1-1-2010

Design Space Analysis and a Novel Routing Agorithm for Unstructured Networks-on-Chip

Neha Parashar
Portland State University

Design Space Analysis and a Novel Routing Algorithm for Unstructured Networks-on-Chip

by

Neha Parashar

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
Christof Teuscher, Chair
Douglas V. Hall
Alaa Alameldeen

Portland State University

ABSTRACT

Traditionally, on-chip network communication was achieved with shared medium networks where devices shared the transmission medium with only one device driving the network at a time. To avoid performance losses, it required a fast bus arbitration logic. However, a single shared bus has serious limitations with the heterogeneous and multi-core communication requirements of today's chip designs. Point-to-point or direct networks solved some of the scalability issues, but the use of routers and of rather complex algorithms to connect nodes during each cycle caused new bottlenecks. As technology scales, the on-chip physical interconnect presents an increasingly limiting factor for performance and energy consumption. Network-on-chip, an emerging interconnect paradigm, provide solutions to these interconnect and communication challenges. Motivated by future bottom-up self-assembled fabrication techniques, which are believed to produce largely unstructured interconnect fabrics in a very inexpensive way, the goal of this thesis is to explore the design trade-offs of such irregular, heterogeneous, and unreliable networks. The important measures we care about for our complex on-chip network models are the information transfer, congestion avoidance, throughput, and latency. We use two control parameters and a network model inspired by Watts and Strogatz's small-world network model to generate a large class of different networks. We then evaluate their cost and performance and introduce a function which allows us to systematically explore the trade-offs between cost and performance depending on the designer's requirement. We further evaluate these networks under different traffic conditions and introduce an adaptive and topology-agnostic ant routing algorithm that does not require any global control and avoids network congestion.

Acknowledgements

I am indebted to Dr. Christof Teuscher for his advice and suggestions throughout the course of this research. There were numerous meetings and discussions that led to successful completion of this thesis. I am also grateful to the committee members, Dr. Douglas V. Hall and Dr. Alaa Alameldeen for reviewing this document and suggesting key changes.

Most importantly, I am thankful to my family, Teuscher-lab members and all my friends for their continued moral support and encouragement.

Contents

Abstract i

Acknowledgements ii

List of Tables vi

List of Figures vii

1 Introduction and Related Work 1

1.1 Trends and Challenges 1

1.2 Evolution of Network-on-chip 2

1.3 Related Work 4

1.4 Problem Description 6

1.5 Organization of the Thesis 6

2 Basic Definitions and Framework 8

2.1 Abstract Networks 8

2.2 Network-on-chip Components 9

2.2.1 Processing node 10

2.2.2 Switch node 10

2.2.3 Type of switching fabric 10

2.2.4 Interconnects . . . 12
2.2.5 Virtual channels . . . 12
2.3 Network Properties . . . 12
2.3.1 Connectivity, k . . . 13
2.3.2 Network topology . . . 13
2.3.3 Rewiring probability, p . . . 17
2.3.4 Power-law distribution . . . 20

3 Design Space Analysis **22**
3.1 Systematic Approach to Obtain a Network . . . 22
3.2 Comparison of Networks in the Four Corners of α and p Space . . . 26
3.2.1 Neighbor distribution . . . 27

4 Traffic and Routing **29**
4.1 Traffic . . . 29
4.1.1 Random traffic . . . 30
4.1.2 Hot-spot traffic . . . 30
4.2 Routing . . . 31
4.2.1 Basic routing taxonomy . . . 32
4.2.2 Ant routing . . . 34

5 Performance Metrics **39**
5.1 Measure of Network Performance . . . 39
5.1.1 Wire cost . . . 39
5.1.2 Average shortest path length and average number of hops . 39
5.1.3 Energy and power . . . 40

5.1.4 Throughput 41
5.1.5 Latency 42
5.2 Performance Evaluation 42
5.2.1 Scalability 42
5.2.2 Local and global connections 48
5.2.3 Regular and random graphs 53
5.3 Performance Evaluation with Traffic 57
5.3.1 Ant routing 62

6 Small-World Power-Law Networks 64
6.1 Experiment 1: Wire length and average number of hops 65
6.1.1 Summary 68
6.2 Experiment 2: Total power and average number of hops 69
6.2.1 Summary 72
6.3 Experiment 3: Total power, total wire length and average number of hops 73
6.3.1 Summary 76
6.4 Optimal Network with Hot-Spot Traffic 77
6.5 Optimal Network with Different Routing Algorithms 78

7 Conclusion and Future Work 79
7.1 Conclusion 79
7.2 Future Work 80

References 81

List of Tables

3.1 Neighbors list before rewiring for figure 3.1. 24

3.2 Neighbors list after rewiring for figure 3.2. 26

6.1 The table gives examples of different optimal parameters for different values of a. For $a = 0$, the optimal network is a mesh whereas for $a = 0.5$, the optimal network is a small-world network. 68

6.2 The table gives examples of different optimal parameters for different values of a. For $a = 0$, the optimal network is a mesh, whereas for $a = 0.5$, the optimal network is a random network. 72

6.3 The table gives examples of different optimal parameters for $a = 0.5$ and $b = 0$. The optimal network obtained is the 5×5 2D mesh network with $R = 90$, $\alpha = 0.8$ and $p = 0.2$. 76

6.4 Comparison of a mesh and the optimal network. 77

List of Figures

1.1 Delay versus feature size [3]. 1

1.2 Different communication structures for SoC applications [6]. 2

1.3 4×4 2D mesh network. 3

1.4 3×3 Mesh network-on-chip(NoC) architecture [21]. 4

2.1 A directed network with a set of nodes and links. 8

2.2 5×5 2D mesh network with 25 SNs and PNs. 14

2.3 $3 \times 3 \times 3$ 3D mesh network with 27 SNs and PNs. 15

2.4 1D ring with 27 PNs and SNs. 16

2.5 3D random network where PNs and SNs are randomly placed. . . . 17

2.6 Small-world network obtained by rewiring the links. a) A ring network with no rewiring, i.e., $p = 0$. b) A small-world network obtained with $p = 0.1$. c) A ring network at $p = 0.9$. d) A random network obtained with $p = 1$. 19

2.7 Global and local interconnected networks. 21

3.1 3×3 2D mesh network with 9 SNs and PNs with $R = 0$, $\alpha = 0$, and $p = 0$. 24

3.2 3×3 2D mesh network with 9 SNs and PNs with $R = 0$, $\alpha = 0$, and $p = 1$. 25

3.3 Comparison of networks in $\alpha - p$ regime of a 5×5 2D mesh network with 25 SNs and PNs($R = 0$). At $\alpha = 0$ and $p = 0$ we get a mesh. At $\alpha = 0$ and $p = 1$ a random network with global connections is obtained. At $\alpha = 2$ and $p = 0$ we get a mesh again and at $\alpha = 2$ and $p = 1$ we get a random network with local connections. 26

3.4 Comparison of networks in $\alpha - p$ regime of a 8×8 2D mesh network with 64 SNs and PNs ($R = 0$). At $p = 0$, we get mesh network with no rewiring. As $p \to 1$, more and more links are rewired. At $p = 1$, we get a more distributed network. 27

3.5 Comparison of networks in $\alpha - p$ regime of a 8×8 2D mesh network with 64 SNs and PNs ($R = 100$). At $p = 0$, where no links are rewired, the network is a mesh network. As $p \to 1$, more and more links are rewired and hence at $p = 1$ we get a very distributed network. 28

5.1 Wirelength as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs. 44

5.2 Average number of hops as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs. 46

5.3 Total power as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs. 47

5.4 Wire length as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs. 49

5.5 Average number of hops as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs. 51

5.6 Total power as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs. 52

5.7 Wire length as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs. 54

5.8 Average hops as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs. 55

5.9 Total power as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs. 57

5.10 Throughput and average latency of networks with varying α and p values. Figures 5.10a and 5.10b uses 8×8 2D mesh networks with different α, p and R values. The data is averaged over 10 runs. . . . 59

5.11 Throughput and average latency of networks with varying hot-spot probability, h. Figures 5.11a and 5.11b use a 5×5 2D mesh network with $p = 0$, $R = 0$, $h = 0.9$ and varying *hotspotperc*. The data is averaged over 10 runs. 61

5.12 Throughput and average latency of the mesh network with different routing algorithms. The data is averaged over 10 runs. 63

6.1 Average number of hops and total wire length as a function of α and p for a network with $R = 10$. 65

6.2 Aggregate objective function y. 5×5 2D mesh network with $R = 10$ and $a = 0.2$. 66

6.3 Minimum y and R scaling plots for average number of hops and total wire length 67

6.4 Average number of hops and the total power as a function of α and p for a network with $R = 40$. 69

6.5 Aggregate objective function, y. 5×5 2D mesh network with $R = 40$ and $a = 0.2$. 70

6.6 Minimum y and R scaling plots for average number of hops and total power 71

6.7 Average number of hops, total wire length and total power as a function of α and p for a network with $R = 90$. 74

6.8 Aggregate objective function y. 5×5 2D mesh network with $R = 90$ and $a = 0.5$ and $b = 0$. 75

6.9 Minimum y value as a function of a and b. 75

6.10 Throughput and average latency of mesh and the optimal networks under hot-spot traffic. 77

6.11 Throughput and average latency of the optimal network with different routing algorithms.The network used is 5×5 2D mesh with $\alpha = 0.8$, $p = 0.2$, $R = 90$. The data is averaged over 10 runs. 78

Chapter 1

Introduction and Related Work

1.1 Trends and Challenges

It is well known that technology scaling works better for transistors than interconnects [15]. Figure 1.1 shows that as the transistor size reduces, the gate delay decreases, but the wire delay increases. Wire scaling leads to issues like clock skews, clock synchronization, increased power consumption and cost due to clock distribution trees.

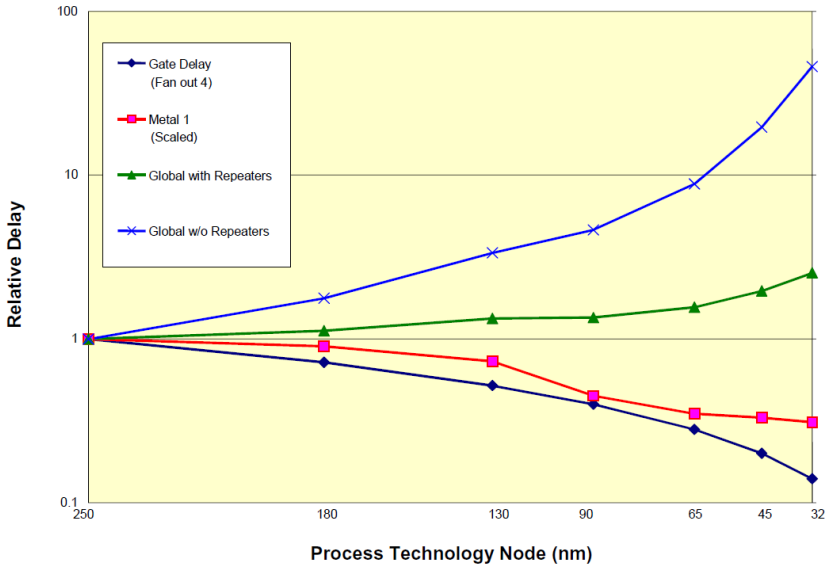


Figure 1.1: Delay versus feature size [3].

Fields like, mobile communication, consumer electronics and multimedia have shown increasing design issues, which were partly addressed by System-on-chip (SoC) designs. By placing different computing resources on a single chip, the interconnection between the components becomes a more challenging issue. Most SoC application use a shared-bus interconnect, where a communication link is shared among multiple resources. This requires an arbitration logic, which has simple control logic and costs less, but is less scalable. In a shared-bus architecture, an additional resource requires modification of the bus architecture and thus results in performance degradation since the same bus is shared among more resources. Figure 1.2 shows different communication structures in System-on-chip designs [6]. The growing computation-intensive applications and a need for low-power, high-performance systems led to an increase in the number of on-chip computing resources. This is because the current VLSI technology and scaling can support a very high integration of transistors on chip. Hence, the demanding computational power of these industries can not be fulfilled by traditional process architectures and shared bus-type interconnects. As the overall system performance not only depends on scaling technology, we need innovative computation and communication architectures. There are classical techniques to increase the computational performance of the system, like instruction-level parallelism, thread-level parallelism and data-level parallelism, but poor on-chip communication restrict the overall system performance significantly. Thus, there is a need for a technology which could improve the communication performance significantly while lowering the cost.

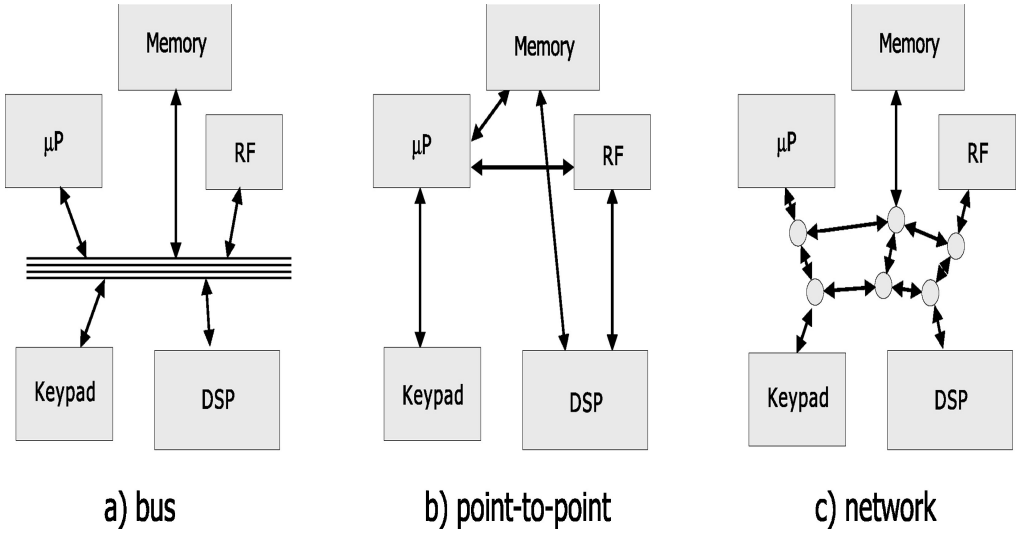


Figure 1.2: Different communication structures for SoC applications [6].

1.2 Evolution of Network-on-chip

As seen above, the shared-bus architectures constitute both a power and a performance bottleneck. Transistor scaling, an increasing clock frequency to several GHz and a need for reliable on-chip communication led the designers to discover a new paradigm called Network-on-chips (NoC) [5]. In simple terms, Network-on-chip is a mere application of network theories on chip. It is an emerging design standard for communication within large VLSI systems implemented on a single silicon chip. Unlike shared-bus architectures, in NoC, an addition of a resource means more communication capacity because of an increase in the number of switches and

links. The data packets are transfered simultaneously on these links in a NoC, thereby making the communication highly parallel. Network-on-chip provide solutions to the challenges faced by shared-bus architectures. They offer re-usable and predictable properties [19], which make them more favorable than shared-bus, point-to-point, segmented buses with bridges, or any other on-chip communication techniques. They have the potential to reuse application parts and components. The increase in popularity of NoCs has led researchers to study networks with a large number of nodes. Overall, NoCs provide enhanced performance for complex integrated systems as compared to traditional communication techniques. Every NoC relies on a specific network topology. Figure 1.3 shows an example of a regular 4*times*4 2D mesh network where the switches are connected in a mesh format.

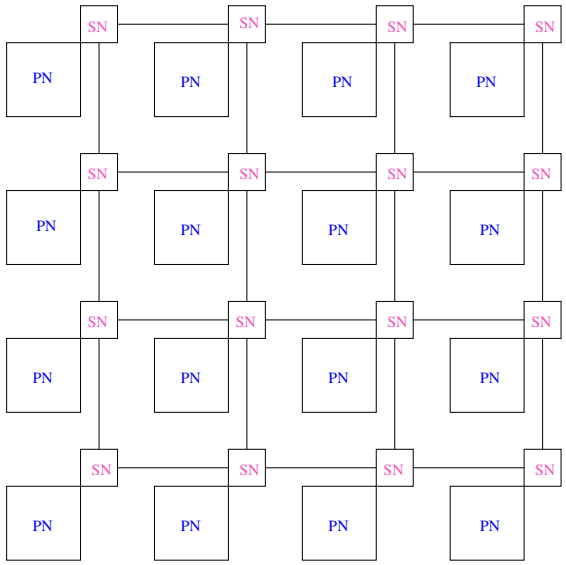


Figure 1.3: 4×4 2D mesh network.

As shown in the figure 1.3, the three main building blocks of a typical NoC are the processing element (PE), switches and the links. The processing elements can

be specialized Intellectual Property (IP) blocks , memory unit, control unit, cache, processor cores, etc. These processing elements send and receive data packets from other PEs via the links. The switches route these data packets with the help of a built-in router. Figure 1.4 shows a 3×3 mesh NoC architecture with the processing elements, switches and links.

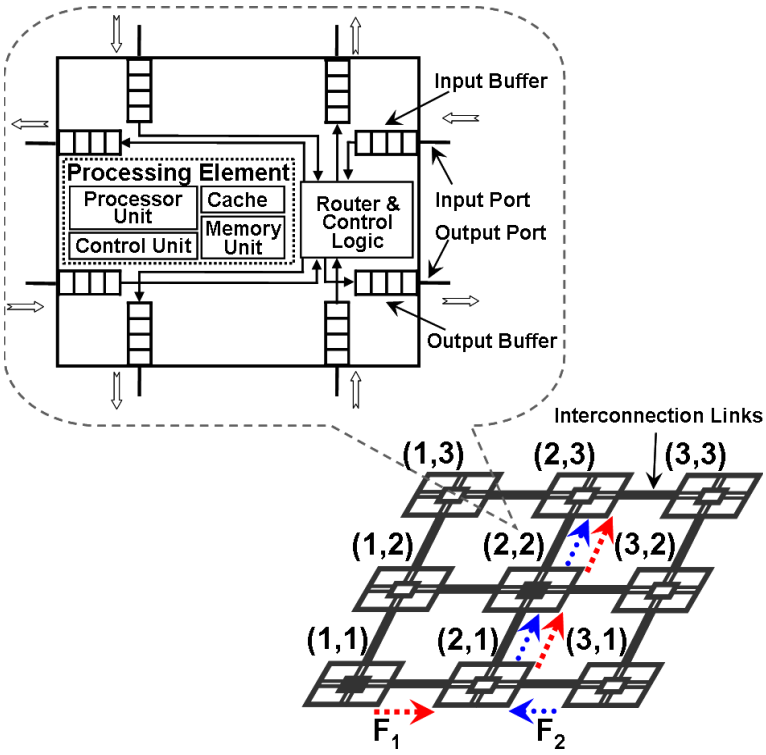


Figure 1.4: 3×3 Mesh network-on-chip(NoC) architecture [21].

1.3 Related Work

Current NoC designs use a regular mesh-like topology which exhibit good performance. But, employing a mesh architecture with a large number of complex on-chip heterogeneous components can increase the delay introduced by multiple hops. In the past, researchers have studied the properties of nature-inspired interconnects to improve the network's performance. For example, the small-world property by Watts and Strogatz [40] is observed in most real networks like neural networks [14], brain functional networks [12], [36], Internet [11], etc. Watts and Strogatz assume uniform rewiring probability over all the nodes, independent of the distance between nodes. Networks based on this concept introduce 'short-cuts' between random distant nodes [40]. But, they do not consider the spatial aspect of the nodes. Ogras et al. [25] did not insert the short-cuts at random locations, but where they were most useful for increasing the critical traffic workload. Their work showed a significant reduction in the average packet latency by adding a few long-range links to a standard mesh network.

Scale-free networks form another class of networks, which have a degree distribution that follows a power-law curve. The robustness of the scale-free networks have led some researchers to scrutinize the application of this theory at the architectural level. Oshida and Ihara [26] have investigated the packet traffic of scale-free and large-scale networks-on-chip designs. The robustness and scalable properties of these networks make them highly reliable for communication. Work done by Oshida and Ihara [26] shows that scale-free topologies achieve short latencies and low packet loss ratios. However, they have not considered the wiring cost.

Our model is based on a physically realistic small-world network approach as explored by Petermann and De Los Rios [30]. Our approach is more practical and addresses the question of how much and what type of interconnect is needed by emerging electronics. We apply the small-world and power-law properties over regular mesh networks and show that the distance-dependent rewiring can improve the performance with little cost overhead.

Network performance is judged not only by the average shortest-path, but also by the latency and throughput under different traffic conditions. Thus, to compare and contrast different networks, our performance measuring techniques are inspired by the work done by Pande, et al. [28]. A routing algorithm's task is to route the data efficiently and quickly. With fast asynchronous communication on-chip, we need a routing algorithm that takes the shortest path when needed and adapts the routing strategy in case of node or link failures. A good routing algorithm should thus balance out the traffic load to achieve ideal throughput. One of the routing techniques in the literature of network-on-chip designs is the application-specific routing algorithm by Palesi, et al. [27]. DyAd-smart routing by Hu and Marculescu [16] uses a combination of both deterministic and adaptive routing. At low traffic conditions it uses deterministic routing and at high traffic conditions, it dynamically switches to an adaptive routing scheme. The routing schemes used in the past are mostly centralized, where a main controller decides the routing scheme. Inspired by Di Caro and Dorigo [7], we implemented a naive ant routing algorithm which is distributed and can adapt to different traffic conditions.

1.4 Problem Description

As stated above, a fixed topology of the network with reusable components can prove to perform better than other communication techniques. Designs with homogeneous processing cores give predicted layouts with regular mesh-like topologies, but for Multi-Processor System-on-Chip (MPSoC) with heterogeneous cores, these regular mesh structures give poor performance with large power and area overhead [4]. The goal of our thesis is to analyze the design trade-offs of different irregular network topologies. We explore this design space of irregular networks with heterogeneous components because future bottom-up self-assembled fabrics are believed to be highly unstructured as opposed to conventional regular architectures. We demonstrate that a class of small-world networks give better performance than a regular mesh structure. We analyze network properties like wire length, average number of hops, latency and throughput under random and hot-spot traffic. We also implement a novel routing algorithm which is adaptive and compare it to other routing techniques, like random and shortest path routing.

1.5 Organization of the Thesis

At the outset, the basic definition of the network and its components are detailed in chapter 2, which highlights the network parameters and the types of networks used for our experiments. The design space is explored in chapter 3, where a pictorial view of networks with different network parameters is given. The traffic models and the implementation of a novel routing algorithm are detailed in chapter 4. This is followed by a discussion of performance evaluations and different experiments in chapter 5 and 6 respectively. Lastly, chapter 7 concludes the work and outlines

further research.

My contributions: All experiments shown in the thesis have been performed on a novice nano-toolbox simulator at the Teuscher-lab. The initial work in the thesis involved setting up the simulator to carry out the experiments correctly. I wrote functions to check for graph connectivity, implemented the power-law theory over the small-world network and introduced the parameter R, which is discussed later. We tested different wire-growth models on the nano-toolbox simulator, which were first presented by Teuscher et al. [39]. The networks were tested under random traffic implemented in the simulator. I implemented a hot-spot traffic pattern and tested the networks under the same. There are different routing techniques given in the simulator like random, shortest-path and ant routing. I further extended the ant routing in the simulator to avoid congestion.

Chapter 2

Basic Definitions and Framework

2.1 Abstract Networks

A functionally correct system with reliable communication between every component on chip is a requirement of any network application. A network, in general, can be defined as a set of nodes or vertices interconnected with links. Figure 2.1 shows a directed network structure with nodes interconnected by links.

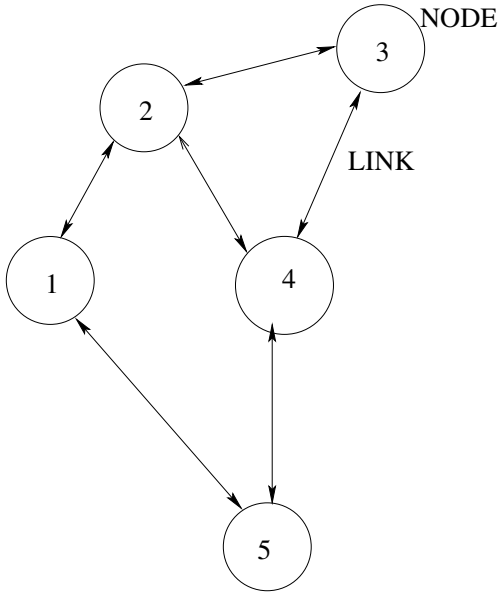


Figure 2.1: A directed network with a set of nodes and links.

Definition 1 *A node in the network can be a terminal node acting as a source*

where a packet is generated or a destination where a packet is received. It can also be a switch node where the packet is just forwarded from an input port to an output port.

Definition 2 *A link is an element interconnecting two nodes or vertices.*

These links can be unidirectional or bidirectional. In a unidirectional link, the source node is the origin of the link and the destination node is the end of the link. Whereas in a bidirectional link, either of the two nodes connected by the link can be the source or destination. For example, in figure 2.1 all links are bi-directional.

Definition 3 *The Adjacency matrix is a means of representing which nodes are connected, i.e., adjacent to another node in the network.*

A network with n nodes can be represented by an adjacency matrix of size $n \times n$, where *Adj(i,j)* indicates whether there is a connection between the i^{th} and j^{th} node. A $'1'$ indicates a connection while, a $'0'$ indicates no connection. *AdjMat* gives the adjacency matrix for the graph described in figure 2.1.

$$AdjMat = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

2.2 Network-on-chip Components

A typical network-on-chip architecture consists of the following basic components

1. Processing nodes.

2. A switching fabric consisting of

   (a) Switch nodes.

   (b) Point-to-point interconnects connecting these nodes.

2.2.1 Processing node

Processing nodes (PN) are the source and destination of packet traffic. As seen in figure 1.4, a processing node can consist of a RAM, cache, a processing unit, control unit, etc. These nodes are mere source and sink for the data packets, but it is here that the actual processing of the data takes place. For example, if the processing unit at the node (2,3) in figure 1.4 wants to access the data stored in the cache of node (2,2), the source of the data is node (2,2) and the destination of the data is node (2,3). Hence, according to the routing and control logic, the data is transfered from the cache of node (2,2) to the processing unit of node (2,3).

2.2.2 Switch node

Each switch node (SN) can store a certain number of messages or data packets and is used to forward these messages in parallel to its neighbors on the available virtual channels. Virtual channels are discussed in section 2.2.5. These switch nodes, which are also known as routers, should be small, fast and energy efficient.

2.2.3 Type of switching fabric

A switching technique determines how the switches connect the input ports to the output ports. There are three principal network-on-chip switching techniques [28]:

circuit switching, packet switching and worm-hole routing.

1. Circuit switching: For this type of switching technique, a connection between the node is established and then the communication process takes place. Thus, a particular link can be used only for the communication between the connected nodes. Though this technique makes optimal use of available bandwidth, the link remains unavailable to other nodes, even if there is no transfer of data taking place [28].

2. Packet switching: Packet switching overcomes the drawbacks of circuit switching by dividing the data in suitable sized packets and letting them choose any link to reach the destination nodes. Here, unlike in circuit switching, the bit-rate of the data stream is variable and depends on the congestion in the network. This switching technique requires a large buffer area at the switch nodes to store and forward the data [28].

3. Wormhole routing: Wormhole switching overcomes the drawbacks of packet switching by dividing the data packets in flits (flow control digits) thereby reducing the buffer size at the switch nodes. Here, the header flit contains all the routing information. The remaining flits follow the path established by the header flit [28].

Similarly, there have been other switching techniques mentioned in the literature. Cosmic cube, iPSC-1, Ametek 14 are a few of the examples of earlier super-computers, which used store-and-forward [24] as a switching technique. In this technique, a data packet is stored at every node before it is forwarded to the next node. Though simple, this technique consumes a lot of memory space and

has a higher latency. To overcome the drawbacks of store-and-forward, virtual-cut through was introduced [24], where a data packet is stored in a node only if the next node's buffer is full. Our simulator uses a packet switching technique, where the data packets travel through the network to reach their destination nodes.

2.2.4 Interconnects

As discussed earlier, all the nodes in the network are connected by the interconnect links that can be either unidirectional or bi-directional. Throughout our experiments, the communication between the nodes is carried out by the bi-directional interconnects on-chip. Each processing node is connected to nearest switch node with only one connection and there can be multiple connections between switch nodes. This will be explained further in section

2.2.5 Virtual channels

Virtual channels are the logical interconnects between the nodes, which share the same physical interconnect. Though they require extra buffer implementation, they are known to reduce latency, avoid deadlocks and improve performance [6]

2.3 Network Properties

In this section we will describe relevant network properties that can be varied and affect the network performance.

2.3.1 Connectivity, k

The connectivity k of any node is defined by the number of incoming links at that node. For example, in figure 2.1, node 4 has a connectivity of 3 ($k = 3$), one each from nodes 2, 3 and 5. In our network-on-chip framework, the average connectivity of a processing node is defined by PN_k and the connectivity of switch nodes is defined by SN_k. The connectivity of the nodes plays an important role in performance and cost evaluations. For simplicity, in our experiments, we have considered that all the processing nodes are connected to the nearest switch node with only one connection and hence, the $PN_k = 1$. Whereas, the switch nodes are connected with an average connectivity defined by SN_kAvg. There are no self-connections allowed, but there can be multiple connections between any pair of nodes.

2.3.2 Network topology

Different forms of interconnecting nodes in a network is referred to as the network topology. The network nodes can be arranged in 2D or 3D space. A few of the representative topologies have been investigated in detail by Teuscher in [37] and are described below:

- 2D Mesh Unfolded: This topology is also known as 2D Cellular Automata (2DCA). All the PNs and SNs are locally interconnected and placed in a 2D arrangement (Von Neumann neighborhood).

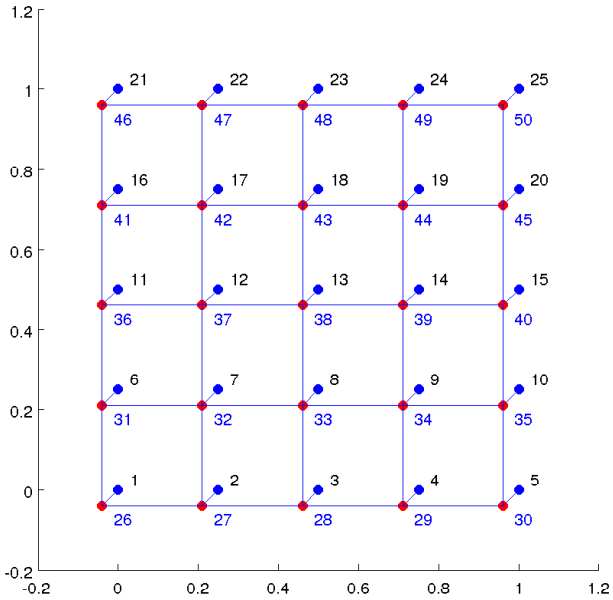


Figure 2.2: 5×5 2D mesh network with 25 SNs and PNs.

Figure 2.2 shows an example of a 5×5 2D mesh network with 25 switch nodes and 25 processing nodes.

- 3D Mesh Unfolded: This topology is also known as 3D Cellular Automata (3DCA). All PNs and SNs are locally interconnected and placed in a 3D arrangement, in a unit cube.

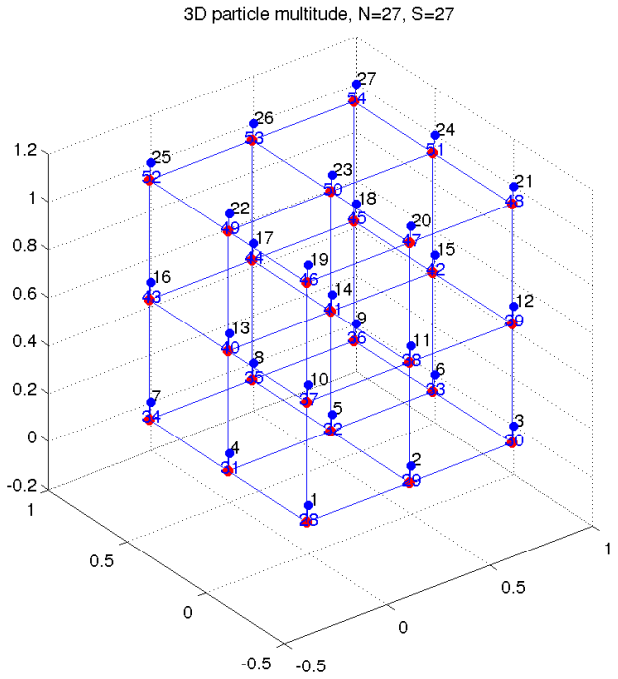


Figure 2.3: $3 \times 3 \times 3$ 3D mesh network with 27 SNs and PNs.

Figure 2.3 shows an example of a $3 \times 3 \times 3$ 3DCA network with 27 switch nodes and processing nodes. Since the nodes are arranged in a three dimensional space, each switch node can have up to six neighbors.

- 1D Ring: All the SNs and PNs are locally interconnected and are placed in a 1D ring arrangement where each SN has only two neighbors (left and right).

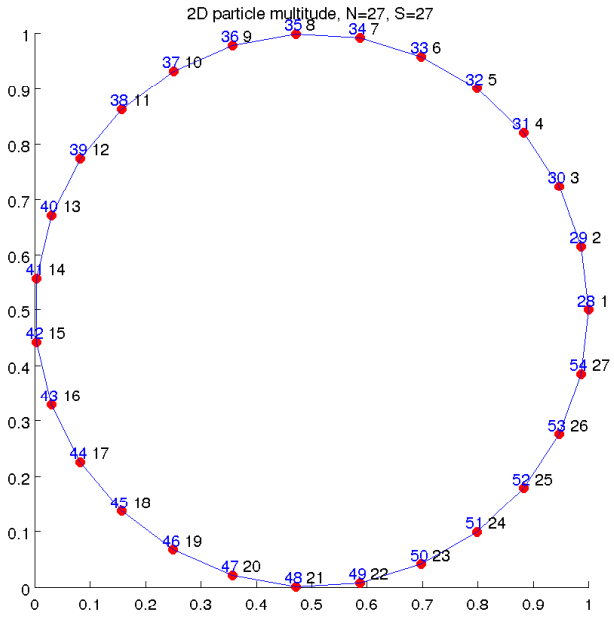


Figure 2.4: 1D ring with 27 PNs and SNs.

Figure 2.4 shows an example of a ring structure with 27 switch nodes and 27 processing nodes. The ring network is described in a unit diameter.

- 3D Random topologies (3DRM): In this topology, SNs and PNs are randomly placed in 3D space and are interconnected with random connections. Figure 2.5 shows an example of a 3DRM network with 27 switch nodes processing nodes. Each PN is connected to one nearest SN ($PNk = 1$) and each SN is connected to 4 other SNs on average($SN_kAvg = 4$). This figure follows a power law distribution where $\alpha = 0$, i.e., there are more long distance connections.

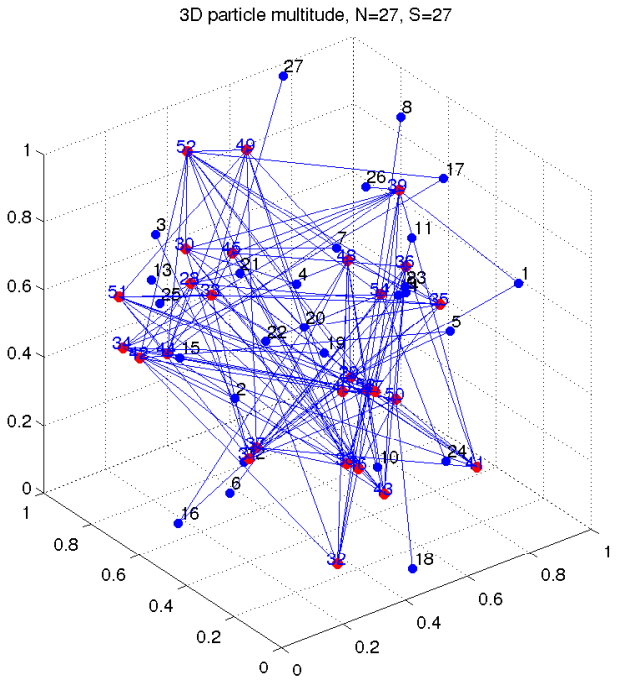


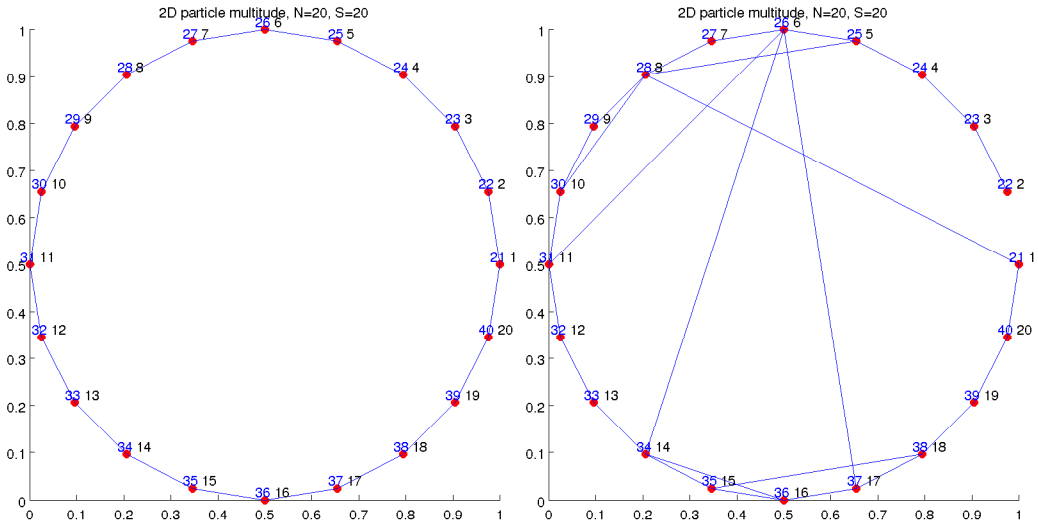
Figure 2.5: 3D random network where PNs and SNs are randomly placed.

For comparison, the 2D and 3D structures are defined in space of 1×1 unit square and $1 \times 1 \times 1$ unit cube respectively. We assume that each switch node is connected to its nearest processing node by only one connection, whereas switch nodes are connected to each other with average number of connections, SN_kAvg. Different values of SN_kAvg have been used to investigate the performance metrics described in chapter 5.

2.3.3 Rewiring probability, p

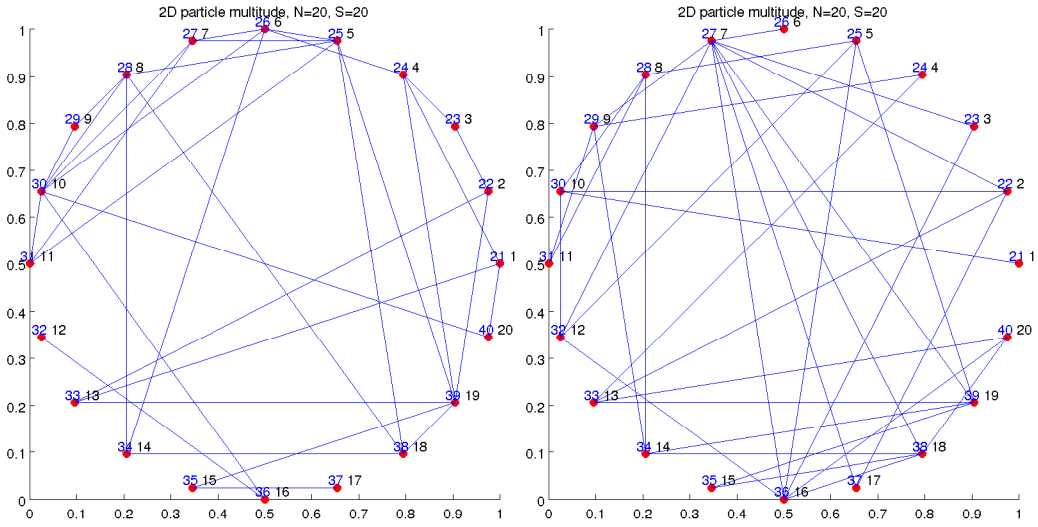
An interesting class of networks introduced by Duncan J. Watts and Steven Strogatz in 1998 is called *small-world network* [40]. The small-world property is observed in most real networks like brain networks [36], electronic circuits [17], the

Internet [11]. The uniqueness of these networks is that they appear highly clustered as of regular lattice and yet have small characteristic average path length like a random graph. As explained by Watts and Strogatz in [40], to obtain a small-world network from an already existing regular lattice or a ring network, each wire is removed and rewired to a random node with a probability, p. Thus, for $p = 0$ we obtain a regular network. On the other hand, for $p = 1$, we get a completely random network. With increasing p, a number of long range shortcuts are added to the network. These shortcuts allow faster transfer of data from source to destination. This reduces the average path length at the global level significantly and does not affect the clustering coefficient at the local level. Adding only a few shortcut links can improve the system performance. However, the network cost becomes an important factor to consider.



(a) $p = 0$

(b) $p = 0.1$



(c) $p = 0.9$

(d) $p = 1$

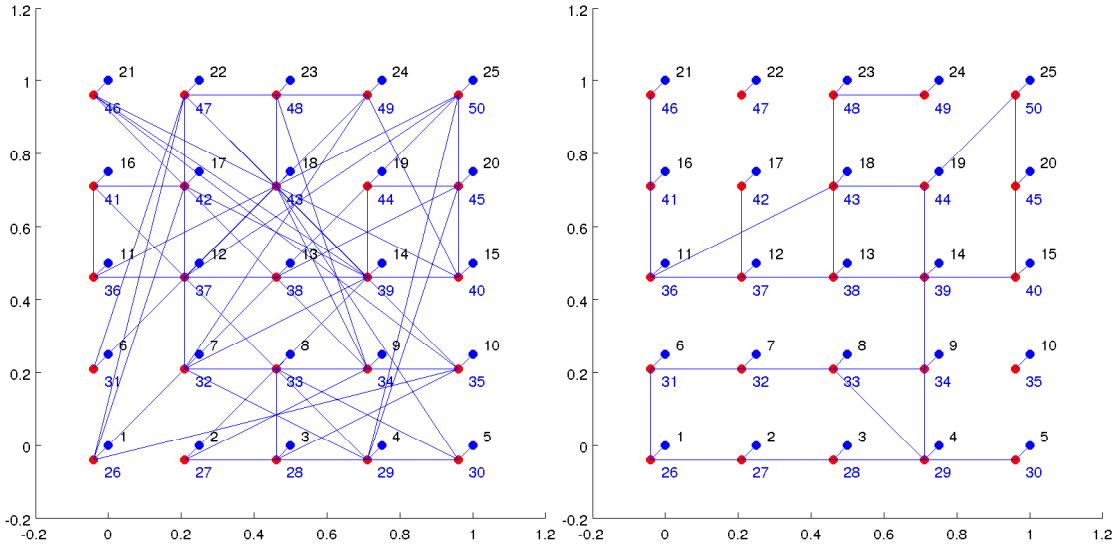
Figure 2.6: Small-world network obtained by rewiring the links. a) A ring network with no rewiring, i.e., $p = 0$. b) A small-world network obtained with $p = 0.1$. c) A ring network at $p = 0.9$. d) A random network obtained with $p = 1$.

Figure 2.6 demonstrates how a regular network is converted to a random network by rewiring the links with probability, p. The network consists of 20 SNs

and PNs arranged in a 2D planar surface. According to the Watts and Strogatz's model, for a network with N nodes, the mean connectivity k should satisfy the following condition: $N \gg k \gg \ln(N) \gg 1$. Analogous to the model, we add a fixed number of links to the regularly connected network. For example, in the ring network in figure 2.6, where each node is connected to its neighbors by a single link, we add 20 additional wires at random locations to the network. We call this parameter R. At $p = 0$, no link is rewired and hence each switch node is connected to only its left and right neighbors. At $p = 0.1$, 10% of the links are rewired. The rewired location depends on the power-law exponent α. Similarly, at $p = 1$, all the links are rewired. The links can be rewired to their previous location too.

2.3.4 Power-law distribution

As opposed to the scale-free networks, where the node degree distribution follows a power-law, we are interested in network with different wire length distributions. A small-world power-law network can be obtained if the rewiring is done proportional to the power law, $l^{-\alpha}$ where l is the Euclidean distance between the nodes and exponent α affects the communication characteristic of the network. Hence, an increase in α creates a locally interconnected network. Figures 2.7a and 2.7b show the variations in networks with $\alpha = 0$ and $\alpha = 2$, respectively. As seen, a lower α results in more long distance connections as compared to a higher α.



(a) $\alpha = 0$. (b) $\alpha = 2$.

Figure 2.7: Global and local interconnected networks.

According to [30], irregularly assembled small-world networks with a power-law decaying distribution of shortcut lengths show major advantages in terms of performance and robustness. For our experiments, apart from regular mesh networks, we have also considered small-world power-law networks since they exhibit the small-world phenomenon and have low wiring cost.

Chapter 3

Design Space Analysis

3.1 Systematic Approach to Obtain a Network

As explained in chapter 2, the three main building blocks of any network-on-chip architecture are the processing nodes, switch nodes and bi-directional links connecting these nodes. In this chapter we will explain how the networks are formed in our simulator. To evaluate any network and compare different performance metrics, the first step is to build a network. We create all the networks by placing the nodes and connecting them via bi-directional links. Certain assumptions are considered before creating these networks. As discussed in chapter 2, all the networks are defined in a unit space for simple comparison.

The following two examples describe the procedure to obtain a network structure before it can be used for performance evaluation.

Example 1: 3×3 2D mesh with 9 PNs and SNs and no additional wires, i.e., $R = 0$, $\alpha = 0$ and $p = 0$.

Referring to algorithm 1, first all the PNs are placed in the two dimensional unit square. This is followed by the placement of the SNs. Each PN is then connected to the nearest SN with a bi-directional link (PN_k). The SNs are connected to its neighbors in a mesh format with the bi-directional links (SN_k).

Algorithm 1 Algorithm to obtain a 2D mesh network.

1: Position the N processing nodes in a 1×1 unit square , where each side contains $\sqrt[2]{N}$ nodes.
2: Position each S switch nodes adjacent to the N processing nodes.
3: **for all** processing nodes N **do**
4: Connect each N to its closest S
5: **end for**
6: **for all** switch nodes S **do**
7: Connect each S to its neighbor in a mesh arrangement
8: **end for**
9: **for all** extra links R **do**
10: Pick a random S and place the link $r \in R$ over an already existing link chosen at random
11: **end for**
12: **for all** bi directional links **do**
13: Randomly pick a link in the network
14: Randomly generate a number *rand* within 0 and 1
15: **if** $rand \leq p$ **then**
16: Select that link to be rewired
17: Rewire that link proportional to the power-law, $l^{-\alpha}$
18: **end if**
19: **end for**

The table 3.1 gives the neighboring nodes for each switch node.

Table 3.1: Neighbors list before rewiring for figure 3.1.

Switch Node IDs	Neighbor SNs
1	11 13
2	10 12 14
3	11 15
4	10 14 16
5	11 13 15 17
6	12 14 18
7	13 17
8	14 16 18
9	15 17

The final network of 9 PNs and SNs with $R = 0$, $\alpha = 0$ and $p = 0$ is given in figure 3.1.

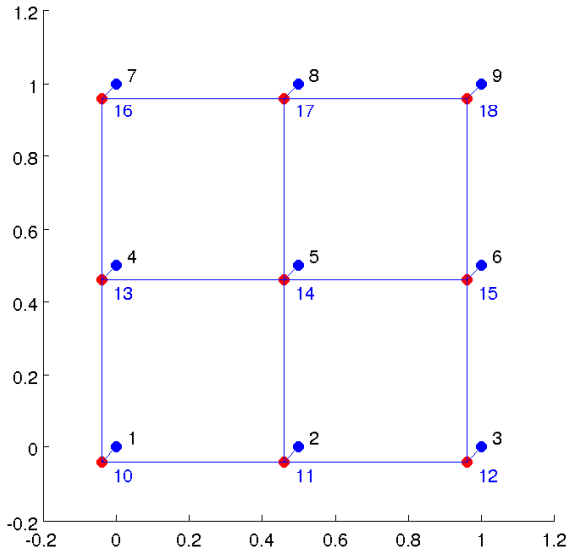


Figure 3.1: 3×3 2D mesh network with 9 SNs and PNs with $R = 0$, $\alpha = 0$, and $p = 0$.

In this example, since $p = 0$, there is no rewiring done.

Example 2: 3×3 2D Mesh with 9 PNs and SNs and no additional wires, i.e., $R = 0$, $\alpha = 0$ and $p = 1$.

As done in example 1, all the PNs are placed in the two dimensional unit square space. This is followed by arranging the SNs adjacent to the PNs and connecting them via bi-directional links. Once a mesh is created, the links are rewired according to α and p. Here, $p = 1$, so each link is rewired. The new location for the link is decided by α value. Since, $\alpha = 0$, there will be more global connections. Table 3.1 gives the neighbor node list of the switch node structure before rewiring is done and table 3.2 gives the neighbor nodes list of the structure after the rewiring is done. Figure 3.2 shows the final figure after the rewiring is done.

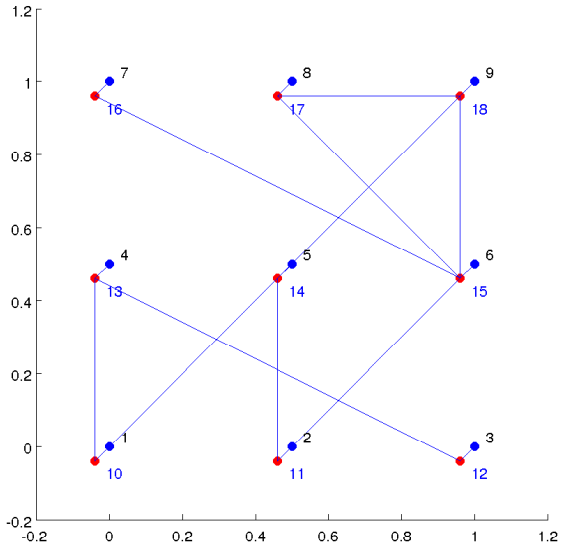


Figure 3.2: 3×3 2D mesh network with 9 SNs and PNs with $R = 0$, $\alpha = 0$, and $p = 1$.

Different values of α and p enable us to further explore the two dimensional design space created by them. These mesh and random networks obtained by varying α and p values can then be evaluated to obtain an optimal network with

Table 3.2: Neighbors list after rewiring for figure 3.2.

Switch Node IDs	Neighbor SNs
1	14 13
2	15 14
3	13
4	12 10
5	10 18 11
6	16 17 18 16 11
7	15
8	18 15
9	15 14 17

good performance characteristics.

3.2 Comparison of Networks in the Four Corners of α and p Space

The networks appear and perform differently in the four corners of the design space covered by α and p. Figure 3.3 illustrates these networks in the four corners.

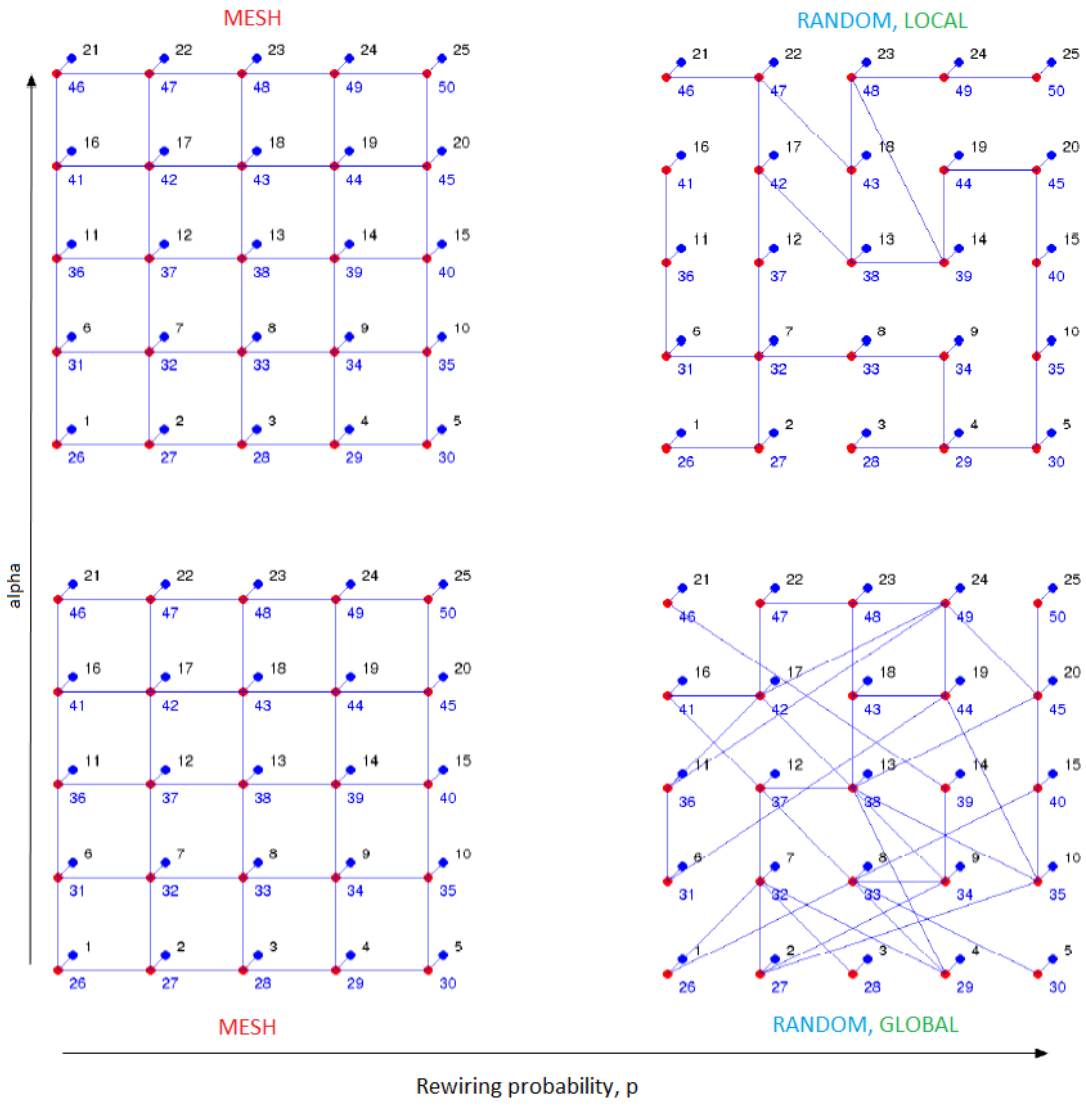


Figure 3.3: Comparison of networks in $\alpha - p$ regime of a 5×5 2D mesh network with 25 SNs and PNs($R = 0$). At $\alpha = 0$ and $p = 0$ we get a mesh. At $\alpha = 0$ and $p = 1$ a random network with global connections is obtained. At $\alpha = 2$ and $p = 0$ we get a mesh again and at $\alpha = 2$ and $p = 1$ we get a random network with local connections.

As can be seen in figure 3.3, at $\alpha = 0$ and $p = 0$, where no link is rewired, the network is a simple mesh-like structure. As we go on increasing p, the network becomes more random. Since, the rewiring is done proportional to $l^{-\alpha}$, at $\alpha = 0$, we get a network with higher number of global links as compared to $\alpha = 2$. As $\alpha \to 2$, the network contains a higher number of local links and a lower number of global links. Not surprisingly, these networks also perform very differently in the four corners of the $\alpha - p$ space. The performance metrics will be discussed in chapter 5

3.2.1 Neighbor distribution

In our experiments, a network is varied from a global to a local network with α values from 0 to 2 with a step size of 0.2 and with the rewiring probability p varying from 0 to 1 with a step size of 0.1. Thus, the network moves from a regular structure to the random structure. The simulations in our framework are very compute intense and hence, we have considered very coarse grained simulations. Since the step size for α and p was chosen as 0.2 and 0.1 respectively, we can thus obtain 121 networks with the different combinations of α and p values. Before evaluating different performance metrics, we analyze the networks in the four corners of $\alpha - p$ space. This will aid the understanding of the networks behavior.

The networks considered for the following two experiments are a 8×8 2D mesh networks with 64 SNs and PNs. Each experiment is averaged over 10 networks.

- With $R = 0$: Since no additional wires are added, at $p = 0$, no rewiring takes place, and hence there are 4 nodes with switch node connectivity (SN_k) of 2, 24 nodes with $SN_k = 3$ and remaining 36 nodes with $SN_k = 4$. There

are a total of 112 bi-directional links in the network, giving us an average connectivity between switch nodes(SN_kAvg) of 3.5, which can be seen in the histogram plot in figure 3.4 . As we go on increasing the rewiring probability p, we get a bell shaped curve with a wider spread. This shows that with increasing p, we get a more distributed network. A longer tail depicts the presence of nodes with connectivity above average. Hence, at $p = 1$, most nodes in the network have connectivity above average.

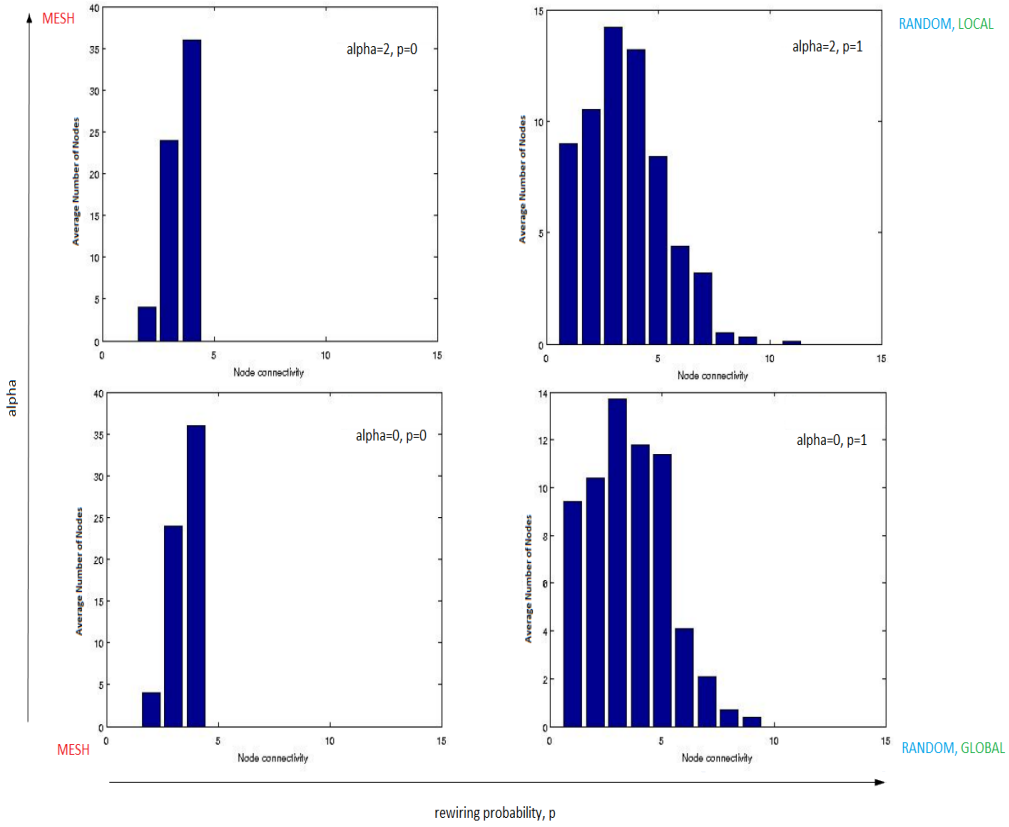


Figure 3.4: Comparison of networks in $\alpha - p$ regime of a 8×8 2D mesh network with 64 SNs and PNs ($R = 0$). At $p = 0$, we get mesh network with no rewiring. As $p \to 1$, more and more links are rewired. At $p = 1$, we get a more distributed network.

- With $R = 100$: An additional 100 bi-directional links are added to the network and hence there are a total of 212 bi-directional links in the network, giving us an average connectivity between switch nodes (SN_kAvg) of 6.625. Referring the figure 3.5, since, these additional bi-directional links are added randomly between the nodes, there is a slight variation in the distributions at $p = 0$. However, as seen, when rewiring probability p increases we get a wider spread, indicating a more distributed network, i.e., more nodes with connectivity above average.

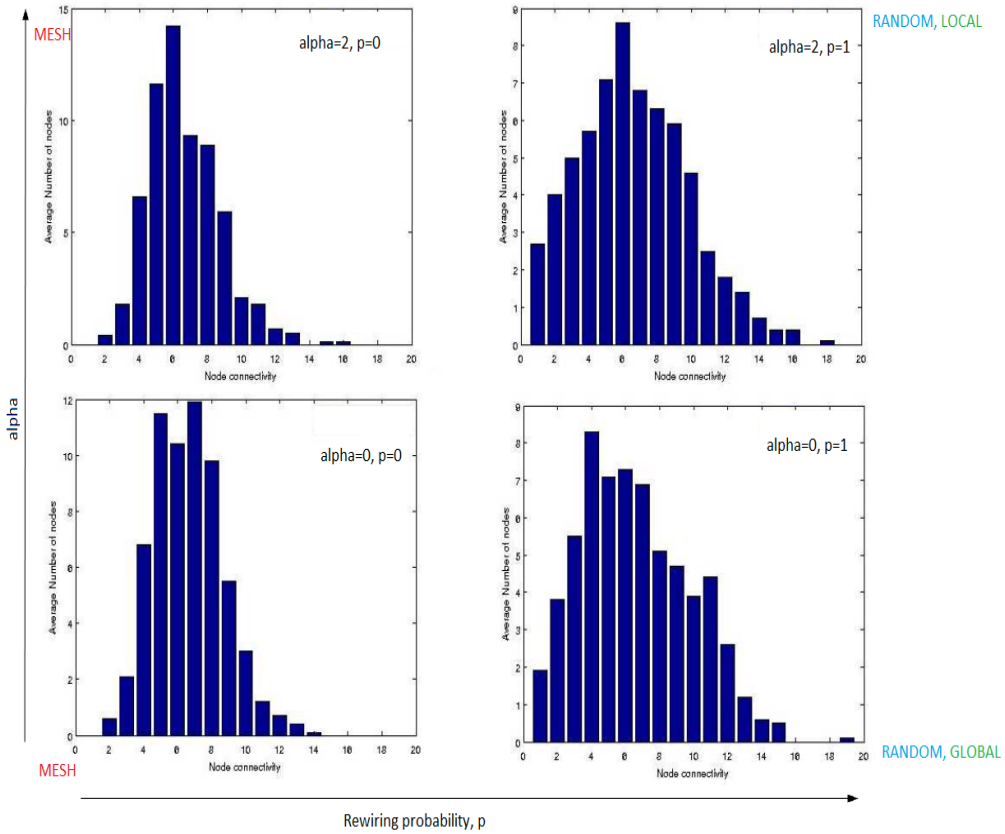


Figure 3.5: Comparison of networks in $\alpha - p$ regime of a 8×8 2D mesh network with 64 SNs and PNs ($R = 100$). At $p = 0$, where no links are rewired, the network is a mesh network. As $p \to 1$, more and more links are rewired and hence at $p = 1$ we get a very distributed network.

Chapter 4

Traffic and Routing

4.1 Traffic

Once a good design with the required trade-off between cost and performance has been obtained, we need to test the network under different traffic patterns to evaluate its performance for realistic applications. The data encapsulated in packets is referred to as traffic in the network. There are several traffic patterns reported in the literature. Different traffic distribution can be classified as *temporal* or *spatial* [7]. *Poisson* is an example of temporal distribution traffic pattern where the injection of data packets follow a Poisson distribution, i.e., the data packets are injected in fixed time interval [7]. Whereas, *random* traffic pattern exhibits the characteristics of spatial distribution because the injection of data is random over the network nodes. Other examples include Transpose [9], Permutation [27], Hot-spot [33], etc. These traffic patterns are used to evaluate the networks.

We have tested our networks under two distributed traffic patterns, *random* and *hot-spot*.

4.1.1 Random traffic

As the name suggests, the traffic generated is of a random pattern. Depending on the traffic intensity, a fixed number of packets is inserted at random sources (PNs) in the network at each time step. These packets are then set out to travel to random destinations (PNs). These packets transfer from one node's output port to another node's input port in one cycle. The random pattern balances the load on the network by injecting the packets at each node with equal probability. Though this traffic pattern is not very realistic, it is a naive pattern which can be used to measure the latency and throughput of the networks with simple routing algorithms.

4.1.2 Hot-spot traffic

In most parallel applications, the traffic is not uniform [18] and hence we need a traffic pattern to test our networks in more realistic scenarios. A hot-spot traffic pattern is one of the realistic traffic patterns observed in many applications like World Wide Web. In such a traffic pattern, a few *hot-spot* nodes receive a larger proportion of the network traffic. We used a hot-spot traffic model inspired by Pfister and Norton [31]. There are two main control parameters used in generating the hot-spot traffic pattern. A *hotspotPerc* decides the number of hot-spot nodes in the network. The nodes are chosen randomly. Once the hot-spot nodes are fixed, a fixed number of messages is inserted in the network at each time step. Each message has a probability h to be injected at the hot-spot node and a $(1-h)$ probability to be injected at a regular (cold) node. Algorithm 2 explains the generation of hot-spot traffic pattern in our simulator.

Algorithm 2 Algorithm to generate a Hot-Spot traffic pattern.

1: Get probability, h.
2: Get the probability of number of hot-spot nodes, *hotspotPerc*.
3: Evaluate the number of hot-spot nodes $H = \text{round}(\text{hotspotPerc} \times N)$.
4: Choose the H PNs randomly.
5: **for all** number of runs, *nbRuns* **do**
6: Generate number of messages *nbMessages* depending on the traffic intensity *tr*.
7: **end for**
8: **for all** *nbMessages* **do**
9: Choose a *Source Node*.
10: Generate random number r between 0 and 1
11: **if** $r < h$ **then**
12: Choose one the H PNs as destination node.
13: **else**
14: Choose one the $(N - H)$ PNs as the destination node
15: **end if**
16: **end for**

4.2 Routing

Routing is considered a critical factor in determining overall network performance in terms of quality (throughput) and quantity (latency) of the data. A naive routing algorithm's task is to guide a certain amount of data from a source node to the destination node. A good routing algorithm performs this task considering issues like avoiding deadlock, live-lock and starvation. Any routing algorithm for a given network is defined by laying out certain goals. A designer can define these goals keeping the application in mind. Based on the application, these goals can vary from robustness and fault-tolerance to scalability. The following a few basic routing algorithms.

4.2.1 Basic routing taxonomy

Centralized and distributed routing

As the name suggests, in centralized routing, the routing decision is centralized and is taken by one controller. Whereas in distributed routing, the decision is distributed over the network and is a collective effort of all the nodes. The main disadvantage of the centralized routing is that the time required to broadcast the updates or information adds to the delay [7]. Shortest path routing is an example of centralized routing, whereas ant routing, where the routing tables are updated by multiple agents, is an example of distributed routing.

Minimal and non-minimal

A minimal routing algorithm, e.g., shortest path routing, is a *greedy algorithm* [9] where the data takes only the minimal path to the destination. On the other end of the spectrum are non-minimal routing techniques, where the data may not always take the shortest path. Examples of the minimal and the non-minimal routing techniques are shortest-path and random routing respectively. Minimal routing is known to give the best latency under low-traffic conditions, but has poor throughput because of the non-optimal load balance over the network.

Deterministic and adaptive

In deterministic routing, the routing path is predetermined and does not consider various network conditions, like faulty nodes or links, congestion, deadlocks, etc. A routing technique, which adapts the routing policy to the varying traffic conditions, is termed as *adaptive routing.* Adaptive routing techniques are particularly

interesting because of their ability to avoid deadlocks and network contention.

Shortest path routing: As the name suggests, this routing algorithm allows the traffic to take the minimal path to the destination. There are different algorithms which can solve the shortest path problem like, the *Floyd-Warshall algorithm*, *Johnson's algorithm*, *Perturbation theory* etc [8]. We implement *Dijkstra's algorithm* [8], which is the most commonly used algorithm to solve the single-source and single-destination shortest path problem. The traffic moves through the network referring the routing table stored in each switch node. Since the message follows the shortest path, the latency of the network is better than any other routing algorithm. But, this is observed only under low network traffic load. Under a high traffic load, there will be hot spots created in the network, thereby congesting the network and increasing the latency. The throughput of the switch nodes scale worse in this routing algorithm because of the poor load-balance in the network. Thus, this routing technique is best for small networks with low traffic because every node needs to store the routing table, which becomes bigger with an increase in the network size.

Random routing: Random routing is simple to implement, but is inefficient for large-scale networks because the message takes random paths to reach its destination. The random wandering nature of the messages in the network makes this type of algorithm the worst in terms of latency. However, this algorithm balances the load better than shortest path routing, thereby achieving better throughput.

Adaptive routing: To obtain an ideal throughput, the network load should be balanced. However, with the shortest path (minimal) routing algorithm, the traffic

load throughout the network is not balanced and the network may not yield the best performance. Since shortest path routing does not consider current traffic conditions, an adaptive routing may be required, which routes the data while looking at the network traffic constantly. So for better performance, a designer may have to add a few longer paths in the network, but a good designer may choose to get the best of both of these worlds.

4.2.2 Ant routing

Adaptive routing overcomes the limitations of deterministic-path routing by exploring all possible paths to avoid deadlocks. One of the many adaptive routing techniques is ant routing. It is an agent-based adaptive routing algorithm. As opposed to the shortest-path routing algorithm, ant routing is distributed and can adapt to the network conditions. Ant-based routing algorithms have been used in the past in various communication networks [7], [32], [35] and are inspired from the collective intelligence behavior of real ants. The ants wander in the network in parallel and collectively discover the routes to the destinations. The ants can find paths in an adaptive behavior looking at the network and traffic conditions. The communication is through the environment typically called *Stigmergy* [7] and is a behavior observed in social insects. Social insects exchange the data by depositing *pheromones* on the path on the way back to their nest when they have found the food. A strong pheromone concentration indicates a valid and a possible shortest path to the food. Our ant routing implementation is inspired by the work done by Di Caro and Dorigo [7] and Liu et al. [20].

There are two types of agents used in ant routing who work collectively to determine the paths. These are *forward ants* and *backward ants*.

Algorithm 3 A simple ant routing algorithm.

1: **for all** number of runs $nbRuns$ **do**
2: Generate forward ants fw_{ant} from each processing node N to a random destination.
3: Update all the SNs and PNs.
4: **for all** fw_{ant} **do**
5: Follow the steps in algorithm 4.
6: **end for**
7: **if** fw_{ant} reaches destination **then**
8: fw_{ant} dies.
9: Generate bw_{ant}
10: **else**
11: fw_{ant} moves to the next node.
12: **end if**
13: **for all** bw_{ant} **do**
14: Follow the steps in algorithm 5.
15: **end for**
16: **if** bw_{ant} reaches source **then**
17: bw_{ant} dies.
18: **else**
19: bw_{ant} moves to the next node.
20: **end if**
21: **end for**

Forward ants: These ants are generated at the source nodes (PNs) with an ant ID and the source and destination Ids. The ant then sets out to the first available switch node. In our experiments, we have considered that a PN is connected to only one nearest SN.

Algorithm 4 Forward ants task.

```
1: for all fw_ant do
2:    Update its arrival time at the node.
3:    if the next node ID is present in the routing table then
4:       get the next node ID information from the routing table.
5:    else
6:       choose any of the neighbors with equal probability.
7:    end if
8:    if first ant then
9:       consider the previous ant delay as 1 and calculate the estimated delay
         from the equation 4.2 .
10:   else
11:      get the previous ant delay from the SN and calculate the estimated delay.
12:   end if
13:   if next node queue size is > threshold and the estimated delay is greater
      than the average path delay then
14:      drop this forward ant
15:   else
16:      put it in the end of queue of the next node.
17:   end if
18: end for
```

Backward ants: These ants are generated at the destination nodes (PNs) with the same ID as that of their forward counterparts along with the path information collected by the forward ant.

Forward ants are inserted in the network and they travel only in the forward direction, i.e., from the source to the destination. Initially, they consider paths in the absence of pheromones, randomly. On every SN that the forward ant crosses, a decision has to be made whether to forward the ant to the next node or not.

Algorithm 5 Backward ants task.

1: **for all** bw_{ant} **do**
2: Fetch the $prevNodeID$ from the via node list stored within.
3: Go to the next SN.
4: Update the $mincost$ and $maxcost$ of the path.
5: Increase the probability of the taken link.
6: Decrease the probability of the links not taken.
7: **end for**

The decision depends on the next node's queue capacity and the estimated delay and is calculated by the equations 4.1 [20].

$$d^i \geq \bar{D}_p(n) \bigcap l > threshold, \tag{4.1}$$

where d^i denotes the estimated delay at the node i of a loop-free path with n nodes and is given in equation 4.2.

$$d_m{}^i = (1 - w) \times d_{m-1}{}^i + (s_m - a_m) \times w. \tag{4.2}$$

a_m and s_m denote the arrival and successful transmission time of the m^{th} packet and $0 \leq w \leq 1$. l is the queue length of the next node. Hence, if the estimated delay at the node is greater than the average path delay and if the next node's queue length is greater than the threshold, the ant is dropped. Equation 4.3 and 4.4 give the average path delay calculation for the loop free path.

$$\bar{D}_p(n) = \frac{D_p(n)}{n} \tag{4.3}$$

$$D_p(n) = \sum_{i=1}^{n} d^i \tag{4.4}$$

When a forward ant reaches its destination, the forward ant dies and a backward ant is generated. This backward ant takes the same path back, i.e., it moves from the destination to the source and its sole purpose is to update the routing tables. The next forward ant is more likely to take the path taken by the previous ant. This is because the routing table indicates a higher probability for that path. This probability denotes the *goodness* factor of the path and was increased by the backward ant. This type of learning is called *reinforcement learning* [29].

Chapter 5

Performance Metrics

5.1 Measure of Network Performance

The performance metrics and evaluations in our simulator are inspired by Pande et al. [28].

5.1.1 Wire cost

In our experiments, the wire cost is defined as the sum of all the wire lengths measured in distance units. There are many applications which demand a large number of nodes in the network. An increase in the number of these nodes will increase the wire length and hence the wire cost. We will further explore this trade-off in section 5.2.1.

5.1.2 Average shortest path length and average number of hops

As discussed in section 2.3.3, rewiring links with probability p affects the characteristic path length of the network. Even a few shortcuts can drastically decrease the average path length of the network, yet the networks appear highly clustered at the local level. The average geodesic (or shortest) path length of the small-world model is an important performance metric. Prior work by M. E. J. Newman [23]

and Barabasi et al. [1] have been focused on this quantity which is denoted by l. In a bi-directed network, the average shortest path length is defined as:

$$l = \frac{1}{n\,(n-1)} \sum_{i \neq j} d_{ij}, \tag{5.1}$$

where d_{ij} is the geodesic distance from node i to node j. Self connections are excluded. The average shortest path is calculated in distance units. Since our networks are described in a unit space dimension, we calculate the average shortest path length as the average number of hops a message would take to traverse over all possible paths in the network. The average number of hops can be *non-dynamic* or *dynamic*. The non-dynamic hops in the network are calculated based on the shortest path of the network and do not consider the network traffic. The dynamic hops are calculated in terms of actual message hopping over the nodes in the network, which may or may not consider the shortest path. The dynamic hops depend on the routing algorithm and are generally proportional to the latency.

5.1.3 Energy and power

Toggling the inter-switch wires and gates in the switches dissipate energy [28]. Pande et al. have investigated the dynamic energy dissipation due to the communication process among nodes. We have only looked at the power dissipation of the network when no communication is taking place. The total power of the network is the addition of the power dissipated due to switch nodes and the power due to the interconnects.

The power due to the switch node is evaluated from a look-up table, whereas the power due to the interconnects is given by equation 5.2 [28].

$$Power_{interconnect} = 0.5 \times Cper\mu m \times V_{dd}^2 \times A \times f \times totalwirelength \qquad (5.2)$$

There are certain assumptions made in calculating the switch node and the interconnect power. These are stated below.

1. The capacitance of the wire is assumed to be 1.0354 pF/m.
2. The Cu unit wire length is $5000\mu m$.
3. Operating voltage, $V_{dd} = 1V$.
4. Operating frequency is assumed to be 115 MHz.
5. Switch nodes operate at a frequency of 4 MHz.
6. Relative switching activity = 0.5.

5.1.4 Throughput

Throughput is an important performance metric of a network and is measured as the rate at which the packets are accepted at every node. In our simulations, the throughput of a switch node is measured in messages/number of updates/switch node. Throughput of a node depends on the type of routing algorithm and the network topology used. For example, shortest path routing in random networks can lead to an uneven usage of nodes, allowing only few nodes to be used for the numerous paths and creating congestion in the queues of those nodes. This reduces the average throughput of the network.

5.1.5 Latency

Latency of a network can be defined as the elapsed time between the injection of the data packet at the source node in the network and the reception of the packet at the destination node. Some overhead at the nodes contribute to the latency too [28]. Like throughput, this performance metric depends on the topology as well as the routing algorithms. For example, as explained in section 4.2.1, the shortest path routing has the least latency because the traffic takes the minimal path to the destination. Random routing, as discussed in section 4.2.1, exhibits worst latency because the traffic takes a random path. In our simulations, the average latency of the network is measured in clock cycles and is the ratio of the total time required by the messages to traverse through the network to the total number of messages received at the destinations.

5.2 Performance Evaluation

5.2.1 Scalability

We need to make sure that the on-chip communication technique scales well with system size. With the increase in the number of on-chip components, one needs to make sure that the communication fabric transfers the data efficiently. The networks considered for this experiment are described in two and three dimensional space with varying number of nodes. The baselines for this experiment are the 1D ring, the 2D mesh, and the 3D mesh network. The other topologies examined are the 3DRM with varying α values, where the nodes are placed randomly in a 3D space and the inter-switch link lengths follow a power-law distribution 2.3.4. Each experiment is averaged over 10 runs, i.e., each network is built 10 times to get good

approximated values and a low standard deviation, which is shown by the error bars.

Wire length

The objective of this experiment is to analyze the wire length (cost) of different topologies as the network scales up. As shown in figure 5.1, the increase in wire length is linear. For all networks except the 1D ring, the wire length increases as the system scales up, i.e., as the number of nodes increase. The wire length of 1D ring remains constant at around 3 because in our simulator, the ring network has a constant unit diameter. As discussed in section 2.3.4, the global connections are the long length links whereas the short length links connecting near neighbors are called the local connections. The figure depicts that a 3D network with random arrangement of nodes with $\alpha = 0$ is the most expensive network because of a large number of global connections. As the network size increases, the increase in cost is much higher as compared to other network topologies. This is analogous with networks with higher α values. As α increases, the cost reduces because of the more number of local connections. Mesh structures, where the nodes are connected only with their neighbors, are cheaper than the network where the nodes are randomly arranged. This is because there are no long range connections in a regular mesh network. The average connectivity of a 8×8 3D mesh network with 64 nodes is 4.5, whereas that of a 2D mesh network with an equal number of nodes is 3.5, thereby making the 3D mesh less favorable in terms of cost.

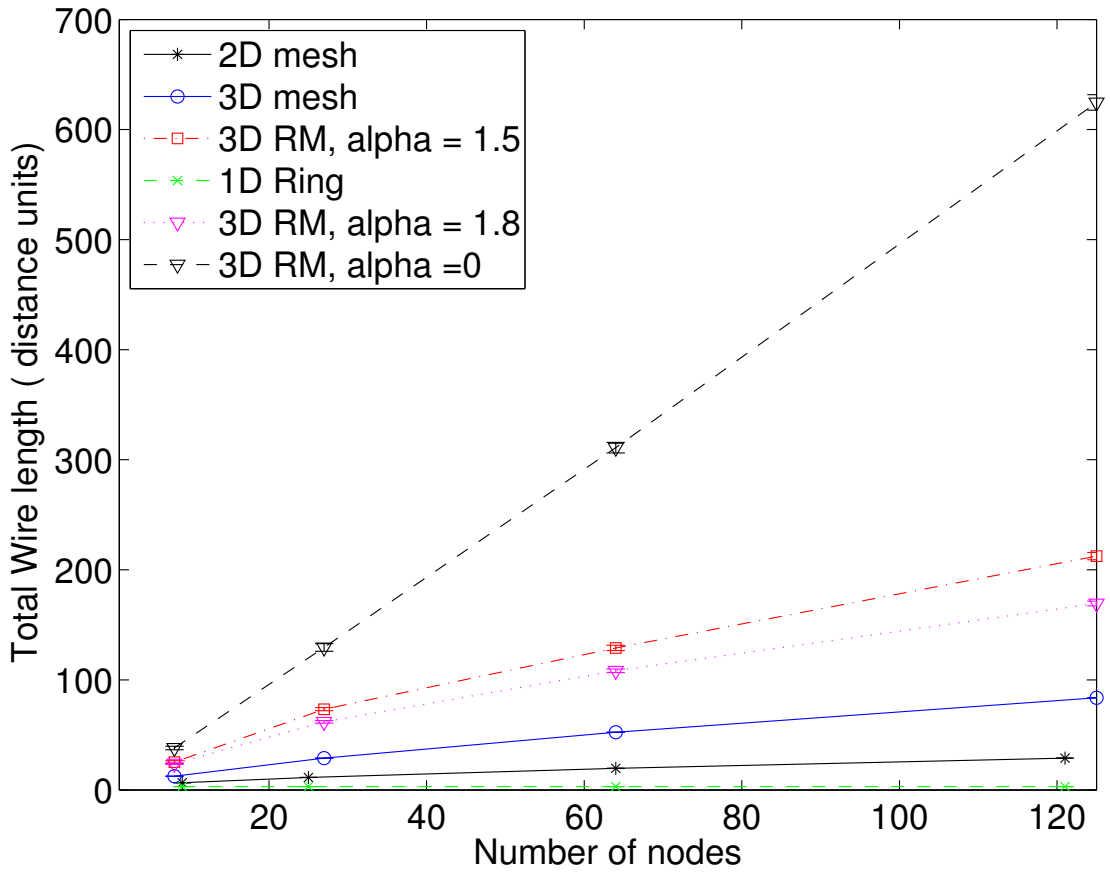


Figure 5.1: Wirelength as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs.

DISCUSSION: A 1D ring scales best in terms of the wire length. Among 2D and 3D mesh networks, the 2D mesh scales better and is more favorable where cost is a concern. A 3D RM network with $\alpha = 0$ is the least favorable option with regards to wire length.

Average number of hops

This experiment is carried out to examine the performance in terms of average number of hops as the system scales up. Figure 5.2 shows that as the network scales up, the average number of hops increases. The increase is very drastic for a 1D ring, where each node is connected only to its two left and right neighbors. When the number of nodes in the network increases, the message has to hop over more nodes to reach its destination. The average number of hops for a 8×8 2D mesh network with 64 nodes is 5.29, which is higher as compared to 3.77, the average number of hops for a 3D mesh network with the same system size. The 3D RM networks have lower average number of hops as compared to the 3D mesh networks with the same number of nodes. This is because of the existence of a few long range short-cuts in the network. Also, the average number of hops increases as α increases.

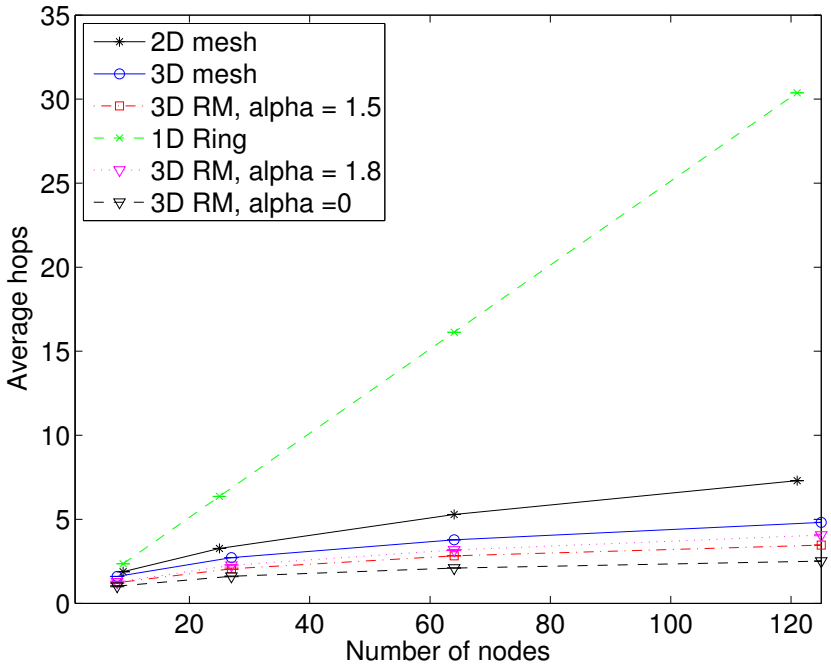


Figure 5.2: Average number of hops as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs.

DISCUSSION: A 3D RM network scales best with the system size, giving the least average hop count, whereas the 1D ring network scales the worst.

Power

As discussed in section 5.1.3, the total power is a summation of the power due to the interconnects and the inter-node connectivity. Since the interconnect power is a higher factor responsible, the total power scales up like the wire length (cost), which can be seen in figure 5.1. As seen in figures 3.4 and 3.5 and discussed in chapter 3, the value of α does not affect the inter-switch node connections and hence the switch node power of RM networks for different α values is almost the

same. However, the total power of 3D RM with $\alpha = 0$ is slightly higher because of the interconnect power, which scales like the wire length. A 1D ring scales best in terms of total power because of only two connections per switch node and a unit diameter. Among 2D and 3D mesh networks, the 2D mesh scales better and is favorable where power is a concern.

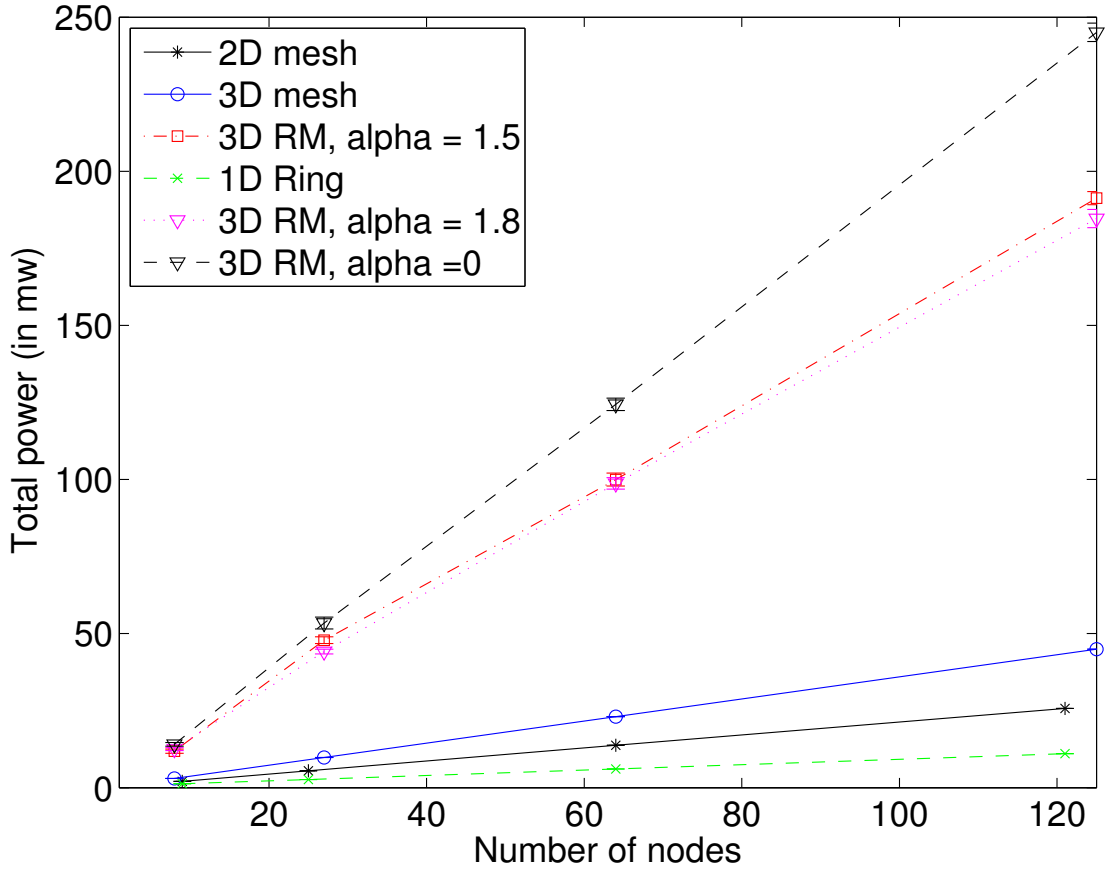


Figure 5.3: Total power as a function of the network size. The networks considered are a 1D ring, a 2D mesh, a 3D mesh, and a 3D random multitude with $\alpha = 0$ (globally interconnected), $\alpha = 1.5$ and $\alpha = 1.8$ (locally interconnected). The data is averaged over 10 runs.

DISCUSSION: The total power scales up similarly to the wire length because of the interconnects power. 2D mesh network is more favorable than 3D mesh or

3D RM networks.

5.2.2 Local and global connections

As explained in section 2.3.4, a small-world power-law network is obtained by rewiring each connection with a probability proportional to $l^{-\alpha}$, where l is the Euclidean distance between the nodes. A large value of the exponent α results in a locally interconnected network, whereas a small exponent α would give a globally interconnected network. The aim of this experiment is to evaluate the networks with local and global connections and its effect on the performance metrics. Since rewiring is done effectively with few extra links in the network, we have considered 2D and 3D mesh networks with varying p values and R = 0, 50, and 100 for the following experiment. All the networks considered are 8×8 2D and 3D meshes with 64 SNs and PNs. The experiments are averaged over 10 runs.

Wire length

This experiment lets us analyze a few trade-offs among different network topologies from the cost point of view. As seen in figure 5.4, the cost reduces as the network becomes more local, i.e., as $\alpha \to 2$. This is because of the lower number of global connections. 2D mesh and 3D mesh networks form the baseline of this experiment with the constant wire lengths of 19.8 and 52.4 respectively. This is because for these networks, since $p = 0$, none of the links are rewired, hence there is no effect of α. The remaining 2D and 3D mesh networks with $p = 0.5$ and 1 show an exponential decrease in the wire length as the network becomes more local. The wire length of 2D mesh is shorter as compared to the wire length of 3D mesh because the average connectivity of the network with the same system size in 2D

is less than the average connectivity in 3D. At $\alpha = 0.8$, we can see a trade-off between a 3D mesh network with no rewiring and no extra wires and a 2D mesh network with $p = 0.5$ and $R = 50$. For almost the same cost as that of a 3D mesh, we can get better performance due to a few global connections in a 2D mesh network with $p = 0.5$, $\alpha = 0.8$ and $R = 50$. Similarly, at $\alpha = 1.2$, we see another trade-off between a 3D mesh with $p = 0$ and $R = 0$ and a 2D mesh with $p = 1$ and $R = 100$.

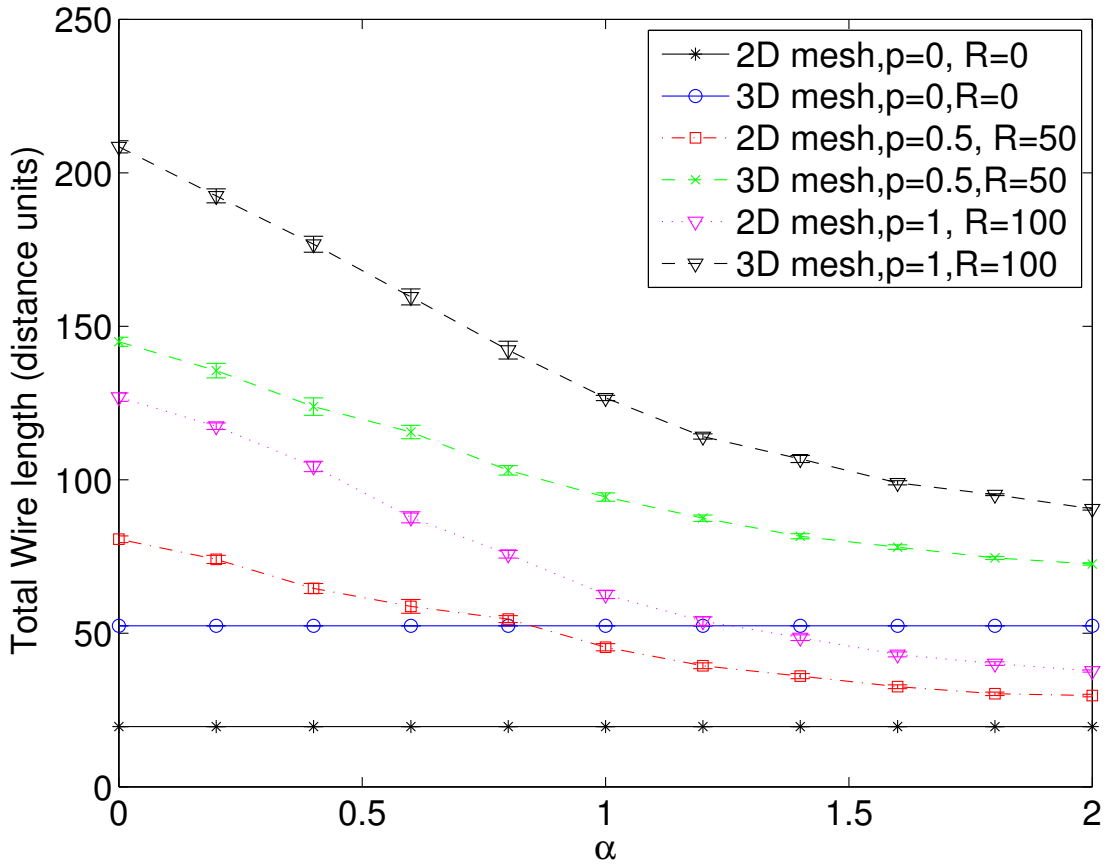


Figure 5.4: Wire length as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs.

DISCUSSION: As shown in figure 5.4, the wire length (cost) of a network

decreases as α is varied from 0 to 2. The cost of the network is high if there is a large number of global connections as compared to the cost of the local connections only. Again, from the cost point-of view, a 2D mesh network with no rewiring would be the best choice, but clearly we can benefit from the performance of a 2D mesh with few longer links with a very low cost overhead.

Average number of hops

According to the small-world property (see section 2.3.3), adding a few longer links in the network can improve the performance in terms of the average number of hops. As seen in figure 5.5, the average number of hops increases as the network becomes more local, i.e., as $\alpha \rightarrow 2$. This is because of the presence of more short length links in the network. The average hop count for a 2D mesh and 3D mesh with no rewiring is constant at 5.29 and 3.77 respectively. Rewiring of the additional wires adds more shortcuts to the network giving a lower average hop count. Thus, the average hop count decreases as the rewiring probability p and the additional wire factor R increases. As shown in figure 5.5, there are certain trade-offs that can be achieved at $\alpha = 0.8$. For almost the same performance, we can trade-off the 3D mesh with the 2D mesh network with $p = 0.5$, $\alpha = 0.8$ and $R = 50$ with very little cost overhead. Similarly, at $\alpha = 1.2$ we see another trade-off between a 3D mesh with $p = 0$ and $R = 0$ and a 2D mesh with $p = 1$ and $R = 100$.

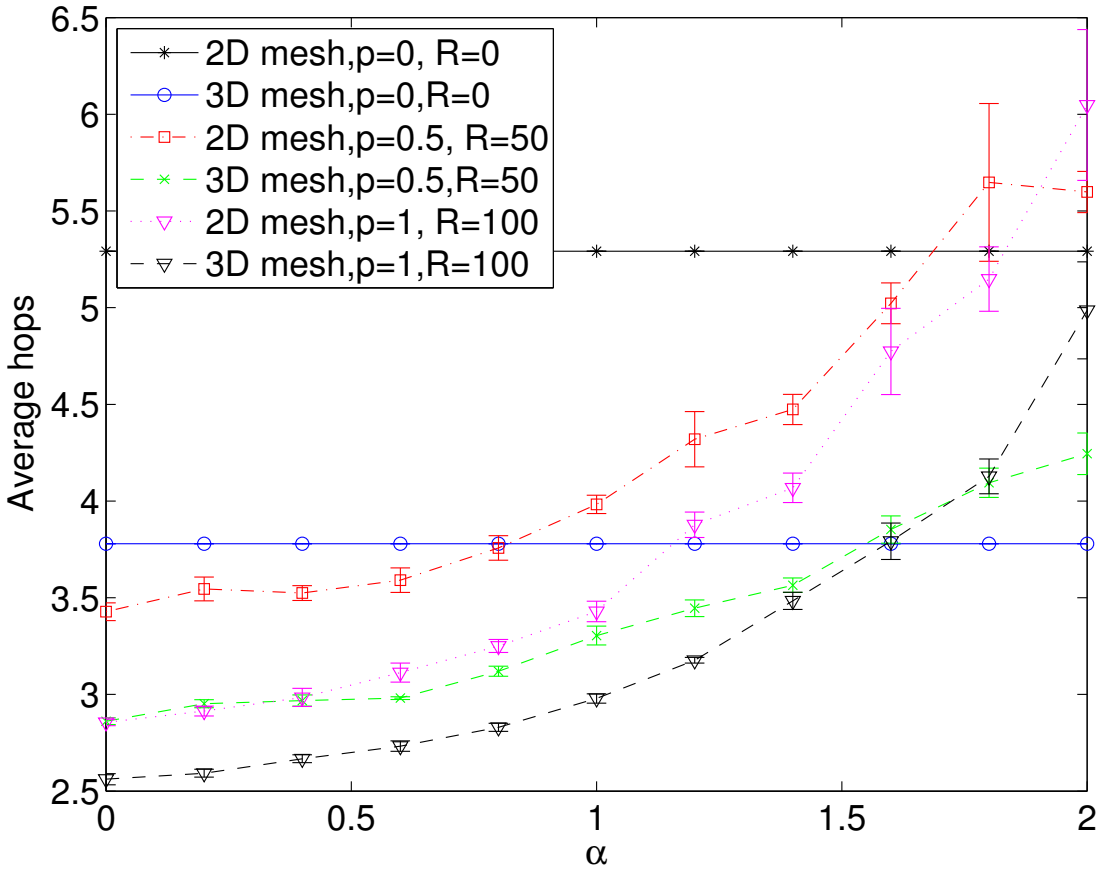


Figure 5.5: Average number of hops as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs.

DISCUSSION: Performance in terms of the average number of hops degrades as a network becomes more local. A 3D mesh network with $p = 1$ and $R = 100$ gives the best performance and a lower average hop count of 2.56 at $\alpha = 0$.

Power

As shown in figure 5.6, 2D mesh network with $p = 0$ and $R = 0$ shows the lowest power of 22.98 mW, followed by that of a 3D mesh with 34.3 mW. Additional R links in the network add to the interconnect power and hence the total power. As

the network changes between a large number of global and local connections, the total power due to interconnects decreases gradually. However, α does not affect the power due to inter-switch connectivity. A 3D mesh network with $p = 1$ and $R = 100$ has the highest total power of 86.1 mW at $\alpha = 0$ and 89.5 mW at $\alpha = 2$.

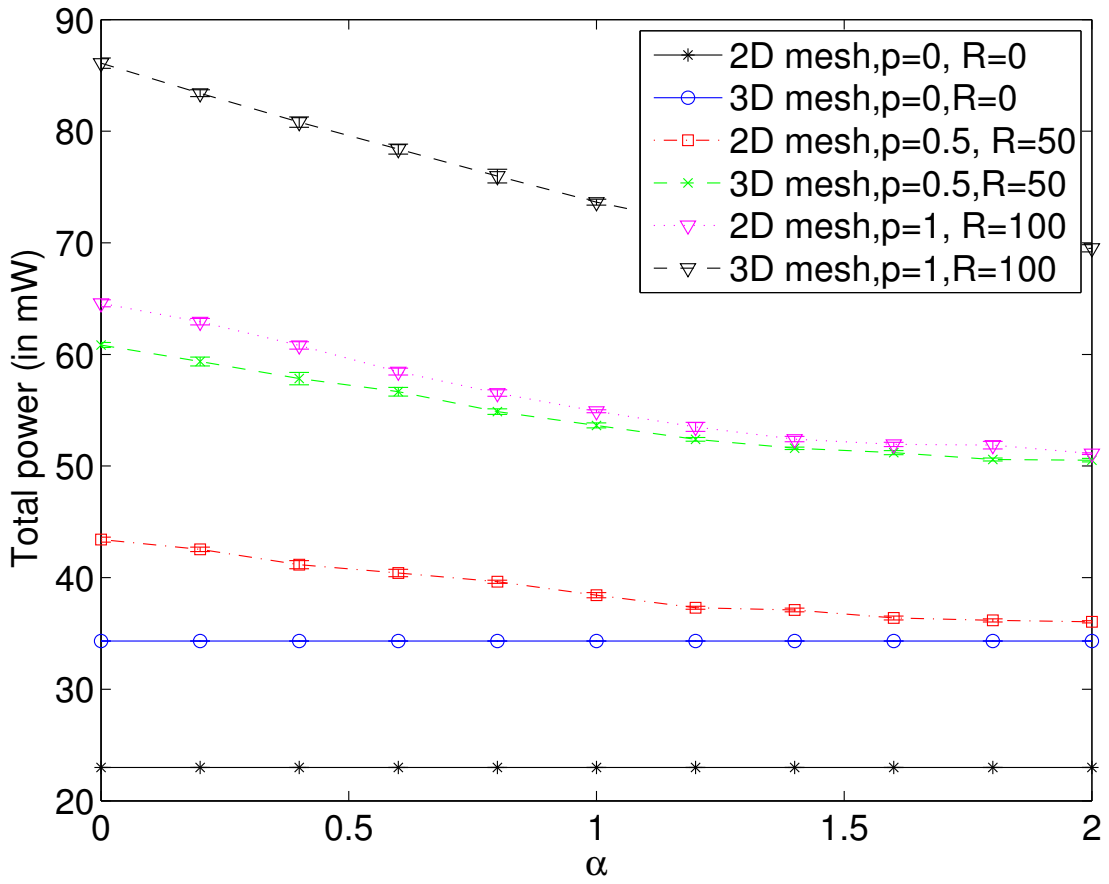


Figure 5.6: Total power as a function of the power-law exponent α. The networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D meshes with $p = 0$, 0.5 and 1. The data is averaged over 10 runs.

DISCUSSION: The total power decreases as α increases and it increases with an increase in the number of additional links in the network. For a design with power concerns, a 2D CA mesh is more favorable because it dissipates the least

amount of power.

5.2.3 Regular and random graphs

As explained in chapter 2, by varying the rewiring probability p from 0 to 1, we allow the network to change between a regular lattice and a random graph. For $p = 0$, we get a regular lattice and as $p \to 1$, the edges are rewired to a new random location and we obtain a random graph. While rewiring, care is taken to avoid any self connections.

Wire length

As shown in figure 5.7, the wire length and hence the wire cost increases as the rewiring probability p, increases. This is due to the introduction of long range 'short-cuts' in the network. The wire length of a 3D mesh with $\alpha = 0$ and $R = 0$ is higher than the wire length of a 2D mesh with $\alpha = 0$ and $R = 0$ because of the higher average connectivity of the 3D mesh network. At a lower value of p, the wire length of a 2D mesh and 3D mesh network with no additional wires is 19.6 and 52.4 respectively, which is less than the networks with $R = 50$. But as the links are rewired, the wire length of the 2D and 3D mesh networks with additional wires become lower with 54.7 and 104.1 respectively. This is because of the more local topology with $\alpha = 1$.

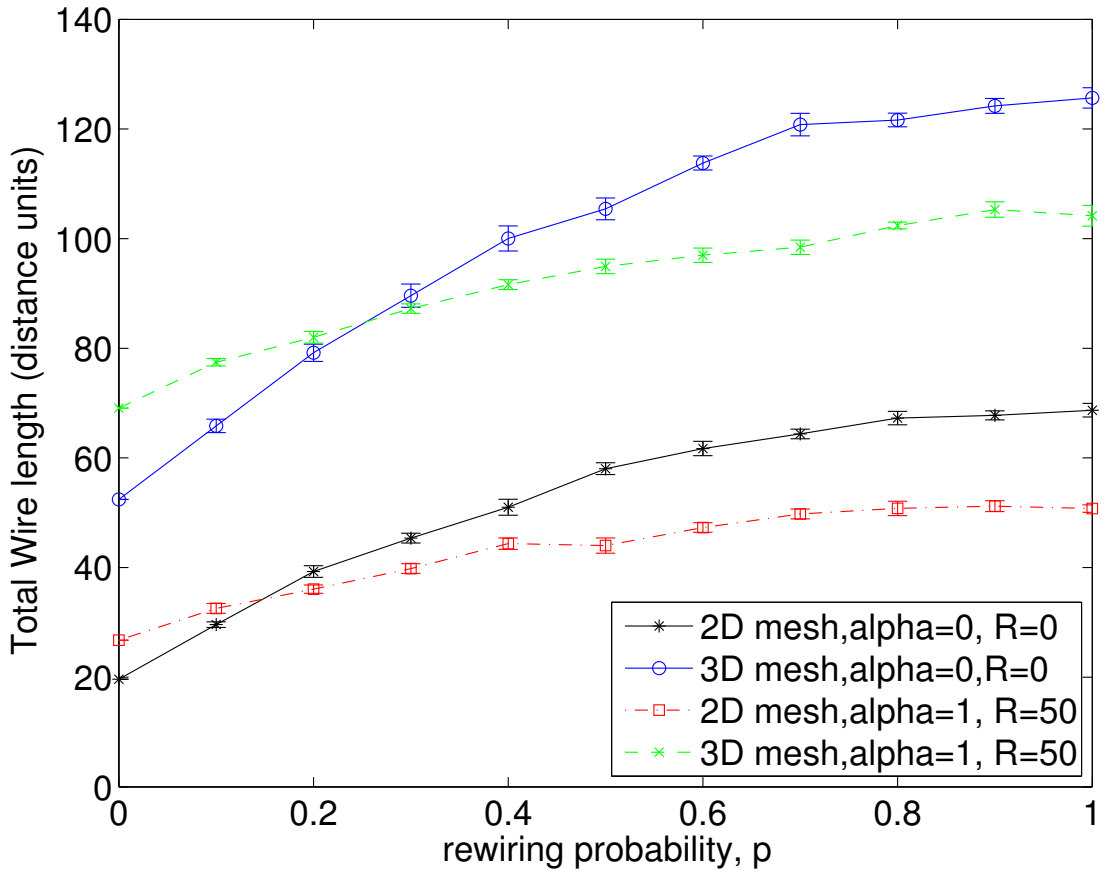


Figure 5.7: Wire length as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs.

DISCUSSION: When no links are rewired, i.e., at $p = 0$, a 2D mesh network with no additional links is preferred since it has the lowest wiring cost. However, at higher p, α plays an important role in selecting an optimal network. A network with $p = 1$ and $\alpha = 1$ is an optimal network from the cost point-of-view.

Average number of hops

With a rewiring probability of $p = 0$, the network becomes a large-world model where the topology is highly regular. By contrast, as $p \to 1$, the system exhibits

the small-world phenomenon. At $p = 1$, the network is more global and hence the average shortest path length decreases logarithmically. To prove the existence of small-world network as the probability p increases, we calculate the average number of hops in the system as the probability p is varied. As shown in the figure 5.8, the average number of hops at $p = 0$ for the 2D mesh is 5.29 and that of a 3D mesh network with an equal number of nodes is 3.77. As $p \to 1$, more and more links are rewired and hence the hop count reduces. The average number of hops for 2D and 3D mesh reduces as α decreases and as R increases. This is because the additional global links in the network reduce the average hop count and hence increases the network performance.

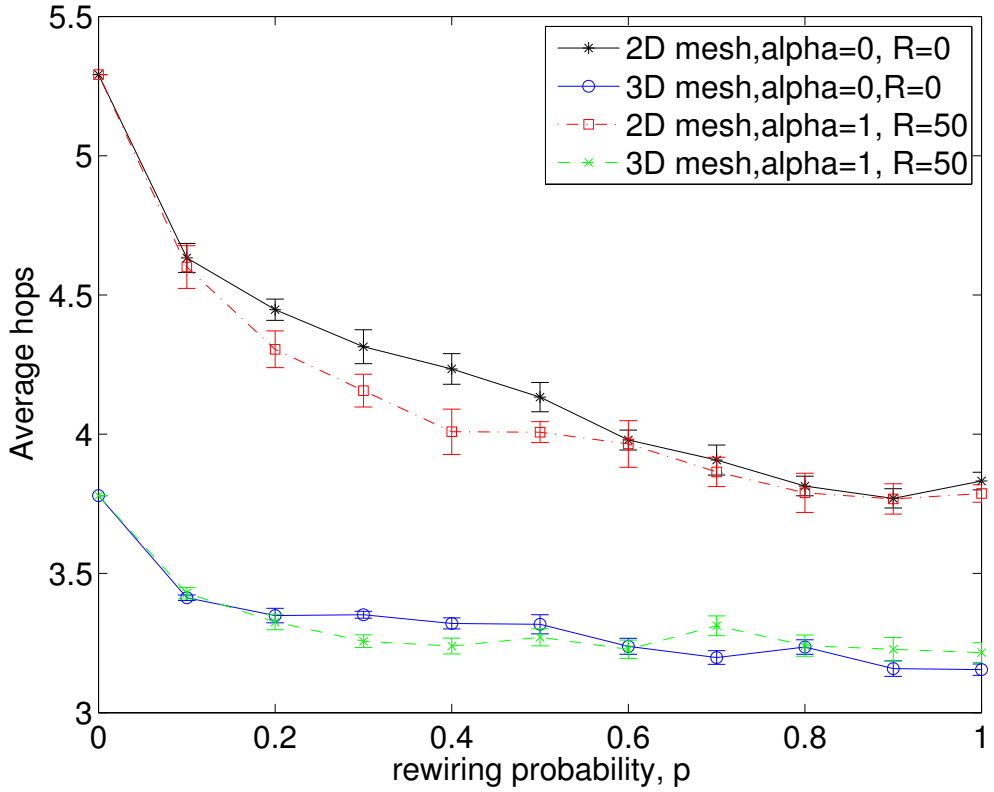


Figure 5.8: Average hops as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs.

DISCUSSION: The average number of hops reduces as the network becomes more random. For a lower average hop count and hence a higher performance, a 3D mesh network with $\alpha = 0$ and a higher R value is beneficial.

Power

Power increases as the rewiring probability p, increases. One of the two factors which add to the total power is the power due to the interconnect lengths. Hence, as $p \rightarrow 1$, most of the links get rewired to distant nodes, thus increasing their lengths and hence the power. As shown in figure 5.9, the total power of a regular 2D mesh network is the lowest (about 22 mW). As the network is rewired and as $p \rightarrow 1$, the long range connections increase the power, thereby resulting in a total power of 31.1 mW. The power due to switch-node connectivity is almost constant as the network changes between a regular and a random network and because the average connectivity in the network thus remains the same. The total power of a 3D mesh network is largest because of the higher average connectivity as compared to the 2D mesh and the additional 50 links in the network.

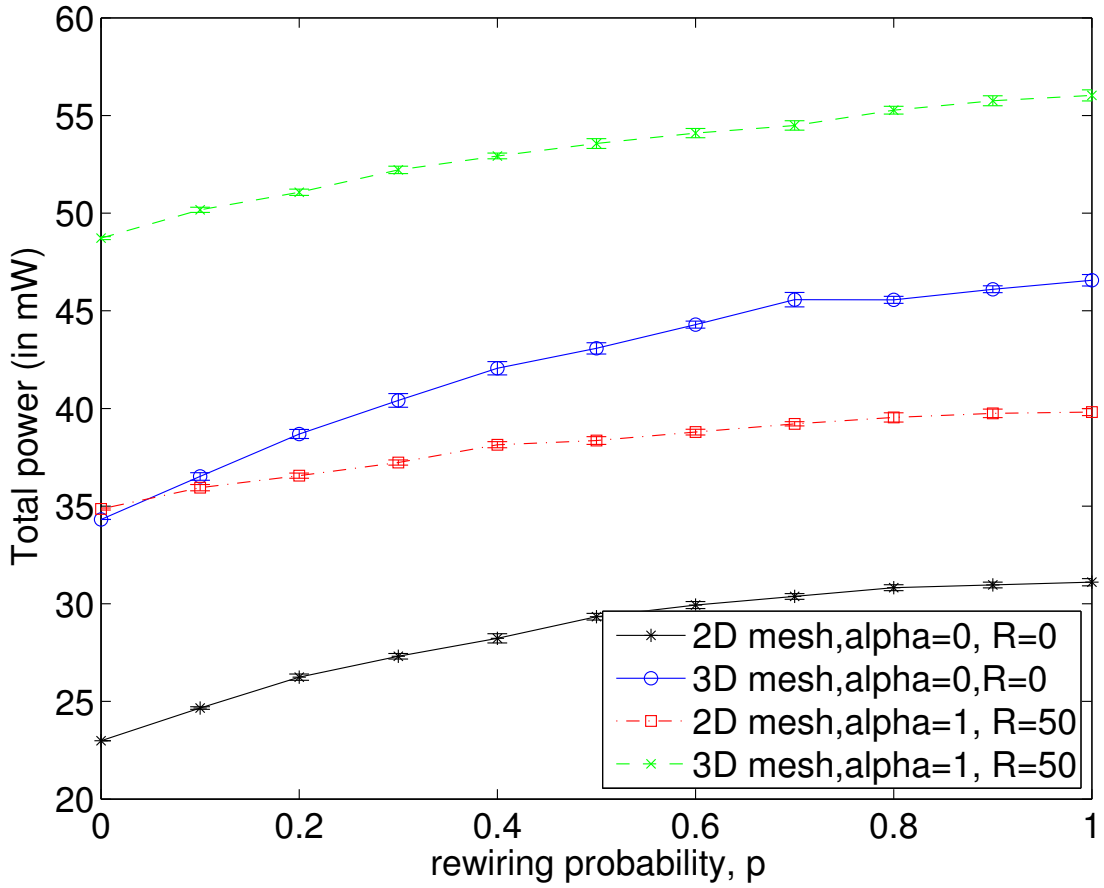


Figure 5.9: Total power as a function of the rewiring probability p. Networks considered are 8×8 2D and $8 \times 8 \times 8$ 3D mesh with $\alpha = 0$ and $\alpha = 1$. The data is averaged over 10 runs.

DISCUSSION: As p is varied, more and more links are rewired and hence the power increases. Thus, a 2D mesh network with no additional wires is more beneficial if power is a concern.

5.3 Performance Evaluation with Traffic

As discussed in chapter 4, we need to test our networks under different traffic patterns and routing algorithms to evaluate and analyze them. The following

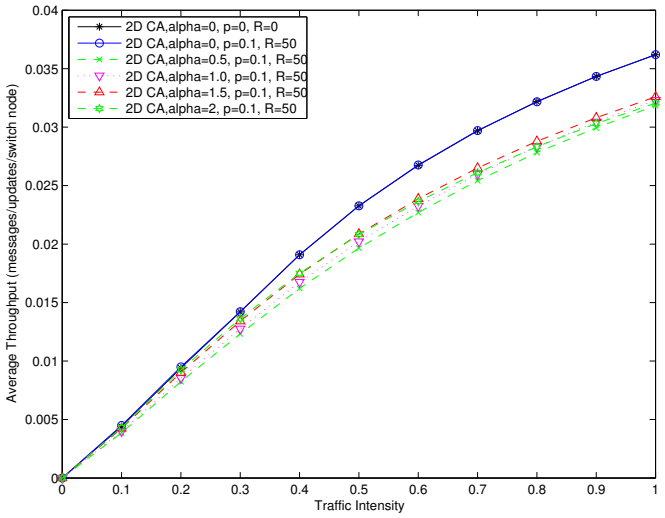
section compares the throughput and latency of different network topologies.

Random traffic

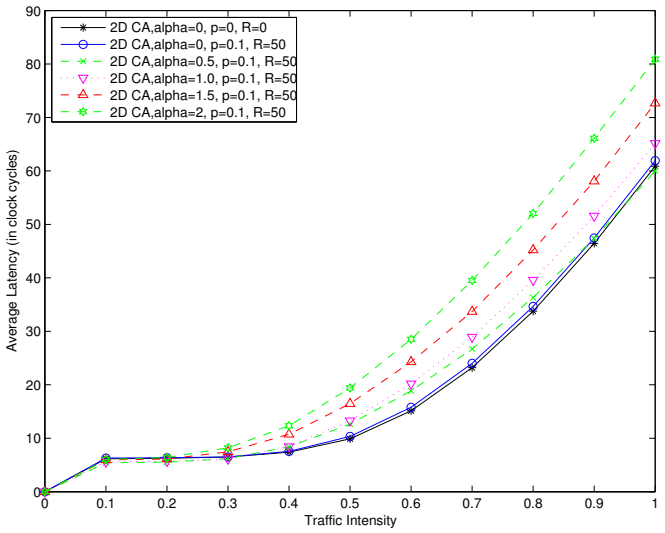
As discussed in section 5.1.4, the throughput of the switch nodes is an important metric and is measured as number of messages/updates/switch nodes. Latency is measured in terms of number of cycles. Figures 5.10a and 5.10b show the throughput and the latency of different networks as the traffic intensity is varied. The networks considered are a 8×8 2D mesh with varying α values and $p = 0.1$. The baseline network for this experiment is the 8×8 2D mesh with $\alpha = 0$, $p = 0$ and $R = 0$. The number of virtual channels considered is 4 [28]. Since the routing algorithm considered is the shortest path routing, a contention in the queues is observed. To avoid any dropping of data packets under heavy network traffic, we do not limit our queue size for this experiment.

As shown in figure 5.10a a regular 2D mesh with no rewiring has the highest throughput of 0.0362. This is due to its regular structure. Adding a few extra links ($R = 50$) and rewiring the network with probability $p = 0.1$ degrades the throughput. As more and more links are rewired, the latency increases because of the contention and a longer delay in the queues. When the network is not congested, the additional links results in less latency. This can be seen at traffic intensity = 0.2. At traffic intensity = 0.3, the rewiring of the links increases the latency and as $\alpha \rightarrow 2$, the average latency increases. As shown in figure 5.10b, the latency of 2D mesh network is about 60 cycles for the highest traffic intensity, making it the most favorable network. Adding more links to the network does not improve the throughput because the throughput is a measure of the switching capacity of the nodes. Hence, both 2D mesh networks with $R = 0$ and $R = 50$ do

not improve the throughput. As the rewiring probability $p \to 1$, the throughput decreases because of the added short-cuts.



(a) Throughput as a function of the traffic intensity.



(b) Average latency as a function of the traffic intensity.

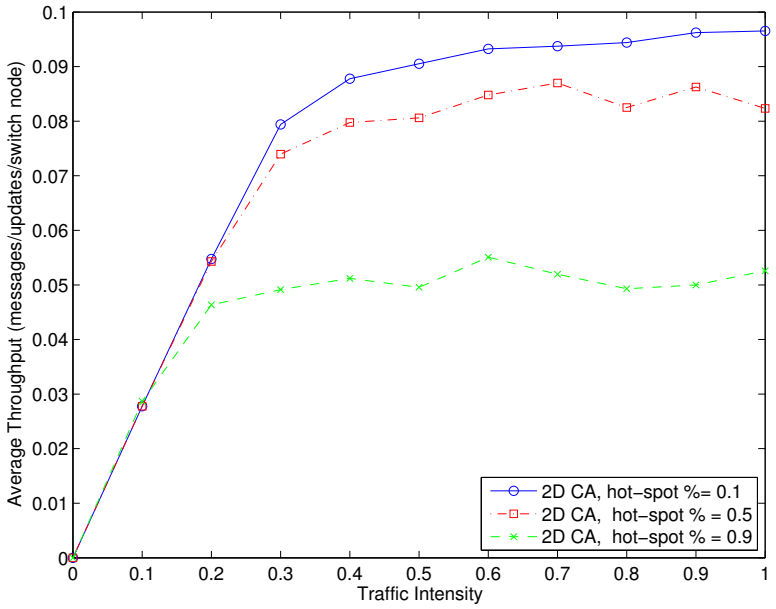
Figure 5.10: Throughput and average latency of networks with varying α and p values. Figures 5.10a and 5.10b uses 8×8 2D mesh networks with different α, p and R values. The data is averaged over 10 runs.

DISCUSSION: A regular mesh performs the best under random traffic pattern and shortest-path routing at higher traffic intensity. This is because of the regular structure of the network and the absence of any long range links. However, at lower traffic intensity, the additional links in the network reduce the latency, thus making a small-world power-law network more favorable.

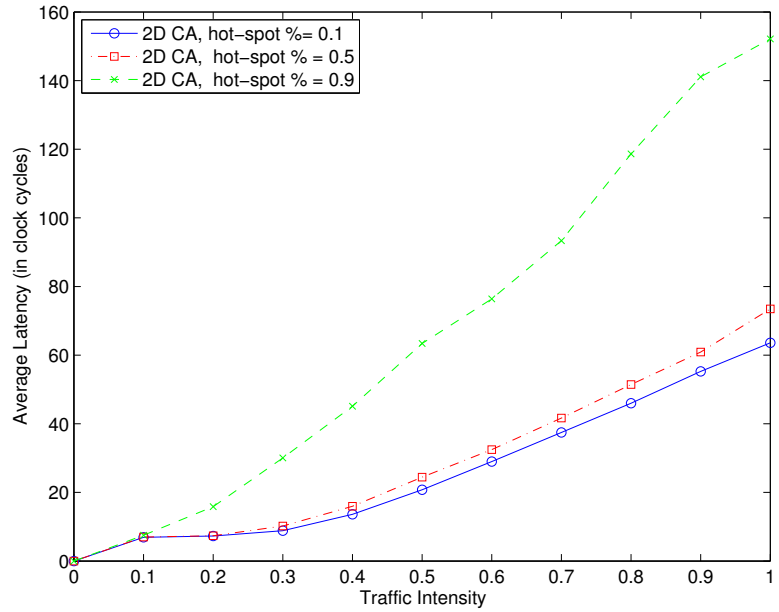
Hot-spot traffic

As discussed in section 4.1.2, hot-spot traffic is more realistic than random traffic. In hot-spot traffic, a few nodes in the network receive the most amount of traffic. In this experiment, we compare the latency and throughput of the networks under hot-spot traffic. The network considered is the 5×5 2D mesh network with 25 SNs and PNs. The experiment is averaged over 10 runs.

As shown in figure 5.11a, a network with $hotspotperc = 0.1$ has about 3 hot-spot nodes in the network. This network has the highest throughput, even with $h = 0.9$. As the number of hot-spot nodes in the network increases, the throughput decreases. For a network with 3 hot-spot nodes, the congestion in the queues is observed at traffic intensity = 0.2. As shown in figure 5.11b, the average latency increases drastically. After the network saturates, the throughput stabilizes. The network with $hotspotperc = 0.9$ has 23 hot-spot nodes in the network. For this network, the congestion is created at a very low traffic intensity of 0.1.



(a) Throughput of a $5x \times 5$ 2D CA network with varying hot-spot percentage



(b) Average latency of a 5×5 2D CA network with varying hot-spot percentage

Figure 5.11: Throughput and average latency of networks with varying hot-spot probability, h. Figures 5.11a and 5.11b use a 5×5 2D mesh network with $p = 0$, $R = 0$, $h = 0.9$ and varying *hotspotperc*. The data is averaged over 10 runs.

DISCUSSION: Hot-spot traffic performs worse than random traffic. As the number of hot-spot nodes increase in the network, the latency increases and the throughput decreases. This is due to the contention in the queues of the hot-spot nodes.

5.3.1 Ant routing

As discussed in section 4.2.2, we implemented an ant routing algorithm inspired by Di Caro and Dorigo [7] and Liu et al. [20]. The network considered for this experiment is a 5×5 2D mesh with 25 SNs and PNs. We have limited the queue size of each node to 20. The nodes are interconnected with 4 virtual channels. In this experiment, different routing algorithms are compared. As explained earlier, under shortest path routing, the data takes only the shortest path to the destination. The use of the minimal path allows data to take the least number of clock cycles to reach its destination. Hence, the latency with shortest path routing is the lowest, as shown in figure 5.12. However, the shortest path routing does a poor job in balancing the load over the network and hence the throughput is worse and increases gradually to 0.078 with an increase in traffic intensity. Random routing does not create any routing tables and hence, the data takes random path to reach its destination. However, this balances the load throughout the network and increases the throughput as seen in figure 5.12. Since the queue sizes are limited, soon there will be contention in the queues and the data packets will have to be dropped. Hence for shortest path, if congestion is observed, we let the data packets drop. In ant routing, the networks consist of forward ants, backward ants and the data packets. In case of congestion, we drop the forward or the backward ants and re-route the data packet to another node. Since, the data packet is re-routed, there

is a delay observed in the path, which adds to the latency. The re-routing of the data also balances the load throughout the network and increases the throughput.

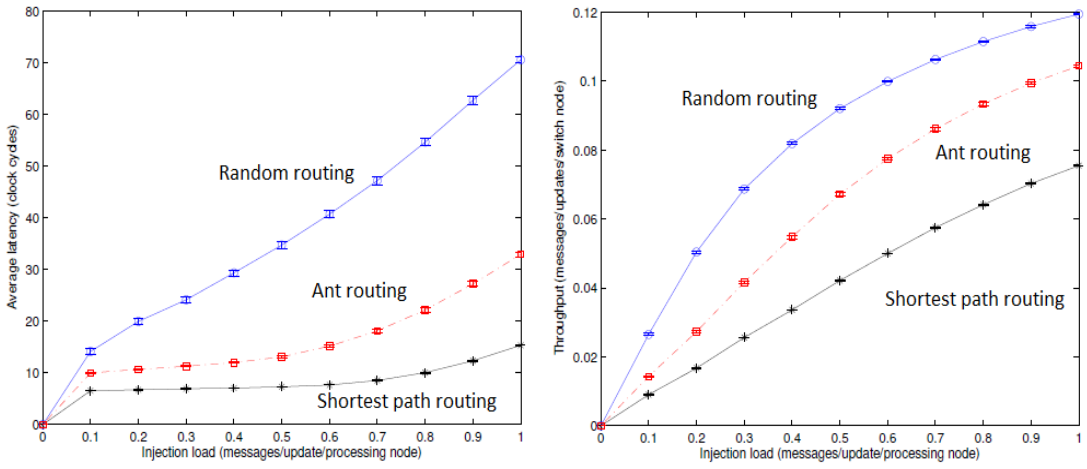


Figure 5.12: Throughput and average latency of the mesh network with different routing algorithms. The data is averaged over 10 runs.

DISCUSSION: The average latency by using ant routing is higher than that of the shortest path routing and lower than for random routing. The throughput of the network with ant routing is better than the shortest-path routing, but not as good as for random routing.

Chapter 6

Small-World Power-Law Networks

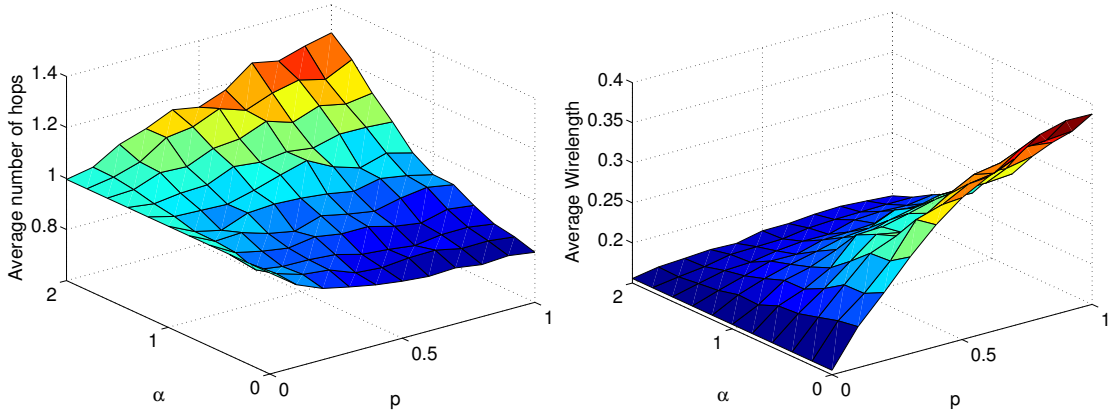
As discussed in chapter 3, the two main parameters, α and p, are used to explore the design space and evaluate different networks and their properties. In chapter 5, we analyzed networks with different local and global connections as well as regular and random graphs. We noticed that there were trade-off between different networks for better performance and lower cost. It was evident that a regular network has the least wiring cost, but a random graph gives better performance. In this section, we analyze the $\alpha - p$ design space to obtain an optimal network with low cost, low power and low hops. We try to answer the question of how much rewiring is sufficient to get a network with better performance with little cost overhead.

For this experiment, we have considered a 5×5 2D mesh network with 25 SNs and PNs. As discussed in chapter 3, we vary the power-law exponent α from 0 to 2 in steps of 0.2 and rewiring probability p from 0 to 1 in steps of 0.1. With different combinations of α and p over the 2D mesh network, we obtain about 121 networks. The third control parameter used in the design space is the additional links R. In our simple simulator there is no systematic way to add these links and hence we vary R from 0 to 100 in steps of 10. With the three control parameters, we obtain a total of 1331 networks to be analyzed. The following three experiments analyze

the trade-offs between different network performances and help us find an optimal network.

6.1 Experiment 1: Wire length and average number of hops

In this experiment, we measure the total wire length and the average number of hops for the above mentioned 1331 networks. Each of these networks either give a better performance in terms of average number of hops or a lower wiring cost. For comparison, we normalize all the networks with a reference network which is a 5×5 2D mesh network with $R = 100$. Figures 6.1a and 6.1b show the normalized average number of hops and total wire length plots for a 5×5 2D mesh network with $R = 10$.



(a) Average number of hops.

(b) Total wire length .

Figure 6.1: Average number of hops and total wire length as a function of α and p for a network with $R = 10$.

To consider both the average number of hops and the wire length together, we

introduce an objective aggregate function given in equation 6.1

$$y = a \times hops + (1 - a) \times wirelength \tag{6.1}$$

The factor a in equation 6.1 is the weight factor, which allows us to give importance to either of the two performance metrics. For example, with $a = 0.1$, 10% weightage is given to the average hop count and 90% to the wire length. With $a = 0.5$, both hops and power are equally favored. We compare the normalized average number of hops and wire length together by putting them in equation 6.1, varying a from 0 to 1 in steps of 0.02. The combined plot is shown in figure 6.2.

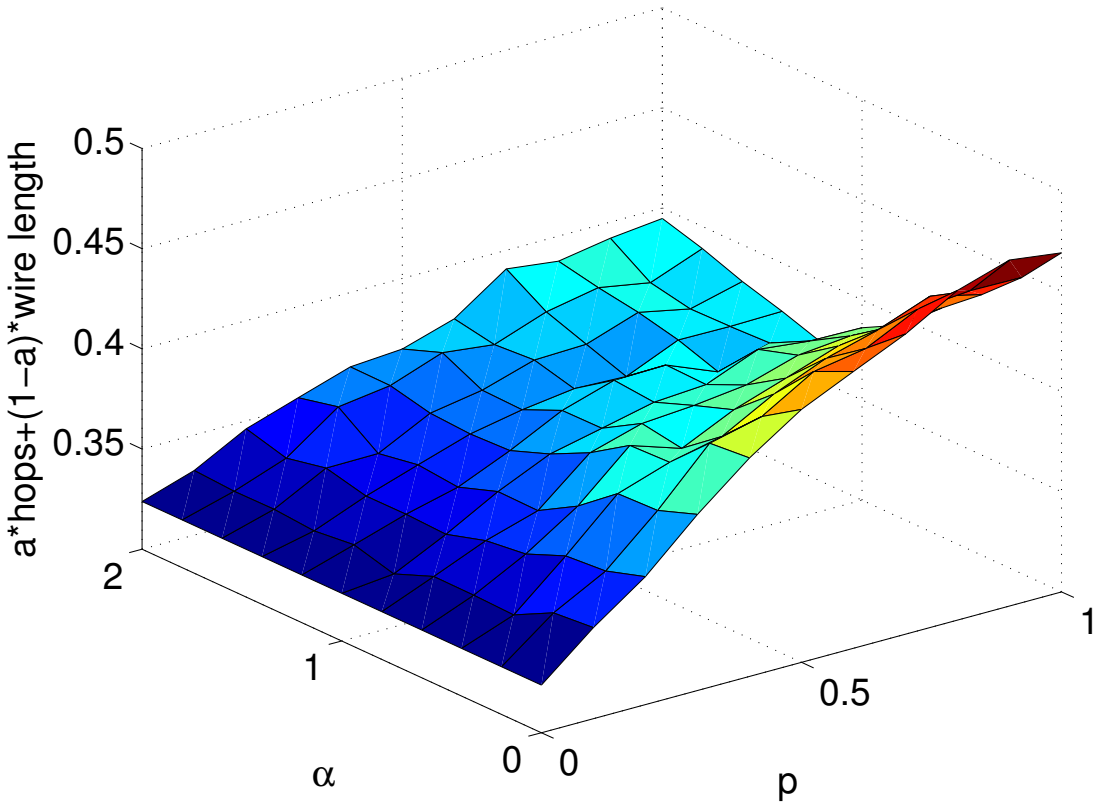
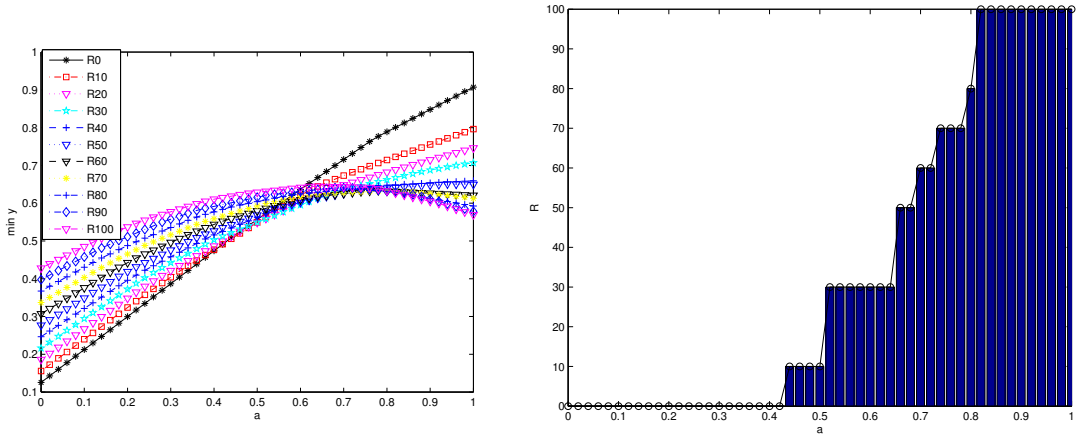


Figure 6.2: Aggregate objective function y. 5×5 2D mesh network with $R = 10$ and $a = 0.2$.

Each of the 1331 combined plots has a minimum surface value called the *sweet-spot*. Figure 6.3a shows a plot with the *sweet-spots* in all the networks for different R values. From the figure 6.3a we can know that if the wire length (cost) is an important factor, i.e., if $a = 0$, the 5×5 2D mesh network with $R = 0$ is the best solution. However, for a network with the best performance, we need a network with $R = 100$. Figure 6.3b shows the minimum value of R required as a varies.



(a) Minimum y value as a function of a.

(b) Extra links R scaling as a function of a.

Figure 6.3: Minimum y and R scaling plots for average number of hops and total wire length

Table 6.1 details some of the optimal network parameters for different a values. As shown, it is evident for $a = 0$ that the optimal network is a mesh with no additional links. Because at $a = 0$, the wire length (cost) is favored and the average number of hops are not at all considered. However, for $a = 0.5$, the optimal network is a small-world power-law network.

Table 6.1: The table gives examples of different optimal parameters for different values of a. For $a = 0$, the optimal network is a mesh whereas for $a = 0.5$, the optimal network is a small-world network.

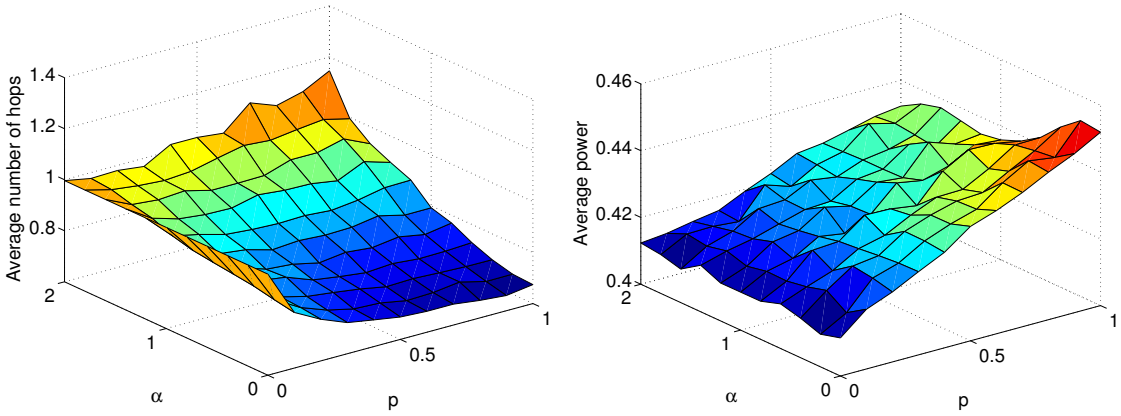
Value of a	R	α	p	Topology
0	0	0	0	Mesh
0.3	0	0	0	
0.5	10	0.4	0.1	Small-world power-law
0.7	50	0	0.3	
1.0	100	0	0.7	

6.1.1 Summary

Table 6.1 states a few examples obtained at different values of a. As it can be seen, for $a = 0$, in equation 6.1 the total wire length (cost) of the network is favored against the average number of hops. Experimentally, at $a = 0$, the optimal R value obtained is 0. With no additional wires, i.e., $R = 0$, we trace the minimum value from the surface plot back to obtain $\alpha = 0$ and $p = 0$. Thus, if complete weightage is given to wire length, the optimal network obtained is a mesh network. For $a = 1$, the average number of hops of the network is favored and thus the optimal network is a small-world network with $R = 100$, $\alpha = 0$ and $p = 0.7$. If equal weightage is given to both average number of hops and wire length, we obtain a network with $R = 10$, $\alpha = 0.4$ and $p = 0.1$, which is a small-world power-law network.

6.2 Experiment 2: Total power and average number of hops

In this experiment, we measure the total power and average number of hops for all the 5×5 2D mesh networks with varying R. As discussed in experiment 1, we normalize all the networks with a reference network, which is a 5×5 2D mesh network with $R = 100$. Figures 6.4a and 6.4b show the normalized average number of hops and the total power plots for a 5×5 2D mesh network with $R = 40$. Here, we have not considered the cost factor since the total power is a summation of the power in the interconnects and the power in the switch nodes. Hence, the cost of the interconnects is implicit in the power measurement.



(a) Average number of hops

(b) Total power

Figure 6.4: Average number of hops and the total power as a function of α and p for a network with $R = 40$.

To consider both the total power and the average number of hops, we use the aggregate objective function given in equation 6.2.

$$y = a \times hops + (1 - a) \times power \tag{6.2}$$

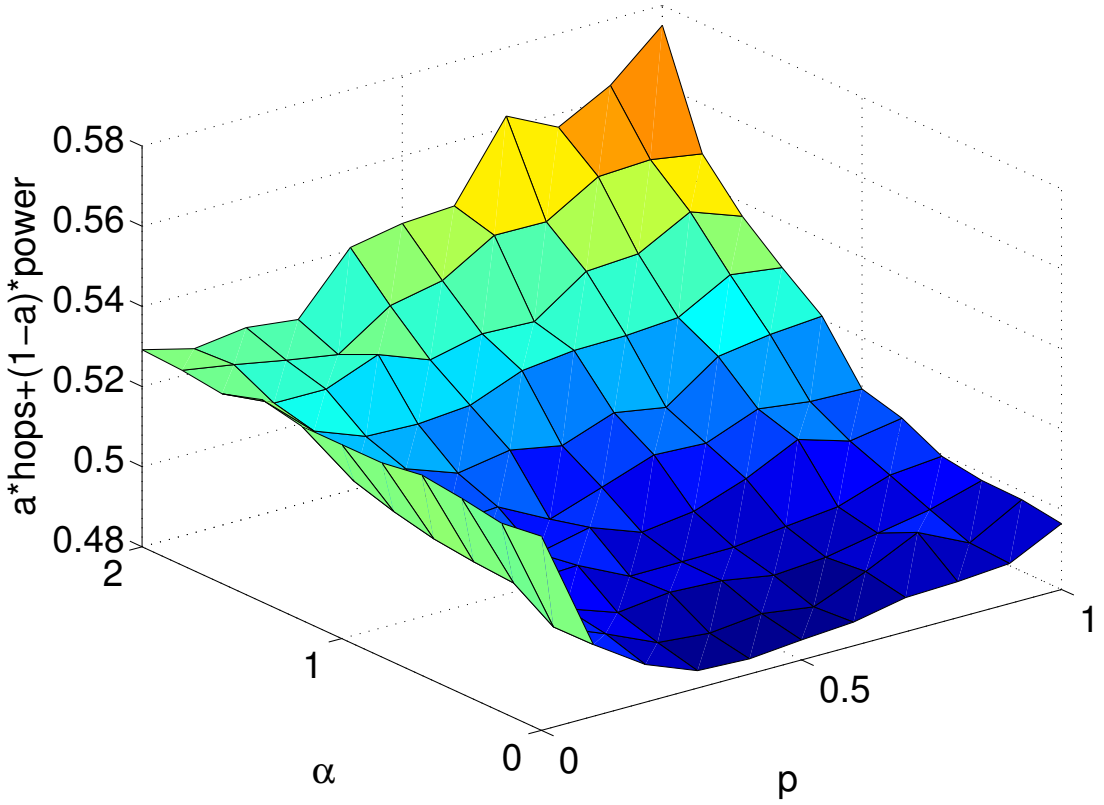
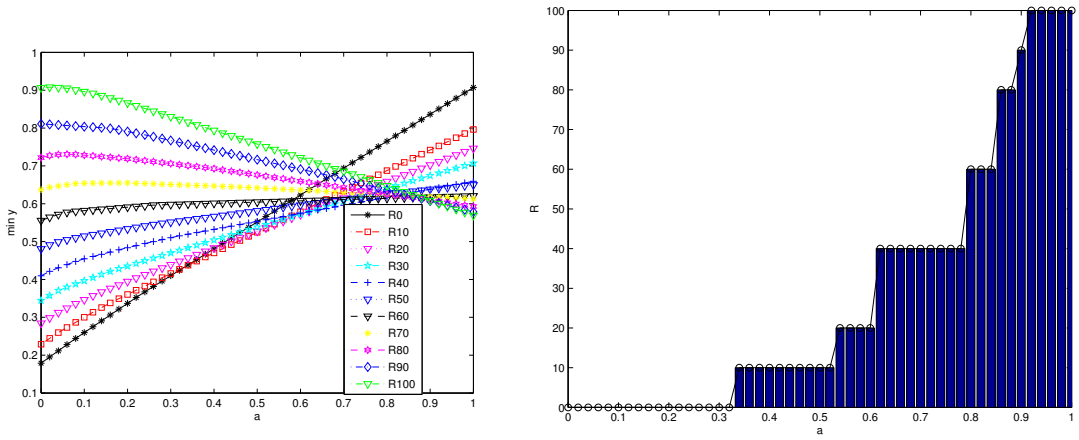


Figure 6.5: Aggregate objective function, y. 5×5 2D mesh network with $R = 40$ and $a = 0.2$.

The weight factor a lets us give weightage to either of the two performance metrics. Figure 6.5 shows an example of the combined plot obtained from the aggregate objective function given in equation 6.2 for a 5×5 2D mesh network with $R = 40$ and $a = 0.2$.

With different combinations of α, p and R, a total of 1331 combined surface plots are obtained. Each of these combined plot has a *sweet-spot*, as explained in experiment 1. Figure 6.6a shows a plot with the *sweet-spots* in all the networks for different R values. Figure 6.6a indicates that if power is an important factor, i.e., if $a = 0$, the 2D network with no additional links is the best solution. However, for the best performance, we need a network with $R = 100$. Figure 6.6b shows the

optimal number of extra links R as the weight factor a varies.



(a) Minimum y value as a function of a.

(b) Extra links R scaling as a function of a.

Figure 6.6: Minimum y and R scaling plots for average number of hops and total power

Table 6.2 shows some of the optimal network parameters for different a values. As shown , it is evident for $a = 0$, the optimal network is a mesh network with no additional links. This is because at $a = 0$, the total power of the network is favored and the average number of hops are not at all considered. For $a = 0.5$, we give equal importance to both average number of hops and total power. As shown in the table 6.2, for $a = 0.5$, the optimal network obtained is the random network.

Table 6.2: The table gives examples of different optimal parameters for different values of a. For $a = 0$, the optimal network is a mesh, whereas for $a = 0.5$, the optimal network is a random network.

Value of a	R	α	p	Hops	Total power (in mW)	Topology
0	0	0	0	3.62	36.65	Mesh
0.3	0	0	1	2.97	40.33	
0.5	10	0	1	2.60	51.82	Random network
0.7	40	0	0.9	2.44	91.72	
1.0	100	0	0.7	1.86	196.63	

6.2.1 Summary

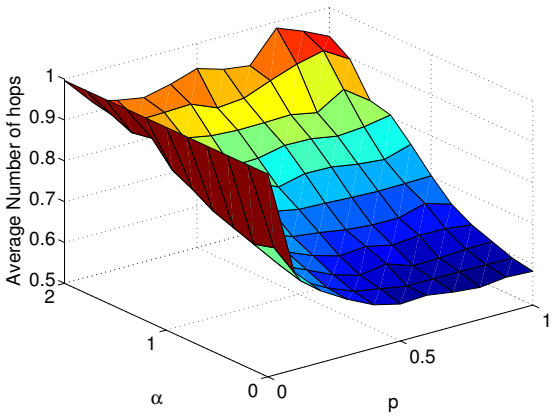
Table 6.2 states a few examples obtained at different values of a. As it can be seen, for $a = 0$, in equation 6.2, the power of the network is favored against the hops. Experimentally, at $a = 0$, the optimal R value obtained is 0. With no additional wires, i.e., $R = 0$, we trace the minimum value from the surface plot back to obtain the α and p as 0. Thus, if complete weightage is given to power, the optimal network obtained is a mesh network. This network proved to dissipate the minimum amount of power of 36.65 mW. However, the average hops for this network is 3.62, which is the highest as compared to any other network. For $a = 1$, the hops of the network is favored and thus the optimal network is a small-world-power-law network with $R = 100$, $\alpha = 0$ and $p = 0.7$. If equal weightage is given to both hops and power, we obtain a network with $R = 10$, $\alpha = 0$ and $p = 1$.

6.3 Experiment 3: Total power, total wire length and average number of hops

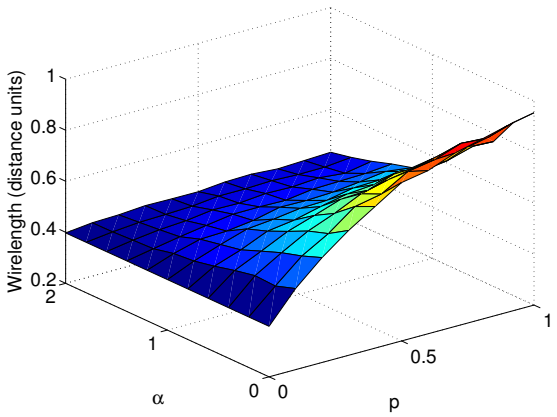
In the previous two experiments we observed that for $a = 0$, the optimal network is a mesh, whereas for $a = 0.5$, the optimal network is either a small-world power-law network or a random network. In this experiment, we consider all three metrics together and find an optimal network. The network considered is a 5×5 2D mesh with varying R. We normalize all the networks, with a reference network which is a 5×5 2D mesh with $R = 100$. Figures 6.7a, 6.7b and 6.7c show the normalized average number of hops, the total wire length and total power plots for a 5×5 2D mesh network with $R = 90$. Here all three performance metrics are considered and hence we use the aggregate objective function given in equation 6.3.

$$y = a \times hops + (1 - a) \times wirelength + (1 - a)(1 - b) \times power \qquad (6.3)$$

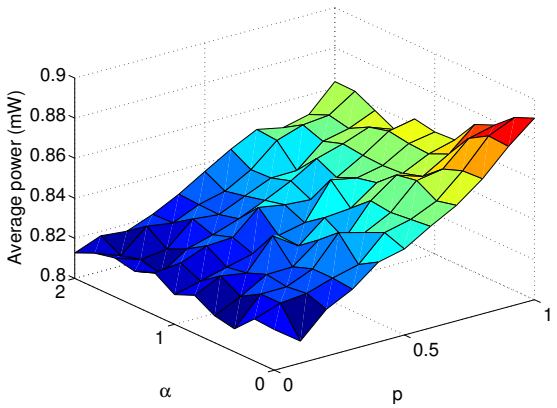
The weight factors a and b lets us give importance to either of the three performance metrics. Figure 6.8 shows an example of the combined plot obtained from the aggregate objective function given in equation 6.3 for a 5×5 2D mesh network with $R = 90$ and $a = 0.5$ and $b = 0$. Thus, equal importance is given to all metrics.



(a) Average number of hops



(b) Total wire length



(c) Total power

Figure 6.7: Average number of hops, total wire length and total power as a function of α and p for a network with $R = 90$.

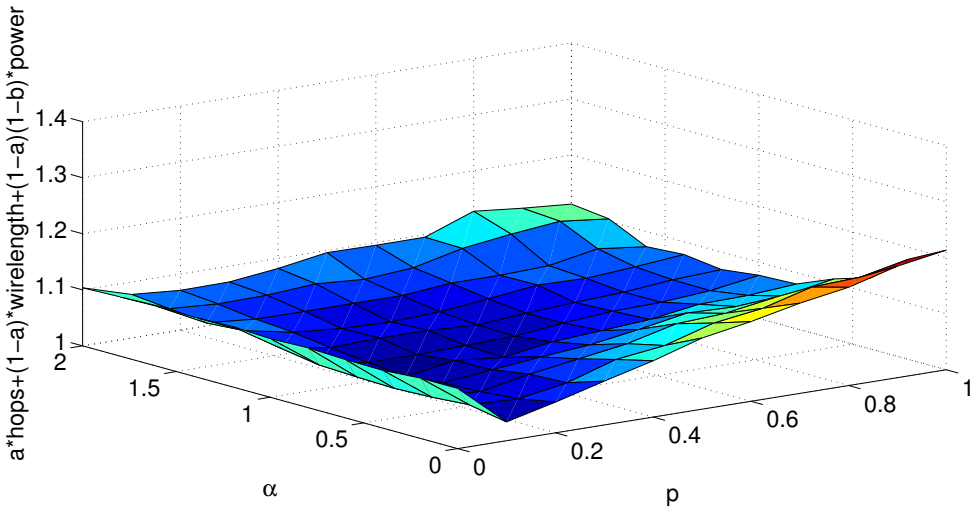


Figure 6.8: Aggregate objective function y. 5×5 2D mesh network with $R = 90$ and $a = 0.5$ and $b = 0$.

With different combinations of α, p and R, a total of 1331 combined surface plots are obtained. Each of these combined plot has a *sweet-spot*, as explained in experiment 1. Figure 6.9 shows a plot with the *sweet-spot* in all the networks for different a and b values and $R = 90$. At $a = 0$ and $b = 0$, the importance is given to wire length and power where as for $a = 1$ and any value of b, the importance is given to the average number of hops only.

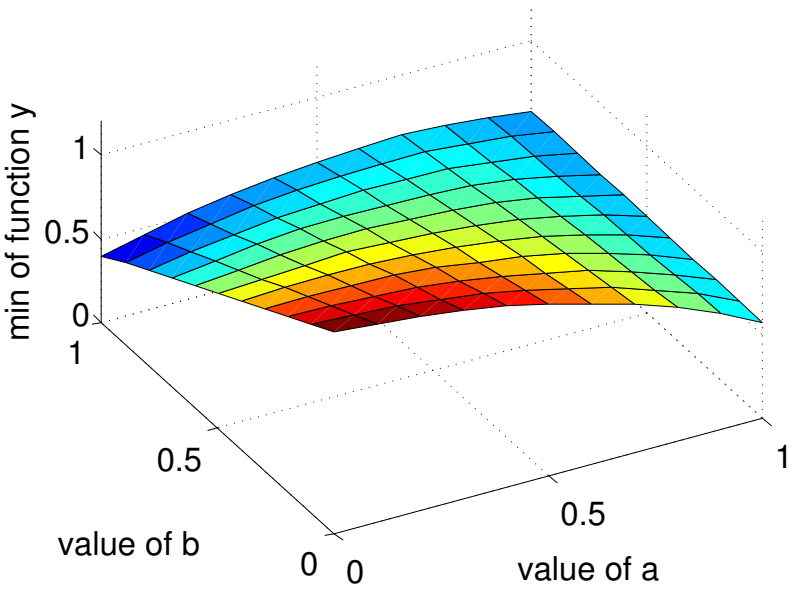


Figure 6.9: Minimum y value as a function of a and b.

Table 6.3 details the optimal network parameters for $a = 0.5$ and $b = 0$. As shown in this table, it is evident that with fixed values of a and b, none of the optimal networks is a mesh.

Table 6.3: The table gives examples of different optimal parameters for $a = 0.5$ and $b = 0$. The optimal network obtained is the 5×5 2D mesh network with $R = 90$, $\alpha = 0.8$ and $p = 0.2$.

R	α	p	Minimum value	Topology
0	0	0.1	0.6481	
10	0.4	0.1	0.6651	
20	0.4	0.1	0.6957	
30	0	0.2	0.7266	
40	1	0.4	0.7729	
50	1	0.3	0.8132	
60	0.8	0.3	0.8654	
70	0.8	0.2	0.9161	
80	0.8	0.2	0.9747	
90	0.8	0.2	0.3977	Small-world power-law
100	0.8	0.3	0.4280	

6.3.1 Summary

Table 6.3 states a few examples of different optimal parameters for $a = 0.5$ and $b = 0$. As it can be seen, the optimal network obtained is the 5×5 2D mesh network with $R = 90$, $\alpha = 0.8$ and $p = 0.2$, which is not a regular mesh network. Hence, from a designer's point-of-view one can look at the combined plots and depending on the design requirement, we can figure out the optimal network. The experiment proved that if equal importance is given to all three performance metrics, the optimal network obtained is not a mesh, but a small-world power-law

network. Table 6.4 shows that with a little cost overhead, and rewiring the network with $p = 0.2$ and $\alpha = 0.8$, we can get a small-world power-law network with better performance than a regular mesh network.

Table 6.4: Comparison of a mesh and the optimal network.

Network	α	p	Average number of hops	Total wire length
Mesh	0	0	3.2653	10.35
Optimal network	0.8	0.2	2.4127	40.88

6.4 Optimal Network with Hot-Spot Traffic

Figure 6.10 shows the average latency and throughput of the optimal network and the regular mesh network under hot-spot traffic with shortest path routing. It is evident that the addition of extra links in the optimal network reduces the latency as compared to the regular mesh network, but the throughput of the optimal network is not as good as the regular mesh network. This is because of added short-cuts in the network result in to uneven distribution of the traffic load.

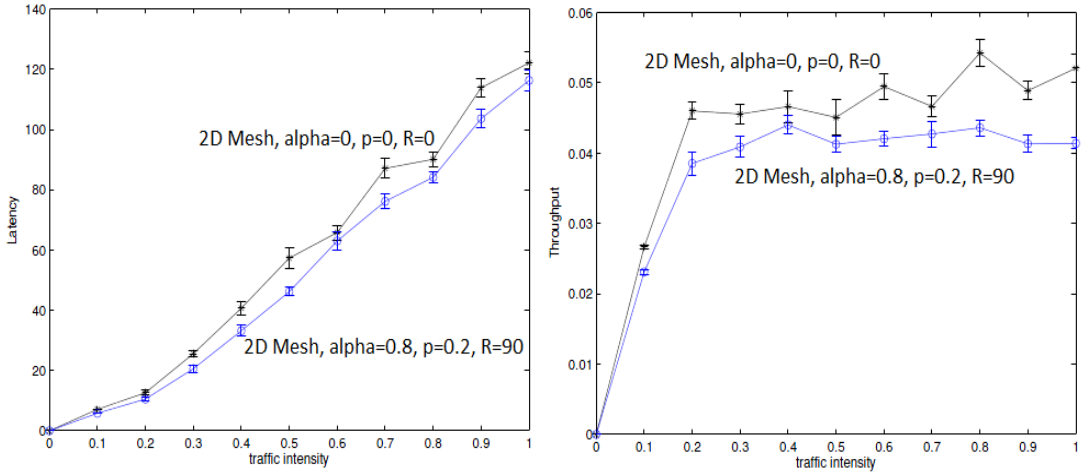


Figure 6.10: Throughput and average latency of mesh and the optimal networks under hot-spot traffic.

DISCUSSION: The latency of the optimal network under hot-spot traffic is better than the regular mesh network because of the additional links in the network, where as the short-cuts worsen the throughput of the network under hot-spot traffic.

6.5 Optimal Network with Different Routing Algorithms

Figure 6.11 show the average latency and throughput of the optimal network under random traffic with different routing algorithms. The figures show that for the optimal network, the shortest path routing has the worst throughput because it does a poor job in balancing the load over the network. The random paths taken by the data in random routing balances the load throughout the network and

increases the throughput, as seen in figure 6.11. The throughput with the ant routing is better than the shortest path routing because once contention is reached in the network, the data takes random paths to reach its destination. In ant routing, the network traffic consist of forward ants, backward ants and the data packets, which adds to the delay in the queues. Ant routing gives a worse latency than the shortest path routing, but is better than the random routing because once the routes have been discovered, the data packets take the shortest path to reach their destination.

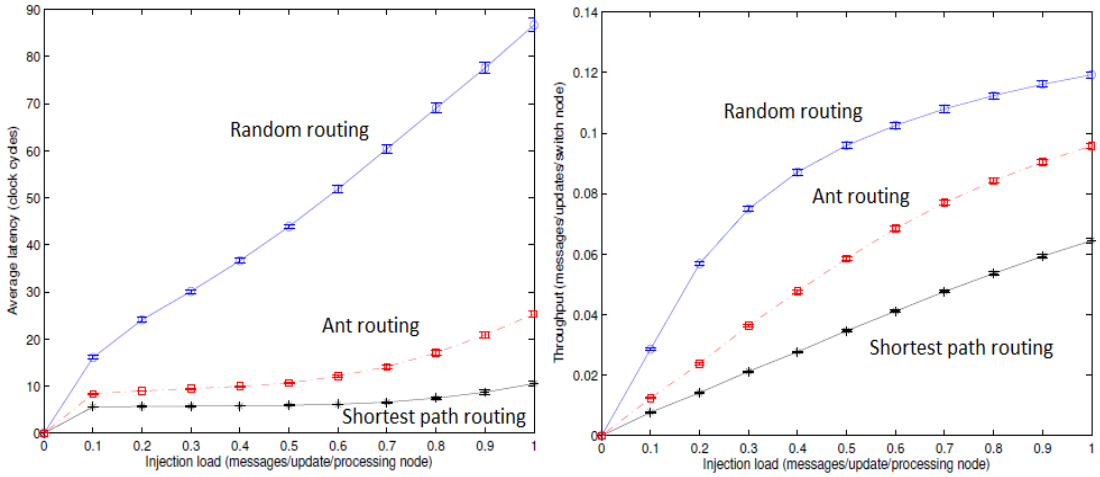


Figure 6.11: Throughput and average latency of the optimal network with different routing algorithms.The network used is 5×5 2D mesh with $\alpha = 0.8$, $p = 0.2$, $R = 90$. The data is averaged over 10 runs.

DISCUSSION: The latency of the optimal network with ant routing is higher than that of the shortest path routing and lower than the random routing. The

throughput of the optimal network with ant routing is better than the shortest-path routing, but not as good as random routing.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We compared the traditional approach of mesh-based interconnects to a new interconnect approach that offers both better performance and lower cost. Choosing an optimal number of links to be rewired with the probability p, a level of non-locality, α, and a total number of wires R, allows a designer to explore parts of the design space that are not accessible with structured mesh-like interconnects. In this thesis, we presented a systematic analysis of the three control parameters. We analyzed different networks in the $\alpha - p$ design space and evaluated different performance metrics. We performed different experiments which proved that the unstructured interconnect topologies offer performance benefits over mesh-like topologies. With given design constraints of $a = 0.5$ and $b = 0$, the optimal network obtained was a 5×5 2D small-world power-law network with $R = 90$, $p = 0.2$ and $\alpha = 0.8$. Compared to a regular mesh network, the optimal small-wold power-law network gave better performance in terms of average number of hops with very little cost overhead. Under hot-spot traffic, the optimal network also showed better latency because of the additional links. We also implemented a simple ant routing technique, which is distributed as opposed to shortest path routing and

which addresses the congestion problem of the network by balancing the load.

7.2 Future Work

We presented a systematic approach to obtain an optimal network according to the given specification in our simulator which is designed in MATLAB. However, an optimization technique is required to obtain these network parameters. There is also a need to develop a mathematical framework to test the network's robustness where the performance against node failures can be assessed. Further, we would like to test the networks under realistic traffic using the Splash-2 benchmark suite. There are other promising alternatives for on-chip communication such as implementing heterogeneous links. Here, the on-chip communication takes place with wired and wireless links. Implementing a routing technique for unknown networks, such as region-based routing is one of the many routing techniques that can be explored. Lastly, there are few open questions regarding the bottom-up self-assembly fabrication techniques, which needs to be addressed in future.

References

[1] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, Jan 2002.

[2] G. Ascia, V. Catania, M. Palesi, and D. Patti. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *IEEE Transactions on Computers*, 57:809–820, 2008.

[3] Semiconductor Industry Association. International technology roadmap for semiconductors(ITRS), 2003.

[4] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, and G. De Micheli. Network-On-Chip Design and Synthesis Outlook. *Integration-The VLSI journal*, 41(3):340–359, 2008. ISSN: 0167-9260.

[5] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35:70–78, 2002.

[6] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Survey*, 38(1):1–51, 2006.

[7] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9(1):317–365, 1998.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, 2001.

[9] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[10] W. J. Dally and B. Towles. Route packets, not wires: on-chip inteconnection networks. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, pages 684–689, New York, NY, USA, 2001. ACM.

[11] S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW (Physics)*. Oxford University Press, USA, March 2003.

[12] V. M. Eguíluz, D. R. Chialvo, G. A. Cecchi, M. Baliki, and A. V. Apkarian. Scale-free brain functional networks. *Physical review letters*, 94(1), January 2005.

[13] J. Flich, A. Mejia, P. Lopez, and J. Duato. Region-based routing: An efficient routing mechanism to tackle unreliable hardware in network on chips. In *NOCS '07: Proceedings of the First International Symposium on Networks-on-Chip*, pages 183–194, Washington, DC, USA, 2007. IEEE Computer Society.

[14] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[15] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, 2001.

[16] J. Hu and R. Marculescu. DyAD: smart routing for networks-on-chip. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 260–263, New York, NY, USA, 2004. ACM.

[17] R. Ferrer i Cancho, C. Janssen, and R. V. Solé. Topology of technology graphs: Small world patterns in electronic circuits. *Phys. Rev. E*, 64(4):046119, September 2001.

[18] C. Izu, J. Miguel-Alonso, and J. A. Gregorio. Evaluation of interconnection network performance under heavy non-uniform loads. In M. Hobbs, A. M. Goscinski, and W. Zhou, editors, *Distributed and Parallel Computing*, volume 3719 of *Lecture Notes in Computer Science*, pages 396–405. Springer Berlin / Heidelberg, 2005.

[19] A. Jantsch and H. Tenhunen, editors. *Networks on chip*. Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[20] R. Liu, W. Guo, X. Zheng, and Y. Tian. On the congestion and shortcut problems of ant-based routing for mobile ad hoc networks. In *Communications, Circuits and Systems*, volume 1, pages 324–328, May 2005.

[21] R. Marculescu and P. Bogdan. The chip is the network: Toward a science of network-on-chip design. *Foundations and Trends in Electronic Design Automation*, 2(4):371–461, 2009.

[22] A. Mejia, J. Flich, J. Duato, S. A. Reinemo, and T. Skeie. Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori. In

Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, pages 84–93, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[23] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[24] L. M. Ni and P. K. Mckinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26:62–76, 1993.

[25] U. Y Ogras and R. Marculescu. "It's a small world after all": NoC performance optimization via long-range link insertion. *IEEE Transactions on VLSI Systems*, 14(7):693–706, 2006.

[26] N. Oshida and S. Ihara. Packet traffic analysis of scale-free networks for large-scale network-on-chip design. *Physical Review E*, 74:026115, 2006.

[27] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. Application specific routing algorithms for networks on chip. *IEEE Transactions on Parallel and Distributed Systems*, 20:316–330, 2009.

[28] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Trans. Comput.*, 54(8):1025–1040, 2005.

[29] L. Peshkin and V. Savova. Reinforcement learning for adaptive routing. In *In Proceedings of the International Joint Conference on Neural Networks*, pages 1825–1830, 2002.

[30] T. Petermann and P. De Los Rios. Physical realizability of small-world networks. *Physical Review E*, 73:026114, 2006.

[31] G. F. Pfister and V. A. Norton. 'Hot Spot' contention and combining in multistage interconnection networks. *IEEE Trans. Comput.*, 34(10):276–281, 1994.

[32] S. Rajagopalan and C. C. Shen. ANSI: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *Journal of Systems Architecture*, 52(8-9):485–504, 2006. Nature-Inspired Applications and Systems.

[33] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie. Analytical modeling of wormhole-routed k-ary n-cubes in the presence of hot-spot traffic. *IEEE Transactions Computers*, 50(7):623–634, 2001.

[34] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker. Autonet: A high-speed, self-configuring local area network using point-to-point links. *IEEE Journal on Selected Areas in Communications*, 9:1318–1315, 1991.

[35] K. M. Sim and W. H. Sun. Multiple ant-colony optimization for network routing. In *CW '02: Proceedings of the First International Symposium on Cyber Worlds (CW'02)*, page 0277, Washington, DC, USA, 2002. IEEE Computer Society.

[36] O. Sporns, D. Chialvo, M. Kaiser, and C. Hilgetag. Organization, development and function of complex brain networks. *Trends in Cognitive Sciences*, 8(9):418–425, 2004.

[37] C. Teuscher. Nature-inspired interconnects for emerging large-scale network-on-chip designs. *Chaos*, 17(2):026106, 2007.

[38] C. Teuscher. Topology-unaware routing in irregular self-assembled networks-on-chip: an explorative case study. In *Nano-Net '07: Proceedings of the 2nd International Conference on Nano-Networks*, pages 1–5, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[39] C. Teuscher, N. Parashar, M. Mote, N. Hergert, and J. Aherne. Wire cost and communication analysis of self-assembled interconnect models for networks-on-chip. In *NoCArc '09: Proceedings of the 2nd International Workshop on Network on Chip Architectures*, pages 83–88. ACM, 2009.

[40] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.