Universidad de Cantabria

Departamento de Ingeniería Informática y Electrónica

Tesis Doctoral

Topologías King como Redes de Interconexión

Esteban Stafford

Dirigida por Ramón Beivide y Jose Luis Bosque

Septiembre 2015

A mis tres mozas,
tres mirlas,
tres musas.

Agradecimientos

Hace años, la idea de hacer una tesis me resultaba inabordable. Ante mis dudas, Ramón me animó diciendo que si fuese a hacerla en el desierto entendería mi preocupación, pero que este no sería el caso en la Universidad de Cantabria. Ahora estoy escribiendo las últimas líneas y veo a qué se refería. Este trabajo no habría sido posible sin las contribuciones de muchas personas. Principalmente Ramón, que desde mis últimos años de estudio me alentó a seguir adelante cuando flaqueaban las ganas, allanó el camino cuando mi brújula apuntaba hacia Alemania y me encontró un hueco para continuar mi carrera en la Universidad, años más tarde. Ahí compartí despacho con un tal Jose Luis, de donde surgió una estrecha amistad y otra fuente de buenos consejos.

A los dos, ahora como directores de esta tesis, quiero agradecerles la dedicación, la paciencia, la profesionalidad y la sabiduría que me han demostrado. Sin duda un ejemplo dificil de seguir.

En mi trayecto por este no-desierto me he sentido arropado por muchos. Mis primeros pasos en teoria de grafos los di de la mano de Carmen. Con Cristóbal y Emilio compartí muchas bromas, scripts y frustraciones de primerizos. Para dudas de navegación marítima, en redes de interconexión y *burrocrática*, Fernando siempre tuvo la puerta abierta. También a Pablo, Borja, Iván y Mariano que avanzan vertiginosamente con sus tesis, siempre encuentran ratos para echarme una mano. Finalmente quiero dar gracias a Enrique, Rafa y al resto del grupo de Arquitectura por aguantar pacientemente mis preguntas y consultas.

El no-desierto de la UC, al igual que Cantabria, se extiende más allá de sus fronteras. En el Barcelona Supercomputing Centre, Miquel y sus compañeros han echado una mano. Y surcando el Ebro a contra corriente ha llegado aliento y consejo de Victor y Darío de la Universidad de Zaragoza.

También quiero resaltar la paciencia del profesorado y mis compañeros del PAS de la Facultad de Ciencias, que durante estos años han tenido que lidiar con un administrador de sistemas distraido por la investigación y la docencia.

Finalmente quisiera dar la gracias a mi familia. A mis padres, que supieron ver en mi el potencial de lo que soy ahora, aunque no siempre les

hiciera caso. Y a Carmen que comparte conmigo las alegrias y frustraciones de esta y otras aventuras.

A todos, muchas gracias.

Resumen

Las redes de interconexion son una parte fundamental de los computadores modernos, independientemente de su marco de aplicación. El diseño de monoprocesadores esta actualmente limitado por varias barreras tecnológicas y físicas. Por ello los diseñadores actuales recurren al paralelismo para satisfacer las demandas de la sociedad actual. Gracias al esfuerzo de investigación que se ha dedicado a las redes de interconexión, el rendimiento de los computadores continúa creciendo. Dependiendo del tamaño de la máquina, una red de interconexión puede ser implementada con cables, ópticos o eléctricos, que conectan los nodos de un equipo multi-rack. O puede consistir en pistas microscópicas dentro de un chip que integra varios procesadores.

Esta tesis presenta una nueva idea para las redes de interconexión futuras. Consiste en una nueva topología y varios algoritmos de enrutamiento. La topología esta basada en las mallas y toros convencionales, a las que agrega enlaces diagonales en dos direcciones. De manera que permiten el movimiento en ocho direciones, como el rey en un tablero de ajedrez. Este hecho es el que les da el nombre de *topologías king*.

El desarrollo de un análisis topológico ha permitido la extracción de un conjunto de expresiones para varios parámetros de redes cuadradas con diagonales. Comparando con las mallas y toros estándar, las topologías king triplican el *throughput*. Esto permite el flujo de mayores volúmenes de tráfico. Además, los enlaces diagonales añadidos tienen el efecto de reducir la distancia media entre nodos. Como consecuencia, la información circula por la red de manera más rápida. Por otro lado, las topologías king tienen unas excelentes propiedades que permiten hacer particionados y distribución perfecta de recursos escasos.

Los algoritmos de enrutamiento descritos en esta tesis aprovechan los recursos topológicos de estas redes para proporcionar un alto rendimiento en diferentes contextos. Hay propuestas para enrutamiento de distancia mínima y no minima, así como un algoritmo de enrutamiento tolerante a fallos. La primera es aconsejable para situaciones en las que se requieran latencias cortas y las condiciones de tráfico no sean excesivamente adversas. En cambio, para situaciones de tráfico adverso, es preferible recurrir al algoritmo de enrutamiento no minimo propuesto. Este da mejores resultados que el

algoritmo de Valiant, ya que reduce la latencia y no requiere un aumento de recursos de almacenamiento.

Las redes king estan particularmente bien dotadas para sistemas grandes en donde se requiere tolerancia a fallos. Esto es debido a la gran cantidad de rutas alternativas que aparecen cuando se relaja la condición de distancia mínima. Estas redes pueden usarse con algoritmos conocidos tolerantes a fallos, pero en esta tesis también se propone uno, llamado KFT, que da un rendimiento similar con un coste inferior.

La posibilidad de poder implementar una malla king en un plano de manera sencilla, hace que éstas se adapten bien a redes dentro del chip. Esta tesis concluye con un análisis de coste y rendimiento que da lugar a una propuesta para una red king integrada en chip, cuyo coste es equivalente al de una malla convencional mientras que mantiene las virtudes de las topologías king.

Universidad de Cantabria

Departamento de Ingeniería Informática y Electrónica

Doctoral Thesis

King Topologies as Interconnection Networks

Cross my mesh and hope to rule

Esteban Stafford

Advisors: Ramón Beivide and Jose Luis Bosque

September 2015

Abstract

Interconnection networks are a fundamental part of modern computers, regardless of their field of application. Increasing the performance of single processors is limited by several physical and technological barriers. Therefore designers rely on parallel hardware to meet the performance demanded by society. It is with the help of extensive research on interconnection networks that the performance of modern computers is continually rising. Depending on the phisical size of the machine, interconnection networks can be implemented with electrcal or optical wires connecting the nodes of a multi-rack computer, or they can consist of microscopic wires whithin a single chip containing several processor cores.

This thesis presents a new offering for future interconnection networks. It consists on a novel topology and several routing algorithms. The topology is based on the well known standard meshes and tori. To these it adds diagonal links in both directions to allow movement in eight directions, like the king on a chessboard. Therefore they are named *king topologies*.

A thorough topological analysis has allowed the formulation of a set of expressions for several parameters of square topologies with diagonals. Compared to standard meshes or tori, king topologies triple the throughput, thus allowing for larger ammounts of traffic to flow through them. Furthermore, the extra links of these topologies also shorten the average distance between nodes. As a consequence, information can be delivered faster. In addition, king topologies present excelent properties that allow perfect partitioning and resource distribution schemes.

The routing algoritms proposed take advantage of the added links to achieve high performance in different contexts. A great effort has been made to find routing algorithms that can satisfy the most applications. There are proposals for minimum distance routing, misrouting and fault-tolerant routing. The first is suitable for applications demanding low latency and benign traffic conditions. In the presence of highly uneven traffic it is advisable to use the proposed misrouting algorithm, as it improves on the classic Valiant approach by considerably reducing the latency and using less resources.

For large systems where, failure-safety is a concern, the king networks are particularly well suited. As they boast a large ammount of alternative

routes, when the minimum distance constraint is removed. Fault-tolerance can be achieved with well known approaches, like Immunet and Immucube, but the thesis also proposes a fault-tolerant routing algorithm, called KFT, with similar performance and less resource requirements.

King meshes are particularly suited to the network-on-chip environment as they can be naturally laid out on a plane. The thesis concludes with a cost-performance analysis that leads to the proposal of a king network-on-chip with a cost equivalent to a standard mesh, but retaining the virtues of king topologies.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Interconnection Networks . . . 3
  - 1.2.1 Network topology . . . 5
  - 1.2.2 Routing . . . 6
  - 1.2.3 Fault-tolerant routing . . . 8
  - 1.2.4 Flow control . . . 9
- 1.3 Objectives . . . 10
- 1.4 Methodology . . . 12
  - 1.4.1 Routing algorithm design tools . . . 12
  - 1.4.2 Cost analysis tools . . . 14
- 1.5 State of the Art . . . 14
  - 1.5.1 Topology . . . 15
  - 1.5.2 Routing . . . 16
  - 1.5.3 Fault tolerance . . . 17
- 1.6 Document Structure . . . 18

2 Topological analysis **19**
- 2.1 Description of the topologies . . . 19
  - 2.1.1 Mesh-like topologies . . . 19
  - 2.1.2 Torus networks . . . 21
- 2.2 Distance related parameters . . . 23
  - 2.2.1 Standard meshes and tori . . . 23
  - 2.2.2 Diagonal Meshes and Tori . . . 24
  - 2.2.3 King Meshes and Tori . . . 26
- 2.3 Bisection bandwidth . . . 28
- 2.4 Path diversity . . . 30
  - 2.4.1 Standard meshes and tori . . . 30
  - 2.4.2 Diagonal networks . . . 30
  - 2.4.3 King networks . . . 32
- 2.5 Graph density and resource placement . . . 34
- 2.6 Physical layout . . . 37

2.6.1 Mesh topologies 38
2.6.2 Toroidal topologies 38
2.7 Concluding Remarks 40

3 Routing Algorithms 41
3.1 Minimal routing 41
3.1.1 Minimal routing in Diagonal networks 42
3.1.2 Minimal routing in King networks 43
3.1.3 Evaluation 49
3.2 Misrouting 56
3.2.1 King$\epsilon\delta$ 56
3.2.2 King$\epsilon\delta$ Implementation 57
3.2.3 Evaluation 59
3.3 Collective 60
3.3.1 Broadcast 63
3.3.2 Multicast 63
3.4 Fault Tolerant Routing 64
3.4.1 Adaptation of known routing algorithms 64
3.4.2 King Fault Tolerance Algorithm 65
3.4.3 Evaluation 71
3.5 Concluding Remarks 76

4 Technological issues in router design 79
4.1 Baseline router description 79
4.2 King network area and energy estimation 81
4.3 Proposals to improve the EPP in king networks 85
4.3.1 Equalising pin count 86
4.3.2 Equalising buffer space 87
4.3.3 Concentration 89
4.4 Concluding remarks 90

5 Conclusions and Future work 93
5.1 Topology 94
5.2 Routing algorithms 95
5.3 Router design 96
5.4 Future work 96
5.5 Publications 97

List of Figures

1.1 Example of (a) bus and (b) crossbar interconnects. 3
1.2 Evolution of (a) latency and (b) throughput with increasing traffic load. 4
1.3 Example of (a) mesh and (b) fat tree interconnects. 6
1.4 Illustration of a deadlock situation. 7

2.1 Example of (a) a mesh, (b) a diagonal mesh and (c) a king mesh. 20
2.2 Example of (a) a torus, (b) a diagonal torus and (c) a king torus. 22
2.3 Example of bisection cuts in various topologies. 29
2.4 Paths leading to different vertices depending on their relative angle. 31
2.5 Equivalence of the path diversity using diagonal links. 31
2.6 Example of path diversity calculation for king topologies. . . . 34
2.7 Average path diversity of standard, diagonal and king tori of different sizes. 35
2.8 Representation of several graphs with the vertices reachable at a given distance k. (a) standard networks, (b) diagonal networks, (c) king networks. 36
2.9 Depiction of KM_{15} partitioned into 25 KM_3 or 9 KM_5 with highlighted center vertices. 37
2.10 Folding of a standard 2D torus network. 39
2.11 Folding of king torus network. For the sake of clarity, the orthogonal links are shown in gray. 39

3.1 Depiction of the number of paths connecting pairs of routers, shown in gray, with different routing algorithms. 45
3.2 Representation of the hop-by-hop vector at every router when destination is (0,0). 48
3.3 Benefits of balanced use of different directions on a KT_{16}. . . . 50
3.4 Maximum throughput comparison for 8×8 networks. 52
3.5 Maximum throughput comparison for side 16×16 networks. . . 53

3.6 Latency behavour in 8×8 meshes and tori. 54
3.7 Performance of 8×8 networks. Execution time is normalized to that of the ideal network. 55
3.8 Representation of throughput vs. table multiplicity in a 32 × 32 king torus. The traffic pattern is uniform and load is 0.6 phits/cycle/router. 59
3.9 Throughput and latency comparison of King$\epsilon\delta$ with Knaive Valiant on a 32 × 32 king torus under various traffic patterns. 61
3.10 Throughput and latency comparison of King$\epsilon\delta$ with Knaive Valiant on a 32 × 32 king torus under various traffic patterns. 62
3.11 Behaviour of routers during broadcast. 63
3.12 KFT example. 66
3.13 Ring created around a complex faulty region (a) before pruning, and (b) after pruning. 69
3.14 Two faulty regions joined by one ring after a fault occurring between routers (1,2) and (2,2). 70
3.15 Throughput of 32 × 32 networks with Immucube. 73
3.16 Throughput and latency of 16 × 16 king meshes with one random fault. 74
3.17 Throughput and latency of 16 × 16 king tori with one random fault. 74
3.18 Throughput and latency of 16 × 16 king meshes topologies with eight random faults. 75
3.19 Throughput and latency of 16×16 king tori with eight random faults. 75
3.20 Saturation point of king topologies versus number of faults. . . 77

4.1 Basic router organization for standard meshes and tori. 80
4.2 Basic king router organization. 82
4.3 Energy per phit comparison between baseline routers. 85
4.4 Energy per phit comparison of mesh baseline router with reduced pin-count king router. 87
4.5 Energy per phit comparison mesh baseline router with reduced buffer king router. 88
4.6 Energy per phit comparison of baseline mesh with king mesh with concentration factor 4. 91

List of Tables

2.1 Distance parameters of different network topologies, where s is the side of the network. . . . 28
2.2 Bisection counts of different network topologies. . . . 30
2.3 Path diversity of diagonal networks. . . . 32

3.1 Minimum latency and maximum throughput of toroidal networks under uniform traffic. . . . 50
3.2 Average path diversity 58

4.1 DSENT model parameters. . . . 82
4.2 Area and static power comparison of mesh and king mesh baseline routers. . . . 83
4.3 Comparison of baseline network configurations during an all-to-all collective. . . . 84
4.4 Area and static power comparison of baseline routers with reduced pin-count king router. . . . 86
4.5 Area and static power of baseline routers and king router with half the buffer space. . . . 88
4.6 Area and static power of baseline meshes with king mesh with concentration factor 2. . . . 89
4.7 Area and static power of baseline meshes with king mesh with concentration factor 4. . . . 90

Chapter 1

Introduction

1.1 Motivation

Since the beginning of computing history computer architects have been trying to create faster and more capable machines. This passionate search for higher processing power was spurred by the need to resolve ever more complicated problems with higher volumes of data. If in those days the pressure came from the military and research institutions [Str06, SSSF13], today it is the whole of society that is hungry for computing power. In fact in the late years, new players are demanding increasing amounts of power. The market of hand-held devices, like phones or tablets, is pressing the engineers for ever more powerful gadgets [Tra07]. The latest phones are already requiring more than one processor core in order to meet market demands. These devices are giving people the chance to access on-line services from anywhere, at any time. Thus, companies are compelled to acquire immense computing resources, or *data centres*, to keep up with the requirements of their clients. These data centres, once conceived as specialised rooms to contain a few different machines, are now thought of as a singular computing facilities composed by thousands of machines [HB09]. In fact our society is relying on information technology so much, that obscene volumes of data are generated. The management of the knowledge buried in these data has given birth to a new industry. What is known as Big Data is presenting computers with new challenges that differ greatly from the number-crunching demands seen in the past [MWBA10, And13].

In early computer history, engineers would strive to make the processing units of their computers as fast as possible [HP11]. In the last decades, the technology developments have allowed computer architects to duplicate the performance of their machines every 24 months, following Moore's Law [Sch97]. Nevertheless, when the speed did not meet their current expectations, they would resort to joining the power of several processors to obtain

a faster computer. Despite the challenges that this brings on to computer architects, parallel machines are very popular nowadays, if not omnipresent. The cause is that although Moore's Law is still valid with the latest technological advances, computer architects are faced with several problems that are hindering the development of faster processors. These problems are often pictured as four walls, although some of them are related. The *frequency wall* dictates the maximum frequency at which a processor can operate [AHKB00]. The performance gap between the memory subsystem and the processor is called the *memory wall* [WM95, Jac09]. The *power wall* restricts energy dissipation on a chip [Goe00]. And the *instruction level parallelism (ILP) wall* is defined by the difficulty of extracting parallelism out of sequential code, at an instruction level of granularity [MGC08]. Despite the different views on the wall metaphor, computer architects are struggling to increase the speed of their processors at the same rate as did in the past. For this reason, the increase in the research and development effort devoted to parallel computers is so notorious in the last years [DS97].

The key idea behind parallel computers is workload sharing. Often a problem can be divided into independent parts that may be solved by separate processors. In these cases parallel computers are able to process a large problem in a fraction of the time required by a single processor. The process of creating parallel applications is complex, as problems do not always let themselves be divided into any number of parts easily [AGK03, HS08]. Most parallel programs require that the processors exchange information as fast as possible throughout the computation. Otherwise, parts of the machine stall in order to wait for data from others more than strictly necessary. Therefore, the *interconnection network* that allows the processors to share data is a fundamental part of any parallel computer.

These networks are implemented at different levels in the parallel machine. From those placed far from the processor to provide computer-to-computer connectivity, usually found in big high-performance computers, and supporting a message passing interface. To others that are an intricate part of the memory hierarchy, allowing a number of processors to share cache or memory banks. These are currently being implemented together with the processors and cache in a single chip, which are known as *Networks-On-Chip (NoC)*. In any of these, the properties of the interconnection network heavily condition the performance of the parallel machine. For instance, the scalability of the network dictates the maximum number of processors that can be connected together.

All these reasons have caused an increase in the research and development effort devoted to interconnection networks in the last years. And have also inspired the pursuit of this thesis.

1.2 Interconnection Networks

Due to their growing popularity, the field of interconnection networks has developed into a large body of knowledge. It shares concepts with communication technology, digital design and computer architecture, as well as including topics particular to itself. For this reason, a brief description of some basic topics of interconnection networks is necessary to set the foundation on which to build the rest of this thesis.

An interconnection network is a programmable system that allows processors to exchange information. It is composed of different parts, like switches, buffers, wires or control modules. To travel through the network, the information needs to be structured as messages, or a sequence of packets of a given size. Then, when a processor sends a message to another, the network system is programmed to perform the necessary actions on the different components to deliver the message at its destination. The way in which the different parts are connected and the way the programming is done conditions the performance of the interconnection network. For instance, bus networks (Fig. 1.1a) would not be able to transfer more than one piece of information at a given time between any two processors, as the wire connecting all of them must be shared. On the other hand, with a crossbar (Fig. 1.1b), all the processors can send messages at the same time, as long as they do not choose the same destination [DT03].

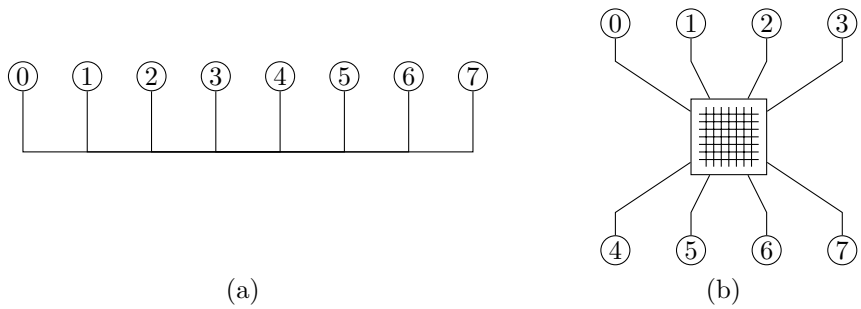


Figure 1.1: Example of (a) bus and (b) crossbar interconnects.

To achieve the best results, the designers must consider the performance requirements of a given problem to be solved, or application, as well as the limitations of the technology in which the interconnection network is to be implemented. With this information, they must choose the right *topology*, *routing* and *flow control* for the network. The routing can be designed with different goals in mind. At the bare minimum, it must be able to guide messages from any source to any destination through minimum distance paths.

Depending on traffic distributions, some areas may be more intensely used than others. In these high traffic areas, or *hot-spots*, messages will be stalled until the necessary resources become available, causing congestion. The development of hot-spots can be reduced with a routing algorithm that sends messages through a longer path, and takes advantage of unused areas of the network. Finally, the design of a routing algorithm might take into account that links can fail over time, and still guarantee that messages reach their destination.

The measure of performance of an interconnection network is tightly coupled to the performance of the parallel machine it belongs to. So better networks will improve the execution time of applications run on the machine. However, predicting this is very complicated and usually only possible through the use of complex simulation environments. For this reason, in the field of interconnection networks, two figures of merit are used instead. First, the delay of the messages, usually referred to by *latency*. And second, the maximum data rate, or *throughput*. These are useful to compare the performance of interconnection networks isolated from the complexity of the parallel machine. Although ideally it would be desired that the latency would remain constant, as the amount of traffic increases, contention occurs rising the latency of messages. When the traffic intensity, also known as *load* is low, it said that the network operates in the *linear zone*. This is characterized by a linear increase of the latency and the throughput, as can be seen in Figure 1.2. As load rises beyond this point, the amount of contention peaks causing an exponential growth of the latency and the throughput to plateau. In general, it is undesirable that networks operate in this *saturation zone*. However, this is inevitable when applications enter intense communication phases, like collective calls. Therefore, networks must perform reliably in both zones.

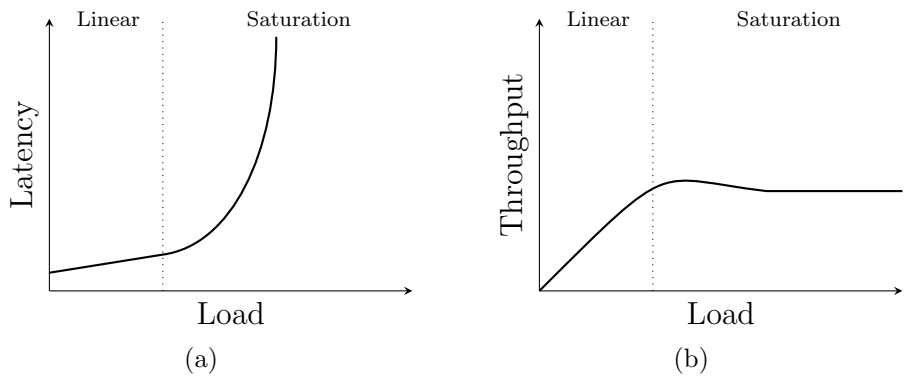


Figure 1.2: Evolution of (a) latency and (b) throughput with increasing traffic load.

1.2.1 Network topology

In general, an interconnection network is composed by a set of routers connected to each other, and to the processors, with communication links in a particular pattern. This pattern is said to be the *topology* of the interconnection network. Designers must take care to adapt their topologies to the characteristics of the technology, as some topologies are more suited for a given environment than others. Notwithstanding, not all topologies might give the performance required by the application. Ideal values for both, the throughput and the latency, can be deduced from the topology using graph theory. Associating graph vertices to the routers of a network, and edges to the communication links, allows a systematic analysis of most topologies.

For instance, the maximum or saturation throughput is obtained by finding the bottleneck of the network. This must be done considering also the traffic pattern, as different patterns might stress different parts of the network. In the context of computer interconnection networks a widely established traffic pattern is *uniform*, where each time a message is sent, the destination is chosen at random with uniform probability. With this pattern the bottleneck is the *network bisection*, which is the smallest set of links that divide the network in two equal parts. Knowing the bandwidth of each of these links, the network bisection bandwidth can be calculated. This is the maximum data rate at which information can flow through the network under uniform traffic.

On the other hand, some applications are highly dependant on short latencies. Considering that a packet takes one cycle to travel from a router to its closest neighbour, the latency of each message, when no contention occurs, is equal to the number of links it must traverse. Then, the average latency is also related to the *average distance* between routers of the network. This can usually be determined analytically with graph methods. Also the maximum latency is bounded by the *diameter* of the network, which is the distance between the two farthest apart routers.

Another consideration to take into account is the number of links leaving or entering a router, which in graph theory this is known as the *degree* of the vertex. In router design forums it is also called *radix*, and has profound implications in the internal design of the router.

The purpose of interconnection networks is to communicate a set of elements, these can be CPU cores, memory modules, computers, file servers, etc. The way these elements are connected to the network separates the parallel machines into two large groups. The first uses *direct networks*, also known as *distributed networks*, where there are equal number of computing elements as routers[1], and each of the former is connected to one of the latter.

[1]This assertion implies that the network does not present concentration, meaning the

Figure 1.3a shows an example of a mesh network, the routers are shown as circles and the computing elements as squares. The routers are interconnected to each other as dictated by the topology. Direct networks can allow a *concentration* factor which states the number of processing elements that share a single router.

The other group employs *indirect networks*, which are characterized by having routers that are not connected to any computing element. Logically, these networks typically arrange the routers in layers. Figure 1.3b shows a fat tree topology with three layers, note that only the first is connected to computing elements. But physically, these networks accumulates the routers of the upper layers in a single spatial point, like a rack, and therefore are also named *concentrated networks*.

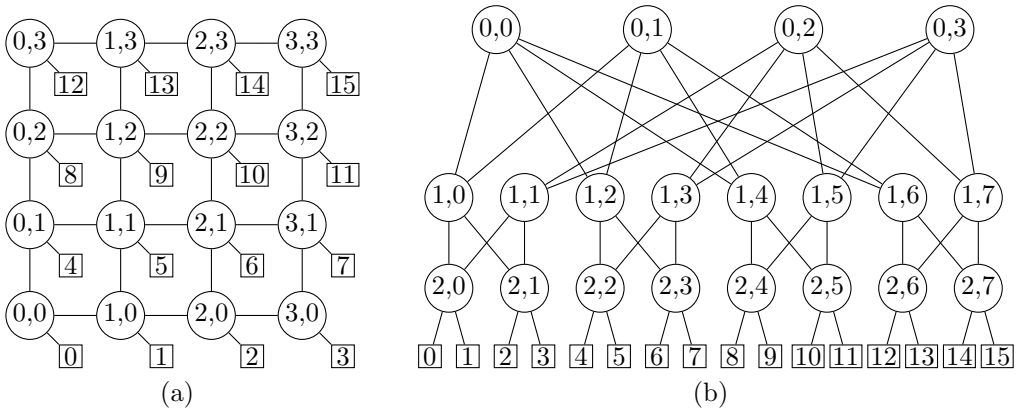


Figure 1.3: Example of (a) mesh and (b) fat tree interconnects.

1.2.2 Routing

Once the topology is set, the next step towards a functioning interconnection network is finding a routing mechanism. This determines the path followed by messages in order to traverse the network from source to destination. It is common to consider only the paths of minimum distance between the routers, thus performing *minimal routing*. Some topologies have a single path connecting any two routers, but usually networks with more than one path are used. In this case, some routing mechanisms always use the same path for each pair of routers, this is known as *deterministic routing* or *static routing*. But a good routing mechanism must balance the use of all the available paths regardless of the traffic pattern. This can be done by using *oblivious routing*, which selects a random path, from the set of minimum distance ones, every time a message is sent. Or it can also be achieved by allowing messages to

concentration factor is 1.

find the most appropriate path as it advances, based on the congestion it encounters, which is then called *adaptive routing* or *dynamic routing*.

Routing might appear simple at first, but the consequences of using a bad mechanism can be worse than just a lack of efficiency. The moment there is more than one message in the network, looped dependencies might appear that will render the network unusable. These so called *deadlocks* are caused by a set of messages that can not advance as each one is in the way of another in a cyclic manner. In Figure 1.4 the four messages are unable to make a left-hand turn because the required link is used by another message. This cycle will cause other messages to stop and these, in turn, will block others until the whole network is flooded. Once a deadlock cycle appears, it can only be broken by removing a message from the network, but this approach is widely considered unacceptable in the context of interconnection networks. Therefore, deadlocks must be prevented instead of resolved, which can be done to a certain extent by an appropriate design of the routing algorithm.

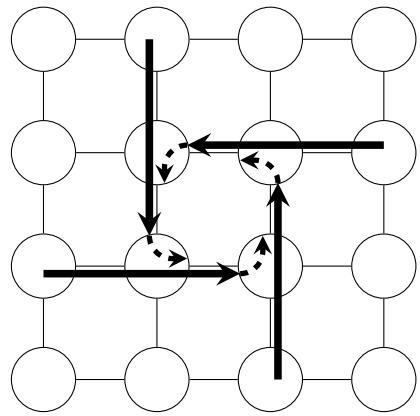


Figure 1.4: Illustration of a deadlock situation.

In contrast to minimal routing, where all messages must use minimum distance paths, *misrouting* permits messages to make detours in order to avoid localised congestion. This has the effect of increasing the latency of the messages in general, but can be beneficial in seriously unbalanced traffic patterns. With misrouting, care must be taken to avoid messages wandering the network indefinitely and never reaching their destination, which is known as *livelock*.

To allow routers to guide messages through the network, these must carry some routing information with them. This information can be their destination, which could allow routers to look up routing tables in order to decide the next hop for the packet. Or it can be a different piece of information. For instance, some systems with grid-like topologies use a *routing record*, which is a vector of integers that specifies the number of jumps the corresponding packet needs to make in each dimension of the grid. The sign of each integer

determines the travel direction. Each time a packet is sent through one dimension, the absolute value of the corresponding component of the routing record is decremented. And, when all of the components are zero, it means that the packet has reached its destination.

1.2.3 Fault-tolerant routing

As in any engineering discipline, interconnection network designers have to face the fact that any system can fail. In fact, the larger the network, the higher the probability of one of its components failing. And in the event of a single failure, the whole network will fail catastrophically because the successful operation of a network relies on all parts being in good working condition. For large systems this is unacceptable, as the *Mean Time Between Failures* (MTBF) could be comparable, or even shorter, than the average application execution time.

Typically, fault-tolerant routing algorithms are designed to cope with one or more link failures. It is accepted that if a whole router fails the application will fail regardless of the capabilities of the routing algorithm. This is because when network loses one of its routers, the connectivity to the corresponding computing nodes is broken, and these become isolated from the rest of the network. Ideally a fault-tolerant routing algorithm will give a performance close to an ordinary routing algorithm when the network is fully operational, and it will allow the least degradation possible in the presence of failures. In addition, the algorithm should require the least extra hardware to perform acceptably when in failure operation.

Despite the above, fault tolerance is typically achieved by *redundancy*. This does not necessarily mean the addition of extra hardware, as networks usually have more than one path to connect any two routers. However, from the routing algorithm's perspective, harnessing this redundancy can be far from trivial. Designers usually apply one of the following two techniques.

If the detection of a failure triggers a reorganization of the network resources, the algorithm relies on *reconfiguration*. This reorganization can mean that the network enters a self-discovery process in which it gathers knowledge of how to avoid faulty links. The drawback of this approach is that during the reconfiguration period, the application traffic is stalled. In contrast, with some topologies, when the routing algorithm detects a faulty link, traffic can be forwarded using a simple set of rules that avoid the failure areas. These are denoted *deflection routing algorithms*.

As with any routing algorithm, great care must be taken to avoid communication anomalies, like deadlock and livelock. But the ways in which these are dealt with in normal routing algorithms can not normally be applied to fault tolerance.

1.2.4 Flow control

Another topic a network designer must deal with is the way messages progress from router to router. The flow control specifies how two fundamental resources of any network are used. The *links*, that transport messages from a router to the next, and the *buffers*, that store the messages within the router until the required link is available. Messages are be divided into more manageable pieces called *packets*, as having arbitrarily long messages traveling through the network can be inefficient. In this way, the network resources are easier to share among the different messages. The packets are further divided into fixed-sized *flits*, which are the smallest data units that the flow control can manipulate. Beyond this, each flit can be actually transmitted as one or more *phits*, which is the amount of information that can be sent in one cycle. Sometimes, and it is the case of this thesis, the flits and phits are the same size.

This structuring of the messages has the advantage of permitting realistic router designs, where the buffer space is known to have room for a number of packets or flits, and the clock cycle can be dimensioned according to the transmission time of a phit. But it also has the effect of increasing the latency. If packets could be transferred between two routers in one cycle, in the absence of contention the average network latency would be equal to the average distance. But as in reality, the packets are sent as a set of flits, their latency is necessarily increased by the number of flits in them. This increase is known as *spooling* or *serialization* latency.

Originally packets would traverse the network one router at a time. With this strategy called *store-and-forward*, a packet can not be sent to the next router until all its phits have been received from the previous one. This implies that the best-case delay of the packet will be proportional to the product of the number of phits and the number of visited routers. Modern interconnection networks usually apply more aggressive flow control methods to improve performance. For instance, *cut-through* flow control allows routers to forward the header phit of the packet as soon as it is received, thus reducing the best-case latency to the sum of packet length and hop count minus one. For a packet to progress from one router to the next, there must be a reservation of buffer space on the receiving router. So that in case of the packet not being able to progress further, it can be stored completely. On contrast, in order to reduce the buffer requirements, *wormhole* flow control makes reservations on a flit basis, instead of packets.

Flow control must manage the links and buffers efficiently, making sure they do not go unused when a packet could advance through it. This happens when a packet is heading towards a highly congested router and can not advance, thus blocking the way for the packets following it that are bound for free parts of the network. Such a phenomenon is known as *Head-*

of-Line(HoL) blocking and can be reduced by using *virtual channels*. By adequately dividing the buffer space associated with a link, several packets can be ready to use it and, if properly organised, packets that can advance are not hindered by others waiting due to congestion. In essence, the division of the buffer space allows more than one stream of packets to use the same physical link in an independent manner, thus the name virtual channels. The reduction of HoL blocking is not the only advantage of virtual channels. They can be used to break cyclic dependencies to avoid deadlocks, or successfully combine different routing mechanisms to benefit from all their virtues. Nevertheless, the use of virtual channels is not without dangers. As a control unit must decide which of the buffers is going to occupy the channel in each cycle, this unit must ensure that a fair chance of success is given to every buffer. Otherwise, the network incurs in *starvation*, and can not guarantee that all packets will eventually reach its destination.

1.3 Objectives

The field of interconnection networks is in constant development. The pressure from the users spur computer architects to devise ever faster machines. This is never an easy task as the designers have to face the reality of technological limitations. In the context of interconnection networks, the size of the routers and the length of the links are constrained by energy and delay issues. Therefore, the choice of topology is of great importance.

In system networks there is a wide variety of topologies with different degree. This is allowed by the high speeds attained by modern signaling cables. However, on-chip networks operate on a tighter energy and delay budget. Furthermore, as they are confined in a two-dimensional substrate, the choice of topologies is much more restricted. Current designs are using low degree topologies, like rings [RH13] or meshes [Mor15], that can easily be laid out on a flat surface minimizing wire length. However, aiming for a latency reduction, some authors propose higher degree solutions, like the flattened butterfly [KDA07], in which routers are arranged in a two-dimensional grid. This solution makes the length of the wires depend on the number of routers, which might lead to scalability issues.

Keeping the idea of a bi-dimensional router arrangement, this thesis offers new topologies that have higher degree than traditional on-chip networks while keeping the length of the links bounded, regardless of the number of nodes. This new family of networks doubles the number of links of standard 2D topologies, like meshes or tori, connecting routers in a diagonal fashion. This permits packets to travel in eight directions, like the king on a chessboard. Thus their name, *king networks*.

The main objective of this thesis, which is to prove that king networks

are a viable substrate on which to base forthcoming *chip multiprocessors (CMP)*, can only be met with a thorough study of the topology from three different points of view. The first takes advantage of graph theory to provide topological facts and figures about the networks, laying the scope of work for the other two. The second allows the proposal of routing algorithms that take advantage of the topological characteristics, and to extract the maximum performance of the networks. From the third point of view, the cost aspects of the network are considered. These include both fabrication and energy consumption during execution. When combining the learnings of the three stages, it can be proved that the king topologies are an interesting competitor in the field of interconnection networks, particularly for NoCs. Not only due to their high performance and efficiency, but also because of their singular characteristics, like scalability and ease of partitioning.

Following is a summary of the individual accomplishments of this thesis.

- Topological analysis
  - A good understanding of the basic properties of king networks was achieved through a theoretical approach, based on graph theory.
  - Precise expressions were found for fundamental topological figures, like diameter, average distance and bisection bandwidth.
  - An evaluation of other interesting features of the topologies was performed, leading to partitioning and folding schemes, as well as an expression for the path diversity.
- Routing algorithm proposals
  - Development of minimal routing algorithms. These take advantage of the topological properties of king networks to reduce the delivery time when used with static routing, and also to improve the throughput in adaptive routing systems.
  - Description of a misrouting algorithm that allows packets a fixed detour, or divergence from the minimal path. This increases the number of possible paths between routers, and gives a better performance when the network encounters highly uneven traffic distributions.
  - A proposal for a fault-tolerant routing algorithm is given, that guides packets toward their destination, even in the presence of one or more faulty links. The algorithm does not disturb the traffic of areas distant to the faults, thus achieving high scalability.
  - As information does not necessarily travel between pairs of nodes, this work also presents specific algorithms for collective communications.

- Cost analysis
  - Basic fabrication and operation cost analysis, based on a modern computational model of interconnection networks. This covered silicon area, as well as static and dynamic energy requirements.
  - Proposal of several strategies to match the cost of king networks to other common topologies while achieving similar performance levels.

1.4 Methodology

To successfully accomplish the above objectives, highly specialized tools have been used, both analytical and computational. For the purpose of establishing the topological advantages and limitations of the king networks, the use of graph theory was fundamental. Based on this large body of work, as well as the constant contributions to the field, it was possible to give precise expressions to most topological features and aspects of the king networks. Also, a great effort has been put in adapting computational tools to the scope of this work. This process is summarised below.

1.4.1 Routing algorithm design tools

Having a good knowledge of the topologies permits designers conceive appropriate routing algorithms. However, although the algorithm might seem correct, it might not work in all possible circumstances. For instance, networks can be easily driven into deadlocks by incorrect routing algorithms. Assessing the correctness of such algorithms from an analytical point of view is seriously complicated, and researches usually revert to software simulation to test them [MMSAZ11].

Interconnection network simulators are built with different degrees of detail. Again, to ensure the highest level of detail, interconnection networks can be simulated with VLSI design tools. However, this approach is extremely time consuming, and only worth the effort for fine-tuning the internal operation of the routers themselves. Slightly more affordable are the cycle-accurate simulators [JBM+13], that have a precise representation of the internal components of the routers, but not as detailed as register transfer or gate level of the VLSI tools. These simulators are adequate to compare different router designs on small sized networks, but it faces serious scalability problems for larger networks.

The last class of simulator is the functional simulator. These further simplify the implementation of the router, to reduce the computing power requirements, and therefore allow focusing on topological aspects of networks

of medium or large size. The findings of this work are based on experiments done with such a functional simulator, called Fsin [NMAPR11]. Although a functioning version of the simulator is available on-line, it was necessary to invest a large number of hours to implement the different networks and routing algorithms shown in this work.

Interconnection network experiments are not only defined by the topology and the details of the routers. These require the definition of a workload that the networks must deal with. The workload is characterized by the traffic that flows through the network. And the traffic can be presented to the simulator with different degrees of precision. Ideally experimentation should be done with the most precise source, which is derived from real execution of applications. For this, the interconnection network simulator must be coupled to a computer-system architecture simulator [BBB+11], that is able of running a real application, and producing communication events which can be processed by the network simulator. The applications used are usually selected from a set of common benchmark suites. Needless to say that this kind of experiment demands high computing power, and can take a long time to conclude.

A less costly approach is *trace driven simulation*. Traces are lists of timed events recorded from the execution of a real application. Feeding this trace to an interconnection network simulator is a reasonable approach when topology considerations are being researched. Apart from the large amount of statistics that the network simulator can give, the most important figure of merit is the number of simulation cycles required to process the trace.

The last traffic source is the synthetic traffic, which is often used in early stages of network developments. This is a set of algorithms that try to emulate real applications. To this aim, they fabricate packets with a given rate, and with a particular spatial distribution. The fact that these simulations are reasonably fast, even with large networks, allows the exploration of broad design spaces, as well as stressing routing algorithms with pathological traffic patterns. In this scenario the simulations are run until a steady state is reached, and then statistics are collected. The most important statistics are the average latency and throughput.

In order to compare the different topologies presented, as well as to test the algorithms devised for them, an iterative process has been adopted. Each novel idea, topology or algorithm had to be implemented in the simulator. These were validated by extensive use of synthetic traffic. Based on the results of the simulations, new ideas were proposed, or alterations were made, in order to satisfy the objectives of this thesis. In the end, some final simulations with traces were performed to further corroborate the findings of this work.

1.4.2 Cost analysis tools

Traditionally, cost analysis of digital systems has been confined to the tools used for VLSI design. Making use of the precise descriptions of the systems, these tools can give accurate cost estimations, both in area and energy. The drawback is, however, that the time and skill required to use such tools is enormous [GRR+08]. Furthermore, once a device is developed, even small changes in the implementation can require a great effort. As a consequence it is not worth using when in an early stage of development where the design scope is still wide open, and numerous alternatives are evaluated.

In recent years several estimation models have appeared that are adapted to single device classes. For instance, there are models that evaluate the cost of processors [LAS+09], and allow comparing different implementations. These models enable researchers to predict the cost of their designs with less effort, and therefore, explore the design scope in shorter time. There are several models for interconnection networks [SCK+12, KLN12], which are available as source code so researchers can tune them to their needs. For this work, the DSENT model was chosen [SCK+12] for its good balance between complexity and precision. Although, some modifications were necessary to include the new topologies.

Once the analysis tool was selected, a typical iterative research procedure was adopted, in which phases of evaluation were interleaved with phases of improvement proposal or tool refinement.

1.5 State of the Art

The main objective of this thesis is to propose a topology that could fill the gap when considering the degree of the topologies used currently in the on-chip network scene. Observing the high-performance computing machines from TOP500 [Str06], it is easy to see that the fastests machines are designed mainly around low or moderate degree topologies. Examples of these are [CEH+11, BWM+12, ASS09], and [KDSA08, FBR+12].

However, technological and scalability constraints within the chip make it difficult to increase the degree of the topology. This has favoured the use of multi-ring networks [RH13] and 2D mesh networks. In recent years, Tilera has proposed a 64 core chip [WGH+07], and Intel presented its Teraflop Research Chip [VHR+08], both based on standard meshes. This trend is continuing today as new players are making also mesh based proposals [Zha15] to compete with the latests designs from Tilera [Gre15] and Intel [Ant13].

Evolving from the current rings, 2D tori have been also considered, [MWLJ15]. On-chip networks with higher degree than traditional 2D meshes or tori have

also been explored [KDA07, BD06]. In addition, multi-level trees are also being considered for forthcoming many-core chips using up to thousands of simple execution engines orchestrated by a much smaller number of control engines, [CAB+13]. Such networks entail the use of long wires in which repeaters and link pipelining are needed. Forthcoming technologies such as on-chip high-speed signaling and optical communications could favor the use of higher degree on-chip networks [KAYC10].

The networks proposed in this thesis are direct, and although their degree is higher than that of meshes, they still preserve a bi-dimensional layout suitable for the on-chip network environment.

1.5.1 Topology

There have been several attempts to improve the characteristics of common 2D topologies by means of considering diagonal links. Without increasing the degree, [TP94] proposed diagonal tori and meshes, and their corresponding routing algorithms, that could be described as rotated versions of the common 2D topologies. Increasing the degree, adding diagonal links in one direction to a 2D torus has been proposed by [SD90, Shi91], the resulting network had an hexagonal shape, in contrast to the square networks proposed here. Square, degree six tori were studied by [NLMA+09], which were tuned to run neural network applications with particular timing constraints. There have also been proposals of degree eight interconnection networks. In [HLB08] the authors present a degree eight mesh-like topology. However, one of the directions has duplicate links, thus being in practice a degree ten network. This duplication permitted deadlock-free, adaptive routing. In contrast, the topologies proposed in this thesis do not require extra links to be deadlock-free. Also, [NSLK14] propose a mesh based topology with extra links allowing diagonal movement of packets. The resulting networks would be similar to the ones proposed in this work, if not for the extra switches that are placed in the crossings of the diagonal links, and the mesh sub-network that joins these switches.

Most of the proposals mentioned above are based on square networks, that is, the routers can be arranged in a square shape if the router separation is constant. However, they are particular cases of larger families of networks, that share the same link connectivity pattern, yet can have a different shape. There are some papers that analyse generalisations of these topologies. Standard meshes and tori are well-known members of a broader family named k-ary n-cubes. Their topological properties applied to interconnection networks were presented in [Dal90] and [Aga91]. Generalisations of degree six and eight graphs have not been considered in the field of interconnection networks. However, some papers from coding theory do

study these topologies, and some of their findings are applicable to interconnection networks. In [FB10], the authors analyse interesting aspects of *Eisenstein-Jacobi* graphs, like distance distributions. These have degree six and the square restriction is removed. Also, [MSB+08] studies the topological features of *diagonal gaussian* graphs. These are a generalisation of king networks, as their connection pattern is equal, and their shape is defined by two attached squares, one of which can have size zero.

From a technological point of view, the proposal of topologies using diagonal links has been considered in the past. In [IML+02], it is argued that if VLSI tools allowed diagonal paths, tighter timing restrictions could be met. To this aim they study the routing of the paths of a RISC processor.

1.5.2 Routing

Since the first machines using standard meshes [SBM62] [BBK+68], the routing algorithms for these topologies have not varied much. Although it is true that generalization and enhancement efforts have been made. When the k-ary n-cubes were proposed in [Dal90], a general routing algorithm for them was presented. For the planar, higher degree networks cited above, each proposal included its own routing. But due to the particularities of the topologies, none were employed in this thesis.

Many interconnection networks have used source routing. Meaning that source routers compute the path packets are going to take. This path is calculated from the source and destination routers [DS86]. Yet some interconnection networks use source routing tables instead [SH94] [BCF+95]. This is usually the case when it is difficult to determine the path arithmetically. However these solutions lack efficiency in large networks, and router-table routing is preferred. This is the case of [Gal96], [Pfi01] and the Internet, where the intermediate routers decide which neighbour is the most convenient destination for the packets they receive.

Deadlock safety is a key issue in interconnection networks, and although some efforts have been made in deadlock detection and resolution [Pin04], the preferred solution is deadlock avoidance. In mesh networks this is achieved by applying *Dimension-ordered Routing* which selects a single path for each pair of routers [DT03]. Other less restrictive methods include the *turn model* [GN92] and *o1turn* [SAL+05]. However for tori, which is composed by rings, an additional mechanism must be applied to avoid deadlocks in the latter. Some authors have used virtual channels to achieve this in wormhole flow control networks [DS86]. While for cut-through flow control [KK79] other authors, based on [Ros87, CO90] developed an injection restriction method called *bubble flow control* [CBGV97], later used in the Bubble Router [PBG+99], and in the BlueGene line of IBM supercomputers [A+02]. Re-

cently, some efforts have been made to implement the bubble routing in wormhole routers [CWP11, MWLJ15, WCP13, CP13].

Deadlock free routing can be very restrictive, and does not allow traffic to take advantage of the topology. For this reason some authors have explored the possibility of adaptive routing. This can be achieved with several virtual channels. One manages traffic in a deadlock-free manner, called static, while the others do not impose restrictions on the packets, called adaptive. Satisfying a set of conditions, packets can change from the adaptive to the static virtual channel if required, thus guaranteeing that the network remains deadlock-free [Dua96].

Interconnection networks usually have to deal with highly uneven traffic patterns that cause local congestion and loss of performance. Some situations can be so severe that minimum distance adaptive routing is ineffective, and some authors propose misrouting as a solution. For instance, in [VB81] the authors balance any traffic pattern by selecting a random router for each packet transmission, and forcing it to visit this intermediate router before proceeding to its final destination. The drawback of such strategies is that they impose a performance penalty when traffic conditions are favourable. For this reason some authors combine minimal routing with misrouting in a dynamic manner, based on traffic observation, like UGAL in [Sin05].

1.5.3 Fault tolerance

As the number of components in modern computers increases, fault-tolerance becomes a key aspect in design. Despite the defects or faults that can appear in an interconnection network, it is likely that it can still be used with an appropriate routing algorithm. The first ideas in fault tolerant interconnection networks appeared in high performance computers at a system level [DA93, CC01]. In these, when a packet encounters a fault on the way to its destination, the routing algorithm tries to find an alternative path. A particularly interesting technique that allows to minimize performance loss without using many extra resources is [GNF+06]. Here, the alternate path is constructed by selecting an intermediate router and sending the packet, first to this router and then to the destination. This algorithm uses at least two virtual channels, but it can require more in order to tolerate more than one simultaneous fault.

In recent years many approaches have been specifically targeted to NoCs. In [RFR+10], the authors propose an efficient routing mechanism to support any irregular topology derived from 2D meshes. Although initially they aim to offer full fault tolerance, the complexity of the routers grows with the number of faults to handle, and a compromise must be reached.

Example of fault-tolerant routing algorithms based on deflection, rather

than reconfiguration, have appeared. For instance, in [SD90], the authors propose an algorithm that allows packets to circumvent faulty areas in hexagonal torus networks. For meshes, [KR09] proposes that when a package finds a fault, it attempts to turn in the orthogonal direction. If not possible, it returns to the previous router where the mechanism is applied again. This is repeated until it succeeds.

Also, in [KMAMP08], a fault-tolerant mechanism that allows routers to be bypassed is described. The routers are designed in a manner that if the crossbar or other parts fail, packets can be forwarded to the next router. These paths, along with a minimal subset of normal network links, form a Hamiltonian ring that provides network-wide connectivity to all processor cores. The mechanism is deadlock-free with only two virtual channels.

From the solutions that can deal with any topology with faults, and are based on reconfiguration, [PGVB04] is able to cope with several faults by finding in the healthy topology a hamiltionian ring, and use it as a safe, deadlock-free subnetwork. A refinement of this algorithm for k-ary n-cubes was presented in [PG07].

1.6 Document Structure

The reminder of this thesis is divided into chapters as follows:

- In Chapter 2 the king networks are defined from a topological point of view, considering different aspects like distance related parameters or path diversity. It analyses each feature comparing it with other related topologies in order to extract conclusions about performance.

- The Chapter 3 is devoted to understanding the implications of routing in performance. Therefore, it presents a set of routing algorithms that allow exploiting the topological features of the king networks in different contexts. Like minimum-distance routing, misrouting or fault-tolerant routing.

- Chapter 4 presents a study of the more technological aspects of router design. Paying particular attention to area footprint and energy consumption, it proposes several alternatives to implement king networks on chip in an efficient configuration.

- Chapter 5 contains a summary of the achievements of this work, and some future lines of work worth pursuing.

Chapter 2

Topological analysis

This work proposes new topologies for interconnection networks. A good starting point in their study is to conduct a thorough topological analysis. As it is fundamental to assess the suitability of the topology as an interconnection network at an early stage. The analysis considers different concepts that determine the overall boundaries of the performance of the network. This chapter summarises the findings and conclusions drawn by the topological analysis.

2.1 Description of the topologies

The main requirement for the topological analysis is formally describing the topologies under study. This section presents such a description of standard 2D topologies, as well as diagonal and king topologies. The combined study of all these will provide interesting knowledge about them.

The best mathematical entity to model interconnection networks is the graph [Har69]. The use of graph theory allows the analytical extraction of general properties of the interconnection networks under study [HL08]. To model these networks, it is straightforward to represent routing elements as vertices, and communication links as edges. In this work only square networks will be considered, as sometimes networks with sides of different length exhibit an unbalanced link usage across dimensions [CMV+10]. The adjective *square* will be assumed for the rest of the thesis. As a consequence, the number of vertices of all the networks considered will be $N = s^2$, for any integer $s > 1$, which represents their side.

2.1.1 Mesh-like topologies

The *standard mesh*, or 2D mesh, of side s will be denoted as M_s. A depiction of this topology is shown in Figure 2.1a. This is a very well-known topology

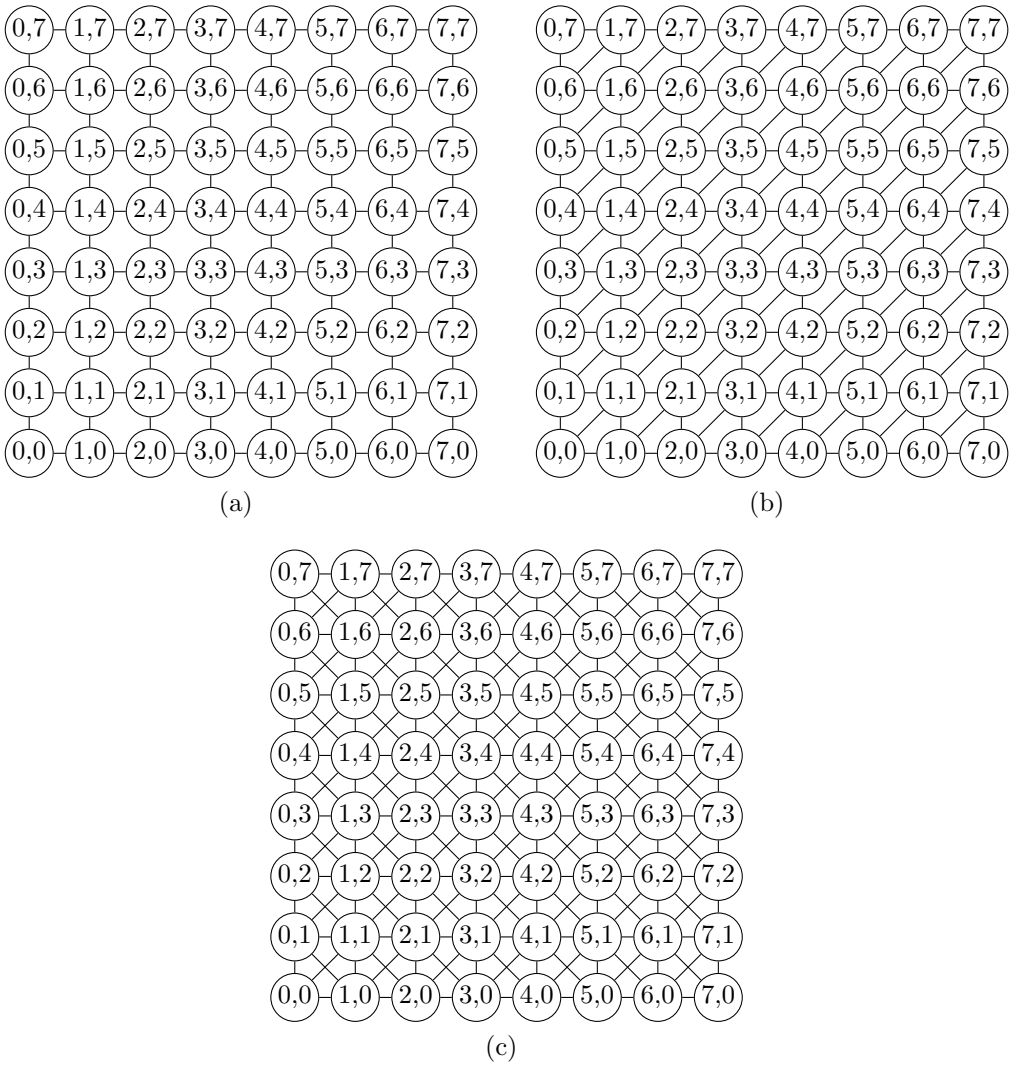


Figure 2.1: Example of (a) a mesh, (b) a diagonal mesh and (c) a king mesh.

which has been deeply studied throughout the years, and continues to be employed in modern on-chip designs (See Section 1.5). In a mesh all the vertices that are not in the periphery have four different neighbours, that is, they have degree four. Similarly, the vertices on the sides have degree three while those at the corners have only two. Since mesh vertices have degrees four, three and two, it is said that its maximum degree is four.

It is well known that a mesh encounters scalability problems with the size of the network [AV94]. To address this, higher degree networks are required. In the context of system networks this has usually been done with 3D meshes. However, this work is focused on planar, or pseudo-planar, networks that can be straightforwardly deployed on flat substrates, like silicon.

A first approach to increase the degree of a mesh without requiring an

extra dimension is to add diagonal links. Initially, these will be added in one direction, resulting in a topology which will be named *diagonal mesh*. Note that there are two possible diagonal meshes, one with north-east links and another with north-west links. In both cases these are denoted as DM_s, since both graphs are isomorphic. An example of a diagonal mesh is shown in Figure 2.1b, where it can be seen that the maximum degree of a DM_s is six.

A mesh-based network of maximum degree eight can be obtained by adding both diagonal links (north-east and north-west). This will be denoted as *king mesh* or KM_s, and is shown in Figure 2.1c. In other words, a KM_s is obtained by adding both sets of diagonal links to a M_s. The next definition gives a formal description of the three topologies.

Definition 1 *Let* $V = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x, y = 0, \ldots, s-1\}$, *be the set of vertices of the three topologies. To complete the definition of the network each topology has a different rule to establish the edges. These rules define which vertices are reachable from a given vertex* (x, y) *by applying jumps to it in different directions:*

- $(x, y) \pm (1, 0), \pm(0, 1)$ *in the* M_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1)$ *(respectively* $\pm(-1, 1)$*) in the* DM_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pm (-1, 1)$ *in the* KM_s.

A condition applies to all rules. The destination vertex must belong to the set of vertices V. *Since some jumps will be invalid, some vertices will have less neighbours than others.*

Note that the jumps represent the coordinate changes that take place when moving through an edge in one of the eight directions or orientations that can occur. Throughout this thesis the four directions of edges are denoted by letters. They can be X or Y to represent the horizontal and vertical directions, Z for the diagonal advancing in the north-east direction, or T for the diagonal advancing in south-east direction. Depending on the way the link is traversed, there can be eight orientations represented by the direction letter followed by a plus (+) or minus (–) sign depending on if it is done forwards or backwards.

2.1.2 Torus networks

The design of interconnection networks is in some ways simplified, if the topologies meet two requirements. The condition of regularity is given to graphs whose vertices all have the same degree. This allows a unified design of the router hardware, as all have the same number of ports. On the

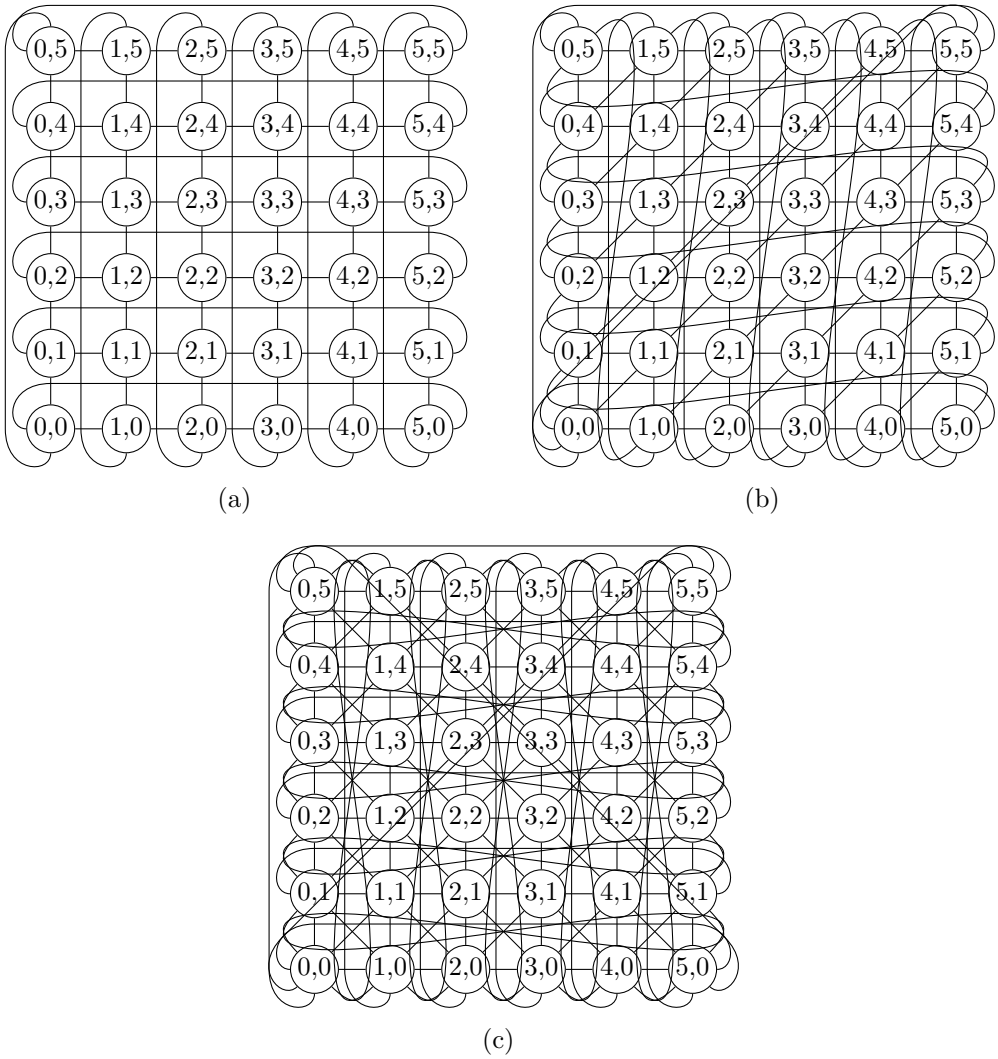


Figure 2.2: Example of (a) a torus, (b) a diagonal torus and (c) a king torus.

other hand, a graph can be vertex-symmetric. An informal definition of this property is that all the vertices of these graphs share the same "view" of the graph. As a consequence, observers situated on a vertex are unable to identify their position based on the appearance of the graph. From a design point of view, vertex symmetry means that the routing algorithms that guide messages through the network can be shared by all vertices, as there is no need to consider particular cases.

Note that all the previous mesh-based networks are neither regular nor vertex-symmetric. A way to, make them satisfy these conditions is to add wrap-around links. These complete the degree of the periphery vertices, so that all have the same number of neighbours. As a consequence each mesh is converted into a torus.

The *standard torus* of side s will be denoted as T_s. Then, DT_s will denote the *diagonal torus* of side s, that is, the degree six graph obtained by adding wrap-around links to the corresponding diagonal mesh. Such diagonal tori can be seen as particular cases of Eisenstein-Jacobi graphs, which were introduced in [MSBG08]. Finally, KT_s will denote the *king torus* network, that is, a king mesh with the wrap-around links required to obtain a degree eight, vertex-symmetric and regular network. Another way to see this network is as a torus with extra diagonal links that turn the degree four torus into an degree eight network. All these topologies are illustrated in Figure 2.2, and a formal description is given in the next definition.

Definition 2 *Let* $V = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x, y = 0, \ldots, s-1\}$ *denote the set of vertices of the three topologies. The definition of the topologies is completed with the rule to determine the links between two vertices. Therefore, any vertex* (x, y) *will be connected to the vertices:*

- $(x, y) \pm (1, 0), \pm(0, 1) \pmod{(s, s)}$ *in the* T_s [1].
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pmod{(s, s)}$ *(respectively* $\pm(-1, 1)$*) in the* DT_s.
- $(x, y) \pm (1, 0), \pm(0, 1), \pm(1, 1) \pm (-1, 1) \pmod{(s, s)}$ *in the* KM_s.

2.2 Distance related parameters

Once the formal definition of the topologies is presented, a study of the distance relationship between vertices can be conducted. This allows determining key aspects of the topology that condition the performance of the interconnection network.

In a simplistic network, with packets of unit length, base latency is proportional to the number of links traversed. Hence, in a first approach, transmission delays in the network can be inferred from distance properties. The maximum packet delay is given by the *diameter* of the graph. The diameter is the length of the minimum-distance path connecting the most distant vertices. The average delay is proportional to the *average distance*, which is computed as the average length of all minimum paths connecting every pair of vertices in the network.

2.2.1 Standard meshes and tori

The diameter and average distance of standard meshes and tori are well-known values, as these topologies have been used in early supercomputers

[1] The notation $\pmod{(s, s)}$ means to take modulo in each component, that is, $(4, 16) \pmod{(8, 8)} = (4 \pmod 8, 16 \pmod 8) = (4, 0)$.

like [BBK+68]. To calculate their distance parameters, the concept of graph product is of great use. In graph theory, it is possible to create complex graphs by multiplying simpler ones. For example, the mesh originates from the Cartesian graph product of two *path* graphs. Just as well, a torus can be obtained by applying the same operation on two *ring* graphs. Furthermore, the distance properties of the graphs obtained through the Cartesian graph product are the sum of those of the operand graphs. The details and properties of different graph products are covered in [IK00].

Due to the above, it is important to know the distance properties of the path and ring graphs. The diameter of a path P_s of s vertices is trivially $k_{P_s} = s-1$. The average distance can be computed by individually averaging the distance of every vertex to every other in the graph.

$$\bar{k}_{P_s} = \frac{1}{s^2}\sum_{j=0}^{s-1}\Big(\sum_{i=0}^{j-1} j-i + \sum_{i=j}^{s-1} i-j\Big) = \frac{1}{3}\Big(s-\frac{1}{s}\Big)$$

With these results, the distance properties of a square mesh M_s can be obtained by adding those of two paths P_s of equal length, in other words:

$$k_{M_s} = 2k_{P_s} = 2s-2 \qquad \bar{k}_{M_s} = 2\bar{k}_{P_s} = \frac{2}{3}\Big(s-\frac{1}{s}\Big)$$

Similarly, the diameter and average distance of a ring R_s of s vertices are:

$$k_{R_s} = \Big\lfloor\frac{s}{2}\Big\rfloor \qquad \bar{k}_{R_s} = \frac{1}{4}\Big(s-\frac{s \pmod 2}{s}\Big)$$

These expressions lead to the corresponding values of a square torus.

$$k_{T_s} = 2k_{R_s} = 2\Big\lfloor\frac{s}{2}\Big\rfloor \qquad \bar{k}_{T_s} = 2\bar{k}_{R_s} = \frac{1}{2}\Big(s-\frac{s \pmod 2}{s}\Big)$$

When the side of the networks is sufficiently large, the above expressions can be approximated for the following:

$$k_{M_s} \approx 2s \qquad k_{T_s} \approx s \qquad \bar{k}_{M_s} \approx \frac{2s}{3} \qquad \bar{k}_{T_s} \approx \frac{s}{2}$$

Note that the values of k_{T_s} and $\bar{k}_{T_s}$ are not approximations when s is even.

2.2.2 Diagonal Meshes and Tori

A convenient way of generating a diagonal mesh or torus through the multiplication of paths of rings has not been found. Therefore, it is necessary to find the diameter and average distance by applying the definitions on the graphs themselves.

The diameter of a graph is defined as the distance between the vertices farthest apart, or the largest distance between any two vertices:

$$D = \max_{0 \leq i,j < N} d(n_i, n_j)$$

Where N is the number of vertices in the graph, and $d(n_i, n_j)$ is the distance between vertices n_i and n_j. Similarly, the average distance of a graph is:

$$\bar{k} = \frac{1}{N} \sum_{j=0}^{N-1} \Big(\frac{1}{N} \sum_{i=0}^{N-1} d(n_i, n_j) \Big) = \frac{1}{N^2} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} d(n_i, n_j)$$

If each vertex n_i is defined by its Cartesian coordinates (n_{i_x}, n_{i_y}), the distance in a diagonal mesh is defined as follows.

$$d_{DM_s}(n_i, n_j) = \begin{cases} max(|n_{i_x} - n_{j_x}|, |n_{i_y} - n_{j_y}|) & \text{if } (n_{i_x} - n_{j_x})(n_{i_y} - n_{j_y}) \geq 0 \\ |n_{i_x} - n_{j_x}| + |n_{i_y} - n_{j_y}| & \text{if } (n_{i_x} - n_{j_x})(n_{i_y} - n_{j_y}) < 0 \end{cases}$$

Note that if n_j is in quadrants 2 and 4 relative to n_i, the distance is the same as in the standard mesh. Whereas if it lies in quadrants 1 or 3, the distance is the same as in a king mesh, as will be shown later. With the aid of a symbolic computation software package it is possible to operate on the diameter and average distance expressions until the following are obtained [MGH+05].

$$k_{DM_s} = 2s \qquad \bar{k}_{DM_s} = \frac{17s^2 - 5s - 8}{30s}$$

For the diagonal torus the process of attaining these values is somewhat simplified, as a formula for the *distance distribution* is available. This is an expression of the number of vertices that can be found at a given distance from a given vertex. The expression applies to any vertex, as diagonal tori are vertex symmetric. The distance distribution of the diagonal torus is obtained by particularizing the one appearing in [FB10]. In that paper the authors presented several properties of the Eisenstein-Jacobi graphs, which are known to be a generalization of the diagonal tori that are being analysed. The Eisenstein-Jacobi graphs are vertex-symmetric tori defined from two parameters a and b, where $a > b \geq 0$. The number of vertices is $N = a^2 + b^2 + ab$. For the particular case of $a = s$ and $b = 0$, a diagonal torus of side s is obtained. Then the distance distribution is:

$$W_{DM_s}(d) = \begin{cases} 1 & \text{if } t = 0 \\ 6d & \text{if } 1 \leq d < T \\ 3s - 3 & \text{if } d = T \\ 18(M - d) & \text{if } T < d < M \\ 2 & \text{if } s \pmod 3 = 0 \text{ and } d = M \\ 0 & \text{if } d > M \end{cases}$$

Where $T = \frac{s}{2}$, $M = \frac{2s}{3}$, and $s \pmod 3 = 0$ means that s is a multiple of 3. In this case the diameter is the maximum distance for which the distribution gives values above zero, that is $k_{DT_s} = \lfloor M \rfloor = \left\lfloor \frac{2s}{3} \right\rfloor$. And the average distance results of averaging the distance distribution:

$$\bar{k}_{DT_s} = \frac{1}{N} \sum_{d=0}^{k_{DT_s}} dW_{DM_s}(d) = \frac{1}{18(s^2 - 1)} \begin{cases} 7s^3 - 3s & \text{if } s \equiv 0 \pmod 3 \\ 7s^3 - 3s - 4 & \text{if } s \equiv 1 \pmod 3 \\ 7s^3 - 3s + 4 & \text{if } s \equiv 2 \pmod 3 \end{cases}$$

For convenience, approximate values of the above expressions can be obtained when a sufficiently large network is considered.

$$k_{DM_s} \approx 2s \qquad k_{DT_s} \approx s \qquad \bar{k}_{DM_s} \approx \frac{17}{30}s \qquad \bar{k}_{DT_s} \approx \frac{7}{18}s$$

2.2.3 King Meshes and Tori

To study king networks, it is useful to note that they can be obtained by using the *strong graph product* on paths and rings. In [IK00], it is stated that the distance in a graph obtained by this product is the maximum of the distances in the operand graphs.

Then, making use of the expressions of the distance parameters of paths, it is straightforward to obtain the same for king meshes. For instance, the diameter is:

$$k_{KM_s} = \max(k_{P_s}, k_{P_s}) = k_{P_s} = s - 1$$

However, it is not so simple for the average distance. Because of the strong product, it can be said that the distance between vertices in a king mesh is the maximum of the distance in the operand path graphs.

$$d_{KM_s}(a, b) = \max(d_{P_s}(a, b), d_{P_s}(a, b)) = \max(|a_x - b_x|, |a_y - b_y|)$$

After substituting the above in the average distance expression, becomes apparent that it can not be written in terms of the average distance of paths.

$$\bar{k}_{KM_s} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} d_{KM_s}(n_i, n_j) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \max(|n_{i_x} - n_{j_x}|, |n_{i_y} - n_{j_y}|) \neq$$

$$\neq \max(\bar{k}_{P_s}, \bar{k}_{P_s}) = \bar{k}_{P_s}$$

Therefore, to reach a simplified expression, it is required to evaluate the average distance expression directly. Again, with a symbolic computation package the following expression was obtained:

$$\bar{k}_{KM_s} = \frac{7s^2 + 2}{15s}$$

In [MSB+08], the authors presented the *diagonal gaussian graphs*, denoted here as $DG_{a,b}$. These are a family of degree eight, vertex symmetric graphs defined by two parameters a and b. The vertices of these graphs can be arranged in two adjacent squares of sizes a and b. Thus, the number of vertices is $N = a^2 + b^2$. The authors were able to determine the distance distribution, and using it they derived the expressions for the diameter and average distance when $0 \leq a < b$.

$$k_{DG_{a,b}} = \left\lfloor \frac{a+b}{2} \right\rfloor \qquad \bar{k}_{DG_{a,b}} = \begin{cases} \dfrac{2a^3 + b^3 + 3ab^2 - 2a + 2b}{6(N-1)} & \text{if } N \text{ is odd} \\ \dfrac{2a^3 + b^3 + 3ab^2 + a - b}{6(N-1)} & \text{if } N \text{ is even} \end{cases}$$

Knowing that a diagonal gaussian graph can degenerate to a square when the a parameter is zero. Then, the expressions for square king tori can be obtained by substituting $a = 0$ and $b = s$.

$$k_{KT_s} = \left\lfloor \frac{s}{2} \right\rfloor \qquad \bar{k}_{KT_s} = \begin{cases} \dfrac{s^3 + 2s}{6(s^2-1)} & \text{if } s \text{ is odd} \\ \dfrac{s^3 - s}{6(s^2-1)} & \text{if } s \text{ is even} \end{cases}$$

Considering sufficiently large values of s, the expression for the average distance can be approximated by:

$$k_{KM_s} \approx s \qquad k_{KT_s} \approx \left\lfloor \frac{s}{2} \right\rfloor \qquad \bar{k}_{KM_s} \approx \frac{7}{15}s \qquad \bar{k}_{KT_s} \approx \frac{s}{3}$$

For convenience and future reference, the Table 2.1 summarises the approximate values of the diameter and average distance of the six topologies considered in this work. The table shows that the topologies with higher degree have less diameter and average distance.

Network	M_s	DM_s	KM_s	T_s	DT_s	KT_s
Diameter	$2s$	$2s$	s	s	$\lfloor \frac{2s}{3} \rfloor$	$\lfloor \frac{s}{2} \rfloor$
Avg. Distance	$\approx \frac{2}{3}s$	$\approx \frac{17}{30}s$	$\approx \frac{7}{15}s$	$\approx \frac{s}{2}$	$\approx \frac{7}{18}s$	$\approx \frac{s}{3}$

Table 2.1: Distance parameters of different network topologies, where s is the side of the network.

2.3 Bisection bandwidth

In addition to the maximal and average delays or latencies, the most important figure of an interconnection network is its maximum throughput under uniform traffic. This expresses the capacity of the network to send simultaneous packets to random destinations. Its value, under certain conditions, can be inferred from the *bisection bandwidth*. This section presents the bisection bandwidth for the different topologies considered, and draws conclusions around its impact on the maximum throughput under uniform traffic.

For uniform traffic, where routers send packets to random destinations with uniform probability at a constant rate, the throughput is bounded by the network bisection bandwidth. According to [DT03], in networks with homogeneous link bandwidth, as the ones considered here, the bisection bandwidth is proportional to the link count across the smallest cut that divides the network into two equal halves. This value represents an upper bound for the maximum throughput under uniform traffic. It is assumed that the bandwidth of a single link is one phit per cycle. In Table 2.2, bisection values for mesh and torus are shown. As an example, the links that belong to the bisection of a mesh are shown in red in Figure 2.3a. If the mesh has side $s = 8$, then the number of links in the bisection is $B_C = 2s = 16$, this corresponds with the figure as each line represents a bidirectional link.

The bisection bandwidth for diagonal and king meshes can be obtained by counting the number of links that need to be removed to cut the network in half. However, to obtain a general expression is not as simple as with the mesh, as the degree of all the vertices is not equal. Then, a horizontal cut in a king mesh of side $s = 8$ affects s vertical links and $2(s-1)$ diagonals, as can be seen in Figure 2.3b. So the bisection count, considering bidirectional links, is $B_C = 6s - 4$. Similarly, diagonal meshes have a bisection count of $B_C = 4s - 2$.

From a topological perspective, cutting a torus in two separate bodies requires two cuts, as only one turns it into a cylinder. This has the consequence that in a torus network the bisection links are found in two groups,

and therefore torus networks have double as much bisection than their mesh counterparts. In a king torus of side $s = 6$, as the one depicted in Figure 2.3c, each cut affects $3s$ bidirectional links, so the total bisection count is $B_C = 12s$. The figure shows both link groups in red and blue for clarity. The same reasoning can be applied to diagonal tori to obtain that the bisection count for these is $B_C = 8s$.

Table 2.2 summarises these values for all the considered topologies. It is noteworthy that a king network doubles the number of links of its degree four counterpart but has three times the bisection bandwidth.

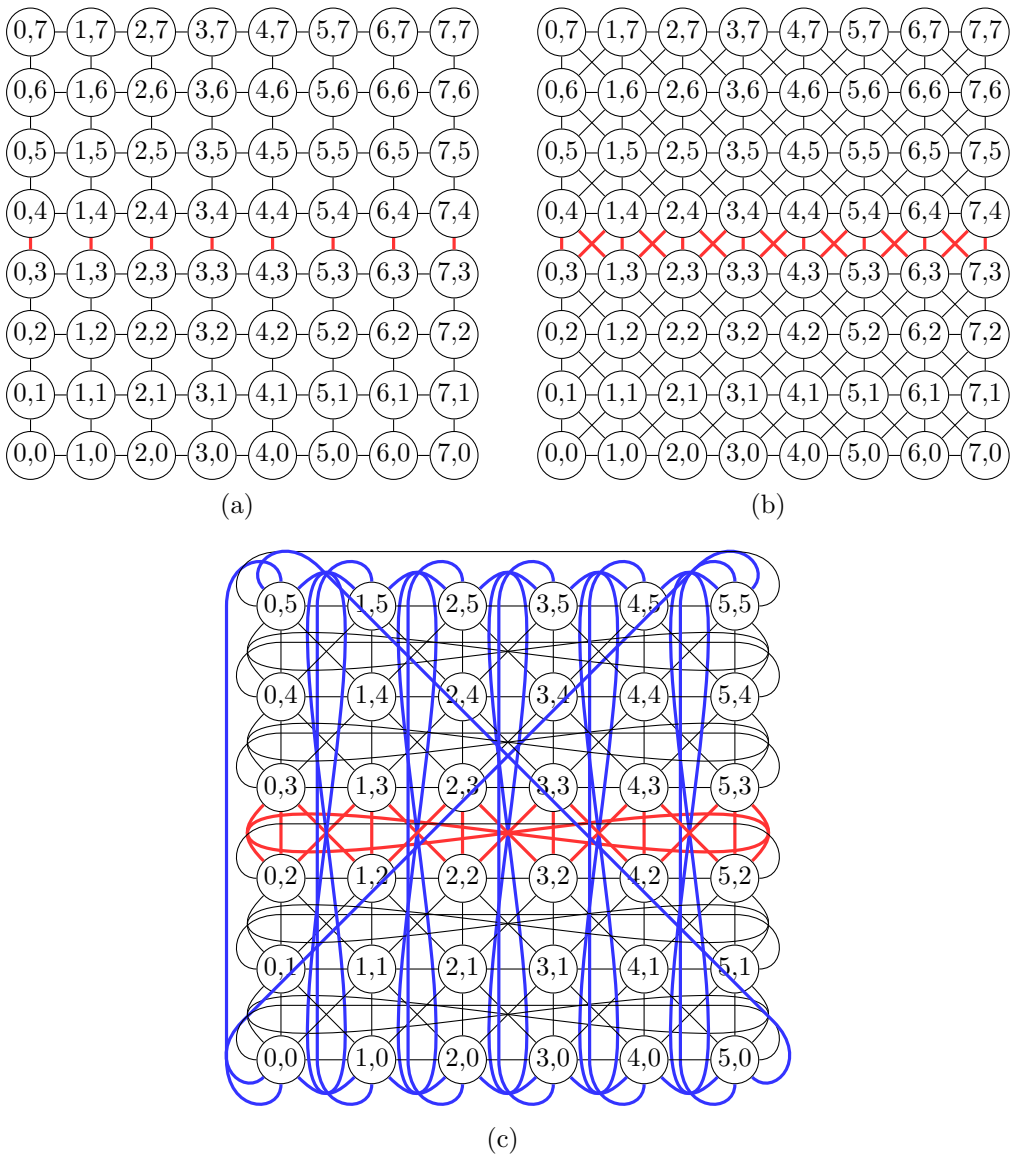


Figure 2.3: Example of bisection cuts in various topologies.

Network	M_s	DM_s	KM_s	T_s	DT_s	KT_s
Bisection Bw.	$2s$	$4s-2$	$6s-4$	$4s$	$8s$	$12s$

Table 2.2: Bisection counts of different network topologies.

2.4 Path diversity

It is known that interconnection networks must manage any kind of traffic, not only uniform. The traffic patterns produced by real applications can vary quickly in time and will stress different areas of the network. These highly congested areas, or hot-spots, constitute bottlenecks that slow the global execution of the application. It is important to estimate the ability of a topology to provide peak performance, even in situations with very uneven traffic. The susceptibility of a network to hot-spots is inversely proportional to its *path diversity*. Which is the number of different paths connecting any two vertices. This section determines this value for the different topologies.

2.4.1 Standard meshes and tori

These topologies have many different minimal paths connecting most pairs of vertices. In fact their path diversity has been studied in the past [DT03], and a closed expression of its value for a given pair of vertices has been obtained. The value of $|R_{ab}|$, the path diversity between vertices a and b, is given in terms of Δ_x and Δ_y, which are the difference of the coordinates of both vertices, i.e. the number of jumps in each dimension separating them.

$$a = (x_0, y_0), \quad b = (x_1, y_1)$$
$$\Delta_x = x_1 - x_0 \quad \Delta_y = y_1 - y_0$$
$$|R_{ab}| = \binom{|\Delta_x| + |\Delta_y|}{|\Delta_x|}$$

This expression counts the number of different ways of choosing $|\Delta_x|$ jumps in paths of length $|\Delta_x|+|\Delta_y|$. It shows that, in order to have $|R_{ab}| > 1$, both $|\Delta_x|$ and $|\Delta_y|$ must be greater than zero. In other words, the vertices can not be in the same row or column. From the expression of $|R_{ab}|$ it can be noticed that the greater the distance between the vertices, the larger the path diversity.

2.4.2 Diagonal networks

The path diversity of diagonal networks is not as straightforward as that for standard meshes and tori. It must be noted that depending on the relative

angle of source and destination vertices, the path diversity can be very different, as illustrated in Figure 2.4. If the destination is in the first quadrant, i.e. $\Delta_x > 0$ and $\Delta_y > 0$, the shortest path uses the diagonal links as well as one of the orthogonals. In contrast, when the destination is in the second quadrant, $\Delta_x < 0$, $\Delta_y > 0$, the diagonal links are of no use. In this last case, the path diversity is the same as for the standard networks. But for the first, the path diversity must be studied in detail.

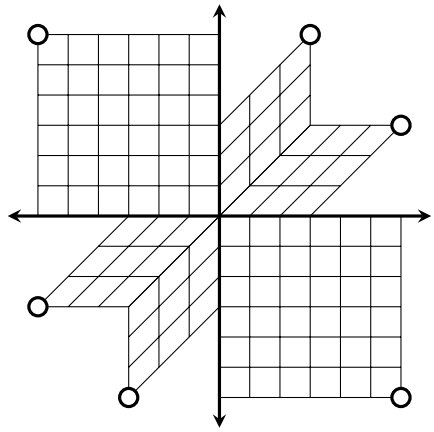


Figure 2.4: Paths leading to different vertices depending on their relative angle.

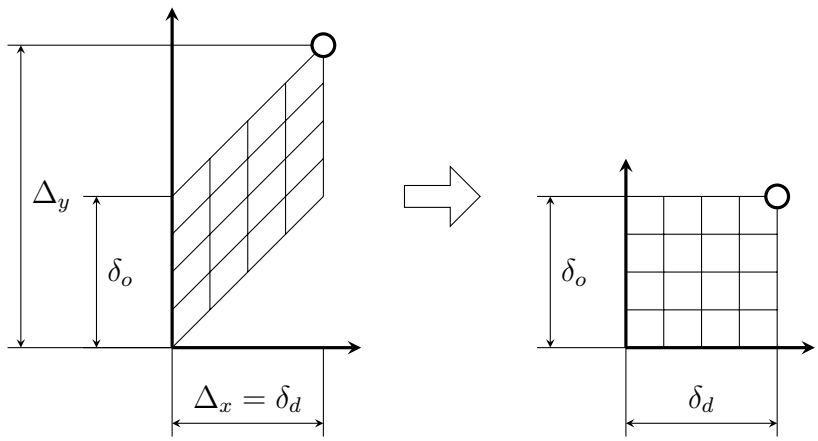


Figure 2.5: Equivalence of the path diversity using diagonal links.

By seeing Figure 2.5 it can be concluded that the path diversity when using diagonal and one sort of orthogonal links can be obtained from the same expression used for the standard networks above.

$$a = (x_0, y_0), \quad b = (x_1, y_1)$$

$$\Delta_x = x_1 - x_0 \quad \Delta_y = y_1 - y_0$$

If $|\Delta_x| < |\Delta_y|$ the number of jumps in the orthogonal δ_o and the diagonal δ_d is as follows:

$$\delta_d = |\Delta_x|, \quad \delta_o = |\Delta_y| - |\Delta_x|$$

$$|R_{ab}| = \binom{\delta_d + \delta_o}{\delta_d} = \binom{|\Delta_y|}{|\Delta_y| - |\Delta_x|}$$

Similarly, if $|\Delta_x| > |\Delta_y|$:

$$\delta_o = |\Delta_x| - |\Delta_y|, \quad \delta_d = |\Delta_y|$$

$$|R_{ab}| = \binom{\delta_o + \delta_d}{\delta_o} = \binom{|\Delta_x|}{|\Delta_x| - |\Delta_y|}$$

These two expressions can be combined in one that gives the path diversity for any destination in quadrants 1 and 3, where Δ_x and Δ_y have the same sign.

$$|R_{ab}| = \binom{max(|\Delta_x|, |\Delta_y|)}{\big||\Delta_y| - |\Delta_x|\big|}$$

Table 2.3 summarises the expressions for any destination, as well as the path length in each case. It reveals that, in quadrants 1 and 3, where the diagonals are useful, the path length is shorter, but the path diversity is also reduced.

Quadrant	Path Length	$\lvert R_{ab}\rvert$
1, 3	$max(\lvert\Delta_x\rvert, \lvert\Delta_y\rvert)$	$\binom{max(\lvert\Delta_x\rvert, \lvert\Delta_y\rvert)}{\big\lvert\lvert\Delta_y\rvert - \lvert\Delta_x\rvert\big\rvert}$
2, 4	$\lvert\Delta_x\rvert + \lvert\Delta_y\rvert$	$\binom{\lvert\Delta_x\rvert + \lvert\Delta_y\rvert}{\lvert\Delta_x\rvert}$

Table 2.3: Path diversity of diagonal networks.

2.4.3 King networks

To assess the path diversity in king networks the binomial coefficients are not useful. In general, counting the number of paths that arrive at a given

vertex can be done recursively by adding the paths reaching the previous vertices. In standard networks the previous vertices are either one or two. The binomial coefficients are well suited to perform this calculation, as shows the next recursive formula:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

This expression states that, the number of paths leading to a vertex (n, k) is the sum of those leading to vertices $(n-1, k-1)$ and $(n-1, k)$. In contrast, for king networks the previous vertices of a minimal path can be up to three. For this reason, the *trinomial coefficients* are required [Com74]. These have the following recursive formula:

$$\binom{n}{k}_2 = \binom{n-1}{k-1}_2 + \binom{n-1}{k}_2 + \binom{n-1}{k+1}_2$$

For these coefficients there is also an explicit formula:

$$\binom{n}{k}_2 = \sum_{j=0}^{n} (-1)^j \binom{n}{j} \binom{2n-2j}{n-k-j}$$

With these it is straightforward to calculate the path diversity for any pair of vertices in a king topology. As with the previous topologies, the coordinates of the vertices are subtracted to obtain the number of jumps in each dimension. Then the path diversity is the trinomial coefficient where n is the largest number of jumps and k the smallest.

$$|R_{ab}| = \binom{n}{k}_2 = \binom{max(|\Delta_x|, |\Delta_y|)}{min(|\Delta_x|, |\Delta_y|)}_2$$

Figure 2.6 shows an example for $\Delta_x = 7$ and $\Delta_y = 4$. The Figure shows the source vertex at the left, and in the intermediate vertices, the number of paths reaching them. This illustrates the recursive nature of the trinomial coefficients, and gives an intuition of which destination vertices have more path diversity. The largest will be found at orthogonal directions from the source vertex, whereas vertices in diagonal directions will have no path diversity at all.

This situation is exactly the reverse of the standard 2D networks. They have large path diversity in the diagonals and none in the orthogonals. Due to the large number of links in king networks, it might be thought that they have larger path diversity than the standard networks. However this is not true. To illustrate this fact, the Figure 2.7 shows the average path diversity of tori of the three topologies considered. The average is taken to have a general idea of the path diversity, and avoid comparing it in particular cases that

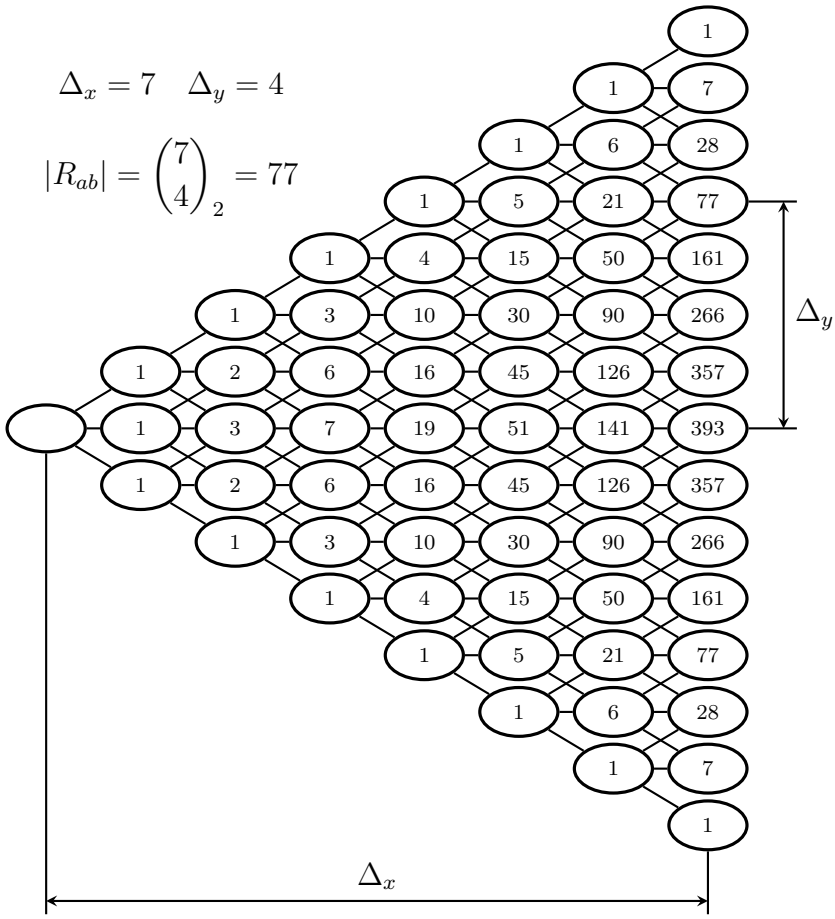


Figure 2.6: Example of path diversity calculation for king topologies.

might benefit one over the others. The graph clearly shows that standard tori have better average path diversity than the rest, and that the difference increases exponentially with the side of the torus. It is also noticeable that the diagonal tori have the least path diversity.

2.5 Graph density and resource placement

In order to build more powerful systems, the tendency nowadays is towards small devices connected by large networks. This allows the creation of systems nearing the exaflop goal [RCG+13], even when processor design is tightly limited by the effect of the different walls (Frequency, memory, energy and ILP). However, the scalability of applications is also limited. Few are the problems that lend themselves to be solved efficiently on any number of cores [San89, GHR95]. Therefore, even though building bigger machines is important, it is also vital to know how to conveniently divide them into

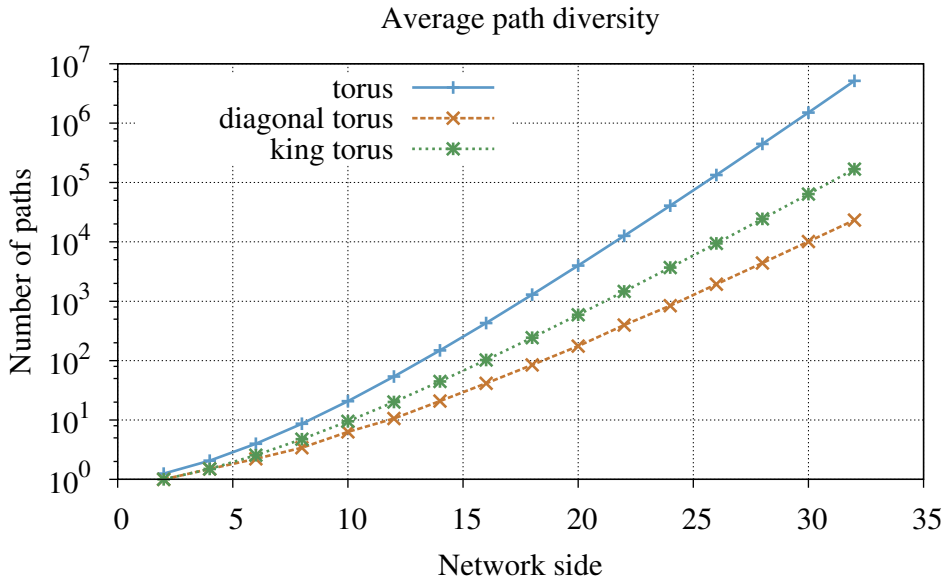
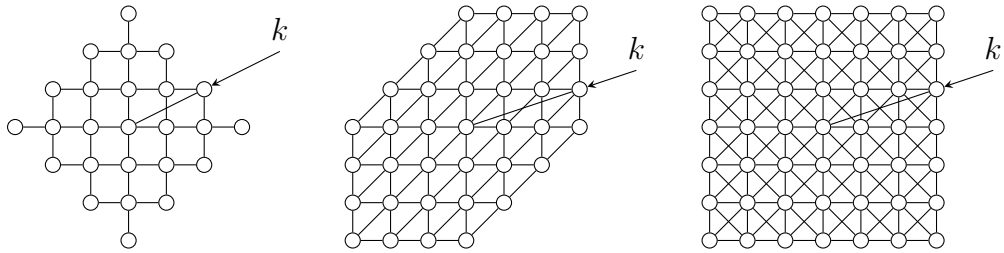


Figure 2.7: Average path diversity of standard, diagonal and king tori of different sizes.

smaller parts. Thus allowing several smaller applications to run efficiently, as in IBM BlueGene machines [ABC+05]. Furthermore, a common problem in large machines is the distribution of scarce hardware resources. These could be hard-disks, memory controllers, power sources, gateway to other networks, etc. In general it is wished that any router has a fair chance to access these resources, so a set of special vertices are chosen such that any vertex of the network can find a special one at a given distance or less. This section gives a solution to the resource placement problem in king networks, and shows that the partitioning scheme induced by it preserves most of the properties of king networks.

However, before studying the resource placement problem on king topologies it is worth considering the idea of graph density. A graph is said to be *dense* when the number of reachable vertices for a given diameter is maximum. This allows a dense graph to deliver a packet to more vertices, in the same time, as one that is not dense. This is synonymous to saying that in dense graphs the vertices are more tightly packed, and therefore reached in less hops. For instance, a torus of side s, and diameter $k = s$ where s is even, consists of just k^2 vertices. An example of this is depicted in Figure 2.8. Notwithstanding, in an infinite standard torus, $4d$ vertices can be found at distance d from any given vertex. By adding the reachable vertices at distances 1 through $\frac{k}{2}$ the expression $2k^2 + 2k + 1$ is obtained, which gives the number of vertices in the dense mesh-based graph of diameter k.

Or, in other words, the ball of radius k in standard 2D networks contains $B_{2D}(k) = 2k^2 + 2k + 1$ vertices. Note that the number of vertices in a square standard 2D torus is less than half $B_{2D}(k)$, meaning that it is far from being dense. A more in-depth study of dense tori can be found in [BMI+03].



$B_{2D}(k) = 2k^2 + 2k + 1$ $B_d(k) = 3k^2 + 3k + 1$ $B_k(k) = 4k^2 + 4k + 1$

(a) (b) (c)

Figure 2.8: Representation of several graphs with the vertices reachable at a given distance k. (a) standard networks, (b) diagonal networks, (c) king networks.

The same conclusion can be reached when considering the square diagonal networks, with degree six. As with a standard torus, a diagonal torus of side s will have diameter $k = s$ and k^2 vertices. Again, considering an infinite hexagonal grid, the maximum number of vertices reachable in k hops is $B_d(k) = 3k^2 + 3k + 1$, which is significantly larger than k^2.

However, in the case of king topologies, all the square tori with odd side s are dense. Note that, in these topologies, at any distance d, $8d$ different vertices can be found. Therefore, for diameter $k = \left\lfloor \frac{s}{2} \right\rfloor$, a king torus has the following number of vertices:

$$B_k(k) = 1 + \sum_{d=1}^{k} 8d = 4k^2 + 4k + 1 = (2k+1)^2 = s^2$$

Which is exactly the number of vertices in the network. Hence, in the case of square king topologies, with odd side, the maximum achievable number of vertices for a given diameter is always attained. The reason behind this is the shape of the ball of the different topologies. For standard and diagonal networks the ball shape is different to that of the network. Whereas the ball in king topologies is square, like the network.

It is important to note that such vertex density is preserved across network partitions for king networks of odd side. For example, consider a king mesh of odd side $s = 15$. Its number of vertices will be $s^2 = 225$ and its diameter will be $k = s = 15$. Such network can be decomposed into 25 (5×5)

equal dense sub-networks with diameter 1, each one composed of 9 vertices. Conversely, the network can also be partitioned into 9 (3×3) equal dense sub-networks with diameter 3, each one composed of 25 vertices. These two partitions are depicted in Figure 2.9.

Such partitions of odd sided networks are especially interesting as the network, as well as any partition, have center vertices that can be considered the center of a ball with radius the sub-graph diameter. Furthermore, finding these center vertices is straightforward, they are evenly distributed across the network, and every vertex is closest to only one of the center vertices. The graphical representation of a partitioned king mesh where the center vertices are highlighted can be seen in Figure 2.9. This property is of great value for hierarchical designs in which the center vertices represent a singular device (memory controller, complex core, network bridge, I/O device, etc.) Current examples of such designs are the Runnemede project from Intel [CAB+13] in which the central vertices would be complex cores executing system code and other CMP designs in which central vertices would be memory controllers, [AEJK+09].

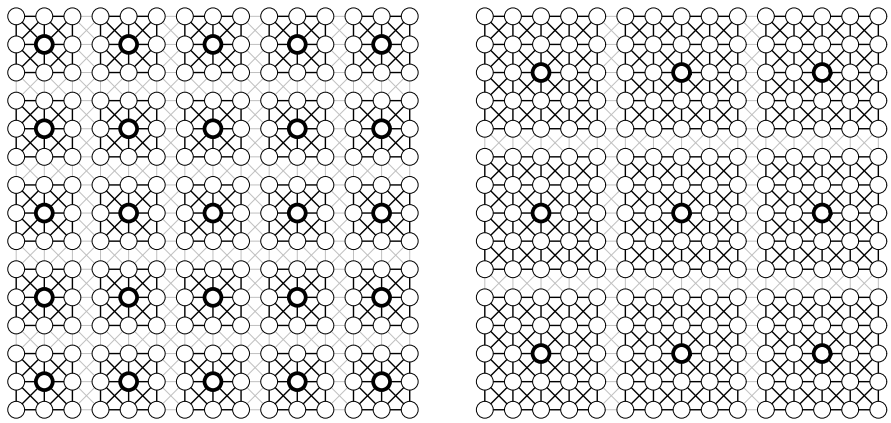


Figure 2.9: Depiction of KM_{15} partitioned into 25 KM_3 or 9 KM_5 with highlighted center vertices.

2.6 Physical layout

The purpose of the topological analysis of an interconnection network goes beyond the assessment of its performance limits. In fact, efficiently deploying an interconnection network is not a simple matter. The topological knowledge of the network is needed when the physical location of the various elements of the network is considered. Regardless of whether the network will be implemented as a system network or an on-chip network, the topology has significant implications in the cost and performance of the final system.

2.6.1 Mesh topologies

When considering the physical implementation of a topology, the ones under study can be divided in two major groups. The first, mesh topologies, are simpler to implement, whereas the second, tori, have an added complication.

System networks are implemented using different technologies depending on their size and, therefore the length of their links. From longer to shorter, these technologies include: optical links between racks, copper wires within each rack and copper strips in backplane PCBs [DT03]. In order to keep the network balanced, it is advisable to assign each dimension of the topology to one technology. For instance, in a multi-rack system the Y dimension of a mesh runs on optical links and the X on the copper wires within each rack. Then all the wires on each dimension have similar delay. Implementing a king mesh in this context means doubling the amount of links. Then, designers must simply allow for the extra cables in the rack and off-rack cable ducting.

At the other end of the size spectrum are the on-chip networks [MHLS14, GD25]. These implement the links between routers with the same metal wires that connect the microscopic devices in any chip. As with PCBs, these wires may not cross each other at the same level. Therefore chip technology allows a number of metal layers in order to satisfy the connectivity requirements of the average integrated device. In contrast to the system networks, all metal layers have similar propagation properties, and from the delay point of view can be considered equal. Ordinary meshes are trivially implemented on a chip as they are drawn on paper. A single metal layer is enough to accommodate the links of different dimensions, and all of them are the same length. In contrast, the diagonal links of king meshes cross each other, then an extra metal layer is required to achieve the connectivity. From a delay perspective, normalizing the length of all links to one of the orthogonals, it appears that the diagonal links will have a length of $\sqrt{2}$.

2.6.2 Toroidal topologies

Toroidal networks present significant advantages over meshes, due to their added wrap-around links. However, they do present a problem when their physical implementation is considered. Especially when this is done in a bidimensional substrate, like tiles on a silicon die, or racks on the floor of a room. If the routers a torus are straightforwardly placed on the substrate, the wrap-around links will have a length proportional to the number of routers on the side of the network. This is undesirable because the latency of these links will be far larger than the rest, breaking the symmetry of the network, and causing unnecessary congestion.

To overcome this problem, a well known technique is *graph folding* [DT03]. For tori, it consists on interspersing the routers of each dimension as shown

in Figure 2.10. This breaks the dependence on the length of the links from the size of the network. However, it has the cost of elongating each link, increasing their delay. For a standard torus, normalizing the length of the links to a non wrap-around one before the folding, the length of the links after turns out to be double, [DT03]. For system networks this implies an extra cost, both in delay and wire length. But for a folded on-chip torus, the links in different dimensions cross, therefore a second layer of metal is required.

Folding a king torus can be done in the same manner, but because of the diagonal links, four metal layers are required. As a consequence of the folding process, the length of the links is between two and $\sqrt{8}$. Figure 2.11 shows a 8×8 folded king torus.

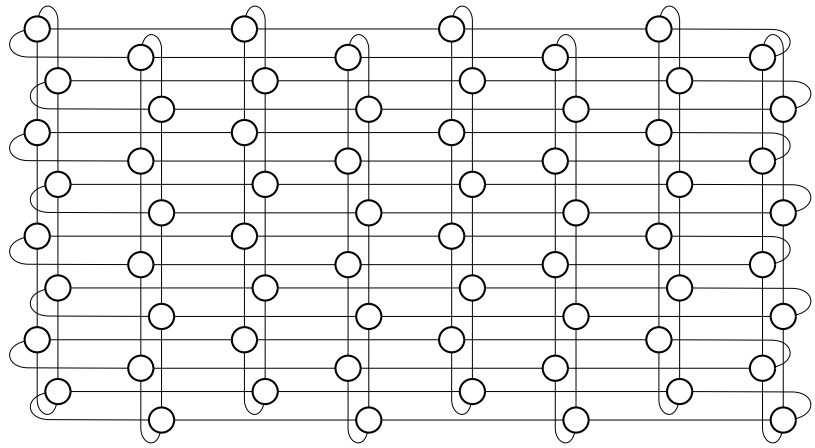


Figure 2.10: Folding of a standard 2D torus network.

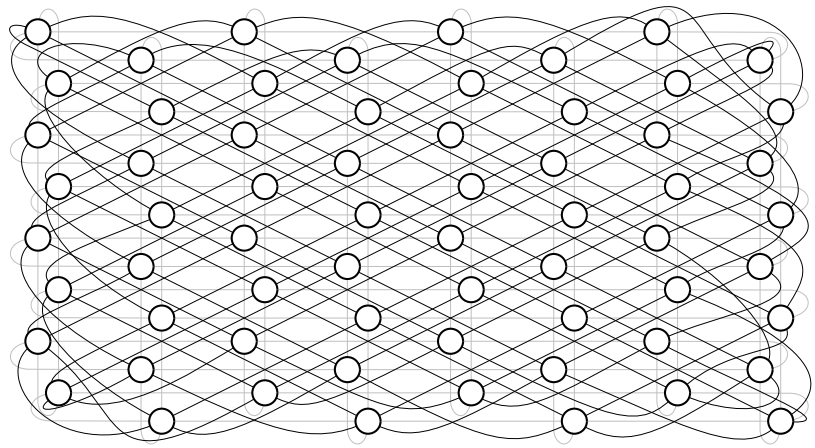


Figure 2.11: Folding of king torus network. For the sake of clarity, the orthogonal links are shown in gray.

2.7 Concluding Remarks

This chapter presented an analysis of different topologies that can be implemented in bi-dimensional substrates. It showed that the extra degree of the king networks confers them better distance properties than the rest. Through the expressions derived for various metrics, it suggests that these topologies will exhibit shorter latencies and greater throughput.

It is worth noting that although a king mesh has twice as many links as a standard torus, they both have the same diameter. Furthermore, the average distance of the king mesh is 7% lower, which promises proportionally shorter latencies at low load, and the bisection bandwidth is 50% higher, consequently allowing it to handle higher traffic rates. And because it does not have wrap-around links, it does not require a folding scheme, like the standard torus.

The chapter also showed that the king networks do not have the best path diversity. This is a direct consequence of them having less average distance and higher density. The added diagonals, that initially were supposed to give better path diversity, in reality reduce the average distance and also the path diversity. As a consequence, it appears that the length of the paths and the path diversity are directly related.

An interesting feature of the king topologies is that they can be easily partitioned in shapes that resemble the original network. Furthermore, for some network sizes, the partitioning can be made into tiles with a center router. Scarce resources, like memory controllers, power sources, or gateway routers, can be placed on these center routers. As a consequence, the resources are uniformly spread throughout the network, meaning that each resource is available to a constant number of nodes, and they can reach it at a given number of hops. This is a feature from which modern CMPs could benefit.

Chapter 3

Routing Algorithms

As a road network without signposts, an interconnection network without routing is difficult to exploit. The routing algorithm is a key component of any network as it guides packets through it. This can be done in an efficient manner, considering alternatives and detours to avoid congestion, or wastefully, by always using the same paths. And even worse, an inadequate routing algorithm will render the network unusable due to deadlock or other anomalous behaviour.

This chapter proposes different routing algorithms aimed at extracting the maximum performance out of king networks. They are presented in growing degree of complexity as more realistic considerations are taken. Starting with minimal routing, and following through misrouting to fault tolerant routing.

3.1 Minimal routing

A first approach to routing packets through networks is to consider minimal routing. Such algorithms impose the restriction that packets should not advance to another router unless it is closer to the destination. They are the typical initial strategy to routing as they are usually the most efficient with low traffic, and they are less complex than others.

The aim of this section is to compare the performance of king networks to that of other similar ones. Therefore, before delving in the matter of finding an acceptable routing algorithm for king networks, this section first presents minimal routing algorithms for the diagonal networks. The baseline topologies for the comparison are the standard 2D networks, meshes and tori, but their minimal routing algorithms have been deeply studied in the last decades [DT03].

3.1.1 Minimal routing in Diagonal networks

To devise a routing scheme for diagonal networks, it is important to notice some differences and similarities with the standard bidimensional networks. Like the latter, diagonal networks have their routers arranged in a square, and each router can be addressed by its Cartesian coordinates. On the other hand, standard networks have vertical and horizontal links corresponding to the two dimensional plane in which they are contained. This contrasts with diagonal networks, which are also contained in a plane, but have the extra diagonal links. Then, it can be said that there are three link orientations but only two spatial dimensions. Which is inconvenient, because so far routing records have equal number of components as the dimension of the topology. To allow packets to use the diagonal links, the routing records have to have an extra component. For the remainder of this work, it is established that routing records must have as many components as the different orientations the links can take.

Once established that the routing record should have three components, a mechanism must be found to derive them from the source and the destination addresses of incoming packets. For standard meshes the routing record (Δ_x, Δ_y) is found by subtracting the source address (S_x, S_y) from the destination (D_x, D_y):

$$(\Delta_x, \Delta_y) = (D_x - S_x, D_y - S_y)$$

This routing record can be used as a basis for determining the three component routing record $(\delta_x, \delta_y, \delta_z)$ required for diagonal networks. Considering that a packet is due to a router in the first or third quadrants, that is Δ_x and Δ_y have the same sign, then each pair of hops in x and y directions can be swapped for a hop in the z direction in the following way:

$$(\delta_x, \delta_y, \delta_z) = \begin{cases} (\Delta_x - \Delta_y, 0, \Delta_y) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_y - \Delta_x, \Delta_x) & |\Delta_x| < |\Delta_y| \end{cases}$$

In any other case, there is no benefit in using the diagonal links. Therefore, when the destination is found in the second and fourth quadrants,

$$(\delta_x, \delta_y, \delta_z) = (\Delta_x, \Delta_y, 0)$$

An interesting fact of this routing approach is that there is no routing record that can have three non-zero components. This is a consequence of the z direction being a linear combination of x and y.

Although this basic algorithm can be useful for diagonal tori, it can not be directly applied. It can be seen that packets bound for the opposite router in a standard torus can have four different minimal routing records. If one path is always chosen over the other three, the network will not be well balanced

under uniform traffic. The same happens for diagonal tori, but in this case the opposite routers can not be trivially found, as with standard tori.

With this in mind, a proper routing mechanism for diagonal tori is shown in Algorithm 1. It computes the length of the four candidate paths connecting source and destination. Then, it finds which one of them is the shortest. In the case of finding more than one, it selects one at random. Once the path is chosen, it is just a question of applying the diagonal mesh algorithm on the chosen one. Experimentation shows that the use of all directions is balanced. That is, under uniform traffic, the average number of packets that traverse the links in a given direction is independent of the direction itself. As with the diagonal mesh routing algorithm, it is also impossible for any routing record to have more than two non-zero components.

3.1.2 Minimal routing in King networks

The search for an acceptable minimal routing algorithm for king networks has given several alternatives, each with their strengths and weaknesses. Contrary to an initial impression, routing on these topologies is not trivial. The fact that the two diagonal links cross each other, makes the topology not strictly planar, and this brings surprising consequences in the routing design. A description of the different minimal routing algorithms considered most interesting is given below.

Knaive

The first approach is heavily based on the diagonal mesh routing algorithm. The king topologies have diagonals in both orientations so, unlike the diagonal networks, there is no need to have substantially different algorithms for king tori and for king meshes. Both of them start by obtaining the routing record for the corresponding bidimensional counterpart (Δ_x, Δ_y). From then on, both algorithms are identical. But as king networks have higher degree than the diagonal networks, another extension of the routing record is necessary. Because the links are set in four different orientations the routing record will have four components $(\delta_x, \delta_y, \delta_z, \delta_t)$. The computation of the different components is similar to the method proposed for diagonal meshes. If the destination is in the first or third quadrants, that is Δ_x and Δ_y have the same sign, an attempt will be made to use orientation z as much as possible:

$$(\delta_x, \delta_y, \delta_z, \delta_t) = \begin{cases} (\Delta_x - \Delta_y, 0, \Delta_y, 0) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_y - \Delta_x, \Delta_x, 0) & |\Delta_x| < |\Delta_y| \end{cases}$$

Whereas if the destination is located in the second or fourth quadrants, Δ_x and Δ_y have different sign, direction z is useless and it is direction t that has to be exploited:

Input: $s = (x_0, y_0)$:source, $d = (x_1, y_1)$: destination, s: side
Output: $r = (r_X, r_Y, r_Z)$: routing record
begin
  $(x, y) \leftarrow d - s$
  if $x < 0$ **then**
    $x \leftarrow x + s$
  if $y < 0$ **then**
    $y \leftarrow y + s$
  $p_0 \leftarrow max(x, y)$
  $p_1 \leftarrow max(s - x, s - y)$
  $p_2 \leftarrow s - x + y$
  $p_3 \leftarrow s - y + x$
  `// Find index of minimum paths`
  $I \leftarrow \{i \in 0..3 / p_i = min(p_0, p_1, p_2, p_3)\}$
  `// Randomly select minimum path`
  $s \leftarrow randomelement(I)$
  if $s = 0$ **then**
    if $x > y$ **then**
      $r_Z \leftarrow y$
      $r_X \leftarrow p_0 - y$
    else
      $r_Z \leftarrow x$
      $r_Y \leftarrow p_0 - x$
  if $s = 1$ **then**
    if $x > y$ **then**
      $r_Z \leftarrow x - s$
      $r_Y \leftarrow s - x - p_1$
    else
      $r_Z \leftarrow y - s$
      $r_X \leftarrow s - y - p_1$
  if $s = 2$ **then**
    $r_X \leftarrow x - s$
    $r_Y \leftarrow y$
  if $s = 3$ **then**
    $r_X \leftarrow x$
    $r_Y \leftarrow y - s$
end

Algorithm 1: Minimal routing algorithm for a diagonal torus of side s.

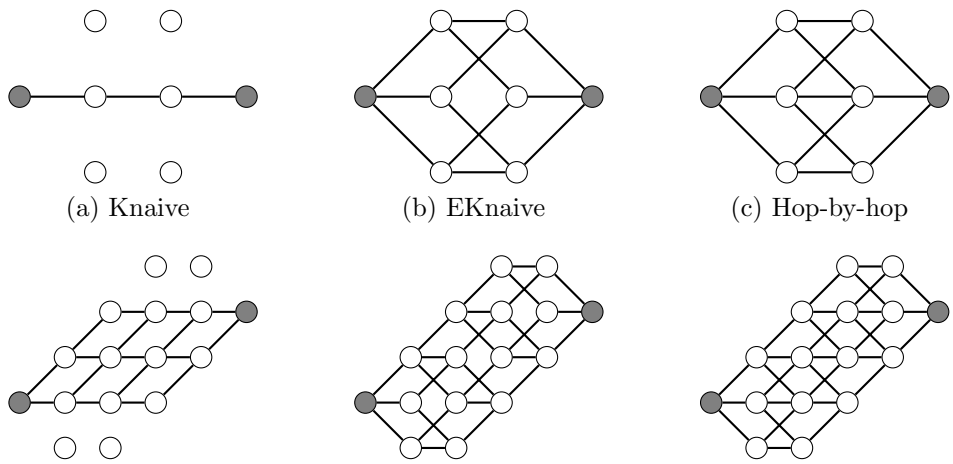
$$(\delta_x, \delta_y, \delta_z, \delta_t) = \begin{cases} (\Delta_x + \Delta_y, 0, 0, \Delta_y) & |\Delta_x| \geq |\Delta_y| \\ (0, \Delta_x + \Delta_y, 0, -\Delta_x) & |\Delta_x| < |\Delta_y| \end{cases}$$

Due to its elegant simplicity and straightforward implementation, this algorithm is denoted *Knaive*. Its main advantage is that, as experimentally

reported later in this chapter, it perfectly balances the use of all directions in uniform traffic.

Unfortunately, network traffic is not always uniform. There are other traffic patterns that direct packets through a reduced set of links, therefore congesting small areas of the network while leaving others unused. These adverse situations cause a diminished performance with Knaive as will be shown later. This effect is caused by the fact that Knaive does not exploit the full path diversity available in the topology.

In Subsection 2.4.3 it was shown that king topologies did not have the best path diversity. But to make matters worse, Knaive never has more than two non-zero components. The path diversity expressed by such routing records is always smaller than that of the king network. Then Knaive allows packets to traverse the network only in a subset of the minimal paths available on the underlying topology. In order to visualise this fact, observe Figure 3.1a. It shows the path followed by packets traveling from a router (0,0) to another (3,0). Computing the routing record is trivial (3,0,0,0), it represents a single path. However the path diversity of the king network between these two routers is 7. So Knaive is ignoring most of the minimal paths connecting these two routers. The same conclusion can be drawn from Figure 3.1d, that shows a router (0,0) and the paths leading to router (5,2). In this case the Knaive routing record (3,0,2,0) also gives only a subset of the minimal paths available in the topology. As a conclusion, it can be said that Knaive would give good performance in uniform traffic, due to the balanced use of directions, yet it would show poor performance in adverse traffic, because it does not use all the path diversity.



(a) Knaive (b) EKnaive (c) Hop-by-hop

(d) Knaive (e) EKnaive (f) Hop-by-hop

Figure 3.1: Depiction of the number of paths connecting pairs of routers, shown in gray, with different routing algorithms.

EKnaive

After considering the shortcomings of Knaive, it is necessary to devise a better routing algorithm. It should make use of more, if not all, the path diversity to show a better behavior in adverse traffic. This proposal is a refinement of Knaive and therefore named Enhanced Knaive, or *EKnaive*.

Observing Figures 3.1a and 3.1d, it becomes apparent that Knaive is restricting the paths to a rhomboid defined by the source and destination routers, with two sides in an orthogonal direction and the other two in a diagonal direction. Notwithstanding, several minimal paths lie outside this rhomboid. In fact, all minimal paths are contained by a rectangle rotated 45 degrees with opposite vertices on the source and destination routers. Thus EKnaive needs to make use of both diagonals in order to increase the path diversity. Note that a pair of orthogonal hops in the same orthogonal direction can be converted to two diagonal hops in different directions. Using this rule, the orthogonal portion of a Knaive routing record can be converted to diagonal hops. As a consequence, the routing record will have at most three non-zero components, one orthogonal and two diagonals. The question remains as to how much of the orthogonal part is transformed to diagonals. By means of an empirical study, it was determined that the best results were given when converting two thirds of the orthogonal component into diagonals.

Of course, EKnaive gives more path diversity, as can be seen in Figures 3.1b and 3.1e, but it also tends to use the diagonal links more than the orthogonals. This breaks the balance of Knaive and justifies a reduced performance on uniform traffic shown later, as forcing an increased use of diagonal links causes them to become the network bottlenecks. In addition, the full path diversity predicted in the topological analysis is not reached. Observe that there are unused orthogonal links in Figures 3.1b and 3.1e. This means that a routing algorithm with better path diversity is still to be found.

However, coming up with a routing record that covers all the path diversity of the king network is impossible. As two directions are linear combinations of the other two, the routing record can not express the variety of paths connecting two routers. Any routing record can be transformed, using the above rule, into others with the same destination and length. But all of them will have at most three non-zero components and none will cover the whole path diversity. Remember that two orthogonal hops can be converted to diagonals, but the converse is not true. Two diagonals in the same direction can not be substituted by any number orthogonals resulting in a path with equal distance. Hence, using routing records with four non-zero components means that misrouting is allowed. A thorough study of this kind of algorithms is shown in Section 3.2.

Hop-by-Hop

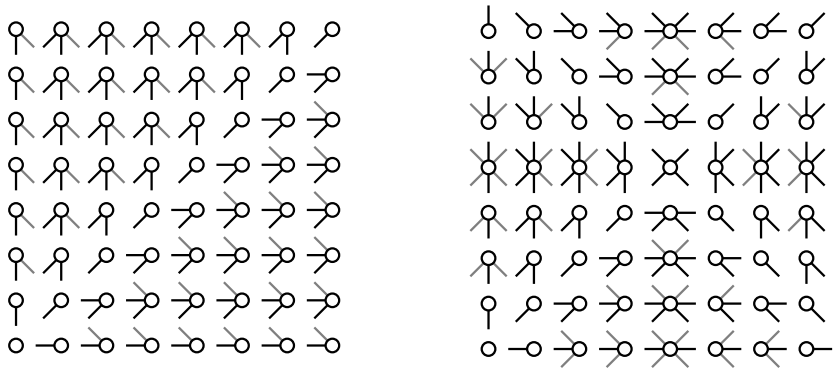
The algorithms presented above are all based on routing records computed at injection time, similar (but not the same) to a source routing mechanism. That is, the decision on which set of paths a packet can follow is predetermined at the source router. This set of paths has been expressed as a routing record, which is a vector holding the number of hops remaining in each direction to reach the destination. But it has also been shown that, because of the linear dependence of the different directions in the king networks, the routing record can not express the full variety of paths connecting any two routers. Then a new approach is going to be adopted in order to exploit the topological richness of these networks.

This new approach abandons the concept of computing routing records at the source and gives freedom to every router on how to direct the packets they receive. This is similar to the way packets are routed across the Internet. When a packet arrives at a router, regardless of it coming from a neighbour or the computing element, the router reads the destination from the packet header. Based on this, it decides to which neighbour it is sent or if it should be consumed. In every hop, the packet always gets closer to its destination, thus keeping the routing minimal. In contrast to the previous routing algorithms, where the packet header carried the routing record, with this approach it bears only the destination address.

In TCP/IP networks each router constructs a table to help it decide on the way to send packets. But, because the topology of king networks is regular, there is no need for this. An algorithm can be derived to calculate the profitable directions for a packet in each hop, knowing the current and destination routers. This algorithm returns a vector with eight integers $(x_+, x_-, y_+, y_-, z_+, z_-, t_+, t_-)$ which are set to one if the corresponding port nears the packet to its destination. With this information the router can apply the adaptive bubble routing mechanism easily [CBGV97, PBG+99]. If the packet is going to be adaptively routed, a port is selected at random from those that are set to one in the vector. Whereas if the packet needs to use the escape channels then the first port set to 1 is selected. This ensures that DOR still governs the escape network.

Figure 3.2 shows the profitable port of each router of a king mesh, and king torus, when they receive a packet bound for router (0,0) on the lower left corner.

In the routing algorithms based on routing records, the number of non-zero components of the routing record has been a concern, as it made a distinction between the naturally balanced Knaive and the increased path diversity of EKnaive. To make a similar analysis with the hop-by-hop algorithm, the vector integers are grouped in pairs, one pair for each orientation. Then it can be seen that, like EKnaive it has at most three non-zero pairs.



(a) King mesh (b) King torus

Figure 3.2: Representation of the hop-by-hop vector at every router when destination is (0,0).

But as the vector is calculated in each router, it does not exclude any path, and the full path diversity is attained. Observe that in Figures 3.1c and 3.1f, all the links leading to the destination are drawn.

Notwithstanding, the benefit of using two non-zero components was that it could give the best results for uniform traffic as it perfectly balances network resources. But this is forsaken in the hop-by-hop algorithm, and it shows a diminished performance under such traffic. However, if the way ports are selected could be prioritised, it could be possible to get the best of both worlds. This new scheme requires that the integers in the vector can take three values. Ports can be assigned 0 if they are not profitable, 1 if they are profitable and are in a direction the Knaive would have chosen, lastly, they can be assigned 2 otherwise. A visualisation of this is shown in Figure 3.2 noting that the black lines would represent ones and the gray twos. Observe that only the diagonal links can have the value two.

Now, the behaviour of the router is slightly different. When a packet arrives, the router checks the adaptive ports that have one in the vector. If all are full, it looks into the adaptive ports marked with two. Should these be also full, the escape port would be used, being the first one in the vector marked with one. In a way, this algorithm makes the decision about which adaptive channel is used in two steps. Therefore it is denoted 2S hop-by-hop. With this strategy, packets tend to be routed with Knaive unless local congestion occurs, point at which it falls back to full path diversity routing. Thus this algorithm can give the best results, both in benign and adverse traffic patterns.

3.1.3 Evaluation

So far, a number of minimal routing algorithms have been proposed to take advantage of the different topologies presented. For king networks in particular, there are several algorithms with different strengths and weaknesses. This subsection describes a set of experiments that corroborate the hypotheses of Chapter 2. Furthermore, it gives an experimental validation of the arguments used in Subsection 3.1.2 to develop the different routing algorithms. Lastly, it shows the performance of the different topologies and algorithms running real applications.

This evaluation is supported by experimentation performed with Fsin [NMAPR11], a functional interconnection network simulator. The architecture of the simulated router corresponds to the Bubble Router, described in [PBG+99], with two virtual channels. In general the simulations are fed with synthetic traffic with typical spatial patterns. Statistics are only taken when the network load reachs a steady state. Theoretically, the higher degree of some topologies allows the throughput to rise above one phit per cycle per router. Therefore each router may have more than one injector, where necessary.

Experiments with synthetic traffic

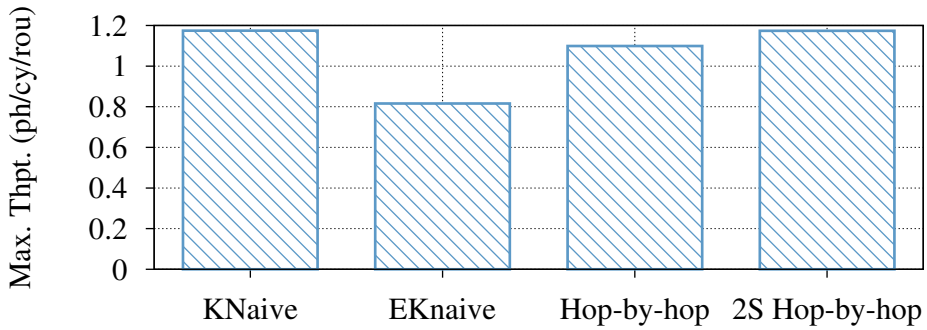
The first experiment tries to validate the simulation infrastructure. This is done by comparing the experimental results with the theoretical expressions presented in Chapter 2. To this aim, Table 3.1 presents various performance metrics corresponding to toroidal topologies of 16×16 under uniform traffic. First, the theoretical average distance and the experimental minimum latency. The latter has been obtained by computing the average packet transmission times when the packets have one phit at minimum load. Under these conditions the latency and the average distance should be the same, and in fact Table 3.1 shows this similarity. On the other hand, the table also presents the maximum throughput, both computed from the theoretical expressions and measured experimentally. To reach the throughput bound of each topology with multi-phit packets, the effect of the *Head-of-Line(HoL)* blocking had to be minimised. Therefore a sufficient amount of virtual channels and injectors was used. It can be seen that the experimental results are close to the theoretical limits, however they are not reached due to the effect of the router architecture.

Throughout this chapter attention has been called on the way the different routing algorithms use the links in different directions under uniform traffic. It has also been pointed out that a balanced use could show better performance. Figure 3.3a shows the throughput of different algorithms running on king tori, and the average use of the different directions in Fig-

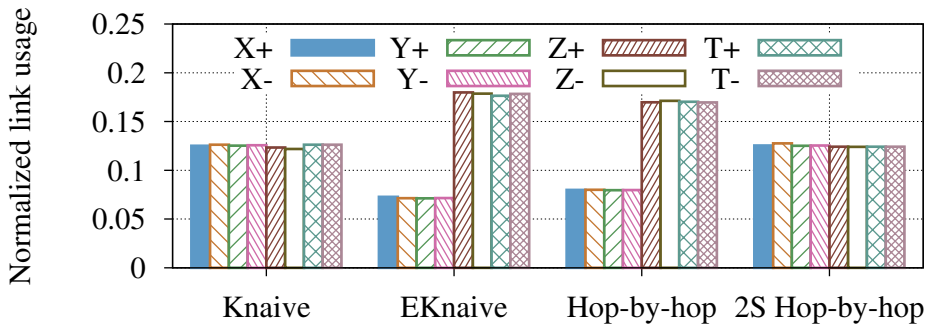
Network	T_{16}	DT_{16}	KT_{16}
Average distance	8	6.22	5.33
Minimum latency	8.13	6.34	5.48
Max. Throughput (Th)	0.5	1.0	1.5
Max. Throughput (Ex)	0.45	0.96	1.49

Table 3.1: Minimum latency and maximum throughput of toroidal networks under uniform traffic.

ure 3.3b. This experiment was made with a KT_{16}, but results are similar with other sizes. It is noticeable that the two algorithms that have an equal use of all channels give better results than the others. Therefore, for the remainder of this Section, king networks will be used only with Knaive and 2S Hop-by-hop algorithms.



(a) Maximum throughput of the different routing algorithms.



(b) Average use of links in different directions.

Figure 3.3: Benefits of balanced use of different directions on a KT_{16}.

Comparing the performance of different topologies is best done by observ-

ing the saturation throughput and the evolution of the average latency, as input stimuli are increased from minimum to medium traffic load. Figures 3.4 and 3.5 show the average saturation throughput of the different topologies, network sizes, routing methods and traffic patterns.

As it has been mentioned in Subsection 3.1.2, the Knaive algorithm perfectly balances the use of all orientations, in uniform traffic, but does not use the full path diversity available on the network. Notwithstanding, it obtains the best results because the combination of traffic in different directions makes an even use of all the network's links. On the other hand, the 2S hop-by-hop algorithm can use the full path diversity, but in a way that is not detrimental to the direction balance, if the traffic pattern is uniform. Therefore its performance is equivalent to that of Knaive.

In general, it can be seen that increasing the number of diagonals gives better results than the standard 2D networks. The best overall performance is given by the 2S Hop-by-hop routing algorithm that is able to exploit the full path diversity of the king networks as can be seen in highly adverse traffic patterns like tornado or shuffle. It is noteworthy that the performance of networks with only one diagonal depends strongly on the mapping of the application. As when the main direction of the traffic matches the added diagonal, a better performance is observed. A good example of this behaviour is the transpose traffic on mesh networks or the tornado traffic. In these situations there is one diagonal network that reaches the same performance as a king network, while the other hardly improves on the standard network result.

On the other hand, an analysis of the average delay of packets can reveal interesting facts about the networks presented. Figure 3.6 shows these results for 8×8 meshes and tori. Those for 16×16 lead to the same conclusions and therefore have been omitted.

Observing the results for uniform traffic, the base latencies for small loads are slightly above the average distance of the topology plus the spooling latency, due to the packet length. This further validates the expressions presented in Chapter 2 for the average distance. Then, adding diagonals significantly reduces the distance between routers, and consequently the packet delay.

Looking at other traffic patterns, there are two extreme behaviours represented by the transpose and tornado patterns. The first obtained similar throughput in all topologies, this is a consequence of all having the same number of links crossing the diagonal of the network, therefore presenting the same bottleneck. But at low loads, as the different topologies have different average distances this fact is reflected in the base latency. The diagonal networks resemble the results of the standard or king networks depending on the added diagonal, thus the aforementioned importance of the application

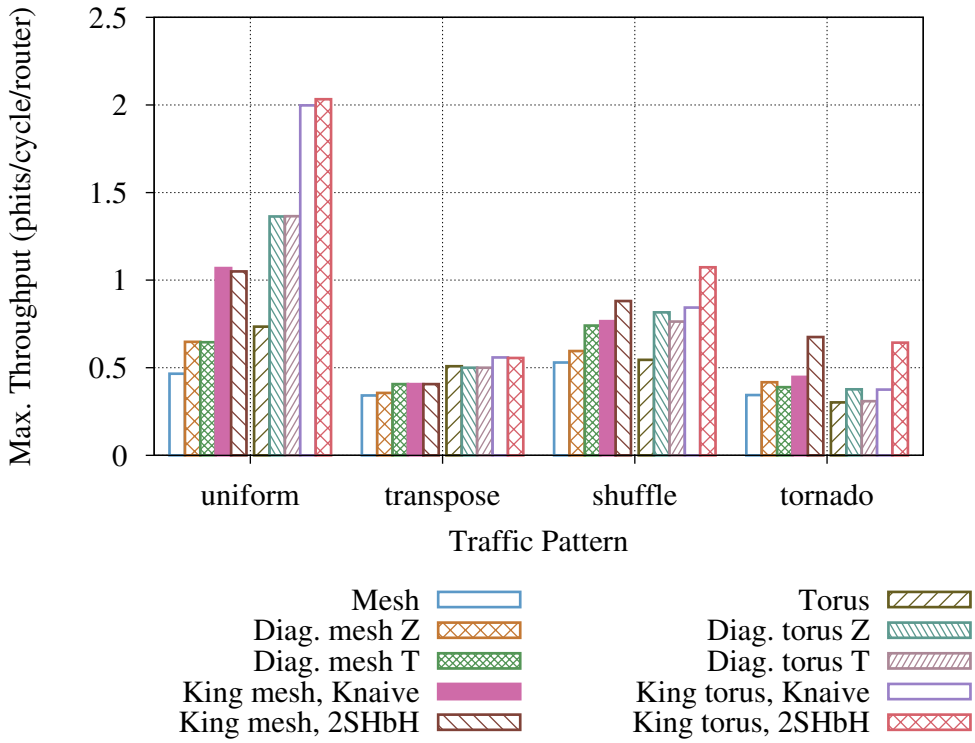


Figure 3.4: Maximum throughput comparison for 8×8 networks.

mapping in these topologies.

The case of the tornado traffic is the opposite, the packets move mainly along orthogonal directions and the added diagonals do not reduce the distance among routers. However they do add path diversity, so at high loads the traffic should spread throughout more paths and improve the maximum throughput. However this effect is only noticeable in the 2S Hop-by-hop algorithm, as it was shown in subsection 3.1.2 that Knaive was not able to exploit the full path diversity.

Experiments with trace-driven simulations

The use of synthetic traffic allows early evaluation of network performance by continually stressing the network with well-known traffic patterns. However, real parallel applications typically combine communication phases, in which the network is used, with computation phases, where no network traffic is observed [Val90]. Therefore an analysis in a more realistic scenario is important to round up an evaluation of the topologies presented in this thesis. The number of routers used for this evaluation suggests tens or hundreds of processing units. Some experimental prototypes with large number of cores do not support cache coherence [HDH+10, Sub13]. Therefore, a set of exper-

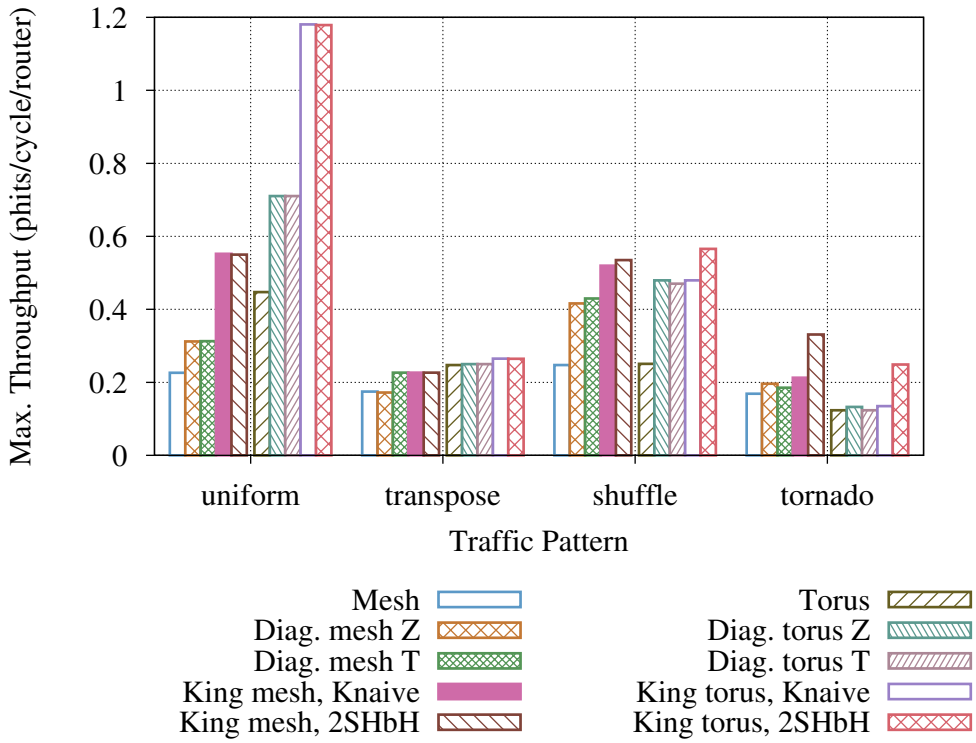
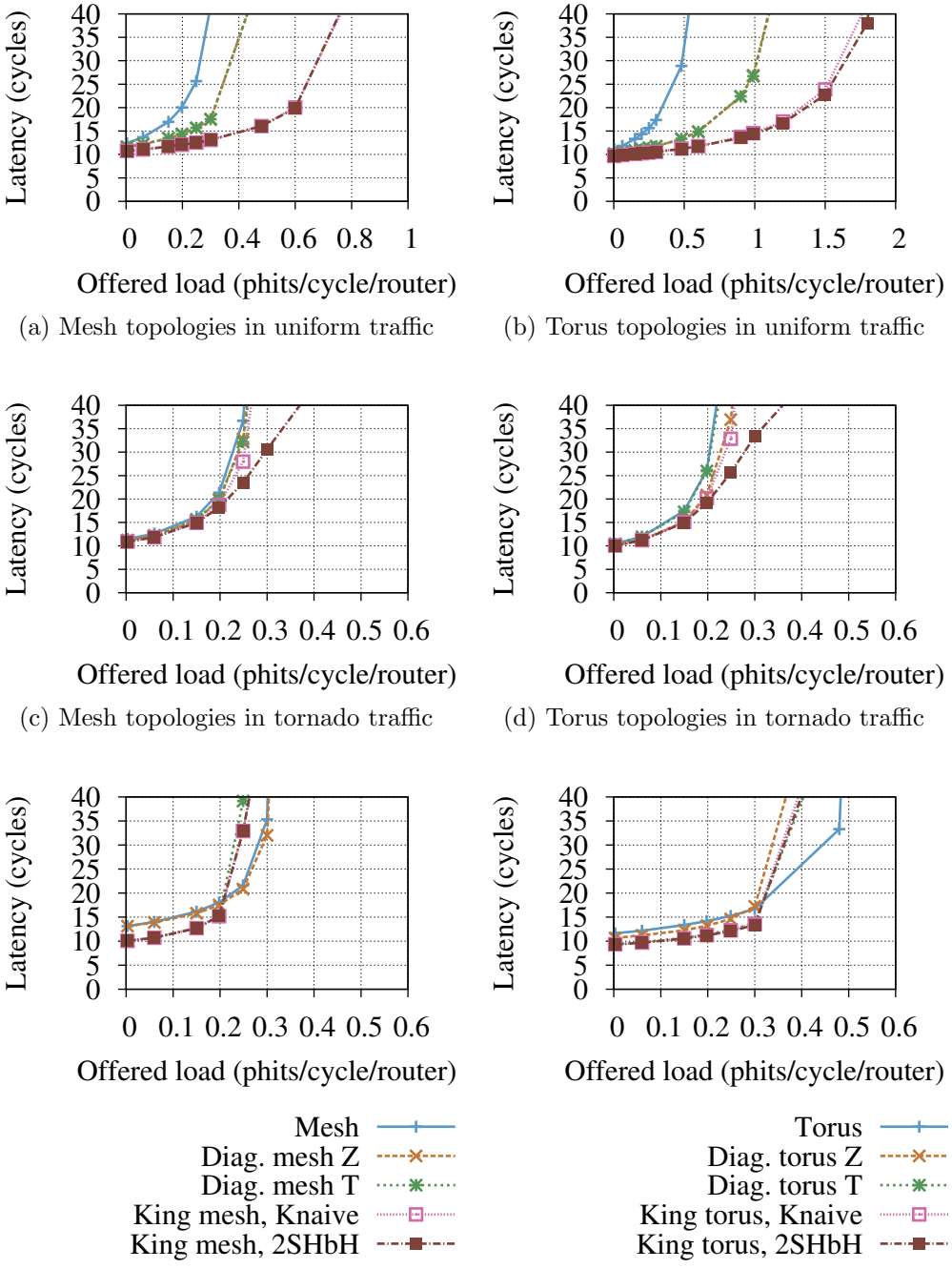


Figure 3.5: Maximum throughput comparison for side 16×16 networks.

iments with traces from the *Nas Parallel Benchmark (NPB)* suite [BBB+91], that uses the message passing paradigm, have been carried out. The results presented in this thesis only cover those benchmarks which have a significant network traffic, as these allow the study of the advantages and disadvantages of the different network topologies.

To present these results in an objective manner, and to emphasize the effect of the network, the simulation time for each topology is compared to the benchmark's ideal best time. Each trace was simulated with an ideal network that has infinite throughput and zero latency. This simulation time expresses the lower bound that can be aspired to when only the interconnection network is optimized. The chart in Figure 3.7 shows the slow-down exhibited by each topology and routing, computed as the ratio of the ideal time to the simulation time.

A first glance at the results shows that the best performance is given by the king topologies with the 2S Hop-by-hop routing algorithm. This can be understood when considering the traffic generated by real applications. Instead of injecting packets at a constant rate with a given pattern, they show communication bursts with irregular use of network resources. As was seen with synthetic traffic the 2S Hop-by-hop algorithm, which uses all the



(a) Mesh topologies in uniform traffic

(b) Torus topologies in uniform traffic

(c) Mesh topologies in tornado traffic

(d) Torus topologies in tornado traffic

(e) Mesh topologies in transpose traffic

(f) Torus topologies in transpose traffic

Figure 3.6: Latency behavour in 8×8 meshes and tori.

minimum distance paths between any pair of routers, spreads the traffic among more network resources. Then, this kind of routing improves the

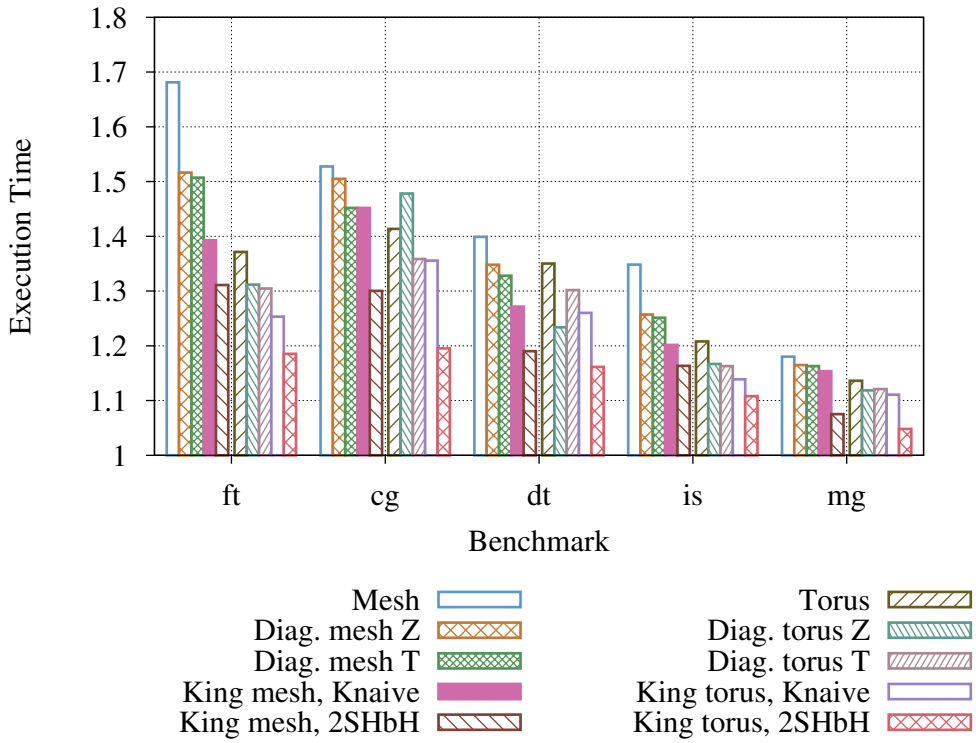


Figure 3.7: Performance of 8×8 networks. Execution time is normalized to that of the ideal network.

traffic balance on the network. A direct consequence is that there is less congestion and thus running the trace needs less time.

Looking at the results in more detail, a series of behavioral patterns are observed. First, when running real applications, adding one diagonal to the mesh or torus gives some improvement. But which diagonal is added does not affect the performance. For instance benchmarks ft, is and mg both in meshes and tori. In all these cases adding a second diagonal, especially when using hop-by-hop routing, gives much better results.

Second, in some cases the election of the added diagonal is important, as was mentioned above. For instance, the performance of diagonal networks with benchmarks cg on meshes and dt is heavily dependent on which diagonal is chosen. One diagonal can give performance close to that of the king topologies with Knaive while the other is barely better than the standard network. Therefore, when using diagonal networks the application mapping is particularly important. Focusing on king topologies, the performance difference between the Knaive and the 2S hop-by-hop routing points out the importance of the routing algorithm, as the use of a more sophisticated one compensates for the increased cost of these topologies.

Finally, the cg benchmark on tori shows that the addition of a diagonal can be detrimental to performance. This is due to the fact that cg has high communication between neighbouring routers on one diagonal. Then, as minimum routing is used, the number of paths connecting the routers is reduced from two to one.

3.2 Misrouting

The previous section showed an in-depth analysis of the benefits of using minimal routing. Subsection 3.1.2 mentioned that there is a possibility of forcing packets to misroute by having non-zero values in the four components of the routing record. This differs from the traditional methods of implementing misrouting. This section presents different solutions based on this concept, and an analysis comparing them to minimal routing and the classic Valiant misrouting approach [VB81].

Typically, the misrouting algorithms aim to balance the traffic traversing the network, regardless of the traffic pattern. The Valiant approach is done by selecting a random router for each packet that is injected, the packet must go to this random router before it can head towards its destination. This technique indeed balances the traffic, but has some drawbacks that will be overcome by the algorithm presented next. First, the latency of the packets is doubled. Second, there is no way to tune the amount of misrouting. That is, the intermediate router is chosen at random from the whole network. In the worst case, a packet travelling to its neighbour could be directed to the farthest corner of the network before it can finally reach its destination. And third, the change of direction that occurs in the intermediate router breaks DOR deadlock avoidance mechanism. To solve this, the packet must be consumed and reinjected, which incurs in an extra delay, or the packet must be sent through a different virtual channel, which increases the cost.

3.2.1 King$\epsilon\delta$

The next algorithm is designed around two ideas. The first is trying to balance the use of all the directions, as it was observed with EKnaive that balance is important for uniform and other benign traffic patterns. Also it is one of the strengths of Valiant routing. To achieve this it is desirable that the absolute value of the components in the routing records are as close as possible. The second idea is to equilibrate the use of all the network by allowing non-minimum routing records, that is, to perform misrouting to balance the use of the links. These concepts are formally described in the following definition.

Definition 3 *Let ϵ, δ be two positive integers. Let a and b be two routers of KT_s at a minimum distance d. Let $(\Delta_X, \Delta_Y, \Delta_Z, \Delta_T)$ be a routing record of the routers a and b. Then,*

- *The routing record is said to be ϵ-balanced if:*

$$\begin{array}{ll} ||\Delta_X| - |\Delta_Y||, & ||\Delta_X| - |\Delta_Z||, \\ ||\Delta_X| - |\Delta_T||, & ||\Delta_Y| - |\Delta_Z||, \quad \leq \epsilon, \\ ||\Delta_Y| - |\Delta_T||, & ||\Delta_T| - |\Delta_Z||, \end{array}$$

- *The routing record is said to be δ-diverted if*

$$|\Delta_X| + |\Delta_Y| + |\Delta_Z| + |\Delta_T| \leq d + \delta.$$

First of all, note that any ϵ-balanced 0-diverted routing record gives minimum paths. Therefore, the search for ϵ-balanced routing records would include the two previous minimal options, that is, Knaive and EKnaive routing algorithms. However, as will be shown later in the results subsection, better performance is obtained when misrouting is allowed. Hence, for a king torus network this subsection explores more suitable values of ϵ and δ so that a better performance in different traffic configurations can be obtained. As it will be shown next, selecting $\epsilon = \delta = \frac{k}{2}$, where k denotes the diameter of the king network, gives the best trade-off between latency and throughput.

The ϵ parameter represents the component balance. The smaller the ϵ, the more balanced are the components, thus increasing the possible paths. Therefore, especially in adverse traffic patterns, the profitable channels are increased, and traffic is spread over a wider area of the network. Whereas, increasing δ allows packets to wander farther from the minimum distance path, therefore allowing potentially empty areas of the network to be used.

Finally, the following example illustrates how this new technique entails an increase of the path diversity between a pair of routers. Consider routers $(0, 0)$ and $(13, 14)$ in KT_{16}. For three different configurations of ϵ and δ, all the ϵ-balanced δ-diverted routing records between both routers are found. In Table 3.2 the average path diversity obtained by these routing records is shown. Note that a different choice of ϵ and δ implies an increase of the average number of paths between any pair of routers. The next section will show how this increase translates to great performance gains.

3.2.2 King$\epsilon\delta$ Implementation

This subsection considers the technical aspects of the implementation of King$\epsilon\delta$. These networks are based on vertex-symmetric graphs. In an intuitive way, this means that the appearance of the network from a vertex is the

(ϵ, δ)	$(3, 0)$	$(4, 5)$	$(6, 7)$
Average path diversity	0.89×10^4	0.46×10^6	0.98×10^6

Table 3.2: Average path diversity

same for any other vertex. Therefore, calculating routing records depends on the relative position of the destination router from the source router, rather than the absolute positions of routers themselves. As this algorithm relies on tables, the vertex symmetry of the network means that there is only one table, of which every router has a copy. Note that the table is calculated off-line once for each network size.

Given the network dimensions, the table is defined by the two parameters, ϵ and δ, and is indexed by the relative position of the destination router. For each, the table has a set of possible ϵ-equilibrated and δ-diverted routing records that will be chosen at random when the packets are injected.

To find the best value for the parameters, an initial evaluation of all the tables with $0 \leq \epsilon \leq k$ and $0 \leq \delta \leq k$, where k is the diameter of the network, was performed. After observing the results, the scope was thought wide enough, as local maxima were found. The results for individual traffic patterns showed a performance improvement in all traffic patterns except uniform, as will be explained later.

From the above exploration it was concluded that, as the parameter values increase, the performance in adverse traffic patterns improves, but the average latency also increases. Analysing these results in detail allows selecting values for both parameters that maximise the throughput and minimise the latency. These values were found to depend on the diameter k of the network, $\epsilon = \delta = \frac{k}{2}$.

An interesting conclusion from this study is that the more adverse a traffic pattern is, the more significant is the improvement with large ϵ and δ. For instance with transpose traffic pattern the best results were obtained with $\epsilon = \delta = k$.

One concern related to the implementation of tables is their size. Presently the selected tables have a large amount of entries per destination in order to attain maximum path diversity. This is impractical, so a way to reduce the size was required. The method devised to reduce the size was to choose a subset of the entries of a table at random, such that the resulting table had a precise average number of entries per destination, or *multiplicity*. The selection was made ensuring that each destination had at least one entry. Then, tests were run to explore the performance vs. multiplicity trade-off. As can be seen in Figure 3.8, the performance stabilises as the multiplicity

increases. For the experiments a value of eight was chosen, since it gave acceptable results with reasonable table sizes. For example, multiplicity eight tables for 16×16 networks have 255 destination routers, and an entry can be packed in two bytes. Then, the total size of the table is almost 4KB. However, slightly better performance can be achieved with larger tables.

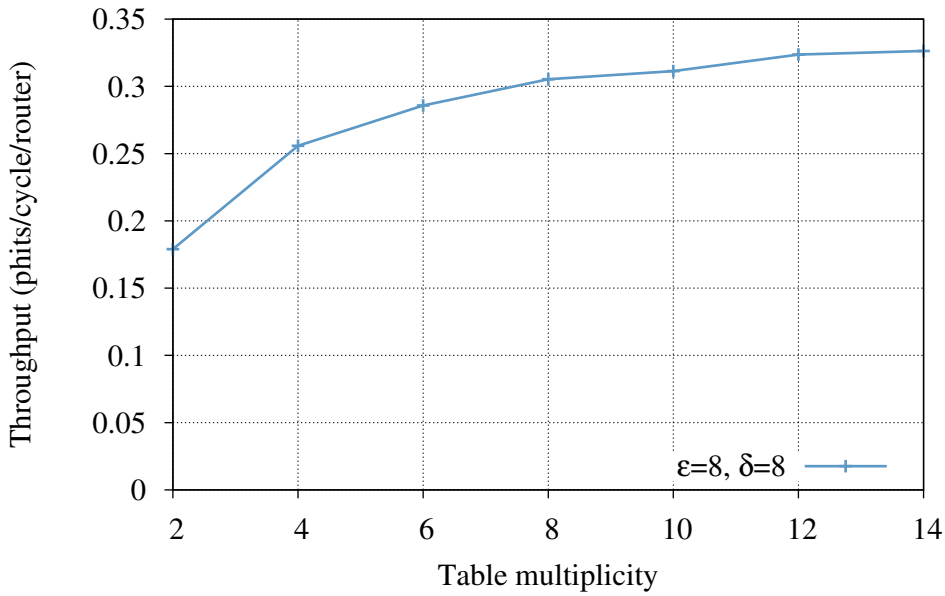


Figure 3.8: Representation of throughput vs. table multiplicity in a 32×32 king torus. The traffic pattern is uniform and load is 0.6 phits/cycle/router.

3.2.3 Evaluation

In this section we present the experimental evaluation carried out to verify the better performance and scalability of the proposed misrouting approach. This is done by comparing it with the 2S Hop-by-hop minimum-distance routing and the well known Valiant algorithm [VB81]. The evaluation will show the advantages of using King$\epsilon\delta$ technique selecting $\epsilon = \delta = \frac{k}{2}$ against the aforementioned algorithms. This study has been made with 16×16 and 32×32 networks. Although only the results for 32×32 are shown, similar conclusions can be extracted from the results of 16×16 networks.

As before, all the experiments have been done with the functional simulator Fsin [NMAPR11]. The router model is based on the bubble adaptive router presented in [PBG+99] with two virtual channels. The evaluation has been performed with synthetic workloads using typical traffic patterns. These were applied on the networks by injecting packets of 8 phits at a constant rate, or load, measured in number of phits per cycle per router. To

ensure the validity of the results, measurements of performance parameters were taken only when the network reached a steady state. The increased degree of some topologies theoretically allows the throughput to rise above one phit per cycle per router. In order to take advantage of this, each router had up to three injectors per router. The metrics considered are throughput, average latency and also maximum latency.

Figures 3.9 and 3.10 show the results in 32×32 networks, thus selecting $\epsilon = \delta = \frac{16}{2} = 8$, when stressed with well known traffic patterns. The tables used have multiplicity eight as was explained in subsection 3.2.2. The values measured are throughput and latency vs. increasing load. As will be shown, the behaviour of the different algorithms is clearly divided in two tendencies. First, in uniform and reversal traffic patterns the best performance is obtained with minimum distance routing, these patterns are considered *benign*. Second, the remaining patterns are considered *adverse* and therefore benefit from the misrouting techniques.

It is known that uniform and reversal traffic patterns naturally balance the use of all the network resources, and hotspots do not appear. In addition, it has been shown in Subsection 3.1.2 that the 2S Hop-by-hop routing algorithm does not disturb this equilibrium. Then, although the misrouting approaches are also balanced, they force each packet to stay longer in the network, so the throughput can not be as high as with a minimal routing.

The Valiant algorithm completely randomizes the communication independently of the used traffic pattern. So it is assumed that it gives an upper bound for the throughput in adverse traffic patterns. Consequently the graphs show that this algorithm always reaches the best throughput in these traffic patterns. It is noticeable that King$\epsilon\delta$ always improves on the minimum distance routing.

The advantage of the Valiant algorithm in throughput is obtained at the expense of doubling the average distance of the packets, and therefore increasing the latency. The diminished throughput of King$\epsilon\delta$ is a trade off of its benefits. Namely, that the average distance is only increased in 50%, thus the latency increase is much smaller. Especially at low loads, which is the most common scenario, the latency is almost equal to that of 2S hop-by-hop. Furthermore, the asymptotic growth of the latency occurs at higher loads than the best of the other algorithms.

3.3 Collective

The previous sections have concentrated in different kinds of routing algorithms. But they all have something in common: they can route a packet beween a single source and a single destination router. However it is usually

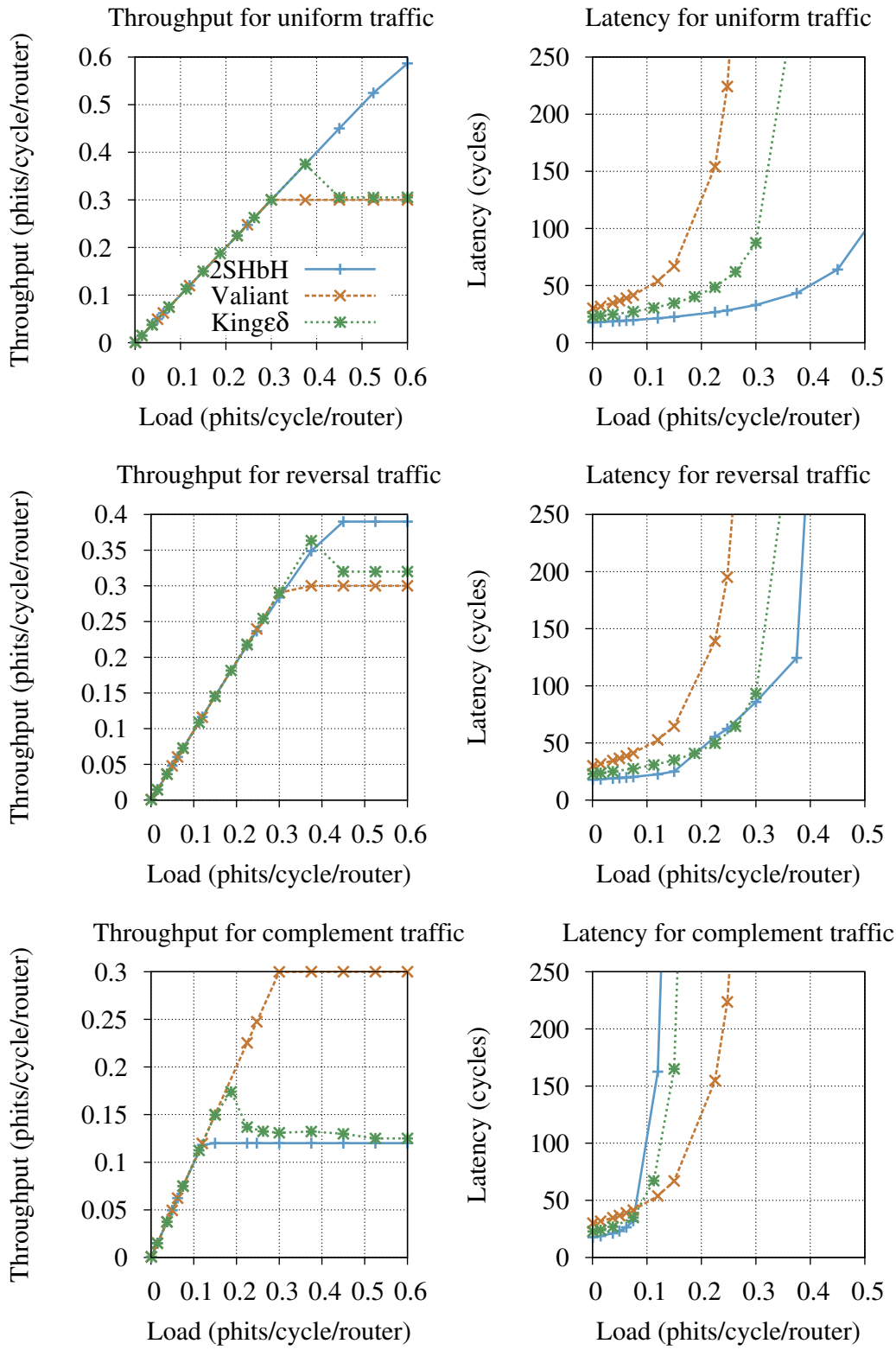


Figure 3.9: Throughput and latency comparison of King$\epsilon\delta$ with Knaive Valiant on a 32×32 king torus under various traffic patterns.

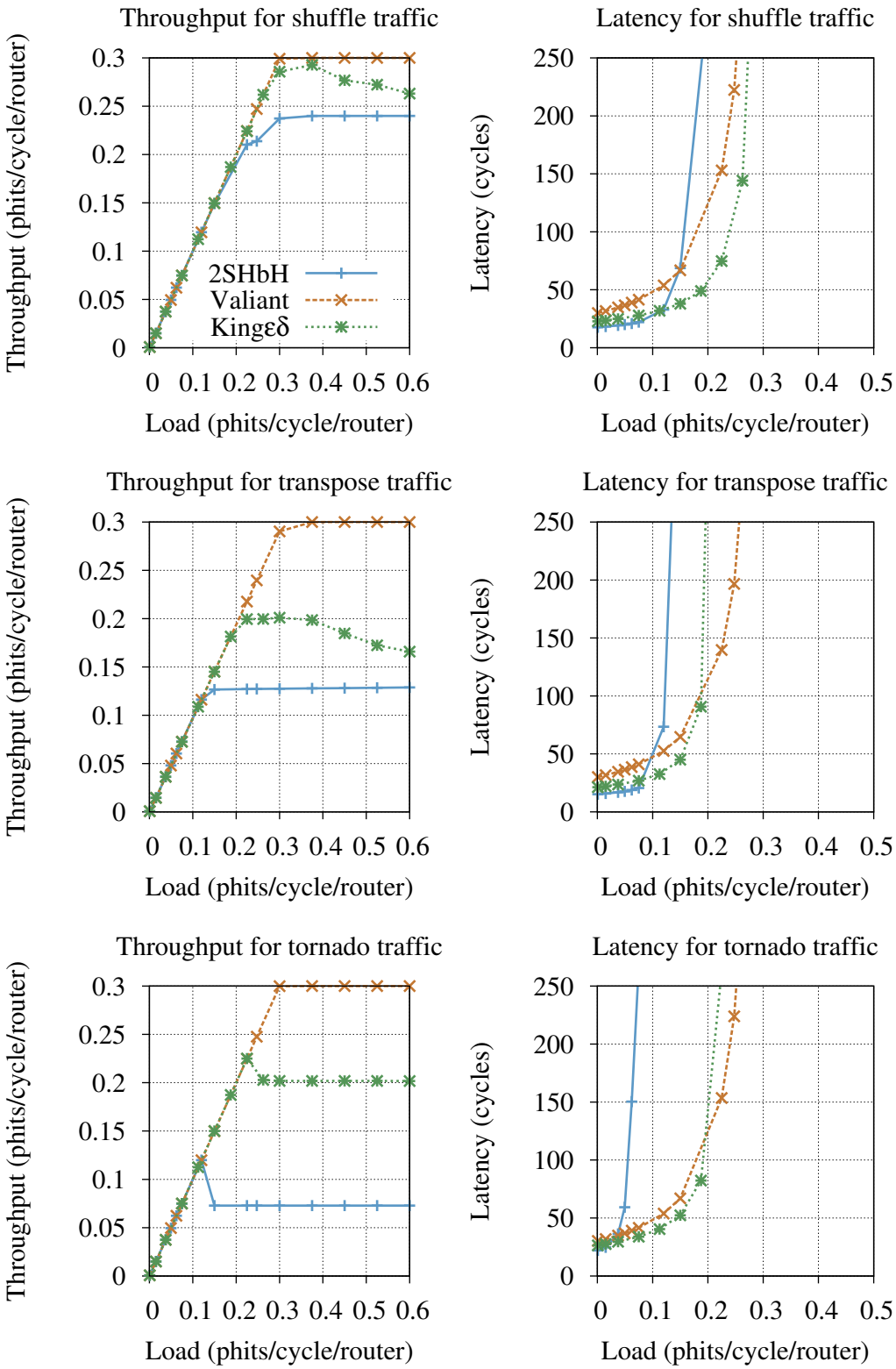


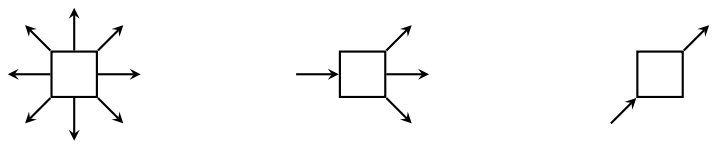
Figure 3.10: Throughput and latency comparison of King$\epsilon\delta$ with Knaive Valiant on a 32×32 king torus under various traffic patterns.

the case that the application program will need to send some information to more than one router, and doing this efficiently is key to achieve the best performance. This section focuses on the implementation of two collective communication primitives, the broadcast and the multicast.

3.3.1 Broadcast

Many parallel applications, both in distributed and shared memory computers, rely on efficiently sending broadcast and multicast messages. In fact, the research for improving the performance of collective communications has received much attention in the last years [TRG05]. Hence, a broadcast algorithm is presented. It relies on routers being able to forward packets through more than one port in one cycle. As the packets are sent under DOR discipline and ABR [PBG+99] restrictions, the algorithm is deadlock safe.

To perform a broadcast, the source router sends the message to all its neighbours (Figure 3.11a). The message is marked with a broadcast flag and a time-to-live (TTL) integer. Then all routers behave in the same fashion. If a broadcast packet is received, the TTL is decremented and consumed. If TTL is greater than 0, the packet is also sent through a number of other ports. So if the broadcast packet was received from an orthogonal port, it is forwarded through the next neighbour in the same direction plus two in the diagonal directions (Figure 3.11b). If the packet is received on a diagonal port, the packet is only forwarded to the next neighbour in the same direction (Figure 3.11c). Thus the TTL value defines the maximum area that the broadcast will reach. Obviously it must be set to a number that covers the whole network. In the case of king meshes, the TTL can be calculated as $max(x, s-x, y, s-y)$, where s is the side of the mesh and x, y are the coordinates of the source router. For king tori the TTL is the diameter.



(a) Source router (b) Ortho. router (c) Diagonal router

Figure 3.11: Behaviour of routers during broadcast.

3.3.2 Multicast

A multicast could be constructed around the same idea. The source router would make a broadcast message including a list of the destination routers. This list should be organised in the same way as the branches of the broadcast

pattern. Then as the broadcast packet spreads and different copies of it take different paths, each copy would have the section of the destination list corresponding to its path. Destination routers that receive the packet should consume it if they are on the destination list. And they should forward it, like the broadcast above, only if the list contains more addresses after removing its own from it. Thus the multicast would not send unnecessary packets to parts of the network where there are no destination routers. Note that this scheme would not allow adaptive routing, as packets would be able to leave their branch.

3.4 Fault Tolerant Routing

As the complexity of computer systems increase, the *Mean Time Between Failures* (MTBF) is reduced. This time could be as small as the average application run-time, rendering computers inefficient, as they are unable to reliably conclude the execution of applications. The fact that computer systems must be aware of the non-zero failure probability extends to all its components, including the interconnection networks. The routing algorithms presented so far consider the network ideal, in terms of failures. This section explores the suitability of King networks for the construction of fault tolerant systems. Therefore it considers different fault-tolerant routing algorithms.

3.4.1 Adaptation of known routing algorithms

Fault tolerant routing is a common topic in the literature, as such, there are many fault-tolerant routing schemes proposed for different topologies. However most of these routing algorithms are bound to a single topology [Shi91] or topology family. Two routing algorithms with no such limitation are Immunet and Immucube [PGVB04, PG07]. This section explains how these algorithms can be used with King topologies.

Immunet

As stated above, Immunet is a fault tolerant routing algorithm that can be used on any topology [PGVB04]. Furthermore it allows any fault distribution at any time, which makes it one of the most robust algorithms available. Also the performance loss is negligible when the number of faults is small.

It is based on Adaptive Bubble Routing (ABR) [PBG+99] and requires only two virtual channels. While the network is fault-free, one of them is used as fully-adaptive, in which minimum distance routing is performed regardless of deadlock considerations. The other is considered an escape channel, which is used with a deadlock avoidance mechanism, and routes the packets that

could not progress through the first channel [Dua96]. In k-ary n-cube topologies like meshes or tori the deadlock avoidance is usually Dimension Order Routing (DOR). When a fault appears, the network enters a reconfiguration phase. The routers affected send fault packets to all the neighbours, and they in turn forward it until all the network is reached. Using only information from the fault packets, a tree that covers the whole network is generated. This tree, that visits all the routers of the network through healthy links, is used as a ring when traversed in preorder. The ring allows packets to visit all the routers of the network without leaving it. This ring replaces the healthy network's deadlock-free routing and allows packets that encounter congestion or faulty links to reach their destination. A more in-depth description can be found in [PGVB04].

The replacement of the healthy-network deadlock-free algorithm with the ring can lead to poor performance when congestion is very high. It has been observed that in these situations the throughput delivered approaches that of a ring instead of the higher degree network containing it.

Employing this algorithm in King topologies is straightforward, as the these can operate perfectly with ABR.

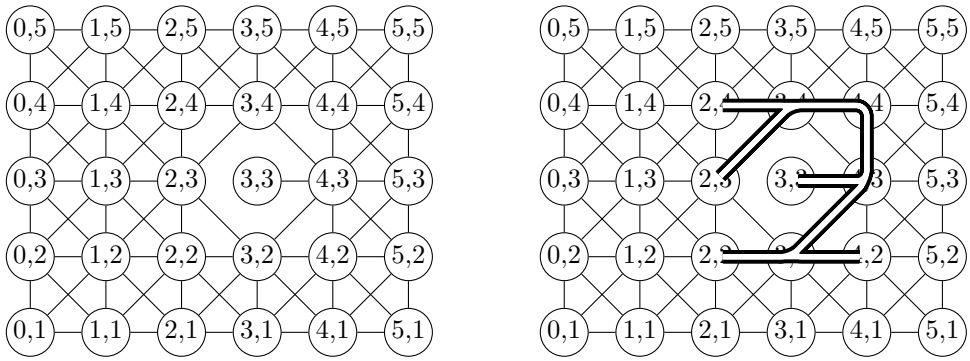
Immucube

Immucube is an evolution of Immunet tailored to k-ary n-cube networks. It solves to some extent the scalability problems of Immunet, whose performance is poor with networks larger than 16×16. When in fault-mode, instead of replacing the healthy-network deadlock avoidance mechanism(DOR) that governs the escape channels with the tree/ring routing, Immucube adds a third virtual escape channel for the ring. When the network is fault-free, the extra channel is used as a second adaptive virtual channel. How the three virtual channels are used, and details on the reconfiguration process are presented in [PG07]. Again, due to the similarities with standard 2D networks, Immucube can be straightforwardly used on king networks.

The great scalability of Immucube is due to the fact that the traffic in areas with no faults use a routing scheme similar to that of the healthy network. The ring channel is almost only used when packets encounter faulty links. Then the probability of the whole network giving the throughput of a ring is much lower. The downside to this algorithm is that it uses an extra virtual channel, thus increasing the cost of the solution.

3.4.2 King Fault Tolerance Algorithm

To further test the fault-tolerance properties of king networks, this section presents a new fault-tolerant routing algorithm custimised for them. Therefore, it has been named King-Fault-Tolerance (KFT). The algorithm was



(a) Faulty network. (b) Logical ring spanning fault area.

Figure 3.12: KFT example.

devised trying to achieve the performance of Immucube at the cost of Immunet. Thus, it uses only two virtual channels and its performance is close to Immucube for a low number of faults. The key feature is that it only affects the area close to faults. Therefore, it does not disturb the traffic in healthy areas. Thus making it more scalable than the other two algorithms.

As in Immunet, KFT uses the concept of a tree traversed in preorder to create a ring. However, instead of one ring covering the whole network, KFT creates a small ring that visits all the nodes with faulty links. An example of a ring covering a faulty area is shown in Figure 3.12. The algorithm takes advantage of the high connectivity of king topologies to allow small trees to cover faulty areas, reducing the number of packets to generate them. The concept could be used in other 2D networks, but the resulting algorithm would be more complex, falling beyond the scope of this work.

The algorithm requires routers to keep some local data.

- The *reconfiguration status*(RS) register contains the concatenation of two integers. On the high order part is the identifier of the router that generated the fault-alert propagation. And on the low order part is the number of reconfiguration processes it has started so far.
- During the reconfiguration process, the *passages table* is used to describe the structure of the tree. Later, it will be used to guide packets through the tree in preorder ring fashion.
- A register with the current amount of active reconfiguration processes in the router.
- The *state* register tells whether the router is in fault mode, depending on the number of faulty links it has.
- A list of pending communications with neighbours.

Also, the routers must exchange some information. This is carried by a set of special KFT packets that only travel among neighbours, so they do not need routing.

```
begin
    if recv fault-alert packet then
        if sending router and current router are fault free then
            send DEC
        if number of reconfigurations in current router > 0 then
            if received RS > local RS then
                stop reconfiguration process
                join tree as child
                send fault-alert packet to neighbours
                send ACC
            else
                send DEC
        if current router has faulty link then
            if received RS > local RS then
                join tree as child
                send fault-alert packet to neighbours
                send ACC
            send DEC
        if current router is fault free then
            if sending router has faulty link then
                if received RS > local RS then
                    join tree as child
                    send fault-alert packet to neighbours
                    send ACC
                else
                    send DEC
            else
                send DEC
end
```

Algorithm 2: Router behaviour on the reception of a fault-alert packet.

- The *fault-alert* packet requests the recipient to become a member of the tree as a descendant of the sender.

- The *accept(ACC)* is a reply packet that conveys a positive answer to the fault-alert packet.

- The *decline(DEC)* is also a reply packet wich gives a negative answer to the fault-alert packet.

The reconfiguration process is driven by two events. The first is triggered by the detection of a faulty link and only happens in the routers connected by that link. The second happens when a KFT packet is received, it can be a fault-alert or a reply.

When a router senses that a link fails, it updates its RS register, increasing the amount of reconfiguration processes and, after stopping all normal traffic, it sends fault-alert packets to all its neighbours. This has the effect of starting two reconfiguration processes, or trees, one for each router connected to the faulty link. The senders of the messages must keep a list of recipients whose answer is awaited.

Depending on the kind of KFT packet received, the router behavior will differ. During the process of reconfiguration, a router can receive fault-alert packets from different neighbours. These packets might request the recipient to join different trees, however, a router can only belong to one tree at a time. Thus, it must choose which request to accept and which to decline. When a router receives a fault-alert packet, it will reply affirmatively only if the RS of the packet is greater than the one stored in the router. Otherwise, the router sends a decline packet back to the sender. This mechanism causes one tree to prevail and the other to be forgotten. As can be seen in Algorithm 2, the acceptance must fulfill some other conditions that prevent the mechanism from covering the whole network.

If a router receives an accept packet, the sender is added as a child to the tree by updating the passage table, declinations are simply ignored. In any case, the reception of a reply causes the sender to be removed from the pending communication list. When the list is empty, the reconfiguration process is deemed complete and normal data communications can be resumed.

It is important to note that the way in which the RS values are used causes a single tree to eventually cover the faulty region. Once this happens, the information stored in the passage table guides the packets through the tree, in a ring-like fashion, avoiding the faulty links.

Optimization: pruning the tree

Once the reconfiguration process concludes, the normal traffic on the network starts moving again. When a packet needs to cross a faulty link it enters the ring and only leaves when it gets closer to its destination. Obviously, the smaller the ring, the less detour the packet must make. With this in mind, KFT has a pruning phase that reduces the size of the tree and thus the size of the ring. As a consequence, the latency of packets affected by a fault is reduced.

As an example consider the ring in Figure 3.13a. It becomes apparent that the ring is covering more routers than necessary to wrap the faulty region. The optimization consists in pruning leaf routers which are not needed. Once

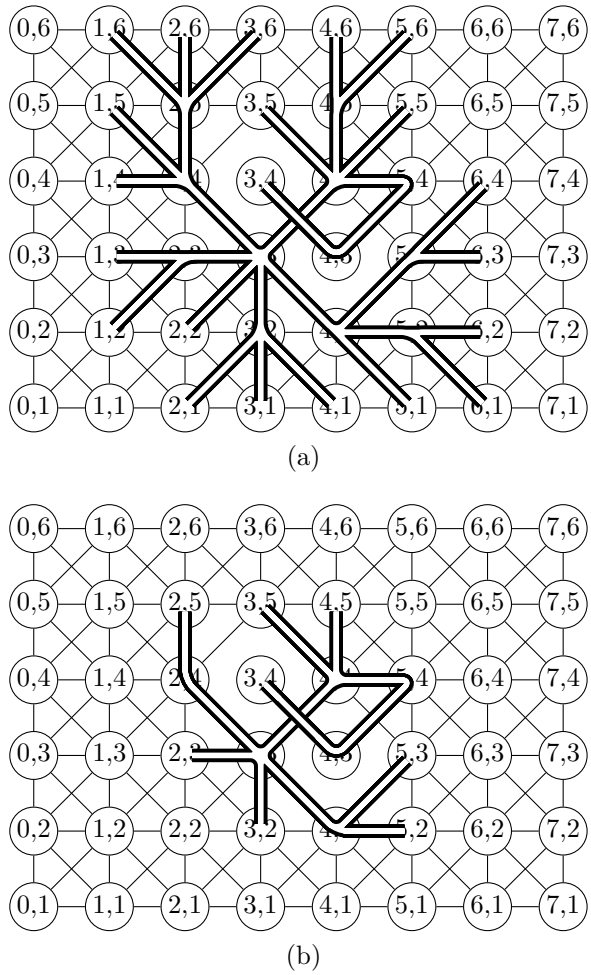


Figure 3.13: Ring created around a complex faulty region (a) before pruning, and (b) after pruning.

a router is in a stable state and is fault-free, it can easily determine if it is a leaf router. If there are no children in the passage table it sends a prune message to its parent and waits for the accept. The parent router clears its child from the passage table and sends the accept back. On reception the leaf router will no longer be part of the ring. Figure 3.13b shows the example tree after the pruning stage.

Pruning must be done once the reconfiguration phase is concluded as some pathological cases require some leaf router to belong to the tree in order to successfully conclude the reconfiguration process.

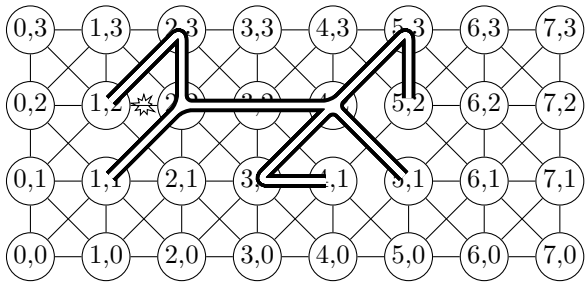


Figure 3.14: Two faulty regions joined by one ring after a fault occurring between routers (1,2) and (2,2).

Multiple simultaneous failures

The scenario presented above with a single faulty link is useful to describe the KFT ring generation algorithm. However, KFT can cope with more complex situations. It can withstand multiple failures happening simultaneously with the reconfiguration process of previous faults. If the new failure occurs in a healthy area, KFT creates a new ring isolated from the existing ones. However, if a new fault occurs close to an existing ring, then it is extended to cover the new fault. If as a consequence of a ring increasing its size it meets another, both are joined to form a single ring covering both faulty regions. An example of this process can be seen in Figure 3.14, where the last link to fail is that shown with ✲ . The process creates a ring that joins two existing and isolated ones.

The scenario in which two rings are joined as above is not contemplated by the previous algorithm. And under some conditions, the two regions will not join. In the case that one of the rings is already stable, and has a higher RS.high than the reconfiguring ring, the stable one will decline all the fault-alert packets from the reconfiguring one. This will lead to the two regions not joining. However this situation can be detected and trigger a reconfiguration process with higher RS.high that will engulf both rings.

The algorithm presented above must be modified to cope with these situations by adding two conditions. First, the fault-free routers should broadcast fault-alert packets if they come from a faulty router and their RS.high is lower. The receiving router should not change its status. This causes that a fault-alert packet will be forwarded to a router from a different ring whose RS.High is higher. According to the algorithm above, this packet should be ignored, therefore a second condition is added. This causes that when a faulty-router receives a fault-alert from a fault-free router, it will respond with DEC and, after increasing its RS.high, start a new reconfiguration process. As it has a higher RS.high, it has a higher priority and will eventually merge the two existing rings.

Routing

The above descripton has illustrated how KFT reacts to fault events and establishes virtual rings that will allow packets to circumvent them. The following lines explain how normal packets behave in a network with faults. Just like Immunet, when the network is healthy, packets are routed using ABR with two virtual channels.

When a failure occurs, all the routers involved in the reconfiguration will stop normal traffic flow until their passage tables are stable. Once this happens, these packets can resume their course. After the reconfiguration, the virtual channels will be reorganized. The escape channels will continue to do static DOR routing. But the fully adaptive channels of the links that conform the ring will be used exclusively by packets in fault mode. The rest of the adaptive channels will remain adaptive. Therefore, the functionality of the fault-free regions is not changed and ABR routing is used as if no failure had occurred.

In faulty regions, when a packet needs to advance through a link that belongs to the ring, it must request the escape channel. While in the ring, packets must use the escape channel unless they need to advance through a faulty link. Then they enter fault mode. To the packets in fault mode a new header will be added. This will indicate their state and also the distance to their destination when they entered fault mode. Then, these packets advance through the routers of the ring until they reach one that is closer to their destination and can exit the ring resuming their normal state.

In order to avoid deadlocks, packets entering or leaving the ring must abide by some rules. Like when a packet changes dimension in ABR, packets entering the ring must satisfy the bubble condition. When leaving, packets must first try to exit through an adaptive channel. If not possible, they should request an escape channel. However, these channels only route packets under DOR. And because the packets exiting the ring might violate this rule, it is necessary to re-inject them.

3.4.3 Evaluation

This section presents a set of experiments that allow the evaluation of the fault tolerance properties of king networks compared with their standard counterparts, conventional meshes and tori. It also includes a comparison of the fault-tolerant routing algorithms described before. This is done evaluating their performance as well as their resource requirements.

Once more, all experiments have been made with Fsin, the functional simulator for interconnection networks belonging to the INSEE Environment [NMAPR11]. They have been carried out on networks of four topologies (mesh, torus, king mesh and king torus) with 16×16 and 32×32 routers. In

order to ensure that king networks of 16×16 reach saturation, two injectors per router have been used.

The traffic used has a uniform distribution, injecting packets of 16 phits at a constant rate. The fault generation model is random with a uniform distribution. Performance is evaluated by considering values of throughput and average latency. To achieve stability in the results, the values presented on the graphs are the average of five individual experiments with different random seeds.

King vs standard topologies

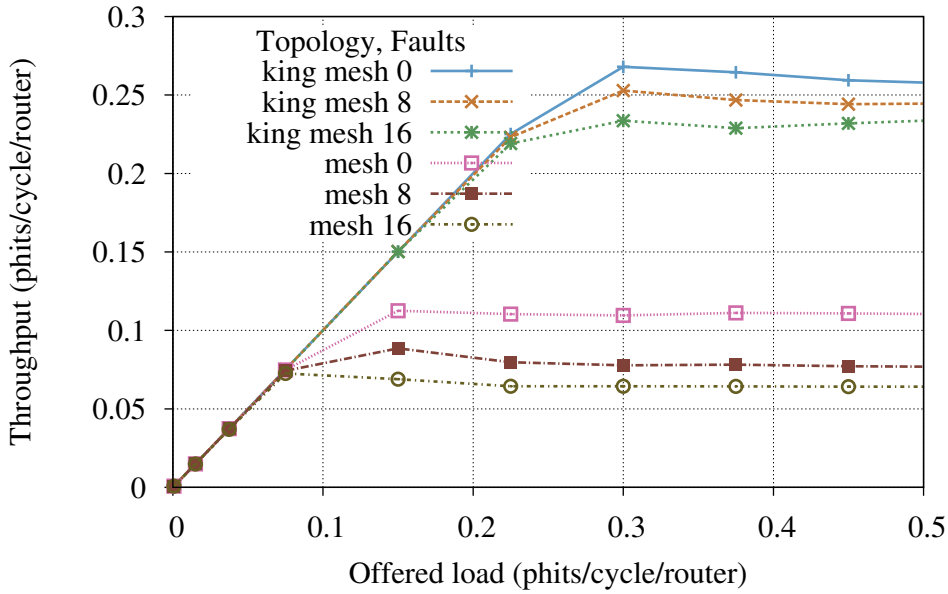
The results of the first experiment show a comparison of the performance of networks different topologies and 32×32 routers. First, in Figure 3.15a, the results for meshes and king meshes are compared and then, those for tori and king tori are shown in Figure 3.15b. Both figures show the performance of healthy networks and two more examples with 8 and 16 faults. The algorithm used in all cases is Immucube, as Immunet is known to give poor performance in large networks and KFT is only applicable to king topologies.

Both figures show that the loss of performance due to faults is much less in king topologies. Meshes show a degradation of 36% for 8 faults and 45% for 16, while king meshes only decrease in 5% and 9% respectively. With tori, the difference is more pronounced as king tori only loose 1.6% and 3.2% and conventional tori 33% and 42% respectively.

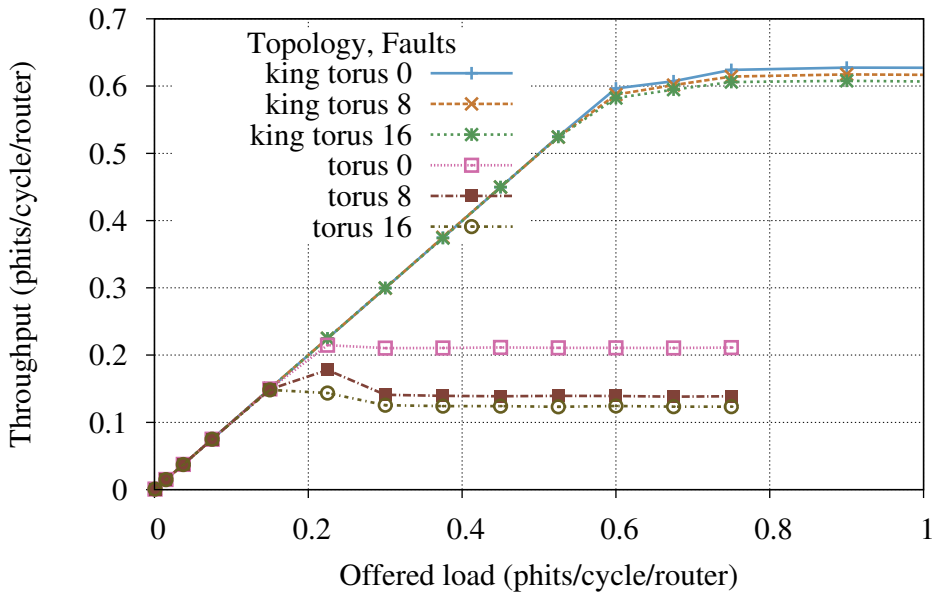
Experiments with a single fault

Focusing on the performance of king networks, the next experiment presents a comparison of the three fault-tolerant routing algorithms working on networks of 16×16. Similar conlusions can be drawn from the 32×32 network results. To visualize the degradation, the performance of the healthy network is also shown. Figure 3.16 shows throughput and latency for meshes and Figure 3.17 for tori. At low loads, before saturation point, the three algorithms give the same throughput and latency as the healthy network. At high load, the performance loss in Immucube or KFT is negligible and quite apparent in Immunet. This effect is more pronounced in tori.

The degradation observed in Immunet is related to the fact that, when a fault is detected, the escape channels stop using DOR and use the ring that covers all the routers, leaving the escape network notably reduced. And as the escape channels are mostly used in saturation, the performance deteriorates. In contrast, both Immucube and KFT have better results because they use DOR in the escape channels. However, the performance of KFT is slightly better as it only affects the vicinity of the failure, while Immucube does not allow packets that have encountered a fault in their way to use DOR. The



(a) Meshes vs. king meshes.



(b) Tori vs. king tori.

Figure 3.15: Throughput of 32×32 networks with Immucube.

latency graphs confirm this behavior; at high loads Immunet is slightly higher than the rest.

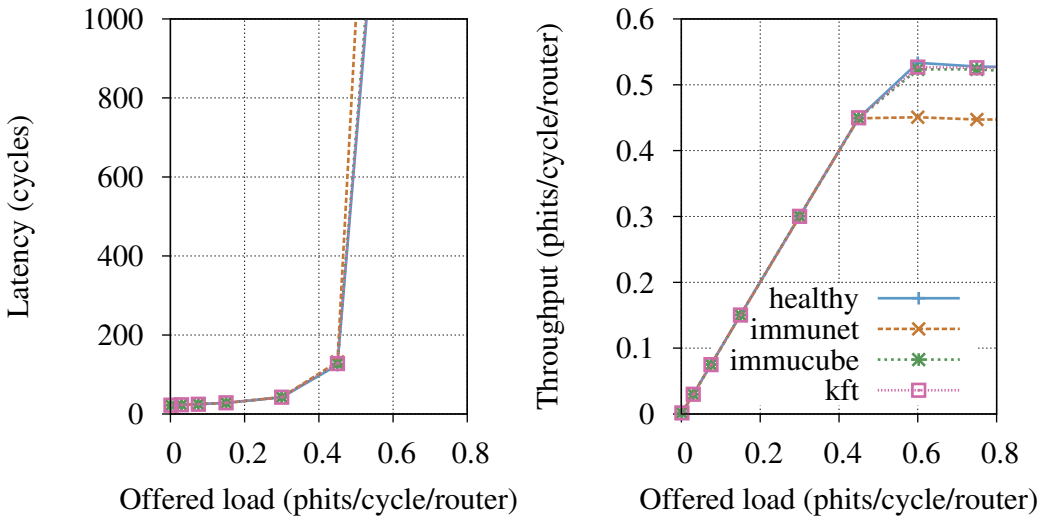


Figure 3.16: Throughput and latency of 16×16 king meshes with one random fault.

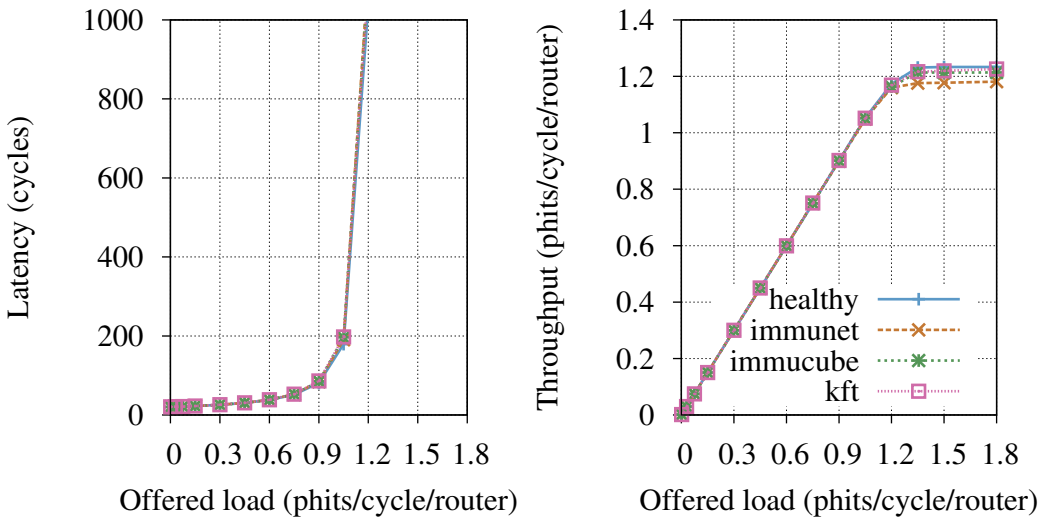


Figure 3.17: Throughput and latency of 16×16 king tori with one random fault.

Experiments with multiple faults

Figures 3.18 and 3.19 show the latency and throughput of networks of sizes 16×16 with 8 random faults and of 32×32 with 32 faults. These illustrate how the algorithms behave when the number of faults increases. The tendencies observed are independent of the size of the networks, thus the evaluation will be done together.

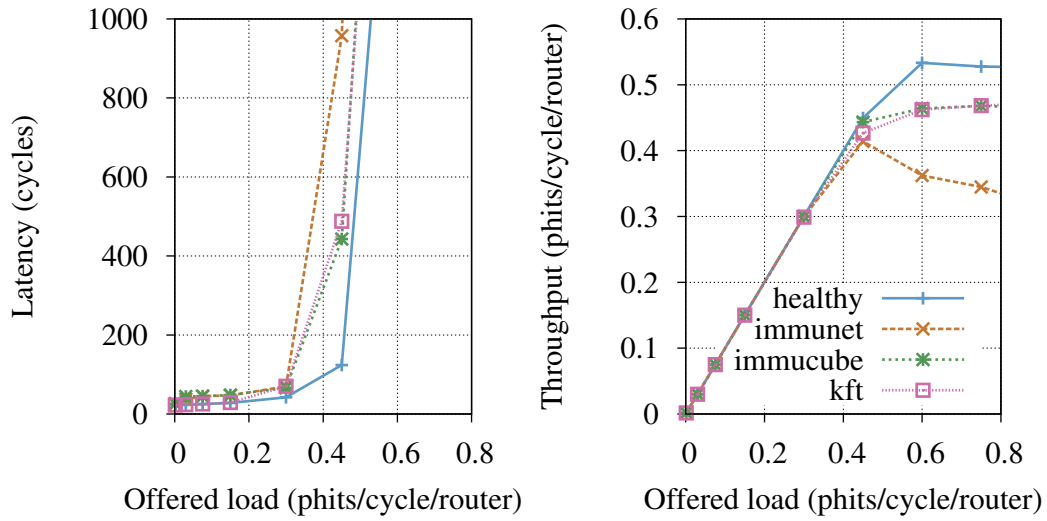


Figure 3.18: Throughput and latency of 16×16 king meshes topologies with eight random faults.

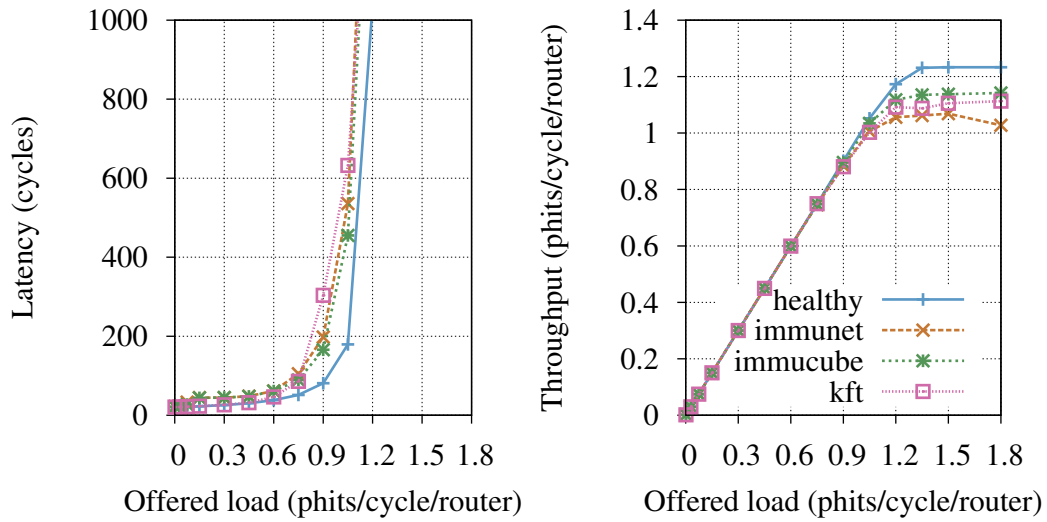


Figure 3.19: Throughput and latency of 16×16 king tori with eight random faults.

On meshes (Fig. 3.18), the higher number of faults is more noticeable as the performance loss is higher than before. Yet Immunet still shows the highest degradation and both KFT and Immucube have similar results. However, the third virtual channel used in Immucube allows a slight improvement over KFT, because it sacrifices adaptive channels to the ring. On tori (Fig. 3.19), the tendency is the same but with a higher difference between KFT and Immucube, again due to the third virtual channel. On Immunet, the

larger amount of faults increase the probability of packets requesting escape through the ring. This causes a faster congestion in it, therefore hampering the traffic flow throughout the network.

At the beginning of the saturation area where latencies are still small, KFT has higher latencies due to the loss of adaptive channels. However as the networks saturate, packets in Immunet and Immucube are forced to make longer detours through their large rings, while KFT is able to keep latencies slightly lower.

Throughput degradation experiments

Finally Figures 3.20a and 3.20b show the evolution of the saturation throughput versus the number of faults. The number of faults is shown as a percentage of the number of faulty links against the total number of links in the networks. The range shown, between 0 and 3%, represent from 0 to 32 faults in a 16×16 network and up to 128 in a 32×32.

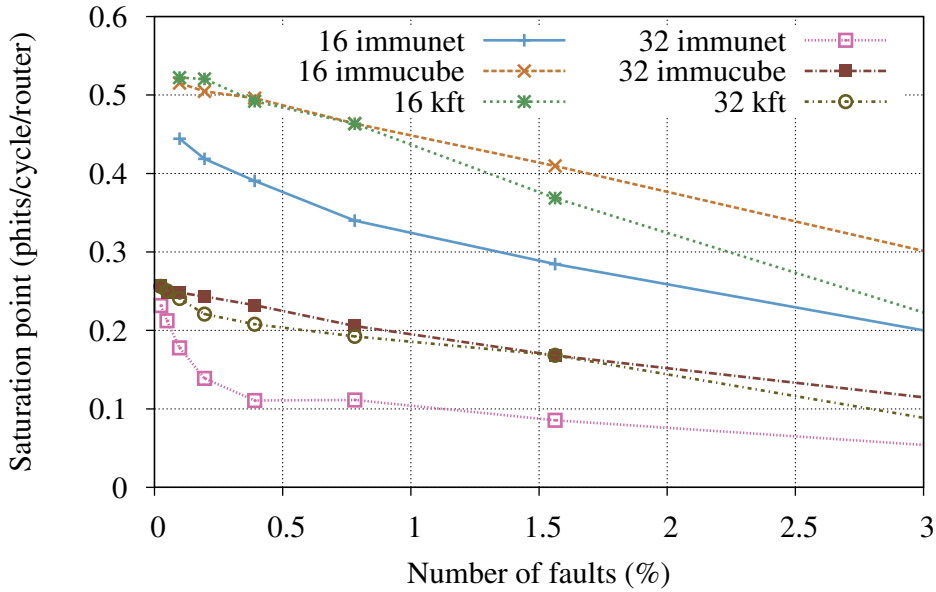
In the mesh, for a low number of faults, the performance of KFT is similar, or even slightly better than Immucube, and always higher than Immunet. This result is more noticeable in 32×32 networks, confirming that the locality of KFT makes it more scalable. However, there is a point at which the relative loss of adaptivity is too big and the performance of KFT begins to decrease. Nevertheless, the network still performs very well with Immucube, showing a graceful degradation despite the large number of faults. The fact that KFT uses 33% less resources than Immucube make it an excellent option if the number of faults is small.

For king tori, the performance is similar for a low number of faults, up to 1%. Beyond this point, the behavior resembles that of the king mesh. In terms of performance loss, Immucube shows a degradation of 20-25%, even with many faults. This confirms the good behavior of king networks in fault-tolerant applications.

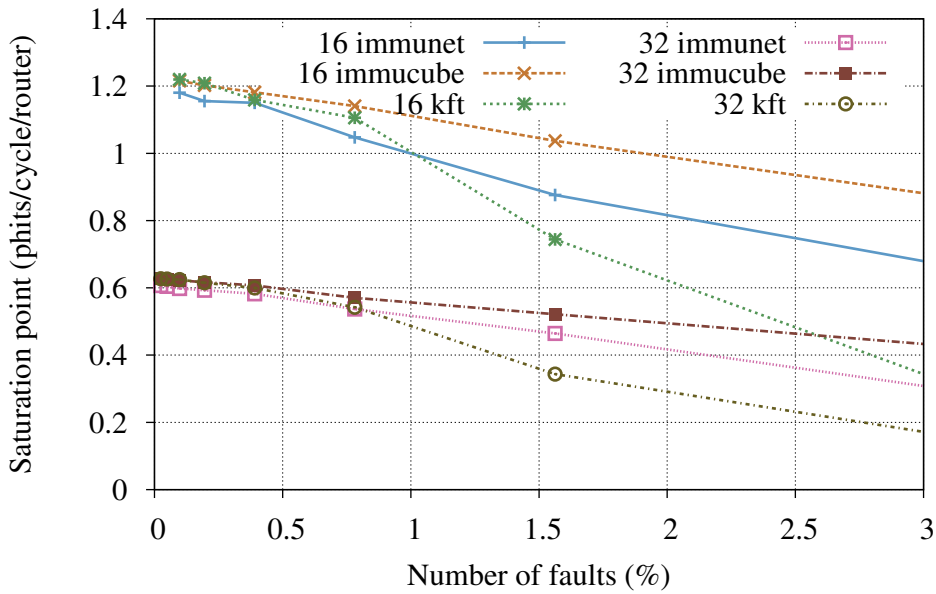
Finally, a conclusion to the experiments shown above is that, the king networks can play an important role in the future of interconnection networks when high fault tolerance is needed.

3.5 Concluding Remarks

This chapter has proposed a set of routing algorithms that allow extracting the best performance king networks. Using the standard networks as baseline, and comparing also to diagonal networks, the best results were shown by the king topologies. The performance of diagonal networks, under particular traffic conditions, was similar to those of the standard networks, and their use does not seem to justify their extra cost.



(a) King meshes of 16×16 and 32×32.



(b) King tori of 16×16 and 32×32.

Figure 3.20: Saturation point of king topologies versus number of faults.

For the king networks there were several proposals, each with their strengths and weaknesses. Thus depending on the scenario, it might be advisable to choose one over the rest.

When the traffic is not high or adverse, and latency and efficiency are a major concern, oblivious Knaive routing is the most adequate. Regardless of the topology being king mesh or torus, this algorithm requires a single virtual channel. For higher levels of uneven traffic, when hotspots are expected to form, it is necessary to use an adaptive routing algorithm. For the king topologies the ideal is the 2S hop-by-hop, as it takes full advantage of the available path diversity without compromising the balanced use of links in uniform traffic. To allow adaptive routing, this solution requires two virtual channels.

Some traffic patterns where packets make an extensive use of the diagonals, like the transpose, the king topologies with minimal-distance routing do not give the expected result. This is because diagonal paths do not have alternatives, so traffic can not spread. The only way to overcome this limitation is to use misrouting. Traditional misrouting approaches like Valiant duplicate the average latency, but the proposed King$\epsilon\delta$ algorithm is capable of giving good throughput results comparable to Valiant, but without increasing the average latency as much. Furthermore, unlike the Valiant algorithm, King$\epsilon\delta$ does not require an extra virtual channel to avoid deadlock.

As the size of interconnection networks grow, so does the probability of one of their components to fail. For this reason this work includes an analysis of fault-tolerant routing algorithms. The best performance is obtained by an adaptation of Immucube to the king topologies, but it requires three virtual channels. Whereas the proposed KFT algorithm offers similar performance but using only two.

Chapter 4

Technological issues in router design

So far, the previous chapters have studied aspects of interconnection networks that are far from the physical world. This mainly mathematical approach is as fundamental as the study of technological issues that arise in the physical implementation of the networks. If in the previous chapters the focus was on performance alone, this chapter bestows a great importance in the efficiency. Thus, a great effort is made to provide reasonable estimations of cost, both fabrication and exploitation. Combining these estimations with performance predictions it is possible to evaluate the efficiency of the different interconnection networks presented in this work.

The king topologies presented in this work are not bound to a particular scenario. They can be implemented as system networks or as networks-on-chip. Taking into account current technology, deploying a king topology in a system context does not pose a significant challenge. Indeed, direct network machines with equal or higher radix are in operation nowadays [ASS09]. The picture is quite different within the chip, where current developments are based on standard meshes [Mor15] or even rings [RH13]. Thus this chapter focuses on evaluating the viability of implementing king topologies in a network-on-chip context.

4.1 Baseline router description

It has been stated that the topology is the most important feature of an interconnection network. Topologies are articulated by edges and vertices, or under a more practical light, links and routers. After the topology, it is this second component, the router, that receives the most attention, since a poorly designed router will reduce the performance of the interconnection network.

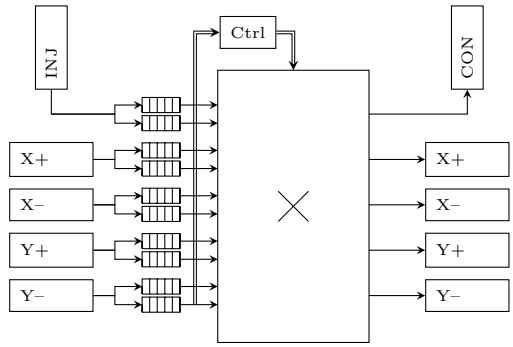


Figure 4.1: Basic router organization for standard meshes and tori.

But from a cost perspective the router has a primordial importance, as it is the part of the network that manages the transmitted data: driving it through the links, storing it when there is congestion... Thus, it is vital to understand the different parts of the router, and how they must be changed to implement new topologies. Figure 4.1 shows the structure of a typical router for a bidimensional network, such as a mesh or a torus. These topologies have degree four, and therefore the router has four input ports devoted to receive packets from the four directions (Labeled X+ through Y–), as well as four output ports that will send packets to the different neighbouring routers. In addition, the router has two extra ports, called injector and consumer, that receive packets from and deliver them to the associated computing element.

The input ports are usually capable of splitting the incoming traffic into virtual channels, and store them in the corresponding queue. Although this can be highly beneficial to performance, the number and size of the queues have a heavy impact on the final cost of the router. The output ports contain the circuits to drive the links between routers. The length of the latter will condition the size of the transistors in the driver circuits. The design of the output ports must guarantee that the timing constraints are met.

Also important when considering the cost of the router is the crossbar. This component acts like a telephone switchboard between the input queues and the output ports. This also accounts for a significant part of the cost of the router. The degree of the topology affects the dimensions of the crossbar in a quadratic manner.

In addition to these data-path components, there is also a set of control components or parts, like the arbiter, segmentation registers or the clock distribution network. The individual cost of these parts is not significant, compared to that of the input ports or the crossbar. However it makes sense to consider their combined influence in the total cost of the router.

4.2 King network area and energy estimation

As in any engineering discipline, technological advances have to be backed up by a sound cost evaluation. For electronic devices, the evaluation usually covers two main moments in the life of the device, the fabrication stage, and during its operation lifetime. In the chip-making industry it is a common assumption to consider the fabrication cost proportional to silicon area used by the device. At an early design stage it is difficult to obtain the area of a given device, thus several efforts have been made to come up with models to give reasonable estimations without the need of complex VLSI tools. These models are usually able to give energy consumptions of the devices, which is what dominates the cost of operation. Each field of computer architecture has a few models highly specialised to it.

The on-chip network field is no exception [SCK+12, KLN12]. These models take a set of design parameters, like topology, bit-widths, clock frequency or buffer sizes, and make an internal representation of all the transistors in the device. Sometimes the latter are adjusted to meet timing specifications. The model then processes the transistor representation and computes the required area. It also calculates the static component of the energy consumption, which does not depend on the data transferred, thus also called *Non-Data-Dependent (NDD)* energy. And finally it can give an expression of the energy cost of different events, like a flit traveling between two routers, or performing a broadcast. Then if this information is combined with a functional simulation of the device, the total energy consumption can be calculated.

Before delving into a detailed cost analysis, its is worth pointing out some differences between the baseline router described above and the one used for king meshes. The first and foremost is the degree of the network. King meshes have double as many edges per router as the regular meshes. This means that the router has to duplicate the number of ports, queues and other internal structures. A graphical depiction of the king router is shown in Figure 4.2.

In addition, the edges in the king networks are of two different lengths. For a given separation between routers, L, the orthogonal links measure L while the diagonal links measure $\sqrt{2}L$. It is assumed that the extra length in the diagonal links has no repercussion in the die area. However it will have an influence in the consumption, as the driving transistors will have to be larger to satisfy timing constraints, or more repeaters will be required.

The following analysis is based on results obtained from the DSENT model [SCK+12]. The Table 4.1 shows a list of the parameters used when executing the model.

As can be seen in Table 4.2, different parts of the router scale differently

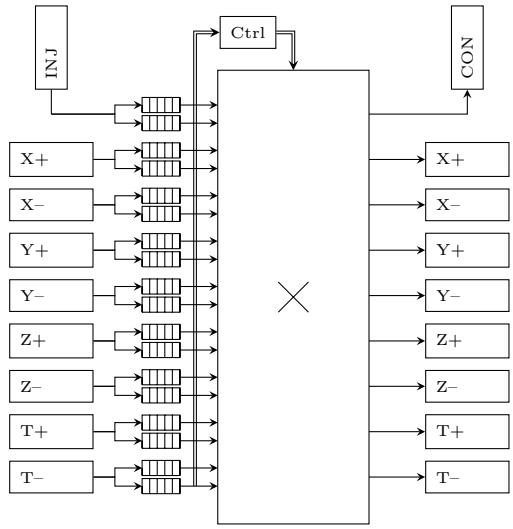


Figure 4.2: Basic king router organization.

Parameter	Value
CPU count	64
CPU separation	1 mm
Flit width	64 bits
CPUs per router	1
Clock frequency	1 GHz
Packet size	16 flits
Virtual networks	1
Virtual channels per port	2
Buffer space per virtual channel	4 flits
Buffer model	DFFRAM
Crossbar model	MultiplexerCrossbar
Arbiter model	MatrixArbiter
Clock tree model	BroadcastHTree
Technology	45nm LVT

Table 4.1: DSENT model parameters.

with the number of ports. Since the king topology duplicates the number of links of the mesh, it follows that the cost of the input ports will be doubled. The table shows that the same happens with the control circuitry. But in contrast, the crossbar is known to grow quadratically with the number of ports.

So far, the area and static energy consumption have been analysed together as the method of estimation is similar. To obtain an estimation of the dynamic energy, the traffic must be taken into account. The model used by

Magnitude		Mesh	King mesh	Increase
Area (mm^2)	Crossbar	0.00593	0.02142	3.61
	Input ports	0.02082	0.03748	1.80
	Control	0.00516	0.00922	1.79
	Total	0.03191	0.06812	2.13
Ndd Power (W)	Crossbar	0.00114	0.00473	4.15
	Input ports	0.00616	0.01109	1.80
	Control	0.00159	0.00285	1.79
	Total	0.00889	0.01867	2.10

Table 4.2: Area and static power comparison of mesh and king mesh baseline routers.

DSENT is optimistic in the sense that it considers that flits traverse the network unaffected by contention. That is, it ignores the fact that a flit might wait several cycles at a router for its destination port to become available. Although it can be argued that such events do not represent a large data transfer, the process of choosing a blocked port does add a small amount to the energy cost. And because of the large amount of such events at high loads, their combined contribution might not be negligible. At any rate, this can account for the fact that DSENT does not give good energy estimations at high loads, when the network is congested.

The dynamic energy estimation of DSENT is based on calculating the cost of moving a flit between two routers at the average distance of the network, this is called a *average unicast event*. Due to this and the above constraints, DSENT can give dynamic energy consumptions for uniform traffic patterns at low load, which is an acceptable assumption in NoCs. To make an estimation of the total dynamic energy of a network without using real traffic statistics, the energy of the average unicast event is multiplied by the average injection rate per router, the number of routers and the frequency. Knowing that the average unicast event is computed for the average number of hops in the topology, the resulting value is the energy per cycle for uniform traffic.

This way of estimating the dynamic energy is adequate when comparing the efficiency of solutions that give the same performance. However, Section 2.2 stated that the average distance in the king networks is 30% lower, so the latency will be proportionally lower, and the execution times of applications will be shorter. In addition, the bisection bandwidth of the king networks is three times as large, so phases of high traffic are going cause lower congestion levels than in meshes, which will have a positive effect on the performance. Therefore, an efficiency metric is required that factors the performance in. The *Energy-Delay Product (EDP)*[GH95] is such a metric. It gives the idea of physical work carried out by a device that spends an amount of energy

during a given time. Then, a device that consumes more but performs better, can have a lower EDP than a device that consumes less but takes a long time to conclude the same task.

To evaluate the EDP of the two routers an experiment is set up in which a typical collective communication takes place. An All-to-all collective is known to have uniform traffic distribution, and the average hop count of the packets is equal to the average distance of the topology. For the network under study with 64 routers, it involves sending more than four thousand packets. The delay of the collective is measured by noting the reception time of the last packet in the network. Then it is possible to determine the dynamic energy consumption of such a communication. Table 4.3 summarises the estimations of the energy cost of meshes and king networks during a single all-to-all exchange, taking into account the dynamic energy and the run time. It is noteworthy that the consumption in terms of dynamic energy is significantly less in the king network. This is a consequence of the shorter distances the packets must travel to reach their destination. Because the static energy is only considered during the time there is traffic, its value is also reduced, even if it is still higher than in the mesh. Combining dynamic and static consumptions shows that the total energy is the same for king and standard meshes. But since the king network performs the all-to-all in less time, the EDP is reduced considerably.

Configuration	Mesh	King mesh	Increase
Avg. distance (hops)	5.33333	3.73333	0.70
Avg. unicast event (J)	3.97633×10^{-11}	3.47215×10^{-11}	0.87
Delivery time (cycles)	1914	1199	0.62
Delivery time (s)	1.91×10^{-6}	1.20×10^{-6}	0.62
Dynamic Energy (J)	2.56521×10^{-6}	2.23995×10^{-6}	0.87
EDP dyn	4.89955×10^{-12}	2.68794×10^{-12}	0.55
Static energy (J)	1.08671×10^{-6}	1.43309×10^{-6}	1.31
Total energy (J)	3.65192×10^{-6}	3.67304×10^{-6}	1.01
EDP total	6.97517×10^{-12}	4.40765×10^{-12}	0.63

Table 4.3: Comparison of baseline network configurations during an all-to-all collective.

The behaviour of real applications is composed of bursts of high traffic, like the all-to-all studied above, interspersed with periods of intensive computation, in which the network is hardly used. In this scenario, the power consumption of the king networks will be higher than for the meshes. Although during the high traffic times the king networks will be the most efficient, this will be overshadowed by the increased static consumption during the computation phases.

It is difficult to model this behaviour with synthetic traffic, and the EDP metric is focused evaluating the efficiency of a given task. In order to make efficiency evaluations with synthetic traffic, a new metric is introduced. This is the *Energy per phit (EPP)*. It denotes the relation of the energy consumed in a given task divided by the number of phits transferred. In Figure 4.3 the EPP of the two networks is shown for different injection rates, or loads. It can be seen that for very low loads, when the dynamic cost is negligible, the king networks require double as much energy per phit as the normal mesh. As the load is increased, in both cases the static energy is amortized over more phits, thus giving lower EPP. However, as king networks have lower dynamic energy consumption, at higher loads they are able to achieve lower EPP values than the mesh.

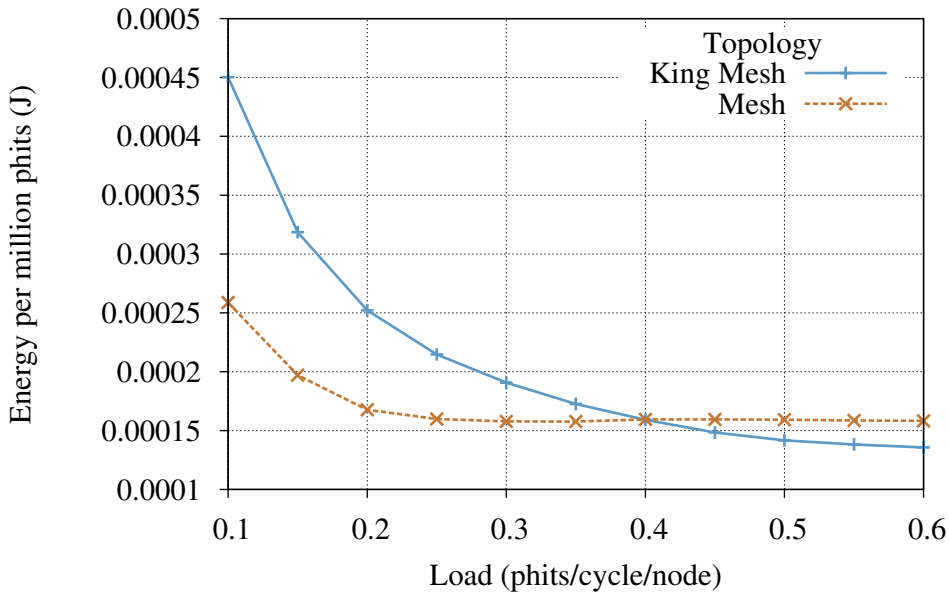


Figure 4.3: Energy per phit comparison between baseline routers.

4.3 Proposals to improve the EPP in king networks

The implementation of the routers in the previous section has been done assigning the same amount of resources per port for both topologies. For instance, from the buffer space perspective, the king router has double as much as the mesh. But it has been seen in Section 2.3 that the bisection bandwidth of the king topologies is three times larger than in the standard topologies. As a consequence they are able to cope with higher levels of

traffic. With this in mind, it can be argued that a king mesh can give the same or better performance as a mesh, but using less resources. This Section proposes alternative ways of implementing the king router, so that its static consumption is equivalent to that of the mesh, while still giving the best performance.

4.3.1 Equalising pin count

As a first approach, it is worth considering the pin counts of both routers [Aga91]. To make it easier to draw conclusions, only the wires of the links connecting routers are counted. With the configuration presented in Table 4.1, the mesh has $64 \times 4 = 256$ outgoing wires, while the king router has double as much. By halving the phit width, it is possible to build a king router with the same pin count as the mesh.

The area and power analysis presented in Table 4.4 shows that the new king router has roughly the same footprint and static consumption as the mesh. It is noteworthy that the crossbar in the king router is still considerably larger than in the mesh, however this is compensated by the smaller size of the input ports and other circuitry.

Magnitude		Mesh	King mesh base.		King mesh	
		baseline	Value	Incr.	Value	Incr.
Area (mm^2)	Crossbar	0.00593	0.02142	3.61	0.01109	1.87
	Input ports	0.02082	0.03748	1.80	0.01913	0.92
	Control	0.00516	0.00922	1.79	0.00465	0.90
	Total	0.03191	0.06812	2.13	0.03487	1.09
Ndd Power (W)	Crossbar	0.00114	0.00473	4.15	0.00323	2.84
	Input ports	0.00616	0.01109	1.80	0.00566	0.92
	Control	0.00159	0.00285	1.79	0.00143	0.90
	Total	0.00889	0.01867	2.10	0.01032	1.16

Table 4.4: Area and static power comparison of baseline routers with reduced pin-count king router.

Taking the dynamic consumption into account, the Figure 4.4 shows the EPP values for the standard mesh and the king mesh with the reduced pin count. It has to be pointed out that the synthetic traffic used for the king network has duplicated the length of the packets to compensate for each phit carrying half the information as before. The figure shows that the low traffic consumption is comparable to that of the mesh, even if slightly higher. But at medium or high traffic, the king network improves the efficiency of the mesh in more than ten percent.

Although this could be considered as a promising result, the truth is that this approach sacrifices one of the key advantages of the king topologies, the latency. The fact that the phits are smaller causes packets to be longer, this duplicates their spooling latency and ultimately delays the delivery.

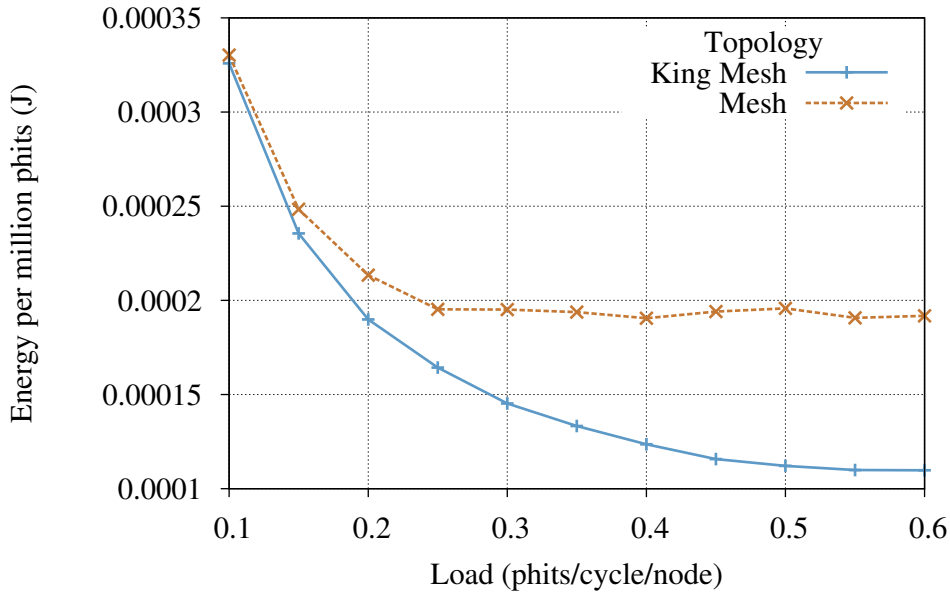


Figure 4.4: Energy per phit comparison of mesh baseline router with reduced pin-count king router.

4.3.2 Equalising buffer space

Observing that the input ports have a high significance in the cost of the router, a different approach is to reduce the amount of buffer space only, and leave the rest of the router untouched. In contrast to the previous solution, this will not affect the size of the packets.

The baseline routers have space for four phits on each queue. In Table 4.5 these are compared with a king router with half the queue length. Obviously, the size of the crossbar and other components does not change. In gray, the table shows that the area and power of the input queues is halved, and as a consequence, the total area and consumption are only around 50% higher than in the baseline mesh router.

Although the static power is higher than in the mesh, it is compensated by the topological advantages of the king network. Even for low loads, the total energy per phit is smaller than in the mesh, as is shown in Figure 4.5. The decreasing tendency of the EPP is maintained by both routers as

Magnitude		Mesh	King mesh base.		King mesh	
		baseline	Value	Incr.	Value	Incr.
Area (mm^2)	Crossbar	0.00593	0.02142	3.61	0.02142	3.61
	Input ports	0.02082	0.03748	1.80	0.01839	0.88
	Control	0.00516	0.00922	1.79	0.00922	1.79
	Total	0.03191	0.06812	2.13	0.04903	1.54
Ndd Power (W)	Crossbar	0.00114	0.00473	4.15	0.00473	4.15
	Input ports	0.00616	0.01109	1.80	0.00547	0.89
	Control	0.00159	0.00285	1.79	0.00285	1.79
	Total	0.00889	0.01867	2.10	0.01305	1.47

Table 4.5: Area and static power of baseline routers and king router with half the buffer space.

the load is increased. However, the king network can actually double the efficiency of the mesh at high loads.

Despite these results, it is important to note that the effect of reducing the buffer space can degrade the performance with bursty traffic [BCGW11].

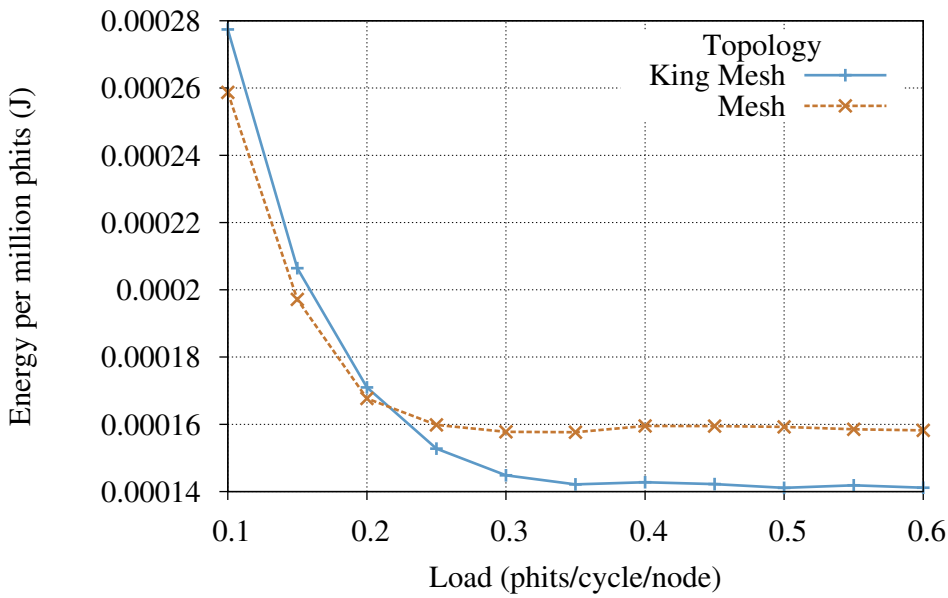


Figure 4.5: Energy per phit comparison mesh baseline router with reduced buffer king router.

4.3.3 Concentration

The shortcomings of the previous proposals are solved when considering the concept of *concentration*. The concentration factor dictates the number of CPUs, or cores, connected to each router. This approach has been used many times in the past, but its application to king networks is full of sense. Section 2.3 stated that the maximum throughput for small king networks will be larger than one. This is ideal for concentration, as the underlying topology is capable of handling heavier traffic coming from more CPUs connected to each router.

From a cost perspective, at a concentration factor of two, the king network is slightly higher than a mesh, as can be seen in Table 4.6. Note that, due to the different number of routers in each configuration, the costs presented in this table combine that of all the routers. This extra static power consumption is surely compensated by the improved delivery times of the king mesh. However, the comparison is unfair, as only one of the networks can be square. The other must have double as many routers on one side. It has been seen in [CMV+10] that the performance of such networks will be conditioned by a smaller bisection bandwidth, as well as an uneven traffic pressure among the different directions.

Magnitude		Mesh	King mesh C=1		King mesh C=2	
		baseline	Value	Incr.	Value	Incr.
Area (mm^2)	Crossbar	0.37952	1.37088	3.61	0.86848	2.29
	Input ports	1.33248	2.39872	1.80	1.33248	1.00
	Control	0.33024	0.59008	1.79	0.32768	0.99
	Total	2.04224	4.35968	2.13	2.52864	1.24
Ndd Power (W)	Crossbar	0.07296	0.30272	4.15	0.19584	2.68
	Input ports	0.39424	0.70976	1.80	0.39424	1.00
	Control	0.10176	0.18240	1.79	0.10112	0.99
	Total	0.56896	1.19488	2.10	0.69120	1.21

Table 4.6: Area and static power of baseline meshes with king mesh with concentration factor 2.

Therefore, the concentration factor has been increased to four, where both networks can be square. For this alternative, the cost values are summarized in Table 4.7. It shows that, as could be expected, the total cost of the king mesh is smaller than for the standard mesh. An interesting fact is that the crossbars do not reduce their total size or static consumption. This is due to the fact that the number of ports is increased to allow the connection of more CPUs per router. Although it seems this should be compensated by having less routers, due to concentration. As is the case with the input ports, for

instance. But, because the crossbars grow quadratically with the number of ports, their total area increases.

Magnitude		Mesh baseline	King mesh C=1		King mesh C=4	
			Value	Incr.	Value	Incr.
Area (mm^2)	Crossbar	0.37952	1.37088	3.61	0.65856	1.74
	Input ports	1.33248	2.39872	1.80	0.79936	0.60
	Control	0.33024	0.59008	1.79	0.19648	0.59
	Total	2.04224	4.35968	2.13	1.65445	0.81
Ndd Power (W)	Crossbar	0.07296	0.30272	4.15	0.15296	2.10
	Input ports	0.39424	0.70976	1.80	0.23680	0.60
	Control	0.10176	0.18240	1.79	0.06080	0.60
	Total	0.56896	1.19488	2.10	0.45056	0.79

Table 4.7: Area and static power of baseline meshes with king mesh with concentration factor 4.

Figure 4.6 presents the energy per phit required by the mesh compared to the king mesh with concentration four. It shows that the efficiency of the concentrated solution is better. This is explained considering several arguments. First, obviously the static consumption is 20% less than the mesh. Second, the smaller size of the network makes the latency shorter, contributing to a better EDP factor. And third, the king networks have a larger bisection bandwidth, allowing them to admit more traffic. Which is important, as one of the shortcomings of concentration is that a smaller network will reduce the bisection bandwidth. For this reason, king networks are specially suited to concentration, as their bisection bandwidth is relatively large.

4.4 Concluding remarks

This chapter presented a preliminary cost analysis of the king networks in the on-chip context. It showed that the extra cost of these topologies is not always compensated by their good performance. As a consequence, a search was conducted to find a lighter configuration of the router, where the cost would be similar to that of a standard mesh router. The best solution found was not to modify the king router and apply concentration. It is known that the bisection bandwidth of the king networks is above one phit per cycle per router, as several injectors were required to fully saturate them. Therefore, reducing the cost by halving or quartering the number of routers is full of sense. The resulting concentrated network still has bisection bandwidth to manage high traffic bursts, but in addition its average distance is smaller, thus further reducing the average latency.

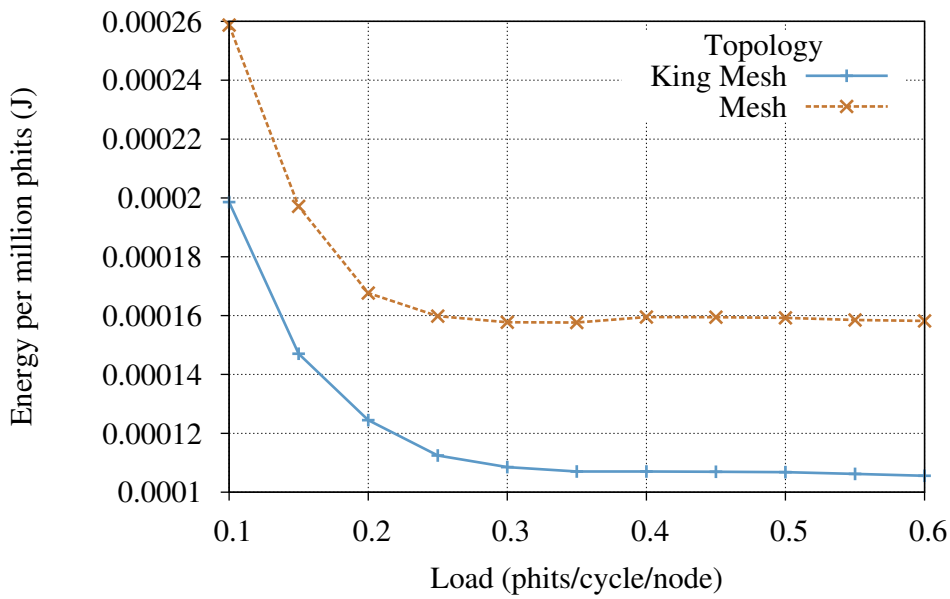


Figure 4.6: Energy per phit comparison of baseline mesh with king mesh with concentration factor 4.

Chapter 5

Conclusions and Future work

Through the completion of this thesis it has become apparent that the field of computer architecture is of utmost importance for the development of faster and more efficient computers. Furthermore, regardless of their scale, modern machines rely heavily on parallelism. From the massive number-crunching machines of the TOP500, to the hand-held devices most people carry, it is difficult to find examples of single processing units. All requiring an interconnection network tuned to their requirements. Due to the effect of the four walls of processor design, thwarting attempts to design faster processors, the importance of interconnection networks is on a constant growth. The scalability of modern and future parallel architectures is highly dependant on the effectiveness of these components.

If the interconnection networks of big machines are mostly implemented with cables, optical or otherwise, the small-scale devices integrate several processors in a single chip, demanding an on-chip network. As the technology improves, it is feasible to fit more and more processors where, only a few years ago, there was room for only one. This only rises the importance of the on-chip interconnection networks.

The design of on-chip networks shares a large amount of knowledge with traditional system networks, but also brings new challenges to engineers. For instance, the current difficulty stacking several layers of processors on a chip, forces designers to choose topologies that can naturally adapt to the plane, as is the case of some of the networks studied here.

This thesis proposes a new topology, the king networks, that could be used in fault tolerant system networks but they are particularly suited for the networks-on-chip environment, due to their planar disposition. Comparing them with other planar topologies, it becomes apparent that they have outstanding characteristics that make them worthy of consideration in future parallel computer systems.

Even though the king networks duplicate the number of links of traditional meshes or tori, they triple the throughput of the latter. Thus being

able to deal with higher traffic volumes. In addition, the reduction of their average distance and diameter procures shorter latencies and faster response times. On top of this, king meshes are easily partitioned in smaller versions of themselves. This property is specially attractive as it allows designers to conveniently organise small tasks, or evenly spread scarce resources, on large networks.

Regardless of the apparent advantages of a topology, it is worthless without an adequate routing algorithm. This thesis devotes a great effort to finding the best routing algorithms for king topologies. Depending on the application, different algorithms, with different advantages are required. For this reason, several algorithms are presented, giving answer to minimal distance routing, misrouting or fault-tolerant routing necessities.

In addition to the previous, less technological study of the king networks, this thesis presents a preliminary cost evaluation in the context of on-chip-networks. Which allows an estimation of the extra cost incurred when using these topologies. In light of this, some implementation alternatives are presented to make an efficient use of resources while preserving the attractive properties of king networks.

These and other interesting findings and achievements are further detailed in the following conclusions.

5.1 Topology

Through the topological analysis, a consistent set of expressions for different parameters of the networks was obtained. These have allowed to formulate predictions of performance, and explain behaviours observed under the effect of different traffic patterns and routing algorithms.

For instance, it was possible to conclude that a king mesh, although it duplicates the number of links, it is noticeably better than a standard torus. They both have the same diameter, but the average distance of the king mesh is 7% less and the bisection bandwidth is 50% more. A further advantage of the king mesh is the implementation complexity, as it does not require a folding scheme.

If at the beginning of the topological study it was thought that king networks, due to their higher number of links, possessed a better path diversity, the study proved the contrary. Rather than considering this a drawback, it is taken as an advantage as the extra links can be seen as shortcuts. This has the effect of reducing the latency, and therefore, allows packets to leave the network sooner, causing less congestion. Reflecting upon this leads to the conclusion that path diversity and latency are coupled values in this kind of topologies, as unit latency implies a fully connected graph, and increasing the path diversity can only be achieved by increasing the number of hops.

Another remarkable property of king topologies is their convenient partitioning possibilities. A common problem in parallel computer systems is resource placement, the resolution of which divides the network in a set of regions with similar shape and a central router. In square standard networks, the shape of these regions contrasts to the shape of the network as they are rhomboid. In fact, in standard meshes the regions are not even the same shape. In contrast, king networks offer an elegant solution to this problem. The regions are square, as the network, and the central routers are also placed on square grid. In a network-on-chip environment, this allows designers to evenly distribute memory controllers, or gateway routers for a hierarchical network topology.

5.2 Routing algorithms

Taking advantage of the singular characteristics of the king networks can only be done with the appropriate routing algorithm. This thesis presents several, each adapted to different needs. All of them have been designed to rule out communication anomalies like, deadlocks, livelocks or starvation.

For fast communication in low traffic conditions, the oblivious Knaive routing is the best choice. It focuses on giving the lowest latencies as well as establishing a natural balance in resource utilisation under uniform traffic. When traffic conditions worsen, and a fair amount of congestion is expected, the hop-by-hop algorithm is best. These try to maximise the utilisation of the path diversity of the network, while still maintaining the minimum distance restriction and homogeneous resource utilisation in uniform traffic.

When the traffic patterns that are expected will cause severely uneven congestion, it is advised to use the misrouting algorithm presented, King$\epsilon\delta$. This algorithm allows packets to diverge from the minimum distance path, thus permitting them to circumvent problem areas. Performancewise it is better than the classic Valiant aproach, as it does not duplicate the average distance, and does not require an extra virtual channel for deadlock avoidance.

Most applications rely on collective communications, both in distributed and shared memory computers. Thus, a broadcast algorithm was proposed, that can be evolved into a multicast algorithm.

The large number of components in a modern computer forces designers to take into account their failure. For this reason, this thesis tests the king network in the presence of faulty links. To overcome the problem of losing links, two well-known fault tolerant routing algorithms have been adapted to operate the king networks. Furthermore, a better scaling alternative was proposed specifically for the king networks, called KFT. This approaches the routing around faulty links in a local manner, thus allowing the rest of the

network to operate normally.

5.3 Router design

This thesis also assessed the cost of the king network routers, and compared them to the cost of a router for standard 2D topologies. By means of this comparison it was pointed out that the improved performance offered by the king networks not always justified their cost. Thus, a series of rearrangements of the routers was made in order to find a more efficient configuration. These include compensating the doubled degree of the router by halving the width of the phit, and reducing the length of the queues. However the best solution was obtained by applying concentration. The king networks do have an excess of bisection bandwidth, this can only be exploited through the use of more than one source of traffic per router. This situation is ideal for concentration, as it allows a reduction of the cost of the network without excessively restricting the throughput.

5.4 Future work

Any new proposal is just an open door to new challenges. At this point in the study of king topologies, they still offer a number of open lines for future research. Due to numerous reasons king meshes are particularly suited to on-chip solutions. Their flat design can be straightforwardly implemented in a flat substrate. Also, their reduced latency hints that they will have a positive impact in the performance on shared memory computers that use king topologies as interconnection networks in the memory hierarchy. Because they convey cache lines from shared caches to memory and processors, a shorter delivery time implies a shorter miss penalty.

Furthermore, the ability of king networks to be partitioned easily in locality spheres with a central router suggests that these routers could be used for special purposes. In the context of shared memory computers, they could be populated with memory controllers, cache directories or accelerators.

It has been seen that a king router roughly duplicates the consumption of a mesh router. This is caused mainly by the influence of the queue space. Assuming that cache traffic is not very intense, it might be worth pursuing a buffer-less, or shared buffer configuration of the router [HY14]. Also, because king meshes have a larger throughput than standard meshes, they can admit higher concentration factors, thus achieving higher efficiencies. It would be interesting to use a more detailed simulation environment to prove these points.

The proposed hop-by-hop routing algorithm distinguished the output

ports of a router in three groups, depending on how they approached the destination router. Only two of these groups were profitable, and only these can be used for a minimum distance routing algorithm. However, the non-profitable ports could be used to devise a misrouting algorithm. The thesis already proposes a misrouting algorithm that can be tuned through two parameters, but it is based on tables. It can be argued that tables do not scale well with the number of routers. The new table-less algorithm could use the two parameters to influence the port choices when routing packets. For example, the δ can be related to the number of times a packet can be routed through a non-profitable port.

The fault-tolerant routing algorithm that is described in this thesis could incur in poor performance in some pathological fault configurations. This lack of performance has been attributed to the unfairness of the DOR routing in the escape channels. Since then, a new routing algorithm has been developed called σDOR that could be used instead of DOR to overcome these problems. Some tests in this direction would be interesting.

5.5 Publications

The research presented in this thesis revolves around the study and evaluation of the king topologies. These have their origin in work developed in the field of coding theory. Which, like the field of interconnection networks, relies on graph theory. As a consequence, findings in one field are often translatable to the other.

The following publications stem from a basic line of research around coding theory. During the development of this research, the idea of using king topologies in interconnection networks became worthy of a more applied research.

- Carmen Martínez, Esteban Stafford, Ramón Beivide, and Ernst Gabidulin. Perfect Codes over Lipschitz Integers. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 1366–1370, June 2007.

- Carmen Martínez, Esteban Stafford, Ramón Beivide, and Ernst Gabidulin. Modeling hexagonal constellations with Eisenstein-Jacobi graphs. *Problems of Information Transmission*, 44(1):1, 2008.

- Carmen Martínez, Esteban Stafford, Ramón Beivide, Cristóbal Camarero, Fernando Vallejo, and Ernst Gabidulin. Graph-based metrics over QAM constellations. In *IEEE Int. Symp. on Information Theory*, page 2494, 2008.

- Carmen Martínez, Ramón Beivide, Esteban Stafford, Miquel Moreto, and Ernst Gabidulin. Modeling Toroidal Networks with the Gaussian Integers. *Computers, IEEE Transactions on*, 57(8):1046–1056, aug. 2008.

- Carmen Martínez, Ramón Beivide, Cristóbal Camarero, Esteban Stafford, and Ernst Gabidulin. Quotients of Gaussian graphs and their application to perfect codes . *Journal of Symbolic Computation* , 45(7):813–824, 2010. Algebraic Coding Theory and Applications .

The idea of king topologies for interconnection networks was deemed worthy of a more applied research, which has led to the following contributions.

- Esteban Stafford, Jose Luis Bosque, Carmen Martínez, Fernando Vallejo, Ramón Beivide, and Cristobal Camarero. A first approach to king topologies for on-chip networks. In *Euro-Par: Parallel Processing*, pages 428–439, Berlin, Heidelberg, 2010. Springer-Verlag.

- Esteban Stafford, Jose Luis Bosque, Carmen Martínez, Fernando Vallejo, Ramón Beivide, and Cristobal Camarero. A First Approach to King Topologies for On-Chip Networks. *XXII Jornadas de Paralelismo*, 2011.

- Emilio Castillo, Esteban Stafford, Fernando Vallejo, Jose Luis Bosque, Carmen Martínez, Cristobal Camarero, and Ramón Beivide. Study of Fault Tolerance for King Topologies. *XXIII Jornadas de Paralelismo*, 2012.

- Esteban Stafford, Emilio Castillo, Fernando Vallejo, Jose Luis Bosque, Carmen Martínez, Cristobal Camarero, and Ramón Beivide. King Topologies for Fault Tolerance. In *HPCC-ICESS*, pages 608–616, 2012.

- Esteban Stafford, Carmen Martínez, Jose Luis Bosque, Fernando Vallejo, Cristobal Camarero, Borja Perez, and Ramón Beivide. Source misrouting in King topologies. In *HPCC-ICESS*, 2014.

- Esteban Stafford, Jose Luis Bosque, Carmen Martínez, Fernando Vallejo, Ramón Beivide, Cristóbal Camarero, and Emilio Castillo. Assessing the Suitability of King Topologies for Interconnection Networks. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–1, 2015.

Bibliography

[A+02] N. R. Adiga et al. An Overview of the BlueGene/L Supercomputer. In *ACM/IEEE Conf. on Supercomputing*, page 60, 2002.

[ABC+05] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken, M. Tsao, and P. Vranas. Blue Gene/L torus interconnection network. *IBM Journal of Research and Development*, 49(2.3):265–276, March 2005.

[AEJK+09] Dennis Abts, Natalie D. Enright Jerger, John Kim, Dan Gibson, and Mikko H. Lipasti. Achieving predictable performance through better memory controller placement in many-core CMPs. *ACM SIGARCH Computer Architecture News*, 37(3):451, 2009.

[Aga91] A. Agarwal. Limits on interconnection network performance. *Parallel and Distributed Systems, IEEE Transactions on*, 2(4):398–412, 1991.

[AGK03] George Karypis Ananth Grama, Anshul Gupta and Vipin Kumar. *Introduction to Parallel Computing.* Pearson, second edition, 2003.

[AHKB00] Vikas Agarwal, M. S. Hrishikesh, Stephen W. Keckler, and Doug Burger. Clock rate versus IPC: the end of the road for conventional microarchitectures. In Alan D. Berenbaum and Joel S. Emer, editors, *ISCA*, pages 248–259. IEEE Computer Society, 2000.

[And13] Mark Anderson. Best Supercomputers Still Not Best for Big Data. *IEEE Spectrum*, 2013.

[Ant13] Sebastian Anthony. Intel unveils 72-core x86 Knights Landing CPU for exascale supercomputing.

http://www.extremetech.com/extreme/171678-intel-unveils-72-core-x86-knights-landing-cpu-for-exascale-supercomputing, November 2013.

[ASS09] Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu. Tofu: A 6D mesh/torus interconnect for exascale computers. *Computer*, 42(11):36–40, 2009.

[AV94] V. S. Adve and M. K. Vernon. Performance Analysis of Mesh Interconnection Networks with Deterministic Routing. *IEEE Trans. Parallel Distrib. Syst.*, 5(3):225–246, mar 1994.

[BBB+91] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks&Mdash;Summary and Preliminary Results. In *ACM/IEEE Conf. on Supercomputing*, Supercomputing '91, pages 158–165, New York, NY, USA, 1991. ACM.

[BBB+11] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The Gem5 Simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011.

[BBK+68] George H. Barnes, Richard M. Brown, Maso Kato, David J. Kuck, Daniel L. Slotnick, and Richard A. Stokes. The illiac iv computer. *Computers, IEEE Transactions on*, 100(8):746–757, 1968.

[BCF+95] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and Wen-King Su. Myrinet: a gigabit-per-second local area network. *Micro, IEEE*, 15(1):29–36, Feb 1995.

[BCGW11] M. Bakhouya, A. Chariete, J. Gaber, and M. Wack. A buffer-space allocation approach for application-specific Network-on-Chip. In *Computer Systems and Applications (AICCSA), 2011 9th IEEE/ACS International Conference on*, pages 263–267, Dec 2011.

[BD06] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of the 20th annual in-*

ternational conference on Supercomputing, ICS '06, pages 187–198, 2006.

[BMI+03] Ramón Beivide, Carmen Martínez, Cruz Izu, Jaime Gutierrez, José Ángel Gregorio, and José Miguel Alonso. Chordal topologies for interconnection networks. In *High Performance Computing*, pages 385–392, 2003.

[BWM+12] Arthur S. Bland, Jack C. Wells, Bronson Messer, Oscar R. Hernandez, and James H. Rogers. Titan: Early experience with the Cray XK6 at Oak Ridge National Laboratory. In *CUG 2012: Cray Users Group*, Stuttgart, Germany, 2012.

[CAB+13] Nicholas P. Carter, Aditya Agrawal, Shekhar Borkar, Romain Cledat, Howard David, Dave Dunning, Joshua Fryman, Ivan Ganev, Roger A. Golliver, Rob Knauerhase, Richard Lethin, Benoit Meister, Asit K. Mishra, Wilfred R. Pinfold, Justin Teller, Josep Torrellas, Nicolas Vasilache, Ganesh Venkatesh, and Jianping Xu. Runnemede: An Architecture for Ubiquitous High-Performance Computing. In *Int. Symp. on High Performance Computer Architecture*, HPCA '13, pages 198–209, Washington, DC, USA, 2013. IEEE Computer Society.

[CBGV97] C. Carrion, R. Beivide, J.A. Gregorio, and F. Vallejo. A flow control mechanism to avoid message deadlock in k-ary n-cube networks. In *Proceedings Fourth International Conference on High-Performance Computing*, page 322, 1997.

[CC01] Chun-Lung Chen and Ge-Ming Chiu. A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults. *IEEE Trans. Parallel Distrib. Syst.*, 12(5):467–475, May 2001.

[CEH+11] Dong Chen, Noel A. Eisley, Philip Heidelberger, Robert M. Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L. Satterfield, Burkhard Steinmacher-Burow, and Jeffrey J. Parker. The IBM Blue Gene/Q interconnection network and message unit. In *Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, pages 26–1. ACM, 2011.

[CMV+10] Jose M. Camara, Miquel Moreto, Enrique Vallejo, Ramon Beivide, Jose Miguel-Alonso, Carmen Martinez, and Javier Navaridas. Twisted Torus Topologies for Enhanced Interconnection Networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(12):1765, 2010.

[CO90] I. Cidon and Y. Ofek. Metaring-a full-duplex ring with fairness and spatial reuse. In *INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE*, pages 969–981, 1990.

[Com74] Louis Comtet. *Advanced Combinatorics: The art of finite and infinite expansions*. Springer Science & Business Media, 1974.

[CP13] Lizhong Chen and T. M. Pinkston. Worm-Bubble Flow Control. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pages 366–377, Feb 2013.

[CWP11] Lizhong Chen, Ruisheng Wang, and Timothy M. Pinkston. Critical Bubble Scheme: An Efficient Implementation of Globally Aware Network Flow Control. In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 592–603, Washington, DC, USA, 2011. IEEE Computer Society.

[DA93] W. J. Dally and H. Aoki. Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels. *IEEE Trans. Parallel Distrib. Syst.*, 4(4):466–475, April 1993.

[Dal90] W.J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775, 1990.

[DS86] William J. Dally and Charles L. Seitz. The torus routing chip. *Distributed Computing*, 1(4):187, 1986.

[DS97] Peter Karsuk Dezso Sima, Terence Fountain. *Advanced Computer Architectures: A Design Space Approach*. Addison-Wesley, 1997.

[DT03] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[Dua96] J. Duato. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Transactions on Parallel and Distributed Systems*, 7(8):841, 1996.

[FB10] Mary Flahive and Bella Bose. The Topology of Gaussian and Eisenstein-Jacobi Interconnection Networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(8):1132, 2010.

[FBR+12] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard. Cray Cascade: A scalable HPC system based on a Dragonfly network. In *Int. Conf. on High Performance Computing, Networking, Storage and Analysis*, pages 1–9, Nov 2012.

[Gal96] Mike Galles. Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip. In *Hot Interconnects*, volume 96, 1996.

[GD25] Ioannis Seitanidis Giorgos Dimitrakopoulos, Anastasios Psarras. *Microarchitecture of Network-on-Chip Routers*. Springer, 1025.

[GH95] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose processors. In *Low Power Electronics, 1995., IEEE Symposium on*, pages 12–13, Oct 1995.

[GHR95] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. Oxford University Press, Inc., New York, NY, USA, 1995.

[GN92] Christopher J. Glass and Lionel M. Ni. The Turn Model for Adaptive Routing. *SIGARCH Comput. Archit. News*, 20(2):278–287, apr 1992.

[GNF+06] M. E. Gomez, N. A. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, and O. Lysne. A Routing Methodology for Achieving Fault Tolerance in Direct Networks. *IEEE Trans. Comput.*, 55(4):400–415, 2006.

[Goe00] Richard Goering. Current physical design tools come up short. *EE Times*, 2000.

[Gre15] Daureen Green. EZchip Introduces TILE-Mx100 World's Highest Core-Count ARM Processor Optimized for High-Performance Networking Applications. http://www.tilera.com/News/PressRelease/?ezchip=97, February 2015.

[GRR+08] G. Guindani, C. Reinbrecht, T. Raupp, N. Calazans, and F. G. Moraes. NoC Power Estimation at the RTL Abstraction Level.

In *Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual*, pages 475–478, April 2008.

[Har69] Frank Harary. *Graph Theory.* Addison-Wesley Series in Mathematics. Addison Wesley, 1969.

[HB09] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan and Claypool Publishers, 1st edition, 2009.

[HDH+10] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. Van Der Wijngaart, and T. Mattson. A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS. In *Enterprise Research*, pages 108–109, Feb 2010.

[HL08] Lih-Hsing Hsu and Cheng-Kuan Lin. *Graph Theory and Interconnection Networks.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2008.

[HLB08] Wen-Hsiang Hu, Seung Eun Lee, and Nader Bagherzadeh. DMesh: a Diagonally-Linked Mesh Network-on-Chip Architecture. *Int. Workshop on NoC Architectures (MICRO-41)*, 2008.

[HP11] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[HS08] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[HY14] Syed Minhaj Hassan and Sudhakar Yalamanchili. Bubble Sharing: Area and Energy Efficient Adaptive Routers using Centralized Buffers. In *2014 International Symposium on Networks-on-Chip (NOCS).*, September 2014.

[IK00] Wilfried Imrich and Sandi Klavzar. *Product graphs.* Wiley, 2000.

[IML+02] M. Igarashi, T. Mitsuhashi, A. Le, S. Kazi, Yang-Trung Lin, A. Fujimura, and S. Teig. A diagonal-interconnect architecture and its application to RISC core design. In *IEEE Int. Solid-State Circuits Conf.*, page 210, 2002.

[Jac09] Bruce L. Jacob. *The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It.* Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2009.

[JBM+13] Nan Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96, April 2013.

[KAYC10] Yu-Hsiang Kao, N. Alfaraj, Ming Yang, and H. J. Chao. Design of High-Radix Clos Network-on-Chip. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 181–188, may 2010.

[KDA07] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly. In *Int. Symp. on Computer architecture*, page 126, 2007.

[KDSA08] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Int. Symp. on Computer Architecture*, page 77, 2008.

[KK79] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3(4):267, 1979.

[KLN12] A. B. Kahng, Bill Lin, and S. Nath. Explicit modeling of control and data for improved NoC router estimation. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 392–397, June 2012.

[KMAMP08] M. Koibuchi, H. Matsutani, H. Amano, and T. Mark Pinkston. A Lightweight Fault-Tolerant Mechanism for Network-on-Chip. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, pages 13–22, april 2008.

[KR09] A. Kohler and M. Radetzki. Fault-tolerant architecture and deflection routing for degradable NoC switches. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, pages 22–31, 2009.

[LAS+09] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, New York, NY, USA, 2009. ACM.

[MGC08] J. L. Manferdelli, N. K. Govindaraju, and C. Crall. Challenges and Opportunities in Many-Core Computing. *Proceedings of the IEEE*, 96(5):808–815, May 2008.

[MGH+05] Michael B. Monagan, Keith O. Geddes, K. Michael Heal, George Labahn, Stefan M. Vorkoetter, James McCarron, and Paul DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.

[MHLS14] Sheng Ma, Libo Huang, Mingche Lai, and Wei Shi. *Networks-on-chip: From Implementations to Programming Paradigms*. Morgan Kaufmann, 2014.

[MMSAZ11] R. Moraveji, P. Moinzadeh, H. Sarbazi-Azad, and A. Y. Zomaya. Multispanning Tree Zone-Ordered Label-Based Routing Algorithms for Irregular Networks. *Parallel and Distributed Systems, IEEE Transactions on*, 22(5):817–832, May 2011.

[Mor15] John Morris. Intel's next big thing: Knights Landing Xeon Phi. http://www.zdnet.com/article/intels-next-big-thing-knights-landing, March 2015.

[MSB+08] C. Martinez, E. Stafford, R. Beivide, C. Camarero, F. Vallejo, and E. Gabidulin. Graph-based metrics over QAM constellations. In *IEEE Int. Symp. on Information Theory*, page 2494, 2008.

[MSBG08] C. Martínez, E. Stafford, R. Beivide, and E. M. Gabidulin. Modeling hexagonal constellations with Eisenstein-Jacobi graphs. *Problems of Information Transmission*, 44(1):1, 2008.

[MWBA10] Richard C. Murphy, Kyle B. Wheeler, Brian W. Barrett, and James A. Ang. Introducing the graph 500. *Cray User's Group (CUG)*, 2010.

[MWLJ15] Sheng Ma, Zhiying Wang, Zonglin Liu, and N. E. Jerger. Leaving One Slot Empty: Flit Bubble Flow Control for Torus

Cache-Coherent NoCs. *Computers, IEEE Transactions on*, 64(3):763–777, March 2015.

[NLMA+09] Javier Navaridas, Mikel Luján, Jose Miguel-Alonso, Luis A. Plana, and Steve Furber. Understanding the interconnection network of SpiNNaker. In *Proceedings of the 23rd international conference on Conference on Supercomputing - ICS '09*, page 286, 2009.

[NMAPR11] Javier Navaridas, Jose Miguel-Alonso, Jose A. Pascual, and Francisco J. Ridruejo. Simulating and evaluating interconnection networks with INSEE. *Simulation Modelling Practice and Theory*, 19(1):494, 2011.

[NSLK14] S. G. Nambiar, K. Swaminathan, G. Lakshminarayanan, and Seok-Bum Ko. Central Switch Noded Mesh architecture (CSNM). In *Electronics and Communication Systems (ICECS), 2014 International Conference on*, pages 1–5, Feb 2014.

[PBG+99] V. Puente, R. Beivide, J.A. Gregorio, J.M. Prellezo, J. Duato, and C. Izu. Adaptive bubble router: a design to improve performance in torus networks. In *Int. Conf. on Parallel Processing*, page 58, 1999.

[Pfi01] Greg Pfister. *High Performance Mass Storage and Parallel I/O*, chapter 42, "An Introduction to the InfiniBand Architecture", pages 617–632. IEEE Press and Wiley Press, 2001.

[PG07] Valentin Puente and Jose Angel Gregorio. Immucube: Scalable Fault-Tolerant Routing for k-ary n-cube Networks. *IEEE Trans. Parallel Distrib. Syst.*, 18(6):776–788, June 2007.

[PGVB04] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide. Immunet: A Cheap and Robust Fault-Tolerant Packet Routing Mechanism. *ACM SIGARCH Computer Architecture News*, 32(2):198, 2004.

[Pin04] Timothy Mark Pinkston. *Deadlock characterization and resolution in interconnection networks*, chapter 13, pages 445–492. CRC Press, 2004.

[RCG+13] Nikola Rajovic, Paul M. Carpenter, Isaac Gelado, Nikola Puzovic, Adrian Ramirez, and M. R. Valero. Supercomputing with commodity CPUs: are mobile SoCs ready for HPC? In

High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for, pages 1–12, 2013.

[RFR+10] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 25–32, May 2010.

[RH13] Sabela Ramos and Torsten Hoefler. Modeling Communication in Cache-coherent SMP Systems: A Case-study with Xeon Phi. In *Int. Symp. on High-performance Parallel and Distributed Computing*, pages 97–108, New York, NY, USA, 2013. ACM.

[Ros87] A. W. Roscoe. Routing messages through networks: an exercise in deadlock avoidance. In *Proceedings of 7th Occam User Group Meeting*, pages 55–79, Grenoble, 1987.

[SAL+05] Daeho Seo, Akif Ali, Won-Taek Lim, Nauman Rafique, and Mithuna Thottethodi. Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks. *ACM SIGARCH Computer Architecture News*, 33(2):432, 2005.

[San89] J. L. C. Sanz. *Opportunities and Constraints of Parallel Computing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1989.

[SBM62] Daniel L. Slotnick, W. Carl Borck, and Robert C. McReynolds. The SOLOMON Computer. In *Proceedings of the December 4-6, 1962, Fall Joint Computer Conference*, AFIPS '62 (Fall), pages 97–107, New York, NY, USA, 1962. ACM.

[Sch97] R. R. Schaller. Moore's law: past, present and future. *Spectrum, IEEE*, 34(6):52–59, Jun 1997.

[SCK+12] Chen Sun, C. H. O. Chen, G. Kurian, Lan Wei, J. Miller, A. Agarwal, Li-Shiuan Peh, and V. Stojanovic. DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 201–210, May 2012.

[SD90] K.G. Shin and G. Dykema. A distributed I/O architecture for HARTS. In *Int. Symp. on Computer Architecture*, page 332, 1990.

[SH94] C. B. Stunkel and P. H. Hochschild. Sp2 high-performance switch architecture. In *Hot Interconnects II, 1994. Symposium Record*, pages 190–195, 1994.

[Shi91] K. G. Shin. HARTS: a distributed real-time architecture. *Computer*, 24(5):25–35, may 1991.

[Sin05] Arjun Singh. *Load-balanced routing in interconnection networks*. PhD thesis, Stanford University, 2005.

[SSSF13] Balaji Subramaniam, Winston Saunders, Tom Scogland, and Wu-chun Feng. Trends in Energy-Efficient Computing: A Perspective from the Green500. In *4th International Green Computing Conference*, Arlington, VA, June 2013.

[Str06] Erich Strohmaier. TOP500 Supercomputer. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[Sub13] Suvinay Subramanian. *Ordered Mesh Network Interconnect (OMNI): design and implementation of in-network coherence*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2013.

[TP94] K.W. Tang and S.A. Padubidri. Diagonal and toroidal mesh networks. *IEEE Transactions on Computers*, 43(7):815, 1994.

[Tra07] Tony Trawick. Multicore communication: today and the future. *Embedded Computing Design*, page 29, 2007.

[TRG05] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103, 1990.

[VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing - STOC '81*, page 263, 1981.

[VHR+08] Sriram R. Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Arvind Singh, Tiju Jacob, Shailendra Jain, Vasantha Erraguntla,

Clark Roberts, Yatin Hoskote, Nitin Borkar, and Shekhar Borkar. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1):29, 2008.

[WCP13] Ruisheng Wang, Lizhong Chen, and Timothy Mark Pinkston. Bubble Coloring: Avoiding Routing- and Protocol-induced Deadlocks with Minimal Virtual Channel Requirement. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, pages 193–202, New York, NY, USA, 2013. ACM.

[WGH+07] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. B. III, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27:15–31, 2007.

[WM95] Wm. A. Wulf and Sally A. McKee. Hitting the Memory Wall: Implications of the Obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, mar 1995.

[Zha15] Charles Zhang. Mars: A 64-Core ARMv8 Processor. In *Hot Chips: A Symposium on High Performance Chips*, 2015.