of filters. Graphs of size 64 and degree 4 were generated and first passed through a diameter filter and then through a fault tolerance filter. The threshold of the diameter filter was set at 7. The networks obtained at the output of the diameter filter were tested for their fault tolerant characteristics by using two filters, the $\Delta(p_f)$ filter and the $\overline{D}(p_f)$ filter. The range of link failure probabilities of interest to us in this example was $[0.0, 0.2]$. Our initial set consisted of 1000 randomly-generated networks. After passing through the diameter filter, we were left with 33%(=330) of the networks. These networks were then evaluated for the two fault tolerance measures and then ranked according to their performance. The thresholds of the filters were set such that only those input networks that lie among the best 5% pass through it. The number of graphs obtained at the end of the filtering process was a respectable 1.5%(=15). It is important to note this "short-list" contains graphs that are better than most graphs published in the literature with regard to diameter and the two fault tolerance measures.

5. Conclusions and Future Work

Synthesizing networks that satisfy a certain set of performance or fault-tolerance requirements is difficult. In our approach, we generate a large number of random regular networks and filter out those that do not comply with the requirements. The choice of random regular networks was motivated by their ease and flexibility of construction and their surprisingly good prop-

erties. The filtering process consists of filters arranged in tandem, one for each requirement to be satisfied. Each filter removes networks that do not comply with the requirement associated with it. 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. We demonstrated the effectiveness of this approach by synthesizing fault-tolerant networks with a small diameter.

Extensions to the current work are ongoing in several directions. Other filters are currently being studied, among them are the filter for embeddability and routability. Other network-generation algorithms are being developed and assessed. A graphical tool to facilitate synthesis of interconnection networks through our approach is also on the anvil.

Acknowledgement

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

References

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

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

[3] E. Bannai and T. Ito. On finite moore graphs. *J. Fac. Sci., Tokyo Univ.*, pages 191–208, 1973.

[4] J. Bermond and B.Bollobas. The diameter of graphs - a survey. *Proc. Congressus Numerantium*, 32:3–27, 1981.

[5] B. Bollobas. Random graphs. *Combinatorics Lect. Note Series London Mathematic Soc.*, pages 80–102, 1980.

[6] B. Bollobas and W. L. Vega. The diameter of random regular graphs. *Combinatorica*, 2:125–134, Feb 1982.

[7] K. Doty. New designs for dense processor interconnection networks. *IEEE Trans. Computers*, C-33:447–450, May 1984.

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

[9] R. Finkel and M. Solomon. Processor interconnection strategies. *IEEE Trans. Computers*, 29:360–370, May 1980.

[10] http://maite71.upc.es/grup_de_grafs/table_g.html. d-k table.

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

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

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

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

[15] R. Read. The enumeration of locally restricted graphs. *J. London Math Soc.*, pages 417–436, 1959.

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