Small-World Interconnection Networks for Large Parallel Computer Systems

Fernando Rodríguez Salazar

A Thesis presented for the degree of
Doctor of Philosophy
to the Department of Electronics and Electrical Engineering,
Faculty of Engineering, University of Glasgow

UNIVERSITY
of
GLASGOW

April 2004

To María and Sofía.

Small-World Interconnection Networks for Large Parallel Computer Systems

Fernando Rodríguez Salazar

Submitted for the degree of Doctor of Philosophy
April 2004

Abstract

The use of small-world graphs as interconnection networks of multicomputers is proposed and analysed in this work. A small-world graph is defined by the superposition of a network which provides the structure of the graph at a local level, and a global network which links remote clusters together. Small-world interconnection networks are constructed by adding (or modifying) edges to an underlying local graph. Graphs with a rich local structure but with a large diameter are shown to be the most suitable candidates for the underlying graph. Generation models based on random and deterministic wiring processes are proposed and analysed. For the random case basic properties such as degree, diameter, average length and bisection width are analysed, and the results show that a fast transition from a large diameter to a small diameter is experienced when the number of new edges introduced is increased. This is similar to the average length reduction which has been analysed elsewhere. Mathematical models for this transition are derived; and existing mathematical models for the average length are modified for an increased accuracy in systems of a small size. Random traffic analysis on these networks is undertaken, and it is shown that although the average latency experiences a similar reduction, networks with a small number of shortcuts have a tendency to saturate as most of the traffic flows through a small number of links. An analysis of the congestion of the networks corroborates this result and provides a way of estimating

the minimum number of shortcuts required to avoid saturation. This work shows that random small-world interconnection networks with a high enough number of shortcuts exhibit rich interconnections and can support traffic very efficiently, with a lower latency, and an increased throughput (in the case of constant pin-out). Unfortunately such graphs are not regular, provide poor algorithmic properties and routing is complicated.

To overcome these problems deterministic wiring is proposed and analysed. A Linear Feedback Shift Register is used to introduce shortcuts in the LFSR graphs, and a Hilbert curve is used to construct the Hilbert graph. Both of these graphs exhibit the same desirable properties of random small-world graphs, such as a small diameter and average length, but present some additional properties, such as regularity, conceptual simplicity and complete determinism. A simple routing algorithm has been constructed for the LFSR and extended with a greedy local optimisation technique. It has been shown that a small search depth gives good results and is less costly to implement than a full shortest path algorithm. The Hilbert graph on the other hand provides some additional characteristics, such as support for incremental expansion, efficient layout in two dimensional space (using two layers), and a small fixed degree of four. It has been compared with meshes and tori and it has been shown that the Hilbert graph provides a reduced diameter and average length, at the expense of a more complicated routing. We anticipate that using an adequate routing algorithm, larger graphs and better expandability can be expected from the Hilbert graph. Small-world hypergraphs have also been studied. In particular incomplete hypermeshes have been introduced and analysed and it has been shown that they outperform the complete traditional implementations under a constant pinout argument, both with a random and a deterministic wiring process, since they provide reduced latency, higher throughput, reduced switch complexity and a lower cost. Since it has been shown that complete hypermeshes outperform the mesh, the torus, low dimensional m-ary d-cubes (with and without bypass channels), and multi-stage interconnection networks (when realistic decision times are accounted for and with a constant pinout), it follows that incomplete hypermeshes outperform them as well.

Declaration

The work in this thesis is based on research carried out at the Department of Electronics and Electrical Engineering, University of Glasgow, Scotland, U.K. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

Thanks go to all the people which with their help and support made this work possible. In particular a big thank you goes to my wife María who had to put up with the late nights, the constant mess and who helped me to organise my ideas and life. To John who gave me the freedom and space to develop this work and who suggested the use of small-world networks to begin with. Scott whom I volunteered to proof read countless drafts and who always provided useful suggestions and to Jeremy who had to put up with a constant bombardment of half solved equations. Thanks also go to my friends who were involved in never ending arguments and helped me to grow both as a person and as a researcher. In particular to Daz, Campbell and Iain. Finally I would like to thank all of the support received from my extended family; some times emotional, some times financial and other times intellectual. In particular to my parents and brothers, and to Rafael, Juan Carlos, Polo and Julio. Finally a big thanks go to Sofía for being here and providing joy to my life.

Furthermore the author wishes to acknowledge the support received from CONACYT and the Mexican government which made this work possible. A big thanks go to them as well!

Contents

List of Publications

During the course of this investigation the following manuscripts have been prepared and submitted for publication:

1. Fernando Rodriguez-Salazar, John R. Barker "Incomplete Hypermeshes: Efficient Interconnection Networks for Pin-Out Limited Systems", Extended Abstract, Proceedings of the 2001 PREP Conference, April 2001.

2. Fernando Rodriguez-Salazar, John R. Barker "Hamming Hypermeshes: High Performance Interconnection Networks for Pin-Out Limited Systems", submitted to Performance Evaluation, 2003.

3. Fernando Rodriguez-Salazar, John R. Barker "Linear Feedback Shift Register Interconnection Networks", to appear in Proceedings of the 2004 Workshop on Massively Parallel Processing, at IPDPS 2004.

Extracts of this thesis have been presented at the following conferences:

1. "Small-World based Interconnection Networks", Invited Talk, Kelvin Nano Conference 2000.

2. "Incomplete Hypermeshes: Efficient Interconnection Networks for Pin-Out Limited Systems", oral presentation at the 2001 PREP Conference, April 2001.

Furthermore the following is a list of manuscripts undergoing final preparations to be submitted:

1. "On the diameter of small-world graphs" to be submitted to Physica E. (This article is based on the material exposed in section 3.5 of this thesis)

2. "Congestion in the Small-World Model" to be submitted to Physica E or Nature. (This article is based on the material exposed in section 3.9 of this thesis)

3. "Hilbert Graph: An Efficient Fixed Degree Interconnection Network" to be submitted to the Journal of Supercomputing. (This article is based on the material exposed in Chapter 5 of this thesis)

List of Figures

List of Tables

List of Symbols

$\mathcal{A}$	An algorithm or function.
β	Inter-cluster distance in a hypermesh.
B	Bisection or bisection width.
B_0	Bisection or bisection width of the underlying graph.
D	Diameter of a graph.
D_R	Routing diameter.
$\mathcal{D}_i(t)$	The number and set of nodes at distance i from node t.
E	The set of edges (links) in a graph.
$\mathcal{G}$	A graph $\mathcal{G} = \{V, E, \iota, \tau\}$.
G_L	Long range graph of a SW network.
G_S	Underlying graph of a SW network.
G_{SW}	A Small World network.
H_n	A Hilbert graph.
ι	Initial mapping function of E into V in a graph.
k	The degree of a node or a graph.
$l_{u,v}$	Length of the shortest path between nodes u and v.
L	Average length (characteristic length) of a graph.
L_R	Routing average length.

$\mathcal{L}(\mathcal{G}, \mathcal{P})$	A Linear Feedback Shift Register (LFSR) graph.
μ	Message generation probability.
N	Number of nodes in a graph.
ϕ	Rewiring probability.
$P_{S \rightarrow T}$	The path followed by packets injected at S and consumed at T.
$\mathcal{P}$	A polynomial.
Ψ	Generating function of a SW network.
$\mathcal{R}_{k,n}$	The k, n ring graph.
R_S	Routing graph for the source node S.
ρ	Density of shortcut ends in a graph ($\rho = 2\phi$).
τ	Terminal mapping function of E into V in a graph.
$\mathcal{T}_s$	Traffic in the subgraph or edge S.
T_d	Service time at dimension d.
T_{dec}	Decision time.
V	The set of vertices in a graph.

Glossary

CIU Communication Interface Unit.

CPLD Complex Programmable Logic Device. A programmable logic technology.

CS Circuit Switching. A switching method.

DCSH Distributed Crossbar Switch Hypermesh. A hypermesh implementation.

DEMUX Demultiplexer.

DMA Direct Memory Access. A transfer technique in which an I/O device writes data directly into the memory without the CPU involvement.

RDCSH Recursive Distributed Crossbar Switch Hypermesh

FPGA Field Programmable Gate Array. A programmable logic technology.

Hypermesh A topology used to interconnect computers.

IN Interconnection Network.

Latency The time elapsed between the instant a message is injected into a network, and the time it arrives at its destination.

LFSR Linear Feedback Shift Register

Multicomputer A parallel computer formed by multiple computers connected through a network.

Multihop An implementation technology in which information is transmitted using different media.

Multiprocessor A parallel computer formed by multiple microprocessors and a shared physical RAM.

Multitasking The ability an operating system has to perform different tasks at what appears to be the same time.

Multiuser The ability an operating system has to work simultaneously with a number of users.

MIN Multistage Interconnection Network.

MIMD Multiple Instruction streams Multiple Data streams.

MISD Multiple Instruction streams Single Data stream.

MPI Message Passing Interface.

MS Message Switching.

MUX Multiplexer.

NI Network Interface. The physical interface to the network.

NIC Network Interface Card. The physical network interface in the form of a plug-in card.

NUMA Non Uniform Memory Access. A loosely coupled multiprocessor, in which the access time to different memory regions is different.

OS Operating System. The set of programs and utilities that allows a computer to run, and which provides basic services.

OSI Open Systems Interconnection standard.

PCI Peripheral Component Interconnect bus.

PVM Parallel Virtual Machine. A computer library which is used to run programs in parallel.

PE Processing Element. The part of a node in a computer network which is responsible of computing.

RSW Random Small-World graph.

SE Switching Element. The part of a node in a computer network which is responsible of the communications.

SIMD Single Instruction stream Multiple Data streams.

SISD	Single Instruction stream Single Data stream.
SMP	Symmetric Multi Processor. A tightly coupled multiprocessor.
SW	Small-World
ULC	User Level Communication.
UMA	Uniform Memory Access.
UDMA	User Level Direct Memory Access.
WDMA	Wavelength Division Multiple Access. A photonic transmission technique in which different nodes can transmit simultaneously using different wavelengths.
WDM	Wavelength Division Multiplexing.
WMCH	Wavelength division Multiple access Channels Hypercube. A hypermesh implementation.
WR	Wormhole Routing. A switching method.
VCT	Virtual Cut-Through. A switching method.

1. Introduction

During their relative short life, general purpose computers have experienced speed-ups of several orders of magnitude. This has been allowed mainly by the creation of integrated circuits, which allow an ever increasing number of transistors to be laid down and packed in a single silicon chip. The industry has been driven by Moore's law, which as originally conceived states that the number of transistors integrated in a single chip would roughly double every year (the current estimate is closer to 18 months) [1].

We have become dependent in an ever increasing computing power to solve a large amount of every day problems. High performance computers are used for instance to predict the weather, real time signal processing (industrial control, radar, video processing), image processing (for instance in medical imaging), computer vision, ray tracing and in the encryption and decryption of data. They are also used in scientific computing, where the main application areas are in fluid dynamics, 2D and 3D Navier-Stokes equations, finite element methods, multigrid methods, lattice gas methods, climate modelling, Monte Carlo methods, n-body problem, molecular modelling, quantum mechanical methods and others.

Unfortunately it is doubtful that Moore's law can be maintained indefinitely. This is due to technological and physical constraints which will eventually limit the number of transistors in a single package [2]. A natural solution is to either incorporate several processors in a single *multiprocessor* computer, or to interconnect several computers to construct a more powerful *multicomputer*. Because of the difficulties in scaling multiprocessors even to moderate sizes, in this work we will only be interested in multicomputer systems.

The central problem in achieving faster communications in a multicomputer has proved to be the way in which the processors are connected together (their topologies). Vast amounts of research has been focused in providing interconnection networks that minimise their cost and complexity, and at the same time provide low latency and high throughput. Networks such as the k-ary n-cube, mesh, toroid, delta, butterfly, omega, hypermesh, DCSH, star, tree and ring have been proposed, analysed and used in different parallel systems. The choice of a particular intercon-

nection network depends on several factors, including application, technological and economical constrains. Furthermore this work is complicated by the fact that it is desirable that the communication structure of the problem to be solved is similar to that of the interconnection network in order that the communication costs are reduced, although this is not always possible, and some level of non-local communications must be supported.

The most popular networks used have been cubes, meshes, trees, tori and hypercubes. Low dimensional structures such as the cubes and tori suffer from a large diameter which increases contention and only communication patterns with similar topologies can be efficiently accommodated. A significant amount of research has focused in extending these low dimensional structures with long range connections in order to support more general communication patterns (see for instance [3–6]). Despite their shortcomings, these structures remain popular due to their simplicity, low cost and expandability. Furthermore in a large number of applications the communication pattern matches these low dimensional networks.

Recent developments in graph theory can be used to provide a new approach in the design of interconnection networks. These have evolved from research in the understanding of complex systems, where a large number of elements interact at different levels of locality. Strogatz and Watts proposed a model for understanding such systems based on the identification of two length scales in their interconnection graphs [7]. At a local level the elements are regular and they form strong local bonds (cliques) where local interactions occur. At a global scale the interconnections are not regular any more; in fact in Strogatz and Watts models global interconnects are random; however, and perhaps because of this reason, they facilitate interchanges between distant cliques very efficiently. The model they proposed is based on the addition of random edges to a locally well connected deterministic graph (but with poor global connectivity). In [8] Newman and Watts show that there exists a small crossover point (of the number of additional edges introduced) in which the average distance is abruptly reduced, while a given measure of locality shows that the graph has not been significantly perturbated in the local level.

This *small-world phenomenon* is of paramount importance for this research, as it suggests that it is possible to generate interconnection networks with a small average length and with a strong local connectivity if a small number of global edges is introduced. A large amount of research has been undertaken in understanding this phenomenon [9]. In [10] Kasturirangan proposed the Multiple Scale Hypothesis, which states that the reduction of average length in the Small-World model is not due to the addition of long range edges, but to the fact that the new edges are of a

multitude of length scales.

These two developments provide a solid starting point for the design of new interconnection networks. A good interconnection network should be able to efficiently exploit the locality present in the communication patterns of the algorithms expected to run in it, while at the same time it should be able to support non-local communications efficiently for cases where the communication pattern does not match the topology of the network (or for cases where there is a dynamically changing pattern). A small world network exhibits precisely these characteristics, as it defines a length scale where local communications can be exploited (in a clique), whereas the global scale is efficient at supporting any non-local communications present in the communication pattern of the algorithm, or which might arise artificially by a mismatch between the communications and topology of the network.

On the other hand the multiple scale hypothesis provides an explicit mechanism for the construction of the networks which does not necessarily involve the addition of randomness into the network, Furthermore it can be used to design networks where not only two levels of locality are defined (short and long range), but a diversity of different length scales can be introduced and exploited, confining the information in all of these levels.

It is important to note however, that due to technological constrains, the introduction of edges of different scales will necessarily introduce more complexity into the design of the system (this is particularly true, but not exclusive, of VLSI). However, according to the multiple scale hypothesis it is not only convenient, but necessary to introduce these edges in order to reduce the diameter and average length of the network. For this reasons the systems proposed in this work are more amenable to be implemented using a set of discrete nodes than in VLSI.

Small-world interconnection networks are introduced, and some basic properties are derived in the remainder of this work. In particular, analytical expressions for the diameter, degree, bisection width, congestion and a latency model are presented, as well as experimental results obtained with the use of a purpose built graph construction, analysis and simulation computer environment. A design methodology based on the direct addition of long range edges to an underlying graph is presented. It will be usually assumed that the underlying graph provides a strong local connectivity but exhibits a large diameter (such as a Cartesian product of an array). Such work is similar to the proposals of the inclusion of bypass or express channels in meshes and cubes, however it provides a more general and powerful method, and can be extended to any other graphs as desired.

Randomness plays an important role in the construction of the proposed intercon-

nection networks. However, as it is not an absolute requirement for the construction of small-world graphs, the construction of small-world deterministic graphs is analysed, and a family of new deterministic small-world interconnection networks called Linear Feedback Shift Register graphs (LFSR) is proposed and analysed. Determinism allows for additional properties, such as the construction of a simple routing algorithm and the guarantee of regularity in the graph which is a very desirable property.

Finally an example of a direct construction where no underlying graph is used is presented, as well as an extension into small-world hypergraphs which although presented towards the end of this work provided the inspiration and starting point of this research.

2. Background

2.1. Parallel Computing

In a parallel system, each processor executes a subset of the original task, and the result is a general speed-up of the problem. Naïvely, one might expect the speed-up to be directly proportional to the number of processors. However in reality this isn't the case. There are numerous reasons for this slow down. According to Minsky's conjecture as the system grows in size, performance becomes bounded by bottlenecks in the communication channels [11]. It is therefore necessary to design the communication network very carefully to ensure it can withstand the traffic presented to it without saturation. Although the physical construction of a parallel system is very important to ensure a speed increase, it is also important to note that fundamentally each task has an inherent serial component which cannot be parallelised. Amdahl's law provide a bound for the maximum speed-up of an algorithm composed of a serial (s) and a parallel (p) fraction as [12]:

$$S = \frac{s+p}{s+\frac{p}{N}},$$

where N is the number of processors.

In a *multiprocessor* several processors share a single memory and other resources. Access to the shared memory is carried out using a single shared bus, a crossbar switch or an interconnection network (usually a multistage interconnection network). Although these systems provide a conceptually simple system, with a single memory image which can be accessed in constant time (if no contention occurs) they can only be expanded to moderate sizes, since to maintain a good performance the system becomes too expensive to build [13–15].

A *multicomputer* on the other hand, is formed by interconnecting a large number of computers through a communication medium. Each computer is formed by a processing element, where information is processed, and a switching or communication element, where packets of data are sent or received from other computers.

Computers communicate only by the explicit exchange of information through the interconnection network [14,16–19]. On this work we are interested in the study and development of interconnection networks for multicomputers, although some of the results can be used to improve the performance of multiprocessor systems.

The central problem in achieving faster and cheaper communications in a large multicomputer has proven to be the way in which the processors are connected together (their *topologies*) [20, 21]. The most straightforward way of connecting them is to simply connect every node to every other node. Unfortunately this *total connection* has some serious disadvantages. For a system with N nodes, it requires that $N-1$ wires reach every node. Therefore the total number of wires is given as $N(N-1) = N^2 - N$. Even a moderate network will require far too many wires. These reasons make this scheme unpractical and very costly.

A large amount of research has been focused in providing alternatives, that while minimising the cost and complexity, still deliver performance levels comparable to this unpractical network. Networks such as the DCSH, k-ary n-cube, mesh, toroid, delta, butterfly, omega, hypermesh, star, tree and ring have been analysed and used in different parallel systems [15, 17, 22–25].

In a parallel computer the time used to solve a problem isn't reduced in proportion to the number of processors, as shown in figure 2.1. There are numerous reasons for the mismatch between the maximum speed-up and the real speed-up. These could be broadly classified into the following [26]:

- Algorithmic Overhead. This is due to limitations in the algorithm which prevent it to be parallelised efficiently. They can be divided into the following:
  - Serial Fraction. Some tasks in the algorithm have to be performed before others in order for the algorithm to work, and so these tasks cannot be parallelised.
  - Work-Imbalance. When partitioning the algorithm in different concurrent processes, different work loads may be assigned to every processor. This has the effect of wasting computing resources and thus slowing down the execution.
- Interaction Overhead: This is the overhead incurred while executing the algorithm in a real, *non-ideal* computer. One of the main reasons for this overhead is the mismatch between the location of the resources in the network (i.e. topology), and the requirements of the algorithm. This overhead can be subdivided into:

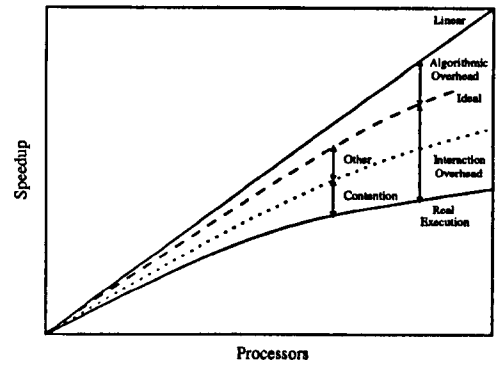


Figure 2.1.: Typical speed-up in a parallel system.

- Latency. The actual transmission time of the information from one processor to the other.
- Contention. The time spent while waiting for a resource to become free in the network.
- Others. Synchronisation, scheduling, cache effects, kernel, network interface, shared memory protocols, etc.

2.1.1. Types of Parallel Computers

Various classification schemes for parallel computers have been proposed . The most widely used is due to Flynn, who classified the systems on the basis of how many streams of instructions and data are present in the machine. His classification is as follows [27]:

- SISD (Single Instruction stream, Single Data stream). This is the conventional von Newman model.
- SIMD (Single Instruction stream, Multiple Data stream). The data is stored on a vector or a matrix and instructions are executed on the whole array at the same time.
- MISD (Multiple Instruction stream, Single Data stream). In these kind of machines, the data is shared between a number of processors, each executing a different instruction on it at the same time. This model of computation is not widely used, since many algorithms do not map naturally into it.

- MIMD (Multiple Instruction stream, Multiple Data stream). In this case each processor operates a different set of instructions into different data sets. These kind of parallel machines have been proven to be the most general and powerful [15].

However, this scheme of classification does not provide an adequate and exhaustive classification of all systems. Therefore other classification criteria has been proposed, based on:

- Synchronisation. Synchronous systems vs. asynchronous systems.

- Memory sharing. In a *multiprocessor* system all the processors physically share a main global memory space. In a *multicomputer* every processor has its own memory and cannot access the other processors memory. All information exchange between them is carried out by passing messages through the interconnection network.

- Coupling. This term applies only to multiprocessor systems. A tightly coupled system has the main memory located at a central location within the machine, so the access time for each processor is the same. This is also known as a UMA (Uniform Memory Access) or SMP (Symmetric Multi-Processor) machine. On the other hand a loosely coupled system places some memory partitions closer to some processors than others, and therefore their respective access times are different. These machines are also called NUMA (Non Uniform Memory Access).

- Granularity. This refers to the level of parallelism, or the *fineness* in which the data and the algorithm can be distributed among the processing units. A coarse-grained system is typically formed by a relatively small number of processing units; a fine-grained system is formed by a large number of processing units.

- ECS (Erlangen Classification Scheme) [28], in which an architecture is defined as a triplet of numbers representing the number of control units, the number of arithmetic units, and the word length respectively. Combinations of triplets with some basic operators are allowed, and therefore almost any architecture can be defined by some combination of triplets. In this scheme, a conventional von Neumann machine with a 32 bit word size is defined by the triplet $t_{conventional} = (1, 1, 32)$.

2.1.2. Multicomputers and Interconnection Networks

In this work we are mainly interested in the development and study of interconnection networks for multicomputer systems. Note however that this subject is of relevance not only for multicomputer systems, but also for a larger class of systems and their modelling. In fact a system is nothing more than a collection of elements or nodes, connected together; and its operation is completely determined by the operation of the nodes and the topology of the interconnections. For this reason the study of interconnection networks is a fundamental problem in the development, understanding and modelling of systems such as power distribution, social networks, disease modelling, communication networks, neural networks, protein reaction networks and others.

A multicomputer system is formed by nodes, which provide computing power, and by an interconnection network (which will also be referred to as IN for shortness), which links the nodes together.[1] The selection and design of the interconnection network is one of the most critical issues in constructing a parallel system, since it will have a direct impact on its performance [4, 29]. An ideal network should have the following characteristics:

- Low diameter: The diameter of a network determines the maximum number of hops a packet needs to make to reach its destination. Although wormhole and VCT routing techniques have made the fly time of a packet somewhat independent of the number of intermediate nodes, networks with small diameters tend to have a smaller latency since packets are less likely to block each other.

- Small, fixed degree: The degree of a network corresponds to the number of input-output lines at each node. A small degree leads to reduced costs since less wires and I/O controller components are required. A small degree is also desirable for VLSI implementation. It is also advantageous to provide an interconnection network that can be incrementally extended without increasing the degree of individual nodes. This is of special importance for networks formed by a collection of discrete components, where each component has a limited amount of I/O capacity.

- Simple Layout: The network should be easy to map into 2 or 3 dimensional space, where the computer will be constructed.

[1] For a good review of such systems see [18].

- Fault tolerant: As in a large multicomputer components are likely to fail with a given regularity, it is important the the interconnection network can support as many faults as possible, without becoming disconnected.
- Universality: It is desirable that the interconnection network can efficiently embed other common topologies, in order to successfully execute algorithms designed for these other networks. The topology should easily map the expected communication pattern of the applications, while at the same time, should be flexible enough to accommodate for other patterns.
- Simplicity and Symmetry: It is desirable that the interconnection network is conceptually simple and symmetric, in order to allow for the easy exploitation of available architectural and algorithmic properties.
- Low cost.

However, in reality it is not possible to provide all of the above at the same time, and a compromise must be made. To try to achieve these goals, a great number of topologies have been proposed [3–6,17,19–25,27,29–63] and used in parallel computers. The selection of an interconnection network is perhaps the most critical decision in the design process for a parallel system; not only because it is the most determinant factor in the cost, performance and power of the system, but also because the comparison of different interconnection networks is an inherently difficult problem, since no universally accepted metrics exists. Networks have to be compared in a per-application basis, taking into account such diverse issues as previously outlined. Obviously this comparison is neither straightforward nor simple [64].

Interconnection networks can be modelled as graphs (or hypergraphs),[2] in which a processor is a vertex and the communication channel is an edge [65]. For this reason graph theory is an essential tool for the design and analysis of interconnection networks. Furthermore any graph can be thought of as the topological structure of an interconnection network. Mathematically there is no difference between the two terms, and they can be used interchangeably.

Interconnection networks can be roughly classified into the following groups [11, 14,15,17,33]:[3]

- Multistage Interconnection Networks (MINs). In these kind of networks the distance between any pair of nodes is constant, since each message has to

[2] In a graph each edge connects only two vertices; while in a hypergraph a hyperedge can link more than two vertices.

[3] It is also possible to have combinations of these classes in a single network.

travel a fixed number of stages before arriving at its destination. Examples of these networks are the SW-Banyan and relatives (omega, delta, butterfly, inverse omega, etc.). These networks cannot be completely modelled with a regular graph or a hypergraph, since intermediate elements are not computing nodes, but switches [15,17].

- Direct, Point to point networks. These networks are modelled as graphs, where vertices represent nodes which are linked by edges (wires). Examples include the Mesh, toroid, hypercube, star, tree, de Bruijn, ring, k-ary n-cube, and others. [15,17–19]

- Bus networks. In these topologies nodes are linked by one or several buses. These networks can be modelled as hypergraphs, where a hyperedge represents a bus, and can join several nodes together. Examples of these networks include the hypermesh, DCSH [51], WMCH [36,37], and De Bruijn and Kautz bus networks [66].

In this work we are mainly interested in direct interconnection networks, and up to some extent in bus based networks.

2.2. Graphs

As previously noted interconnection networks can be modelled as graphs and hypergraphs,[4] in which the vertices or nodes are the processing elements, and the edges are the channels or connections between the processors.

A direct interconnection network can be modelled as a graph $\mathcal{G} = \{V, E, \iota, \tau\}$, where V is the set of nodes (a node is the combination of a switching element and a processing element), E is the set of edges (i.e. connections between the processors), and ι, τ are mappings (called *initial* and *terminal* respectively) such that $\iota(E)$, $\tau(E) \in V$ and $E \rightarrow V \times V$. Sometimes the mappings are not specified and the graph is given simply as $\mathcal{G} = \{V, E\}$. A vertex connected to an edges is said to be incident to that edge and vice-versa. Edges with common incident nodes are called *parallel*, and an edge with the same initial and terminal node is called a *loop*. Graphs with no parallel edges and no loops are called *simple*.

The number of vertices N is simply the cardinality of the vertex set $|V_\mathcal{G}|$. In this work we are mainly interested in undirected graphs; if an edge exists between u, v there is also an edge between v, u (there is no directionality associated with ι, τ).

[4] In a graph an edge links only two nodes, whereas in a hypergraph no such restriction exists.

These graphs have the advantage of being able to exploit some of the locality of information between neighbours; if node u sends data to node v, there is a high probability of node v sending information back to node u.

A *path* of length n is a sequence of edges P_n such that $\zeta(e_i) = \zeta(e_{i+1})$, $\zeta(e_1) = u$, $\zeta(e_n) = v$ for $1 \leq i < n$, where $\zeta \in \{\iota, \tau\}$. The notation $l_{u,v}$ will be used to represent the length of the shortest path between nodes u and v. We will call this the *distance* between u and v. Two nodes u, v are said to be *neighbours* if their distance is one, i.e. if $l_{u,v} = 1$.

The diameter of the graph is the maximum distance in the graph; that is $D = \max\{l_{u,v}\}$, $u, v \in V$. This is an important measures since it determines the maximum latency under no load in the network. The degree of a node is the number of neighbours it has. The degree of a graph Δ is the highest degree of any of the nodes in the graph, and a graph is said to be regular if all of its nodes have the same degree. The degree is an important measure as it determines the cost of an individual node. Furthermore if the graph is regular similar nodes can be used throughout the network, which helps to reduce the cost of the system.

The bisection width is the minimum number of wires cut when the network is divided into two equal halves. This measure is of particular importance when the network has to be constructed in VLSI, since in such systems the wire density is a critical factor [20].

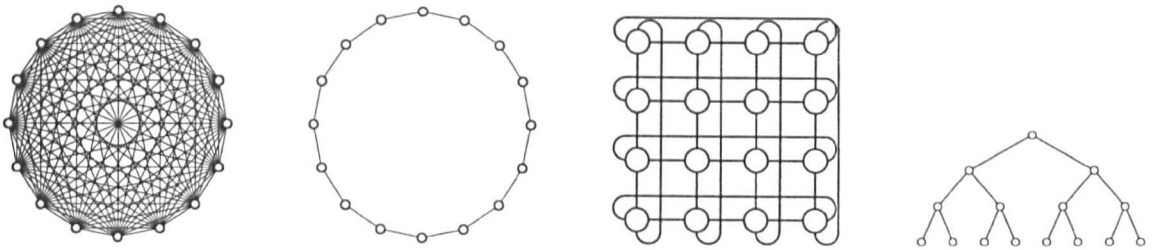
A graph can be drawn by representing its nodes with small dots or circles, and its edges by arrows or lines (where an arrow is used for directed graphs only). Sometimes it is desirable to label nodes or edges in order to reference them directly. In general, there is a large number of different drawings and labellings possible for a given graph. Any two representations are said to be *isomorphic* to each other, and the set of all possible representations is called an *isomorphism class.*

2.3. Direct Networks

2.3.1. Meshes, tori and others

Naively one might consider constructing an IN using a complete graph K_N, such as the one shown in figure 2.2(a). Although such network provides excellent communication facilities, its cost is too high, even for moderate sizes, as the degree is $O(N)$ and the number of edges is $O(N^2)$.

The binary tree depicted in figure 2.2(d) provides an IN with a small diameter, simple routing, and a fixed small degree, all of which are ideal qualities for an



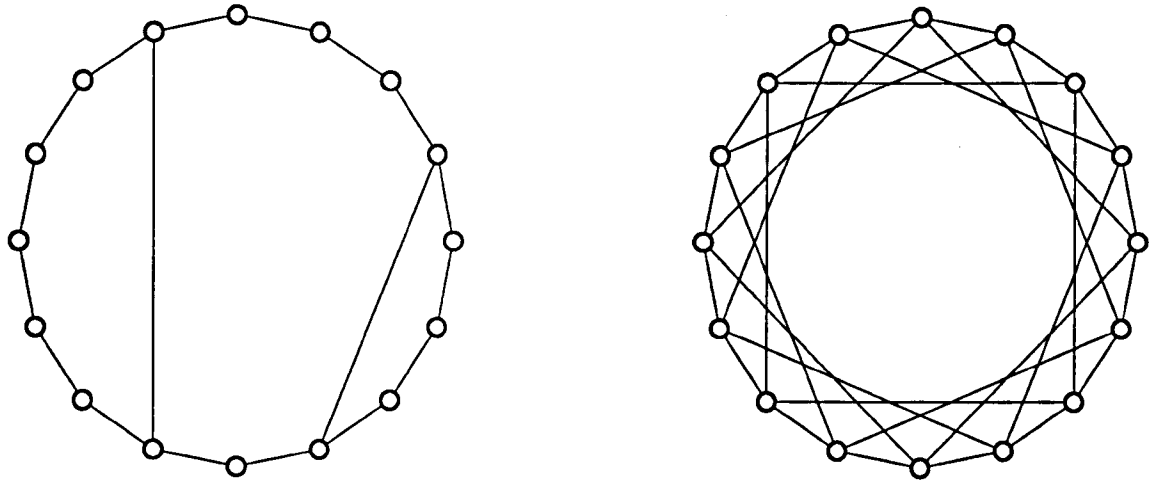
(a) Complete (b) Ring (c) Torus (d) Binary Tree

Figure 2.2.: Common direct interconnection networks. Figure (a) shows the complete graph K_{16}. As expected the degree and number of edges is very large. The ring $\mathcal{R}_{1,16}$ shown in figure (b) is a one dimensional linear array with wrap around connections. The two dimensional Cartesian product of a ring forms a toroid shown in (c), while in (d) a binary tree is shown.

interconnection network. Unfortunately for generalised communication patterns it suffers from congestion close to the root nodes, as a large amount of traffic needs to be routed through the root edges. Furthermore it exhibits the minimum fault tolerance possible, as it is enough to remove one edge to disconnect the network. Variations and augmentations have been proposed to counter this effects, such as the fat tree [60], the k-ary N-tree [62], the recursive fat tree [63] and the generalised fat tree [57].

The linear array, and its Cartesian products are one of the most natural graphs to consider for an interconnection network, since they map naturally into space. The two and three dimensional products of the linear array are the mesh and cube respectively. A linear array with wrap around connections is called a ring or a cycle, and the two dimensional product of a ring is a torus. The m-ary d-cube is a d dimensional product of an m ring, and is described in the next section.

The ring makes a poor interconnection network, due to its large diameter and poor fault tolerance (two edge or node failures are enough to disconnect the network). Chung and others have proposed augmenting the basic ring with one or several non-crossing arcs (chords) such as the one shown in figure 2.3 [6, 67]. The *chordal rings* constructed in such a way preserve most of the desirable properties of the ring, such as simplicity, small degree and planarity, while significantly reducing the diameter and increasing the fault tolerance of the network. A related network is the *circulant* graph where each node in the ring is extended with shortcuts to nodes with a given relative distance, as shown in figure 2.3. In this way the graphs constructed are



(a) (b)

Figure 2.3.: Extended ring graphs. Figure (a) shows a chordal ring graph, while (b) shows the circulant graph $Ci_{16}(1,4)$.

regular and symmetric, with a smaller diameter and much improved fault tolerance (although they are not planar anymore) [68]. Both of these networks are of particular interest to this research as they are predecessors of the small-world interconnections networks proposed herein; in particular a chordal graph can be thought of as a random small-world interconnection network where the underlying graph is a ring (although the small-world graph is not necessarily planar).

2.3.2. Hypercube and m-ary d-cube

The hypercube is one of the most popular interconnection networks. It has many desirable properties, and a large number of parallel machines have used hypercubes as their topology, such as the Cosmic Cube [34], the Connection Machine [29, 61] and the iPSC/2 [22]. The n dimensional hypercube Q_n can be defined as the n-th Cartesian product of the complete graph K_2. Alternatively it can be defined using binary sequences, where two edges are joined iff their binary sequences differ in only one location. The hypercube Q_n has $N = 2^n$ nodes, $E = n\,2^{n-1}$ edges and diameter n. Furthermore it has simple routing properties, can be recursively divided into smaller hypercubes, and can embed other interconnection networks, such as rings, meshes, trees and the complete graph K_{n+1} (with dilation 2). For a review of these and other properties of the hypercube see [17, 33, 45]. Despite its desirable properties, hypercubes exhibit some drawbacks, such as a large, non-

fixed degree and a somewhat large diameter. In an effort to reduce the diameter several variations have been proposed, such as the twisted cube [30] and the Möbius cube [41]. Other variations have been constructed in an effort to provide a bounded and small degree, such as the cube connected cycles (CCC) [69], and the Hierarchical Hypercube [50].

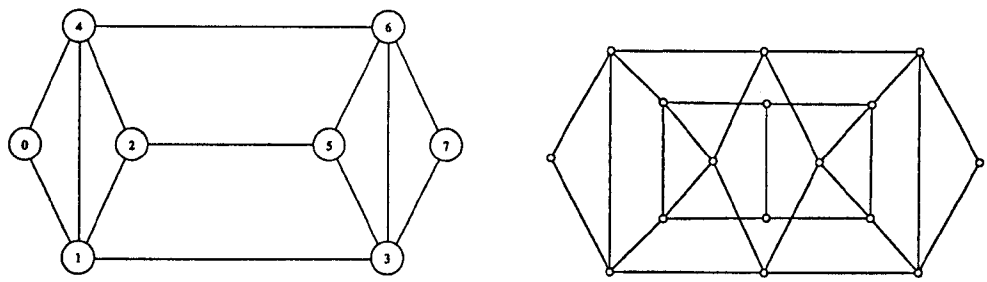
The m-ary d-cube $Q(m,d)$ is a generalisation of the hypercube, where the vertex set is given as $V = \{x_1x_2\cdots x_d,\ 0 \leq x_i < m\}$, and two nodes are joined if and only if their labels differ in one dimension by one unit; i.e. node $x_1x_2\cdots x_d$ is connected to the $2d$ nodes $x_1\cdots x_i \pm 1 \,(\mathrm{mod}\, m)\cdots x_d$. There are $N = m^d$ nodes in the network, with degree $2d$ and diameter $md/2$. The m-ary d-cube is isomorphic to rings, meshes, binary cubes, tori and Omega networks [20], and allows for a tradeoff between network degree, diameter, network size and dimensionality. Simple ordered dimensional routing can be used (e-cube routing); however, and particularly for low dimensional structures, the diameter obtained is large. In an effort to reduce the diameter several bypass strategies have been proposed (see for example [5]). Some researchers have advocated for a complete bypass using a bus based network, the hypermesh [70]; which drastically reduces the diameter at the expense of using buses in the network.

2.3.3. De Bruijn and Kautz

The De Bruijn network $B(d,n)$ consists of d^n labelled nodes $V = \{x_1x_2\cdots x_n : x_i \in \{0,1,\ldots,d-1\}, \forall 1 \leq i \leq d\}$, where for each vertex $x_1x_2\cdots x_n$ there are d incident edges to all vertices $x_2x_3\cdots x_n\alpha$, where $\alpha \in \{0,1,\ldots,d-1\}$. A De Bruijn graph can be thought of as the graph formed by all possible linear feedback shift register states of length n and base d. In this work we are mainly interested in the undirected de Bruijn graph, which is constructed as before, without assigning any directionality to the edges, and by removing loops and parallel edges. Although the directed de Bruijn graph is regular, with degree $2d$, the undirected version is not regular and has minimum degree $2d-2$, maximum degree $2d$ and diameter n.

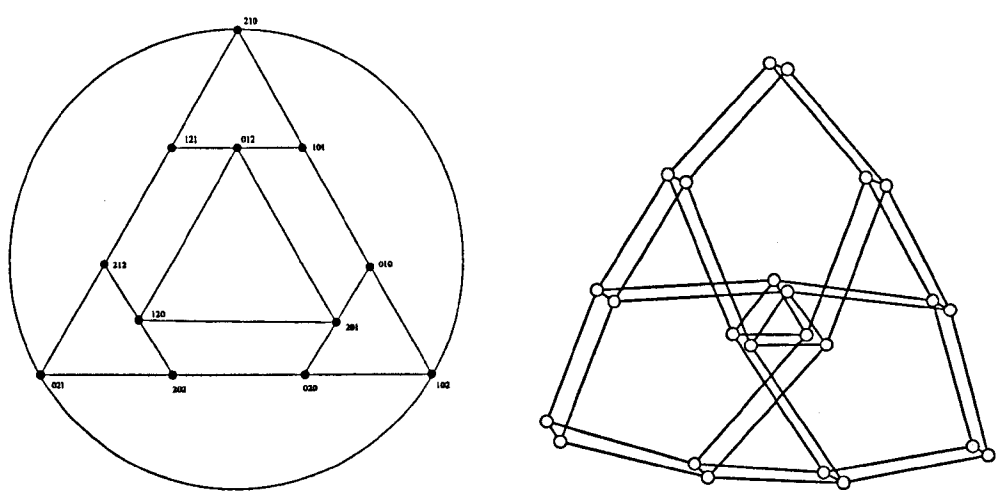
The Kautz network $K(d,n)$ has the vertex set $V = \{x_1x_2\cdots x_n : x_i \in \{0,1,\ldots,d\}\}$ with , $x_i \neq x_{i+1}$, $\forall 1 \leq i \leq d$, and for each vertex $x_1x_2\cdots x_n$ there are d incident edges to all vertices $x_2x_3\cdots x_n\alpha$, where $\alpha \in \{0,1,\ldots,d\}$ and $\alpha \neq x_n$. As it is apparent from the definition, Kautz graphs are very similar to de Bruijn networks, and retain most of its characteristics, while providing a better connectivity [16].

De Bruijn and Kautz networks retain most of the useful properties of the hypercube, such as a simple recursive structure, simple routing algorithm and contain



(a) (b)

Figure 2.4.: De Bruijn Networks. $B(2,3)$ is shown in (a), and $B(2,4)$ is shown in (b).



(a) (b)

Figure 2.5.: Kautz Networks. In (a) $K(2,3)$ is shown, and in (b) $K(2,4)$ is shown.

other topologies as subgraphs. However both of these networks posses some additional desirable features. The degree and diameter are independent, allowing for large networks with a small degree, and for a tradeoff between diameter and degree. Furthermore, the de Bruijn and Kautz networks are more efficient expanders than the hypercube, so larger networks with the same degree and diameter can be constructed using these networks. Both of these graphs exhibit a unique shortest path between pairs of nodes, which provides for simple routing algorithms. However the existence of multiple shortest paths in the hypercube can be used as an advantage to increase the bandwidth and to support a larger amount of traffic before the network saturates by using adaptive routing.

Due to its desirable characteristics, these networks have been widely studied (see [71] for a review) and their hypergraph counterparts have been proposed in [66].

2.3.4. Cayley graph

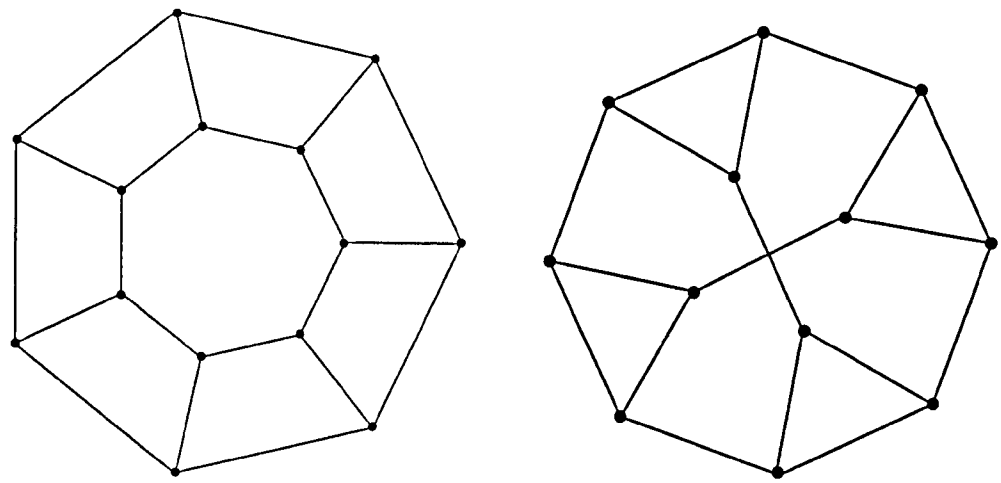
Cayley graphs, proposed by A. Cayley [72], are a class of graphs defined using group theory. It has been shown that large classes of graphs are Cayley graphs, including the de Bruijn and Kautz networks, and the complete and circulant graphs [14]. The Cayley graph $C_\Gamma(S)$ is constructed using the finite group Γ, with the aid of the subset $S \subset \Gamma$, $S \neq \{\}$, which does not posses the identity element I of Γ. The vertex set of $C_\Gamma(S)$ is $V = \Gamma$ and the edge set is:

$$E = \{(x, y) : x^{-1}y \in S \forall x, y \in \Gamma\}.$$

The identity element is not allowed in S to avoid loops, as $x^{-1}x = I \notin S$. The Cayley graph is regular, and the determination of the diameter is simple, as it is only required to calculate the distance between the identity element and the rest of the elements in the group. $C_\Gamma(S)$ is connected iff S is a group generator of Γ. Cayley graphs are general enough to be regarded as a design method for interconnection networks [14]; and some of the (d, k) graphs shown in table 2.1 have been designed using Cayley graphs.

2.4. Bus Networks

Several hypergraph based architectures have been proposed, despite the fact that current technology seems to favour point to point high speed serial links, such as Infiniband and RapidIO (see for instance [75–77]). The use of buses as communica-



(a) (b)

Figure 2.6.: Cayley graphs. Figure (a) shows the Cayley graph with generator $C_{D_7}(7,1,2,3,4,5,6)$ where D_7 is the Dihedral group with 14 elements, and (b) is $C_{A_4}(2,1,4,3)$ where A_4 is the alternating group. See [73,74] for details.

tion channels in interconnection networks provides a way of reducing the number of edges in the network and the node degree, while providing a small diameter, which makes such constructions very attractive. The interest of hypergraphs in this research is limited to the hypermesh, and in particular the DCSH, which provides a very powerful communication structure.

2.4.1. Hypermeshes

The hypermesh is a symmetric hypergraph network. Running along each dimension there are a number of *clusters* or *hyperedges*, which contain k nodes connected in a linear array, and routing capabilities amongst those nodes. The Cartesian product of n clusters forms an n-dimensional hypermesh. Messages can change dimension at any node, and therefore can reach any node in n steps. (See fig. 6.1). Hypermeshes posses some very desirable characteristics. They have a low diameter, high bandwidth, support for efficient broadcast operations, and since they can embed meshes, binary trees and hypercubes, applications that map naturally into these topologies will also do so into the hypermesh [38], [51].

The hypermesh has been studied by different researchers, who have given it different names and have proposed different implementations. The use of shared buses, fully connected clusters and crossbar switches have been proposed (see fig. 2.8).

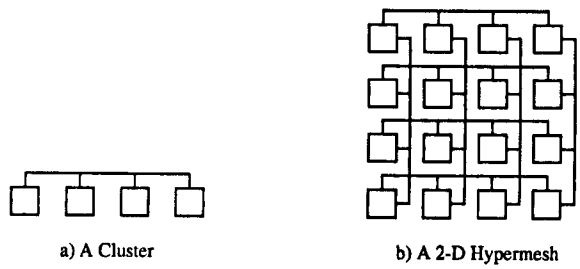


Figure 2.7.: Construction of a hypermesh. The 2-D hypermesh (b) is a 2-D Cartesian product of a cluster (a).

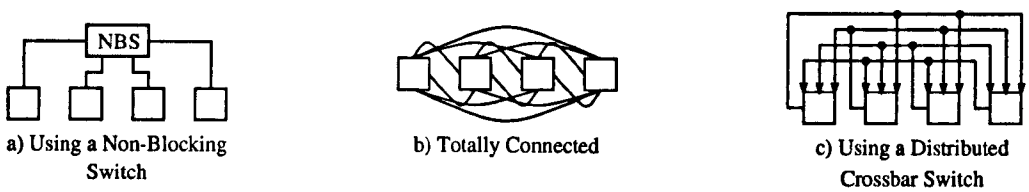


Figure 2.8.: Three different implementations of a cluster. (a) Using a centralised switch within the cluster. (b) Using complete connections. (c) The result of distributing a crossbar switch among the nodes.

However, these implementations suffer from bandwidth and cost limitations as the system is expanded [78].

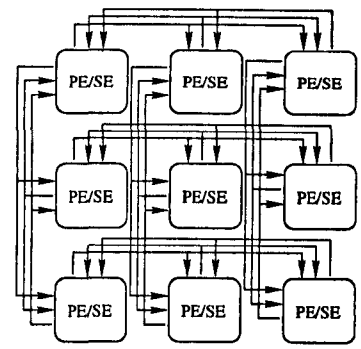
In [3] Dowd *et al* propose a basic hypermesh architecture called Spanning Multi-access Hypercube. Szymanski proposes an optical implementation using distributed optical crossbar switches and other techniques [46]. However the required technology for a full scale system doesn't seem to be available yet.

In [51] Ould-Khaoua proposes an implementation which doesn't suffer from these problems, based in distributing the crossbar switch among the nodes using a layered construction. He called this architecture DCSH (Distributed Crossbar Switch Hypermesh), and it is depicted in figs. 2.8-(c).

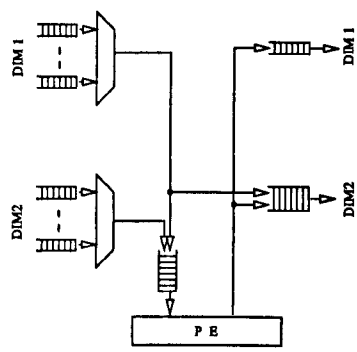
2.4.1.1. Distributed Crossbar Switch Hypermesh

The DCSH architecture has been introduced by Mackenzie, Khaoua and Sutherland in [31]. And subsequently expanded by Khaoua in [51]. In the DCSH a cluster is formed by using a distributed crossbar switch, as shown in figs. 2.8(c) and D.1(a). The proposed SE structure is shown in fig.D.2 (b) for VCT or Packet switching.

Contention is minimise in a DCSH because the buses are driven by only one node in the system. Initial work for this research focused on an implementation of the DCSH and in the I/O mechanisms associated with the construction of the



(a) A 3x3 hypermesh.



(b) SE design using VCT.

Figure 2.9.: A Hypermesh and its proposed implementation.

SE interface. Although the DCSH is a very powerful structure it suffers from a relatively large node degree, and from the required bus structures. In an effort to reduce the node degree of a DCSH the Hamming hypermesh is proposed in Section 6.2. Further work carried out in this direction is described in appendix C.

2.4.1.2. Optical Hypermeshes

Several methods of constructing a hypermesh using optical technology have been proposed. Optical connections offer various advantages over electrical ones, such as low power, electromagnetic noise immunity, large fanout and very high bandwidth [36].

In [37] Dowd proposes an optical hypermesh, the WMCH (Wavelength division Multiple access Channels Hypercube), and proposes different approaches to implement the clusters (or as he refers to them, the multiple access channels). The use of optical bidirectional buses, dual buses, folded buses and star-coupling are proposed.

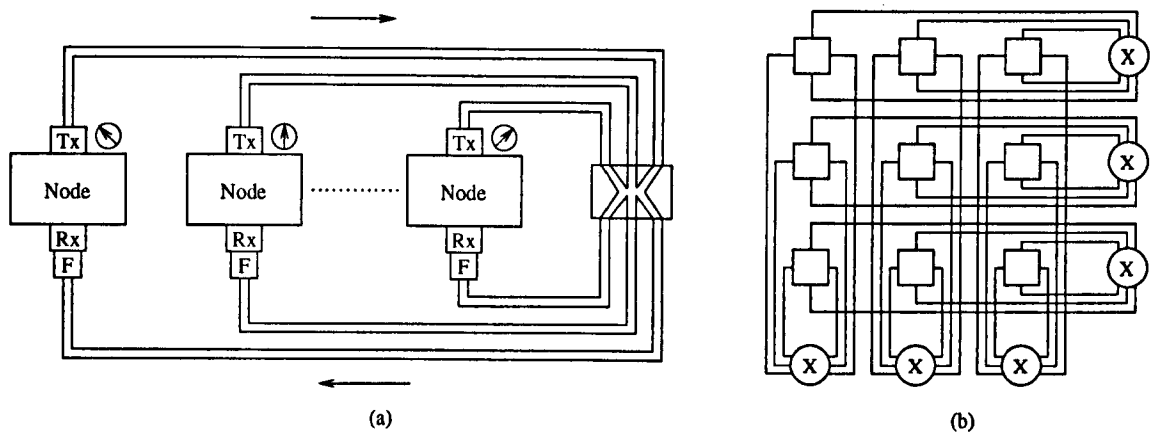


Figure 2.10.: Optical Hypermeshes; (a) WMCH cluster and (b) the WMCH architecture.

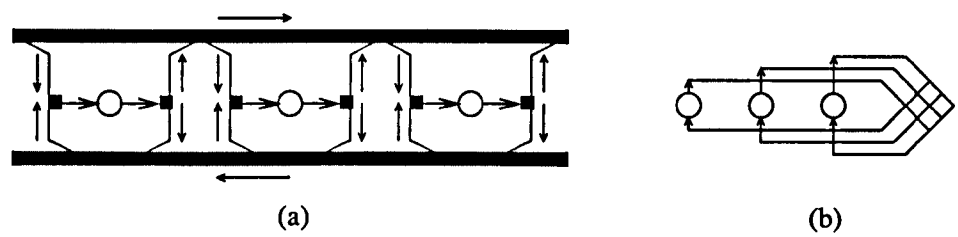


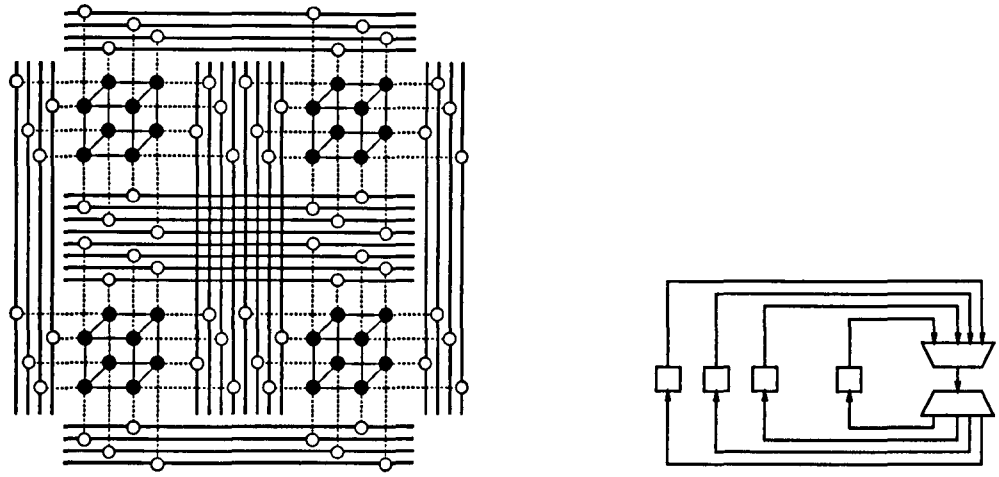
Figure 2.11.: A distributed optical switch hypermesh. (a) Proposed implementation. (b) Logical diagram.

One of the designs uses a star coupler and tunable transmitter at the output of the nodes with a fixed frequency filter at the input (see fig. 2.10), joined by a star coupler.

The system is not purely optical, since a packet must undergo optical-electrical and electrical-optical conversions (the system is said to be *multihop*) to achieve inter-dimensional routing. In order to improve performance and avoid the hopping, in [36] Dowd et al propose a different architecture, called FHA and SWHA. It is of interest since it resembles a 1-D RDCSH (which is proposed by the author of this work in appendix E.4), although the RDCSH has been proposed to solve different issues.

Szymanski proposes an optical hypermesh network, and proposes different implementations for the *hyperedges* (clusters), such as the use of centralised crossbar switches, star coupler using WDM (Wavelength Division Multiplexing) and distributed optical switching (see fig. 2.11) [46].

In [4] Louri et al propose a highly scalable network, the SMHL (Spanning Multichannel Linked Hypercube), which is formed as the hybrid product of a binary



(a) (b)

Figure 2.12.: An SMHL network (a). Thick continue lines represent the hypermesh, while slim lines represent the hypercube. Dashed lines link nodes which are physically the same. (b) Cluster implementation using WDMA and an optical MUX-DEMUX.

hypercube and a hypermesh. An optical implementation using MUX-DEMUX pairs and WDMA techniques is proposed by Dowd [37]. The use of the MUX-DEMUX simplifies the design of each node since the input filter is no longer required. The multiplexer is a k to 1 star coupler, while the demultiplexer is a diffraction grating which separates all the frequencies into their respective channels. The network is shown in fig. 2.12.

2.4.1.3. Other Hypermeshes

Other hypermesh implementations have been proposed and built. In [24], [79] the construction and testing of a parallel system named CP-PACS based on a 3D Hypermesh is described. The cluster design is based on crossbar switches. Other very similar parallel computers have been constructed, the SR2201 [25] and the PRODIGY [43]; all based in a 3D crossbar hypermesh.

2.5. The (D,k) Graph Problem

A fundamental problem in graph theory is the well known (D, k) problem, which emerges naturally in the design and optimisation of interconnection networks [80,81]. The problem is to find the largest graph (graph with the largest order) given the degree k and diameter D. A related problem is to find a graph that will interconnect

N processors with a degree k and the minimum possible diameter. The problem emerges naturally in the design of interconnection networks, because the number of connections that can be attached to any component is limited, and it is important to design networks where packets do not need to traverse a large amount of intermediate components before reaching their destination. This problem has received a large amount of attention in the scientific community, and some graph families with large number of processors for a given diameter and degree have been proposed [80,82–85], such as Cayley, De Bruijn and Kautz graphs. These networks are interesting candidates for use as interconnection networks. The De Bruijn graph has some additional properties (such as the ability to embed rings, binary trees and shuffle exchange networks) [71]. The Kautz network has a very similar structure to the De Bruijn graph, retaining the advantages of the De Bruijn network, with some improvements [16].

It is easy to show that no (D,k) graph can have a larger order than a Moore graph, which is a perfectly expanding graph. Since in a Moore graph all edges are spreading into new territory, the number of nodes in a Moore graph is simply:

$$\begin{aligned} N_{Moore} &= k + k(k-1) + k^2(k-1) + \ldots + k(k-1)^{D-1} \\ &= \frac{k(k-1)^D - 2}{k-2}. \end{aligned} \tag{2.1}$$

Equation 2.1 is called the Moore limit, or Moore bound [86,87]. It is known that Moore graphs do not exist except for $D = 2, 3, 7$ and possibly 57 [88,89]. The Moore bound has been improved in [90], where it is shown that for $k \geq 3$, $N \leq N_{Moore} - 2$. Better bounds have been obtained for specific cases using explicit constructions, and up to January 2004, the best such constructions are shown in table 2.1.

2.6. Routing

In a multicomputer information is exchanged between the nodes by interchanging messages through the interconnection network. The nodes send packets of information (which contain routing information and data), that travel through the network and arrive at the destination. At each of the nodes they visit, the network has to make a *routing* decision in order to place each packet closer to its destination.[5] If

[5]This is not always the case. For instance for some deadlock avoidance algorithms and some adaptive algorithms it is possible for a message to temporarily travel further away from its destination; as long as there are other mechanisms in place that warranty that the package will

d\k	2	3	4	5	6	7	8	9	10
3	10	20	38	70	132	190	330	570	950
4	15	41	96	364	740	1,155	3,080	7,550	17,604
5	24	72	210	558	2,766	5,500	16,956	53,020	164,700
6	32	110	380	1,395	7,908	19,279	74,800	294,679	1,211,971
7	50	148	672	2,756	11,220	52,404	233,664	1,085,580	5,311,566
8	57	253	1,081	5,050	39,672	129,473	713,539	4,039,649	13,964,808
9	74	585	1,536	7,884	75,828	270,048	1,485,466	8,911,766	25,006,478
10	91	650	2,211	12,788	134,690	561,949	4,019,489	13,964,808	52,029,411
11	98	715	3,200	18,632	156,864	970,410	5,211,606	48,626,760	179,755,200
12	133	780	4,680	29,435	359,646	1,900,319	10,007,820	97,386,380	466,338,600
13	162	845	6,560	39,402	531,440	2,901,294	15,733,122	145,880,280	762,616,400
14	183	912	8,200	56,325	816,186	6,200,460	34,839,506	194,639,900	1,865,452,680
15	186	1,215	11,712	73,984	1,417,248	7,100,796	45,000,618	282,740,976	3,630,989,376
16	198	1,600	14,640	132,496	1,771,560	14,882,658	86,882,544	585,652,704	7,394,669,856

Table 2.1.: Largest (k,d) graphs known. Taken from http://www-mat.upc.es/grup_de_grafs/grafs/taula_delta_d.html.

all processors inject their packets concurrently into an empty network, it is said that the injection model is *static*; otherwise it is *dynamic*. Static message injection is normally found in SIMD machines, where permutation routing is required. In permutation routing the goal is to route N packets in parallel (one per processor) to different processors as given by a permutation, where only one packet can traverse a link per unit time. If the permutation is known in advance, the routing can be optimised at compile time, and is referred to as *off-line* routing. Otherwise the routing is done *on-line* at execution time. In this work we will focus on on-line dynamic routing only, as this is a good model for large multicomputers with random communication patterns.

Routing problems are usually categorised as [92,93]:

1. Permutation routing: transfer of messages from a set of nodes, into a permutation of the same (or another) set.

2. Broadcast: transfer of a message from one node to all the nodes.

3. Reduction: recollection of information from all nodes into a single node where

eventually reach its destination. If these mechanisms are not in place livelock can occur. This is a condition where the message never reaches its destination (or takes an arbitrarily long time to do so), and should be avoided in the design of the routing algorithm [91].

the data is reduced with the aid of a combination function (maximum, minimum, etc.). This is the dual problem of broadcast.

4. Gather: assembly of a local vector from data distributed in a group of nodes.

5. Scatter: distribution of a local vector into a group of nodes (dual problem of gather).

6. Multinode Broadcast/Reduction: simultaneous broadcast/reduction of all nodes. In multinode reduction the result of the reduction is distributed to all processors (it is equivalent to a reduction followed by a broadcast operation).

7. Multinode Gather/Scatter: simultaneous gather/scatter from all nodes.

8. Total Exchange (Gossipping): all nodes send different packets to all other nodes.

Several techniques based on spanning trees can be used to implement efficient algorithms for Broadcast, Multinode Broadcast and Total Exchange (and their duals); see [92,94] and references therein. Permutation routing is a very common routing problem which arises naturally in a large number of applications, such as multigrid methods, matrix problems, finite elements and others [93]. On this work we will mainly focus on permutation routing, since this is the most general routing problem. Although this work will not do so, it is important to analyse the other routing problems in the interconnection networks discussed throughout the remainder of this work. In particular broadcast and gossipping occur frequently in common applications, and efficient implementations can be constructed by taking the architecture of the network into account (see for example [51,60]).

In any network, the routing algorithm and hardware have to avoid deadlock (which is caused by a cyclic dependency of requested and used resources which prevents messages to continue their travel), livelock (which is the case where a message never arrives at its destination and continues to travel the network indefinitely) and starvation (when a node is prevented from injecting messages forever) [95].

Routing algorithms can be classified into the following classes [91–93,95]:

1. Oblivious (static) Routing. The route of the message is determined solely by the source and destination addresses of a packet, without any knowledge on the conditions of the network.

a) Restricted routing. In restricted routing there exists only one path between a pair of nodes A,B through where information originating in A and destined to B can flow. If the routing algorithm brings the packet closer to its destination at each hop, then it is also *minimal.*

b) Random routing. In this kind of routing there are various paths through where information originating in A can travel to its destiny B. However, the path selection for a particular message is random, or at least independent on the network conditions.

2. Adaptive Routing. Here the route assigned to each packet is influenced by the conditions of the network.

a) Minimal Routing. The message is always moved closer to its destination.

b) Predictive Routing. The message is allowed to travel further away from its destination (the message can be *miss-routed*).

Oblivious routing strategies are particularly susceptible to faults in the network, as messages can only take a predetermined path in their travel. Although it seems obvious that oblivious routing is more susceptible to traffic hot-spots, it has been argued that adaptive routing techniques are prone to cause the hot-spots in the first place [91,93].

Randomness plays an important role in the design of efficient routing algorithms, since usually the worst case performance of deterministic communication algorithms is much worse than the average. In such cases randomisation of the routing problem can increase the performance, since the randomised algorithm has a much improved worst case performance. This can be done by either introducing randomness into the routing algorithm, or directly into the topology of the interconnection network. For this reason randomised networks such as the ones proposed in this work can prove advantageous.

2.6.1. Switching

Messages are transmitted through the network by "jumping" between switches. Each message is formed by a number of *phits*, which is the smallest unit of information that can travel the network. The switching method dictates the way phits are transfered through the network, and dictates the way flow control is performed. Common switching techniques are:

1. Circuit Switching (CS) is a method in which a channel between the source and the destiny is created when required, and only when the channel is ready the transmission starts. No information travels the network before the channel is ready. The channel is not liberated until the last phit arrives at its destination.

2. Message Switching (MS) is a method in which when a packet is received it is immediately buffered. Only when the whole packet is buffered at the intermediate switch, it is forwarded to the next switch, which will in turn buffer the packet again. This method is also known as "store-and-forward" [15], [96].

3. Virtual Cut-Through (VCT) is similar to MS, however when a switch starts receiving a packet, it can begin to forward it to the next switch if it is free. If not, the message continues to fill the buffer, but as soon as the next switch becomes available, transmission will begin. If the resources continue to be occupied, the whole message is buffered (and at the same time the previous link is removed, therefore freeing network resources) [97].

4. Wormhole Routing (WR) is similar to VCT. However, the packet is divided into a number of *flits*, which consist in a small number of phits of data in which flow control operates. If the next switch is busy, the message retains the switches already gained which can cause blocking [96]. The flit size is very small, and therefore only very small queues are required. This makes implementing WR less expensive than VCT.

5. Other. There have been some non-conventional proposals to implement flow control. In [95] a scheme called *valve routing* is discussed. In [31] another approach called *ELF* is discussed.

2.6.2. Network Interface

The Network Interface (NI) is responsible for transmitting and receiving information between the network and the processing element (be it the memory, caches or registers inside the CPU). That is, it is responsible for transmitting information from the source processing element, through the network, and into the destination processing element via its own NI, and vice-versa,

Current designs of NIs are connected to a fast system bus, such as PCI. In this arrangement, the maximum bandwidth is determined by the bus performance. However, all communications have to be mediated by the kernel which adds a significant overhead for every transfer (typically this consists of a system call and a memory

copy for transmission; and an interrupt, a system call and a memory copy for reception. Each of these steps increase latency, and since they imply a context switch, second order effects such as cache pollution would arise [98]). This overhead imposes a very high penalty for small transfers, and hence limits its usability to long messages (it is suitable only for coarse-grained communications). However, recent research in parallel computing suggests that for scientific applications, the average messages are small in size (average between 19-230 bytes) [99], and hence this kind of applications are severely penalised by the kernel involvement in the transfer, and by any extra latency added, i.e.. PCI bus arbitration, message copying, etc.

Several alternatives have been proposed, such as the use of Coherent Network Interfaces, user level DMA, block transfer mechanisms and others [26, 98–109]. In all of them the kernel is removed from the critical path. These new architectures provide User Level Communication (ULC), since the messages are transfered directly from user space to the NI and vice-versa.

2.7. Random Topologies

2.7.1. Random Graphs

A random graph $G_{k,N}$ is a graph in which τ, ι are functions that map $E \rightarrow V$ by choosing a random vertex from the set $V = \{v_1, v_2, ..., v_N\}$ independently for each edge [87]. For our purposes we will impose the condition that self loops and redundant edges are not allowed. We shall characterise such graphs by their number of vertices N and the average number of edges per vertex, k, where $|E| = kN/2$. Although random graphs have a small characteristic length, they cannot exploit the locality of communications within a cluster. Traffic is not confined at a local level, and has to traverse most of the network. For this reason they are not well suited to be used as an interconnection network of a parallel system.

The performance of random graphs as multistage networks has been studied in [54,110,111]. The results show that random graphs outperform all of the traditional networks studied under an equal hardware cost constrain. Furthermore this work shows that under faulty conditions random graphs outperform traditional networks by large margins, and that latency and congestion are reduced when random graphs are used. It is important to note however, that this results holds for multistage networks, where no locality of information is expected which could be exploited by the traditional deterministic networks.

The diameter of random graphs has been studied in [87,112–118]. A large amount

of this work is focused in the determination of the diameter close to the phase transition undertaken during the emergence of the giant component in the evolution of the graph, as well as in the transition when the graph becomes connected. Most of this work is not relevant for the present research, since we are interested in estimating the diameter of a well connected random graph. This has been achieved by estimating the average size of a nodes' neighbourhood. A lower bound for these measures can be obtained by approximating the graph to a Moore graph (a Moore graph is a perfectly expanding graph) [87]. Such approximation is valid, because if the graph is big enough, then with high probability most of the edges will be "spreading" into new territory. Using such approximation, the diameter is given as [87,118]:

$$D_{\mathcal{R}} = \frac{\ln\left(\frac{k-2}{k}(n-1)+1\right)}{\ln(k-1)} + 1; \ (k > 2). \tag{2.2}$$

The average length can be written as [118]:

$$L_{\mathcal{R}} \approx D - \frac{k\left[(k-1)^D - D(k-2) - 1\right]}{(N-1)(k-2)^2}; \ (k > 2) \tag{2.3}$$

When $N \gg k > \ln(N) \gg 1$, the following approximations can be made: $D_R \approx \ln(N)/\ln(k)$, and $L_{\mathcal{R}} \approx D_{\mathcal{R}} \approx \ln(N)/\ln(k)$. It is important to note that no graph can have a shorter diameter than a Moore graph; and therefore this measure is more of a theoretical limit than a physical reality. Furthermore very few Moore graphs exist, and they cannot be constructed for general cases.

2.7.2. Pseudo-random graphs

Random networks have the advantage over regular networks, of having a smaller characteristic length and diameter (i.e. only a small number of steps are required to reach any vertex in the network) [119]. On the other hand regular networks have the advantage of having a high clustering and can naturally embed regular communication patterns.

In [7] a method of rewiring a deterministic network to generate pseudo-random networks which can be "tuned" between both extremes by varying the randomising parameter ϕ is proposed. By numerical simulation it is shown that there is a fast transition from large characteristic lengths to small characteristic lengths, and a slow transition from a large to a small clustering coefficient. Therefore there is a big interval of probabilities in which the clustering coefficient remains almost as high

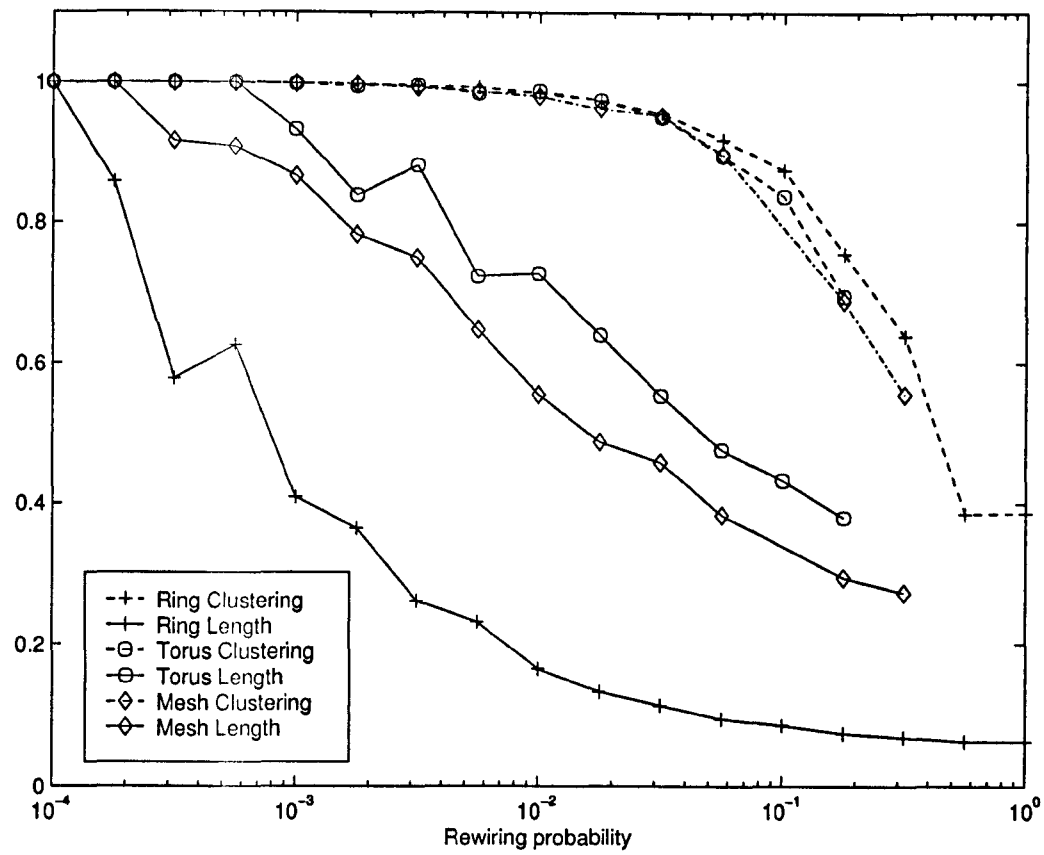
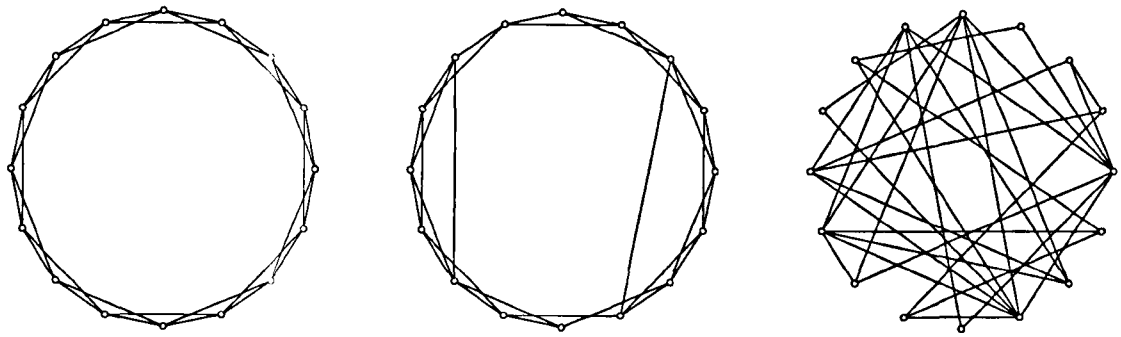


Figure 2.13.: Effects of rewiring Cartesian products of arrays. Different networks (N=1024) are randomly rewired with probability ϕ. Note the steep descent of the characteristic length at very small probabilities (a logarithmic scale is needed to resolve it). Note also, how the clustering coefficient holds until much higher probabilities before falling. All plots are normalised to unity at ϕ=0.0001.

as in the regular network while the characteristic length is almost as short as in the random case (see fig. 2.13). This phenomenon, the *small-world phenomenon*, is what makes them very attractive for their use as the interconnection network of a parallel computer (and also for other systems such as computer networks and general communications networks.).

2.8. Small World Networks

Small world networks have been proposed as a suitable model for a large number of complex graphs, such as social graphs, neural networks, disease spreading networks, electric grid network and others, which are present in natural or man-made systems [7,9,118]. Traditionally large networks have been approximated to random graphs,



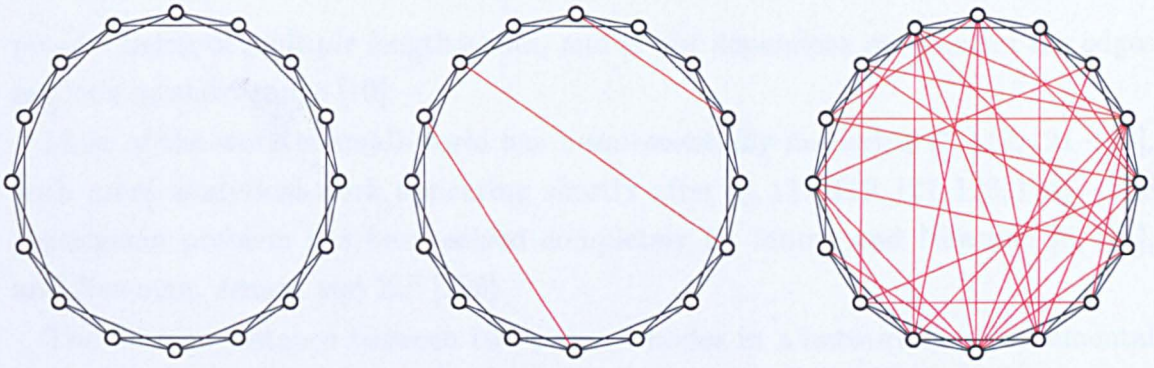
(a) $\phi = 0$ (b) $\phi = 0.1$ (c) $\phi = 1$

Figure 2.14.: Generation of SW Graphs under the conservative model. The regular ring where $\phi = 0$ corresponds to a large world, highly clustered network where L grows linearly with N. On the other hand, $\phi = 1$ is a poorly clustered network, with a small L (it only grows logarithmically with N). It is important to note that with the original rewiring process, when ϕ the resulting graph is not completely random, since the edges have selected only one of their endpoints randomly.

and although they present some resemblance with them, since both have a small diameter (and characteristic length), they differ in that the random graph does not possess any structure at the local level, while small world networks do. For instance in a typical social network of human acquaintances, there is a strong local structure, since if persons A and B know person C, there is a stronger than random chance that A and B know each other (i.e. social networks are highly clustered). In this sense the social network resembles more a regular graph such as a mesh or a ring, than a random network. However, the number of mutual acquaintances between any two randomly chosen individuals in a social network is typically small, as discovered by Milgram's experiment [120], such as in a random network.

Early work on the problem was undertake by Chung and Bollobás [67], however it was Strogatz and Watts who proposed a model for constructing and analysing such networks [7]. Experimental evidence of an abrupt reduction of length is reported in their work, while randomising a small subset of edges in an otherwise deterministic network. The size of the subset is small enough so that no structural changes at a local level are evident.

Small-world graphs have been produced in [7] using a conservative rewiring process as follows. Starting with a ring with N nodes and k connections per node randomness



(a) $\phi = 0$ (b) $\phi = 0.1$ (c) $\phi = 1$

Figure 2.15.: Generation of SW Graphs under the Additive model. Figure (a) shows the original ring. When a small rewiring probability is used as in (b), the results are similar to the conservative model, although the regularity of the underlying graph is fully conserved. As the rewiring becomes large, the graphs diverge from the conservative model, as a much larger amount of edges are introduced, as shown in figure (c).

is introduced by rewiring either end of a randomly selected edge with probability ϕ (see figure 2.14) to a new (random) destination. When the number of rewired edges is small $\phi \to 0$ the network corresponds to a large world, in which the average distance between nodes is large, $\overline{L} \gg \overline{L}_{\mathcal{R}}$. However, crucially when only a small amount of the edges are rewired $1 \gg \phi > 0$, the average distance decreases dramatically, $\overline{L} \to \overline{L}_{\mathcal{R}}$ while the graph remains ordered at a local level $C \to C_0 \gg C_{\mathcal{R}}$ (see figure 2.13). This model has gained acceptance with the scientific community, although in a revised form in which edges are added instead of rewired, with probability ϕ to the original ring, which will be referred here as the additive model (see fig. 2.15) [9].

In a small-world network there are two different length scales [8,118]. At a local level the graph is ordered within a cluster in a straight resemblance of the underlying network. Local edges within this cluster maintain local connectivity. The long range connections which are introduced by random rewiring connect the graph at a different level; the global scale, and according to Watts and Strogatz (and others) are responsible of providing a small diameter and characteristic length [7,118,121]. The mechanics of such a reduction in diameter and length are still a matter of debate, although it is clear that they are a direct result of the additional (or rewired) edges. Kasturiarangan attributes the phenomena to the multiple length scale hypothesis, which states that the phenomena is a result of the edges introduced in the rewiring

process being of multiple length scales, and is not dependent on whether the edges are long or short range [10].

Most of the work in small-world has been essentially numerical [7, 118, 121–123], with more analytical work appearing shortly after [8, 117, 119, 121, 123, 124]. The percolation problem has been solved completely by Moore and Newman [8, 125], and Newman, Jensen and Ziff [126].

The average distance between two pairs of nodes in a network is a fundamental property of it. This average (also called characteristic length) is simply [7]:

$$\overline{L} = \frac{1}{N(N-1)} \sum_{u,v \in V} l_{u,v}. \tag{2.4}$$

A small characteristic length is desired, since it is critical for a low latency network, since this will determine the average number of "hops" a packet has to make to reach its destination.

The average length $\overline{L}$ and the length distribution has been extensively studied, both by numerical and analytical methods, however neither the distribution of path lengths, nor the average length (or diameter) has been determined exactly yet [9].

2.8.1. Average Length with Underlying Lattices

For the special case of a small-world network with an underlying lattice several results have been obtained:

Dorogstev and Mendes [127] provide an exact solution on a related model, in which all shortcuts are connected to a central hub. Their results gives:

$$\overline{L}_{Doro} = \frac{N}{2\rho^2} \left[2\rho - 3 + (\rho + 3)e^{-\rho} \right], \tag{2.5}$$

where $\rho = \phi N$. The distribution of shortest distances is given as:

$$P(L)_{Doro} = \left[2 - 8L^2\phi^2 + 4L\phi(1 + N\phi) \right] e^{-2L\phi}. \tag{2.6}$$

Newman and Watts [116] and Moukarzel [117] have shown that there is only one non-trivial length scale which determines the characteristic length in a SW graph, which is given as:

$$\xi = \frac{1}{(k\phi)^{1/d}}. \tag{2.7}$$

Furthermore, Newman and Watts proved that the scaling function is of the form

$L = \alpha F(x)$, where x is a function of the number of shortcuts present in the network. Kulkarni *et. al.* have shown that the average length is related to the distribution of path lengths of diametrically opposite nodes in the graph [128]:

$$L_{Kulk} = \langle l(\eta) \rangle \left(1 + \frac{1}{N-1}\right) - \frac{\langle l^2(\eta) \rangle}{N-1}, \tag{2.8}$$

where $\langle l(\eta) \rangle$ is the average minimum distance of nodes with Euclidean distance $N/2$. Unfortunately the distribution of $l(\eta)$ is still unknown and evaluating it seems to be as complicated as the determination of the average length itself.

Newman and Watts constructed a third order Padé approximation based on series expansion of a one dimensional small world graph, which is given by [8]:

$$L_{NWP} = L_0 \frac{1 + 1.825\, Nk\phi}{1 + 1.991\, Nk\phi + 0.301\, (Nk\phi)^2}, \tag{2.9}$$

where $L_0 = 1/4$.

Newman and Watts [116] provided another good approximation for the average length in a one dimensional lattice using a continuum mean-field model. Their main result is:

$$L_{NW} = \frac{\xi}{2k\sqrt{1 + 2\xi/N}} \tanh^{-1} \frac{1}{\sqrt{1 + 2\xi/N}}. \tag{2.10}$$

Moukarzel [117] provides an analytical approximation for the generalised case (i.e. for n-dimensional lattices) using a continuum branching process. His result is:

$$L = r_c \left[1 - \frac{1}{d+1} \left(\frac{r_c}{N^d}\right)^d\right], \tag{2.11}$$

where

$$r_c = (2\phi\Gamma_d(d-1)!)^{-1/d} \ln\left(2\phi N^d\right). \tag{2.12}$$

2.8.2. Clustering Coefficient

A clique is a subgraph of $\mathcal{G}$ which has strong local connectivity. To quantify the cliquishness of a vertex u in a graph the following expression can be used:

$$C_u = \sqrt{\frac{2}{|\mathcal{N}_u|\,(|\mathcal{N}_u| - 1)} \sum_{\langle v,w \rangle \in \mathcal{N}_u^2} \frac{1}{\left(l_{v,w_{\mathcal{G}'}}\right)^2}} \tag{2.13}$$

where $\mathcal{N}_u$ is the neighbourhood of u, and $\mathcal{G}'$ is the subgraph $\mathcal{G} - \{u\}$. This is

a measure of how well connected a graph is at a local level around node u. The clustering coefficient of a graph is simply defined as the average of all the clustering coefficients of its nodes $C = \sum_{\mathcal{G}} C_u/N$.

Note that this definition is different than the one given by Strogatz and Watts in [7]. Their definition only takes into account direct connections amongst the neighbours of a node, which leads to problems with regular networks such as meshes and hypercubes which do not share direct connections amongst the neighbours of a node. By taking into account higher order connections we arrive at a more consistent and well behaved model.

The dynamic performance of the network will be affected by modifying its clustering coefficient. By having a high cliquishness, traffic within a cluster can be confined at a local level avoiding the need to travel outside of the clique. Recent measurements of the communication pattern of different applications confirm this idea; processors exchange data more often with their close neighbours than they do with far nodes [39,48]. Furthermore, if a node exchanges information with a number of neighbours, there is a high probability that the neighbours will exchange data between themselves, which is facilitated by networks with a high clustering coefficient. Some algorithms can benefit particularly well from such a topology, such as Multigrid methods where information can be confined at different levels of locality when changing the grid size (typical parallel implementations suffer from high communication costs when cycling through the different grid sizes [129]). Other examples include Iterative methods (for instance SOR), Partial Differential Equations (Navier Stokes, Fluid Dynamics) [130] and Molecular Dynamics where bonded (local) and non-bonded (global) forces need to be calculated (usually a cut-off is applied which artificially induces local neighbourhoods of varying scales in the problem) [131]. In general any method which involves the use of sparse matrices (usually derived from real systems) can benefit from this approach.

3. Small World Interconnection Networks

3.1. Introduction

It has been shown that random networks have the advantage over their deterministic relatives in that they present a network with some very desirable properties: shorter diameter and characteristic length, higher fault tolerance and very good expandability. However, these advantages are somewhat diminished due to the following reasons

- Routing is complicated and leads to the creation of artificial bottlenecks in the network; therefore the available bandwidth is not thoroughly exploited.
- Their physical and logical layouts are completely independent, leading to complex wiring and construction. Networks have to be physically laid down in three dimensional space, which favours Cartesian product networks, such as meshes, tori and cubes, since they map naturally into two and three dimensional space. The layout of a higher dimensional network (such as a random graph) is much more complicated.
- All nodes have physical neighbours which arise naturally in the network. Random networks completely ignore these neighbourhoods. Furthermore, random networks have a low tendency of constructing new neighbourhoods (they have a low cliquishness), therefore locality of information cannot be fully exploited.
- A large number of parallel algorithms are designed with a particular underlying geometry which creates deterministic communication patterns. These are difficult to accommodate in a random network, leading to increased traffic.
- The resulting network is not regular (different nodes have different degrees), increasing the cost and complexity of the network.

Clearly it is desirable to have a network with the benefits of a random network (low diameter and characteristic length and high fault tolerance), but without its drawbacks. Traditional interconnection networks provide the ability to exploit locality of information for a subset of communication patterns; however not all of them can efficiently support generalised communication patterns, and those who do tend to be too expensive.

A good alternative is a small-world network, in which the underlying deterministic graph provides regularity and the ability to exploit locality of a subset of communication patterns, while the over-imposed random graph creates an efficient and rich interconnection which lowers the diameter and average length, adding support for generalised communication patterns. Furthermore these networks scale-up better, than their underlying deterministic networks do [8].

We know that the small-world phenomenon implies that it is sufficient to add a small number of these random shortcuts to a deterministic graph to reduce its diameter and characteristic length dramatically. Therefore the structure of the deterministic graph will remain largely unmodified and will determine, into a large extent, the structure of the resulting small-world graph.

In this chapter random small-world interconnection networks (RSW graphs) are introduced, and some of their basic characteristics are derived. A model of the latency under random traffic using shortest path routing is constructed and validated using computer simulations. The model shows that when a small number of shortcuts are present the RSW networks saturate due to excess contention in them. This analysis is completed with the construction and experimental validation of a congestion model which corroborates the results. With the aid of these models it is possible to estimate the minimum number of shortcuts required to avoid premature saturation in the network. Such constructions are demonstrated for the torus and the hypercube network, although they are not limited to them. It is important to note that the model ignores the propagation delay in the edges, and it assumes an equal bandwidth amongst all edges; however due to technical limitations this might not be the case, as large wires introduce a larger delay and are more difficult to drive at high speeds than shorter wires.

3.2. Generation Model

A small-world network is defined in [7] as a network in which its characteristic length approaches that of an equivalent random graph $L \rightarrow L_{\mathcal{R}}$, while its clustering co-

efficient is much larger $C \gg C_{\mathcal{R}}$. A small world network can be created by the superposition of two graphs; a long range graph (which is not necessarily random) which joins local clusters, with a second graph, which provides local clusters and order (and which usually is deterministic). This graph will usually have other desirable properties, such as regularity and symmetry, as well as a small degree.

For the purpose of this work we will define a small world graph (or SW graph for short) as:

$$G_{SW} = \Psi(G_S, G_L), \tag{3.1}$$

where G_S is the underlying or local graph, G_L is the long range graph, and Ψ is a function. These graphs are restricted to the cases where $|G_{SW}| = |G_S| \leq |G_L|$ and $|E_{SW}| \leq |E_S| + |E_L|$ (note however that Ψ might exclude edges present in either G_S or G_L from G_{SW}.

By selecting different functions for Ψ several generating models can be defined. Two models are particularly interesting, which are defined in the following sections.

3.2.1. Conservative Model

Strogatz and Watts used the model described in [7] for generating SW networks, in which for each edge of the graph, if a Bernoulli trial with probability ϕ is successful, a new randomly chosen vertex is assigned to either of its extremities, with the condition that self-loops and redundant links are not allowed (in such eventualities another random vertex is chosen).

The rewiring process corresponds to deletion followed by addition of an edge. A graph can be associated to each of this processes, such that deleted edges form G_{L^-}, and G_{L^+} is formed by the added edges. From this $G_{L'} = G_{L^+} - G_{L^-}$ can be informally define, and

$$G_{SW} = \Psi(G_S, G_L) = G_S \bigcup G_{L'}. \tag{3.2}$$

Alternatively Ψ is defined as:

$$G_{SW} = \Psi(G_S, G_{L^+}, G_{L^-}) = \left(G_S \bigcap G'_{L^-}\right) \bigcup G_{L^+}. \tag{3.3}$$

This generation model has the characteristic of preserving the number of edges in the graph. However the structure of G_S is not preserved in its entirety. This has the disadvantage that some of the useful properties due to the deterministic nature of G_S cannot be exploited in G_{SW}. This is especially important for the routing

properties of the network, as will be discussed in section 3.7. These characteristic makes networks based on this model less attractive than networks based on the additive model. Furthermore, theoretical modelling of these networks has proven to be more difficult than in the additive model [9], and most of the later theoretical work is based on the additive model [8–10,116,121,124,126–128]. For these reasons we will focus on additive networks throughout this research.

3.2.2. Additive Model

To overcome the difficulties created by the complete lack of determinism in the conservative generation model, the additive model is introduced. This revised model was introduced by Newman and Watts for different reasons in [8].

The model is similar to the conservative case; for each edge in G_S a Bernoulli trial with probability ϕ is performed, and if successful a new random edge is inserted in G_{SW}. For our purposes, neither self loops nor repeated edges will be allowed. In such eventualities a new random edge is selected.

In this case G_L is formed by all edges to be inserted, and Ψ is simply:

$$G_{SW} = \Psi(G_S, G_L) = G_S \bigcup G_L. \tag{3.4}$$

The number of edges in G_{SW} is larger than the edges of G_S and is given as:

$$|E_{SW}| = |E_S| + |E_L| = (1 + \phi)\,|E_S|\,. \tag{3.5}$$

3.3. Some Properties of RSW Networks

The most important properties of RSW networks are their small diameters, small characteristic lengths and high clustering coefficients, since they exploit the small-world phenomenon, as explained before. The following is a characterisation of some other important properties.

3.3.1. Degree and switch complexity

The degree of a RSW will most certainly be higher than that of its underlying network G_S, k_0. Due to the additive rewiring process an average of $\epsilon = \phi k_0 N$ edge ends are created in the network. Assuming that these are distributed evenly through the set of N nodes (which is generally true except for a small factor due to the fact

that self loops are not allowed), the probability $P(k_0+\Delta)$ that the degree of a node is increased to $k_0+\Delta$ is given by the binomial distribution:

$$P(k=k_0+\Delta)=\left(\frac{1}{N}\right)^{\Delta}\left(\frac{N-1}{N}\right)^{\epsilon-\Delta}\left(\begin{array}{c}\epsilon\\ \Delta\end{array}\right). \tag{3.6}$$

The average expected degree increase is easily found to be $E[\Delta]=\epsilon/N$, however we are more interested in obtaining the expected degree of the graph Δ_M, which is the average of the largest degree over a large number of realisations of the RSW graphs. This is easily obtained by noting that the expected number of nodes of degree Δ is given by $NP(k=k_0+\Delta)$, and that the largest expected degree will have an average multiplicity of one over a large set of randomisations. Therefore the expected degree of the graph is $k_0+\Delta_M$, which can be obtained from:

$$\left(\frac{1}{N}\right)^{\Delta_M}\left(\frac{N-1}{N}\right)^{\epsilon-\Delta_M}\left(\begin{array}{c}\epsilon\\ \Delta_M\end{array}\right)=\frac{1}{N}. \tag{3.7}$$

It is possible to extend this model for the conservative rewiring generation model by noting that the process migrates only one of the edge ends, and therefore $\epsilon=\phi k_0 n/2$ in this case.

Figure 3.1 shows a comparison between the models and experimental data for different networks, in which a good fit between both is seen, except at very large values of ϕ. This is due to the fact that at such large probabilities the assumption of evenly distributed edge ends does not hold.

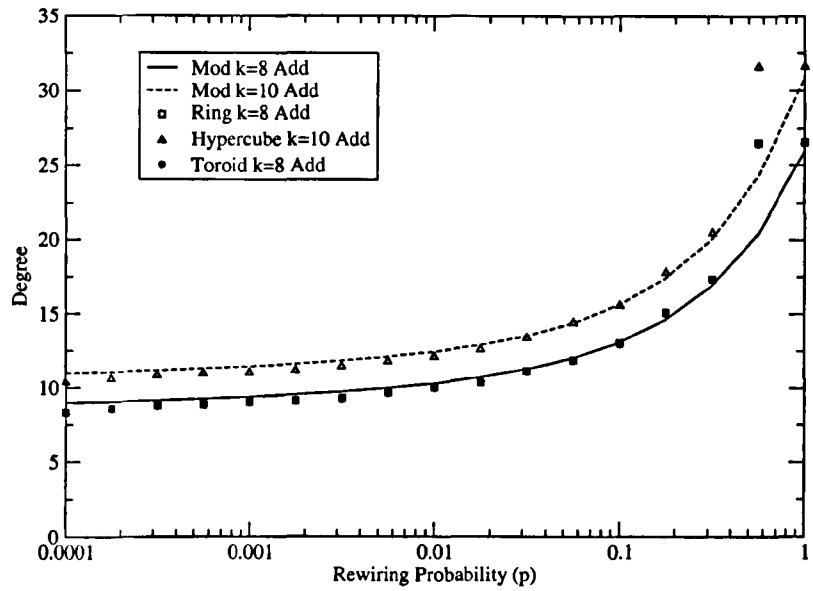
This large degree is perhaps the main drawback of a RSW network, since it implies additional complexity and cost, even if only a small number of nodes have a high degree. An alternative worth considering is to modify the rewiring procedure and constrain the maximum degree of a node to a maximum value (the degree of the graph). Since only a very small number of nodes exhibit a very high degree, it is expected that such process would not disrupt our main results significantly.

The switch complexity of a node will be approximated as k^2, since this is a rough estimate of the number of internal routes within the router. The average switch complexity will be given as:

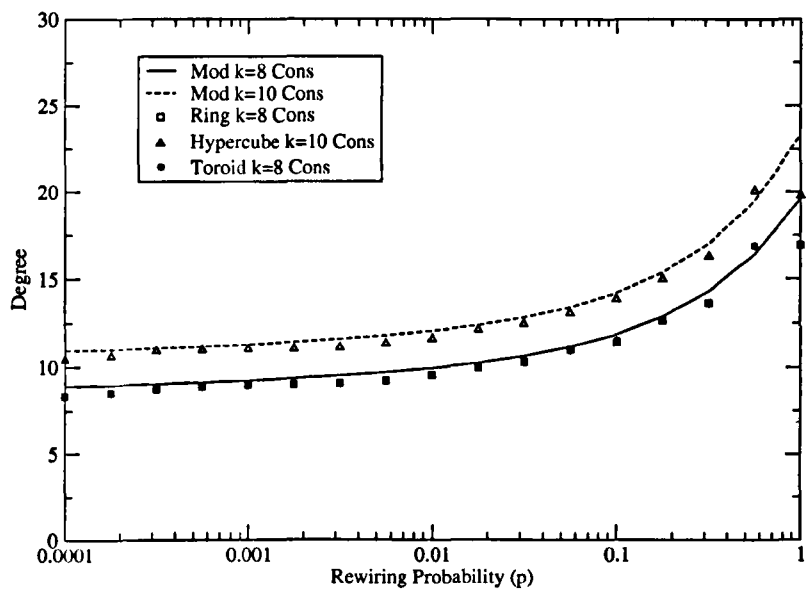
$$X=\left[\sqrt{X_0}\left(1-\frac{\phi k}{2}\right)+\frac{\phi k}{2}\right]^2 \tag{3.8}$$

where X_0 is the average switch complexity of the original network.

These measures are shown in figure 3.2 for different underlying networks.

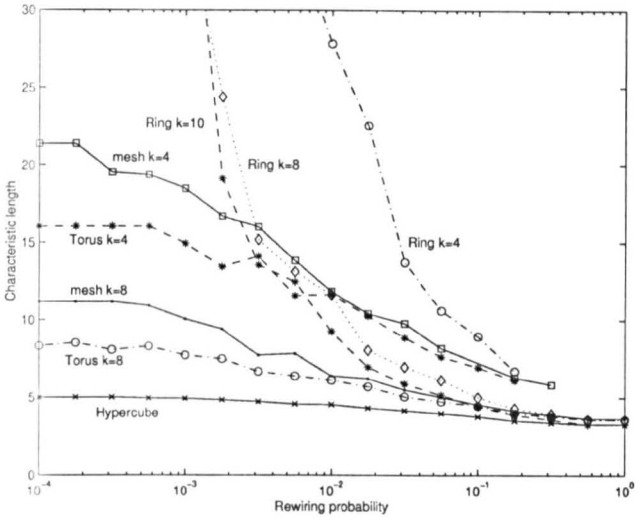


(a) Additive Model

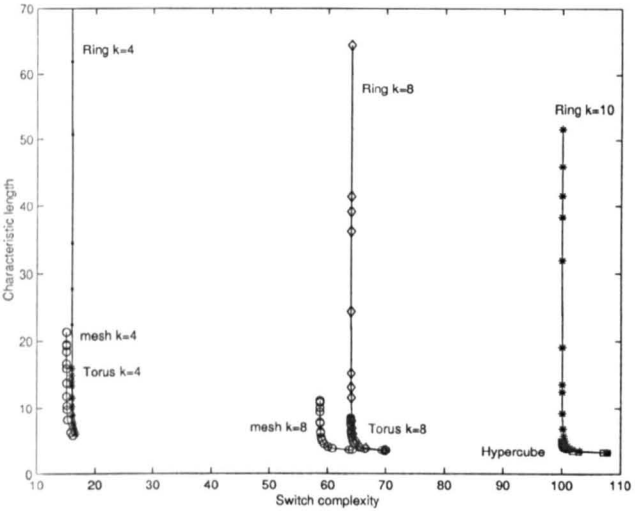


(b) Conservative Model

Figure 3.1.: Effect of rewiring on the degree of various graphs (N=1024). The graphs show experimental measurements averaged over 100 realisations next to the model predictions. The models shows a good fit of data except at very high rewiring probabilities, where the assumption of evenly distributed shortcut ends does not longer holds.



(a) Characteristic length vs. rewiring probability.



(b) Characteristic length vs. average switch complexity.

Figure 3.2.: Characteristics of some RSW networks (N=1024 in all graphs). In (a) the characteristic length of all graphs is lowered with increasing rewiring probabilities (downward direction for all graphs). The final value depends only on N and k, since the final graphs are almost completely random. Note however that some graphs become disconnected before reaching such a value. The switch complexity is not increased considerably throughout the rewiring process as shown in (b).

Graph	Size	Diameter	Reduction
D-Mesh	k^D	$D(k-1)$	$O\left(N^{1/D}\right)$
D-Torus	k^D	$D\left\lfloor \frac{k}{D} \right\rfloor$	$O\left(N^{1/D}\right)$
k-Ring	N	$\left\lfloor \frac{N-1}{2k} + \frac{1}{2} \right\rfloor$	$O(N)$
D-Hypercube	2^D	D	$O(\log\log N)$
k-ary D-cube	k^D	$\frac{Dk}{2}$	$O(\log\log N)$

Table 3.1.: Maximum expected reduction in diameter of some popular graphs.

3.4. Diameter and Average Length in the General Case

Since ϕ will be chosen in a region where $L \to L_{\mathcal{R}}$, we will also have $D \to D_{\mathcal{R}}$. That is $D \approx \ln(N)/\ln(k)$ for $N \gg k > \ln(N) \gg 1$. This is smaller than the diameter of most deterministic graphs. This is a very important characteristic of RSW networks, as a small diameter is a desirable characteristic for an interconnection network, since it can lead to a small latency. However, when only a small number of shortcuts are introduced the effect can be the opposite, as these are likely to become congested, and increase the latency or even saturate the network (this behaviour is studied in sections 3.8 and 3.9).

Although an exact expression for the reduction in diameter and average length for the general case has proven to be elusive, and has not been derived in this work, experimental evidence suggests that this reduction behaves in a similar fashion as that of a ring (which is studied thoroughly in this work), as shown in figure 3.2. The maximum reduction in diameter and length is much simpler to calculate, as on the limit of large ϕ, these measures are simply the ratio of the initial diameter and average length of a graph with the respective measures of a random graph.

The maximum expected reduction in diameter is shown in table 3.1 and plotted in figure 3.3. As expected graphs with a large diameter (such as arrays, including the mesh and ring) benefit the most from the addition of shortcuts, while graphs with a relative smaller diameter (such as the hypercube, de Bruijn and others not shown), have less to gain from the process (however note that the hypercube suffers from a very large degree for such large systems).

In general not all of the edges in the graph form an efficient expander; and for the cases where only a small amount of them will provide the reduction in diameter (such as in the deterministic constructions undertaken in the following chapters), it is possible to approximate the diameter using a somewhat naive approach, in which

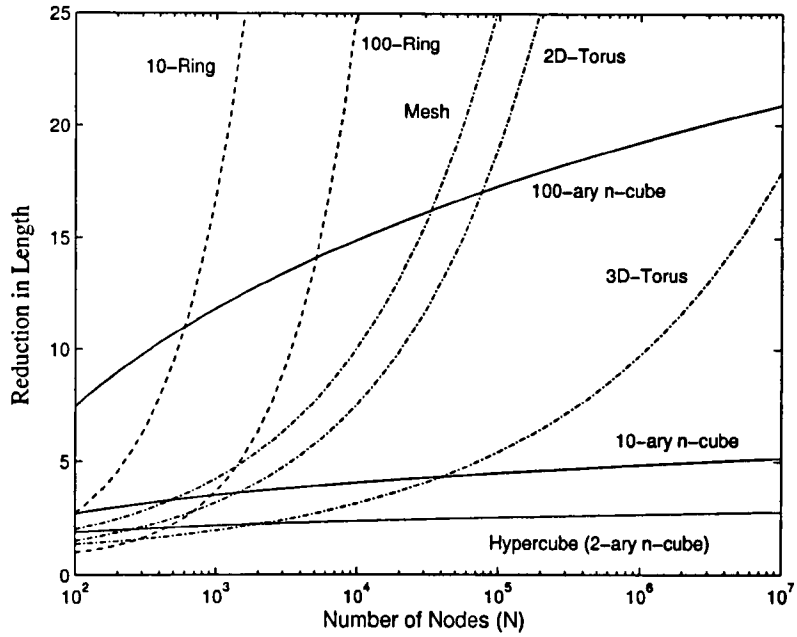


Figure 3.3.: Plot of the maximum reduction in diameter of some popular graphs.

an effective node degree (k_e) for the underlying graph is determined from a Moore graph approximation as follows:

$$k_S(k_e - 1)^{D_S - 1} = N - 1, \tag{3.9}$$

which has the approximate solution:

$$k_e \approx 1 + N^{\frac{1}{D_S}}. \tag{3.10}$$

Assuming that edges in G_L are very good expanders, the diameter of $G_S + G_L$ can be approximated to that of a Moore graph as:

$$(k_S + k_L)(k_e - 1 + k_L)^{D-1} = N - 1, \tag{3.11}$$

and

$$D \approx \frac{\log N}{\log(N^{\frac{1}{D_S}} + k_s\phi)}. \tag{3.12}$$

3.5. Diameter and Average Length with Underlying Lattices

For the case where the underlying graph is a ring or a lattice, several results regarding the average length are known and have been described in section 2.8.1. The diameter on the other hand, has received little attention. This is probably because the estimation of the diameter is complicated by the fact that it is not only required to know the average maximum distance in an average graph, but it is also required to determine the distribution of distances within that given graph, since the diameter is the largest of such distances.

3.5.1. Previous Work

The saturation time of the aggregation of new nodes to the volume of infected nodes in several models can be identified to the mean diameter of the graph. Dorogstev and Mendes provide the following approximations in a simple mean-field like undirected model in [127] for a one dimensional ring (k=1):

$$\begin{aligned} t_{sat}/2 &= \langle D_{Doro} \rangle /2 = -\frac{1}{\log(1-\phi)} + \frac{1}{2\phi} + \frac{N-1}{2} \\ &\quad -\sqrt{\frac{1}{\log^2(1-\phi)} + \frac{5}{4\phi} + \frac{N-3}{2\phi} + \frac{(N-1)^2}{4}} \qquad (3.13) \\ &\sim \frac{1}{2\phi}\left[3 + \phi N - \sqrt{9 + 2\phi N + \phi^2 N^2}\right], \qquad (3.14) \end{aligned}$$

where the last approximation is valid for large N. As it is apparent from fig. 3.5, equation 3.14 does not provide a very accurate fit with experimental data for small ϕ. Furthermore this model does not incorporate the effects of a larger degree (k>1), and therefore can only be used for k=1, and for large values of ϕ.

An expression corresponding to the mean diameter is derived by Moukarzel in [117], which is given as:

$$\langle D_{Mou} \rangle = t_{sat} \sim (2\phi\Gamma_d(d-1)!)^{-1/d} \ln(2\phi\Gamma_d N). \qquad (3.15)$$

This equation provides a good approximation to the diameter of the graph for the cases where $(2\phi\Gamma_d(d-1)!)^{-1/d} N \gg 1$ (see figs. 3.5 and 3.7); however it starts diverging at this point, such that $\lim_{\phi \to 0} D \to -\infty$. We are interested in providing a better approximation which is well behaved for all values of ϕ, and which provides

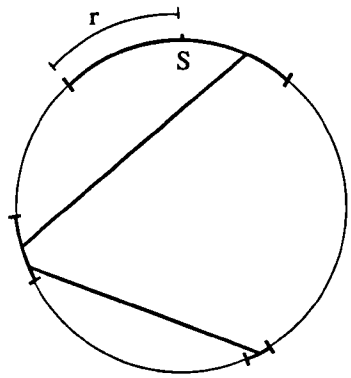


Figure 3.4.: Mean-Field diameter model. Suppose a disease starts spreading from the source node (S) with constant velocity through the underlying ring. As shortcut ends are reached new secondary infections are created at random locations.

an accurate estimate of the diameter of small graphs (where small means $N \ll \infty$ in this context). This is achieved in the following sections.

3.5.2. Diameter in the mean-field approximation

In this section we provide a simple extension to the Newman and Watts mean-field model described in [116], in order to obtain an approximation for the diameter of the networks. Suppose a vertex s is randomly selected in the continuum SW graph shown in figure 3.4.

Furthermore, suppose a disease starts spreading from s and travels a distance r in all directions. The surface area of the disease spread (i.e. the number of nodes infected in the interval $[r, r + \delta r]$) has been calculated by Newman and Watts, and is given by [116]:

$$A(r) = 2k + \frac{4Nk}{\xi}\left(\mu(r) - \mu(r)^2\right), \tag{3.16}$$

where the normalised number of uninfected nodes is approximated as [116]:

$$\mu(r) = \frac{1}{2}\left[1 + \sqrt{1 + 2\xi/N}\tanh\left(\tanh^{-1}\frac{1}{\sqrt{1 + 2\xi/N}} - \frac{2kr}{\xi}\sqrt{1 + 2\xi/N}\right)\right] \tag{3.17}$$

We are interested in obtaining the maximum distance r_{sat} where all nodes have been infected since this corresponds to the diameter of the network. This is given by:

$$\int_0^{r_{sat}} A(r)dr = N. \tag{3.18}$$

By evaluating the integral and solving for r_{sat} we get:

$$r_{sat} = \frac{\xi}{2k\sqrt{1+2\xi/N}} \left(\coth^{-1}\left(\sqrt{1+2\xi/N}\right) \pm \cosh^{-1}\left(\pm\sqrt{1+N/(2\xi)}\right)\right). \tag{3.19}$$

where as usual, trigonometric inverses are represented by functions like $\cosh^{-1}$. We are interested in the largest real solution, which is the one with only positive signs. The mean diameter is therefore given as:

$$\langle D_{MF} \rangle = \frac{\xi}{2k\sqrt{1+2\xi/N}} \left(\cosh^{-1}\left(\sqrt{1+N/(2\xi)}\right) + \coth^{-1}\left(\sqrt{1+2\xi/N}\right)\right). \tag{3.20}$$

Although this approximation is only exact at the limit of a large system size $N \gg 1/(k\phi)$ it does provide a well behaved function, and as expected $\lim_{\phi \to 0} D = N/(2k)$, as it can be seen in figs. 3.5 and 3.7.

3.5.3. Forking Process Diameter Approximation

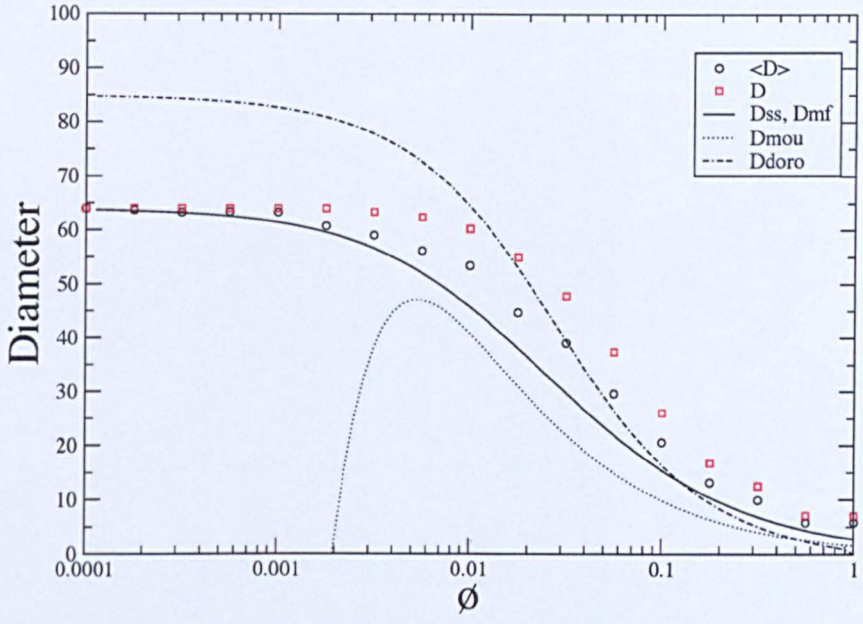
We are interested in providing a better approximation to the diameter of a graph (in particular for small graphs). We will do so by providing an extension to the Moukarzel's model as follows.

Consider a d-dimensional continuous space where nodes are spread evenly. Suppose that a disease spreads with radial velocity 1 from its randomly chosen source (which corresponds to the spread through the underlying lattice). The shortcuts in the system are distributed randomly, such that the density of shortcut ends is $\rho = 2\phi$. The initial infection will grow as a d-dimensional sphere of surface $\Gamma_d t^{d-1}$. As new nodes are reached by the sphere, shortcuts will be found at each unit step with probability ρ and new "secondary" spheres will start to grow at random locations in space (see fig. 3.6).

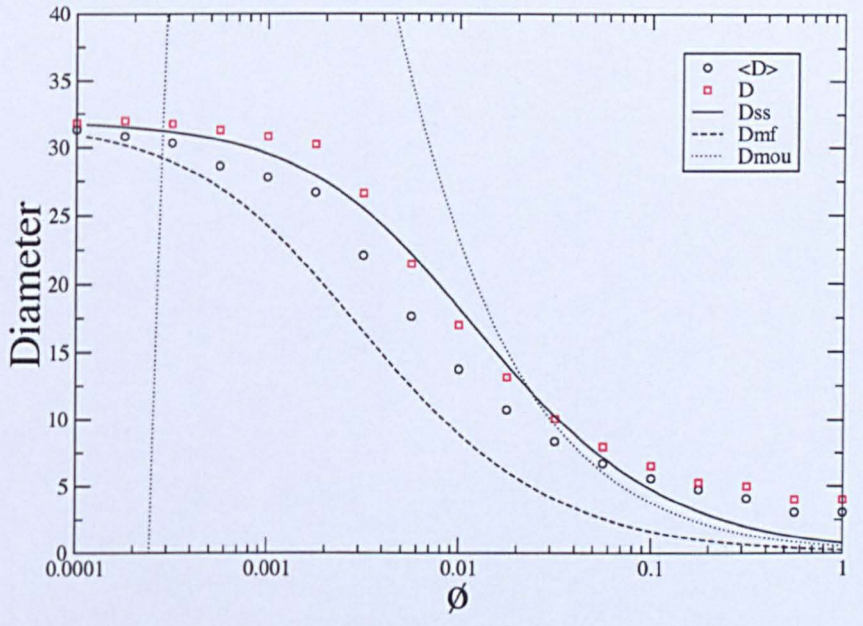
The total infected volume is given by Moukarzel as [117]:

$$V(t) = \Gamma_d \int_0^t \tau^{(d-1)} \left(1 + \rho V(t-\tau)\right) d\tau, \tag{3.21}$$

whose solution is given by the rescaled variables $\widetilde{V} = \rho V$ and $\widetilde{t} = (\rho \Gamma_d (d-1)!)^{1/d} t$



(a) RSW Ring (k=1, N=128)



(b) RSW Ring (k=4, N=256)

Figure 3.5.: Diameter (D) and mean diameter (<D>) of a 128 and a 256 node ring. The graph shows measurements, against model predictions for the proposed diameter models; the mean-field D_{MF} (eq.3.20) and the branching process D_{SS} (eq.3.29), as well as the most precise of the previously known models, which is Moukarzel's model D_{Mou} (eq.3.15).

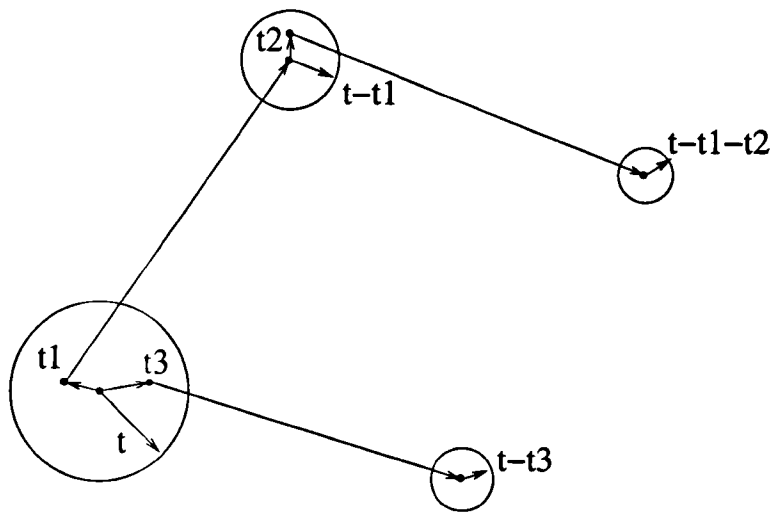


Figure 3.6.: Moukarzel's branching process model. Suppose a disease starts spreading from the source node (centre of large circle), with a constant radial velocity. As the infection grows, shortcut ends are reached with probability ρ, and new "secondary" spheres will start to grow at random locations in space.

as:

$$\widetilde{V}(\widetilde{t}) = \frac{1}{d}\sum_{n=0}^{d-1} e^{\mu_d^n \widetilde{t}} - 1, \tag{3.22}$$

where $\mu_d = e^{2\pi i/d}$. In a finite system, after a certain time the process will stop, when all nodes have been infected. This time t_{sat} is the same as the mean diameter of the network and is given as:

$$t_{sat} \sim (\rho\Gamma_d(d-1)!)^{-1/d} \ln(\rho\Gamma_d N). \tag{3.23}$$

As previously noted this expression diverges from the actual value when there are a small number of shortcuts present, and when the system size is small. Unfortunately as it will be shown in section 3.9, these are precisely the conditions which trigger a large congestion in the network, so we are interested in providing a better approximation for these cases.

3.5.3.1. Small Systems Correction

From eq. 3.21 we see that the total infected volume at time t has been modelled as the sum of the volume of the primary sphere $\Gamma_d \int \tau^{(d-1)} d\tau$ with a contribution

of all $\rho V[t-\tau]$ spheres born at time $\tau \leq t$. This model is accurate for the cases where $N \rightarrow \infty$, however this is not a correct approximation for small system sizes. For these systems the number of spheres born at time τ is not proportional to $\rho V[t-\tau]$ because as the infection grows, although new shortcuts are encountered at a rate ρ, the shortcuts are more likely to lead to an already infected area, in which case no new spheres are born. The rate of uninfected shortcuts (where an uninfected shortcut is defined as a shortcut leading to an uninfected area) is therefore $\rho(N - V[t-\tau])/N$. Furthermore, the growth of the secondary spheres is reduced by the same mechanisms, so we can write:

$$V(t) = \Gamma_d \int_0^t \tau^{(d-1)} \left(1 + \rho \frac{[N - V(t-\tau)]}{N} V(t-\tau)\right) d\tau. \tag{3.24}$$

We proceed to solve eq. (3.24) by changing the integration variable and taking the d-derivative with respect to t to find:

$$\frac{\partial^d}{\partial t^d} V(t) = \Gamma_d (d-1)! \left[1 + \frac{\rho}{N}(N - V(t))V(t)\right]. \tag{3.25}$$

No general solution for this differential equation has been found by the author yet. For the case where $d = 1$, one can write:

$$V'(t) = 2k \left[1 + \rho V(t) - \frac{\rho}{N} V(t)^2\right], \tag{3.26}$$

which has the solution:

$$V(t) = \frac{N}{2} + \frac{N}{2}\sqrt{\frac{4}{\rho N} + 1} \tanh\left[\frac{\rho}{2}\sqrt{\frac{4}{\rho N} + 1}\,(2kt + N\kappa_1)\right]. \tag{3.27}$$

The constant κ_1 can be evaluated by taking the derivative of $V(t)$ and evaluating it as $\lim_{t+\rightarrow 0} V'(t, \kappa_1) = 2k$, which is the initial rate of growth of the infection. The complete solution is therefore:

$$V(t) = \frac{N}{2} + \frac{N}{2}\sqrt{\frac{4}{\rho N} + 1} \tanh\left[k\rho t\sqrt{\frac{4}{\rho N} + 1} + \frac{1}{2}\cosh^{-1}\left(1 + \frac{\rho N}{2}\right)\right], \tag{3.28}$$

The infection will grow until a saturation time, where all nodes have been infected, which can be found from $V(t_{sat}) = L$ and is given as:

$$\langle D_{SS} \rangle = t_{sat} = \frac{\cosh^{-1}\left(1 + \frac{\rho N}{2}\right) + 2\mathrm{sech}^{-1}\left(\frac{2}{\sqrt{4+\rho N}}\right)}{2k\rho\sqrt{\frac{4}{\rho N} + 1}}, \tag{3.29}$$

which is the mean diameter of the graph, and provides a much better fit than the previous models, as can be seen in figs. 3.5 and 3.7.

3.6. Bisection Width

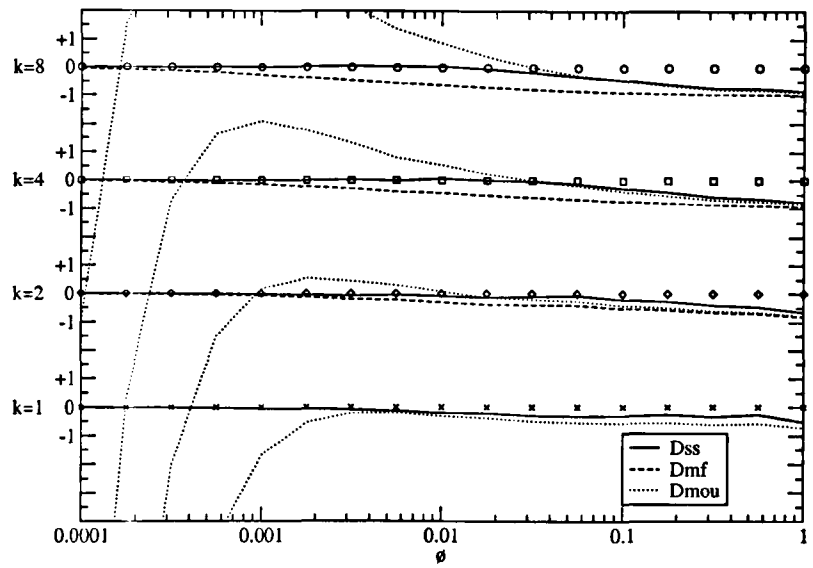
The bisection width of a SW graph depends on both, the original graph, and the randomisation process. The main obstacle in determining the bisection width of a small-world graph, is that the original network partition where the minimum bisection width is obtained is not necessarily the same as the partition where the SW bisection width is obtained. Since there is no a priori knowledge of the underlying network, the bisection width cannot be determined exactly. However we can estimate a higher bound by fixing the partition to that of the original network, where the minimum bisection width is obtained. The width of a bisection $|B|$ is defined as the minimum number of edges linking two partitions G^+, G^- such that $G^+ \biguplus G^- = G$ and $|G^+| = |G^-|$ (plus or minus one). The set of edges linking the partitions is the bisection set B. The bisection width of the underlying network is $|B_0|$ (or simply B_0 when no confusion arise).

An upper bound for the bisection width in the additive model is determined by estimating the number of new edges introduced which span the original partitions where B_0 is determined, G^+ and G^-. Since the addition of edges is random (except for the fact that parallel edges and loops are not accepted), there are $\phi kN/4$ edges introduced to B, giving the bisection as:

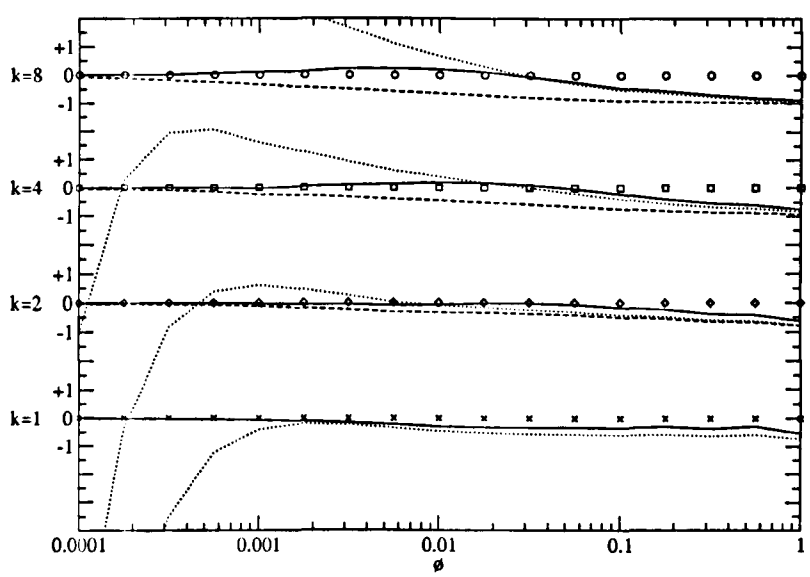
$$B_A = B_0 + \frac{\phi kN}{4}. \tag{3.30}$$

To determine the bisection width in the conservative model it is required to quantify the number of edges deleted from B, and the number of new edges in B. The randomising procedure will select new random endpoints for ϕB_0 of the original edges in B_0. Assuming the selection of endpoints for this process is distributed evenly through G^+ and G^-, it follows that $\phi B_0/2$ edges will migrate from the bisection into G^+ or G^- (and are therefore deleted from the bisection).

Initially there are $\frac{kN}{2} - B_0$ edges not forming part of the partition, which can be selected by the rewiring algorithm and migrate into the partition. If we assume that the new endpoints of the selected edges are distributed evenly through both partitions then half of them will span both partitions, and therefore the bisection width is:



(a) RSW Ring, N=128.



(b) RSW Ring, N=256.

Figure 3.7.: Mean diameter model deviations for a 128 node ring (a), and a 256 node ring (b). Deviations are shown as percentages for k=1, k=2, k=4 and k=8. The newly constructed mean-field, D_{MF} from eq.3.20 and branching process model, D_{SS} from eq.3.29 are shown, as well as Moukarzel's model D_{Mou} (eq.3.15).

$$B_S \approx B_0 \left(1 - \frac{\phi}{2}\right) + \frac{\phi k N}{4}. \tag{3.31}$$

In general the supposition of evenly distributed endpoints does not hold and a better approximation is required (although the errors incurred by it when calculating the number of edges coming out of B_0 is small, this is no longer the case for the large number of edges originally not in B_0).

A better approximation is derived as follows. Suppose that $B_0 = \{\}$, which forces all nodes in a partition to only have edges within the partition. For each node the probability P_k of rewiring each of its k edges to the opposite partition is successively evaluated as follows:

$$P_1 = \frac{\phi}{2} \frac{N/2}{N-k-1}, \tag{3.32}$$

since there are no edges linking the current node into the opposite partition, the factor $\phi/2$ is due to the fact that edge e is considered twice, once in each endpoint of e, ($N/2$ is the number of possible destinations for an edge in the opposite partition, and $N-k-1$ is the total number of possible destinations). For the second edge the probability is:

$$P_2 = \frac{\phi}{2} \frac{N/2 - P_1}{N-k-1}, \tag{3.33}$$

since no repeated edges are allowed. Therefore:

$$P_i = \frac{\phi}{2} \frac{N/2 - \sum_{j=1}^{i-1} P_j}{N-k-1}, \tag{3.34}$$

and the total number of edges crossing the partition from the current node is given by:

$$B_{node} = \sum_{i=1}^{k} P_i = \frac{N}{2} - \frac{N}{2} \left[\frac{N-k-1-\frac{\phi}{2}}{N-k-1}\right]^k. \tag{3.35}$$

Combining this equation with the previously determined expectation of edges migrating out of the partition in eq.3.31 yields the expected bisection width as:

$$B_S = B_0 \left(1 - \frac{\phi}{2}\right) + N B_{node}. \tag{3.36}$$

For both models the bisection width is $O(\phi N)$ which implies that the networks are expandable as they can support (with a constant penalty) the expected traffic increase which grows as $O(N)$.

3.7. Routing in RSW

The addition of long range edges introduces a rapid decrease in the diameter and average length of the graph [117], which can be exploited by the routing algorithm to reduce the delivery time of all packets (provided the conditions described in section 3.9 are met). Unfortunately providing minimal routes in the case of random small world graphs is difficult. Shortcuts are wasted if the routing algorithm is unable to find and exploit them. However there is no simple decentralised routing algorithm which provides minimal (or close to minimal) routing. Shortest path algorithms (Dijkstra, Bellman Ford), or their decentralised counterparts, can be used to provide minimal routing and exploit the small diameter and characteristic length. Unfortunately the complexity of these algorithms is high, $O(E + V \log V)$ and $O(\min(D, V)E)$ respectively (where D is the maximal number of edges of any shortest path) [132], and it is not plausible to solve this problem on-line for each newly arrived packet at an intermediate node. However it is still possible to construct off-line routing tables using these algorithms and use them to perform the routing functions. In this case, the routing tables are constructed only once when the node comes on line, or when the topology of the network changes (perhaps due to a fault).

Jon Kleinberg analysed the performance of decentralised geometrical aware routing algorithms in small-world graphs [124], and showed that there is no decentralised algorithm capable of providing paths of expected short length for the additive RSW graphs. He proceeds to construct a family of graphs, where the expected length of shortcuts is proportional to $d(u, v)^{-r}$. He shows that the optimal exponent is $r = 2$ for routing using a decentralised algorithm, and that no other decentralised algorithm performs better than the simple geometrical greedy one (where a packet is routed to the node closest to the destination, as viewed by the current node, with only local information). In this case the expected routing diameter is proportional to $(\log n)^2$, which is larger than the diameter achievable if minimal routes were assigned, which is proportional to $\log n$. This work is based on the assumption that no global topological information is available, which can be exploited by the routing algorithm. However this is not necessarily the case for RSW graphs.

It is important to note that a basic requirement for a routing algorithm is to be deadlock free [92–94]. Deadlock occurs when packets cannot advance any further because the queues are full, and there is a cycle of ungranted requests for resources [133,134]. A sufficient condition to avoid deadlock is to avoid cycles in the channel dependency graph [133]; however it is not a necessary condition, as deadlock free routing algorithms allowing the existence of cycles in the channel dependency

graph can be designed, as long as there exists a connected subset of channels free of cyclic dependencies [134]. The use of virtual channels can simplify the design of deadlock free routing algorithms, since it allows more flexibility in the design [135]. For Small-World interconnection networks a deadlock free routing algorithm can be constructed by defining a subset of connected channels with no cyclic dependencies in the underlying deterministic graph. One possible approach is to identify a virtual network with these channels and to allow traffic to be routed amongst all channels, with the condition that once a packet traverse a channel which belongs to the cycle free subset, it is not allowed to leave that virtual network (otherwise the subset is not guaranteed to be cycle free).

3.7.1. Greedy Geographical Routing

This routing algorithm is based on routing incoming packets to the closest node to the destination, as measured by the underlying graph. Each node needs only to know the topology of the underlying graph (which is usually regular), and its long range connections; so only local information is used. It is also possible (and convenient) to have a routing algorithm for the underlying graph $\mathcal{A}(G_S)$ which can be used as an advantage (for instance to help implement deadlock freedom).

In greedy routing, when node v receives a packet, it extracts the destination t from within it. At this point node v calculates the distances $l^{G_S}_{\mathcal{N}_v,t}$ (that is the distances between its neighbours and the destination, as seen by the underlying graph), and route the packet to node u, where $l^{G_S}_{u,t} = \min\{l^{G_S}_{\mathcal{N}_v,t}\}$. Ties can be broken by several means (random, largest ID, smallest ID, $\mathcal{A}(G_S)$ has preference, etc.). If a routing algorithm for G_S is available, the assignment can be done using the algorithm where the long range connections are used if at each routing step this provides an advantage.

It is obvious that some (or a significant part) of the shortcuts will not be exploited by this routing technique; and for this reason we are interested in determining the routing distance in order to compare it with that achieved by the full shortest path routing algorithm. The distance distribution of graph G_S will be used to determine the routing distance. Assuming that the graph G_S is symmetrical the distance distribution function is the same for each source node. The set of nodes at distance i from node t is given by $\mathcal{D}_i(t)$. The number of nodes at distance i from a given node t is simply $|\mathcal{D}_i(t)|$, however $\mathcal{D}_i(t)$ will be used to denote both the set of nodes at distance i and the cardinality of this set. It is obvious that $\sum \mathcal{D}_i(t) = N \forall t$.

We will consider the path followed by a packet generated by node s and consumed

by node t. The number of hops for this packet is $L = l_{s,t}^{Gs}$ if no favourable shortcuts are encountered by any node in the path. However an intermediate node j (and most certainly all following nodes) can be removed from the route if a previous node encounters a suitable shortcut. In such a case, the packet is routed to a node u ($l_{u,t} < l_{j,t}$), which is not part of the path $\mathcal{P}_{s,t}$. However if the network is symmetrical it follows that $l_{u,t}^{Gs} = l_{v,t}^{Gs}$ for a suitable choice of $v \in \mathcal{P}_{s,t}$; furthermore $l_{u,t}^{Gs} = l_{w,t}^{Gs}$ for all $w \in \mathcal{D}_{l_{u,t}}(t)$, so the path $\mathcal{P}_{s,t}$ can be thought of as a succession of states, where state j is formed by all nodes $w \in \mathcal{D}_{l_{j,t}}(t)$ (the distance between successive stages in the path is still monotonically decreasing). Because of these reasons the principle of deferred decisions can be applied [136]. The probability that state j is skipped is given by the probability that a node at a stage $k > j$ has a shortcut ending in the set of nodes with a distance smaller than that of j, provided that node k itself is not skipped:

$$PS_{j,L} = \sum_{i=j+1}^{L} \left[\phi k \frac{\mathcal{X}_{j-1}}{N} (1 - PS_{i,L}) \right] \tag{3.37}$$

$$= \frac{\phi k}{N} \mathcal{X}_{j-1} \left[L - j - \sum_{i=j+1}^{L} PS_{i,L} \right] \tag{3.38}$$

where $\mathcal{X}_i = \sum_{j=0}^{i} \mathcal{D}_j$. The derivation of $\mathcal{D}_j$ is graph dependent and will not be attempted here. The solution of $PS_{j+1,L}/PS_{j,L}$ leads to a simpler equation which can be solved numerically by iteration to obtain $PS_{j,L}$:

$$PS_{j,L} = PS_{j+1,L} \left(\frac{\mathcal{X}_{j-1}}{\mathcal{X}_j} - \frac{\phi k}{N} \mathcal{X}_{j-1} \right) + \frac{\phi k}{N} \mathcal{X}_{j-1}, \tag{3.39}$$

with the initial condition $PS_{L,L} = 0$ (note that the second subscript of $PS_{j,L}$ can be dropped in the last two equations).

The expected number of nodes skipped in the whole path of length L is therefore $E(S_{j \le L}) = \sum_{j=1}^{L} PS_{j,L}$, and the expected average length for the routing algorithm is given by:

$$\overline{L_R} = \frac{1}{N-1} \sum_{h=1}^{D} (\mathcal{D}_h [h - E(S_{j \le h})]) \tag{3.40}$$

Similarly the expected routing diameter is given by:

$$D_R = D - E\left(S_{j \leq D}\right). \tag{3.41}$$

Since a general analytical solution for eq.3.37 does not exist, an approximate solution (in fact an upper bound) can be obtained as:

$$PS_{j,L} \leq \sum_{i=j+1}^{L} \phi k \frac{\mathcal{X}_{j-1}}{N}, \tag{3.42}$$

which gives:

$$ES_{j \leq L} \leq \frac{\phi k}{N} \sum_{j=1}^{L} [(L-j)\mathcal{X}_{j-1}] \tag{3.43}$$

$$\overline{L_R} \geq \sum_{h=1}^{D} \left(\mathcal{D}_h \left(h - \frac{\phi k}{N} \sum_{j=1}^{h} [(h-j)\mathcal{X}_{j-1}] \right) \right), \tag{3.44}$$

and:

$$D_R \geq D - \frac{\phi k}{N} \sum_{j=1}^{D} [(D-j)\mathcal{X}_{j-1}] \tag{3.45}$$

3.8. Stochastic Performance of RSW

3.8.1. Latency model

We are interested in investigating the impact of the rewiring process to the dynamic characteristics of the RSW graphs. In particular, it is expected to see a reduction in the average latency of a packet as the average length is reduced, and an increase in the maximum throughput of the graphs. A simple analytical model for the additive RSW graphs using the simple switch shown in figure 3.8 and under virtual cut-through switching has been developed to study these effects.

The model is based in the following assumptions (most of which have been used in similar studies [46,78,137,138]):

1. The rewiring probability ϕ is high enough that at the global scale the network can be approximated to a random network, and the network is not subject to a premature saturation as described in the next section (3.9).

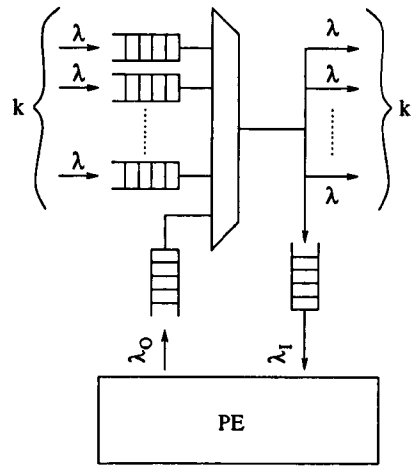


Figure 3.8.: Diagram of the switch used in the model and in the simulations. Each of the k inputs of rate λ as well as the local output from the processing element with rate λ_0 are buffered before entering the multiplexer. The output of the multiplexer can be directed to any of the k outputs (of rate λ), as well as to the local input queue.

2. Packets are distributed evenly through all edges belonging to a given class in the network.

3. All messages generated are of equal length of B phits.

4. Each node generates a message by a Poisson process of rate μ. The destination for each packet is chosen uniformly and independently amongst all possible recipients (uniform traffic).

5. Only one phit per unit time can be sinked by the Processing Element. Similarly, only one phit per unit time can be sourced (this is also known as the single acceptance model). All information interchange amongst nodes is done synchronously at each clock cycle.

6. Queues are of infinite size.

7. Messages are routed along the shortest path.

8. The network uses packet switching, and the switching element shown in figure 3.8.

9. Each edge is implemented by two physical wires (one in each direction).

10. The propagation delay in all wires is ignored, and the bandwidth of all edges is the same (regardless of their size).

An additive RSW has an average degree of $k = k_0(1+\phi)$, and $kN/2$ edges. However the network is constructed in such a way that each edge consists of two channels (one in each direction), so that there are on average k input channels, and k output channels for each node. The traffic rate at each of these channels is λ_i, $0 < i \leq k$. We will assume initially that the traffic in all of the channels is equal. It is important to note however, that in general there is no way to determine the distribution of the traffic in the edges (λ_i) without an exact description of the underlying network. We will refine the model afterwards by the introduction of two classes of edges, based on a small world model.

We proceed the analysis by noting that the queues are of infinite size, so there is no blocking in the network except that which occurs in the input multiplexer (see figure 3.8). As previously stated there are k inputs of rate $\lambda_i = \lambda$ to the multiplexer. The local output queue is also connected to the multiplexer and has a traffic rate $\lambda_O = \mu$ phits per network cycle (initially we will assume that unit sized packets are used throughout the network; we will extend the model to incorporate the effects on non unit packets afterwards). This is also the traffic sinked into the local processing element, λ_L (provided the network has reached a constant steady state). A typical packet injected into the network will traverse the multiplexer of the node where it was injected, and continue its path which goes through $\overline{L}$ other nodes (and multiplexers), before being consumed at the destination, so in total $\overline{L}+1$ nodes are traversed. The traffic coming out of a multiplexer will be ejected into the local processing element with probability $1/(\overline{L}+1)$, which implies that the traffic at any of the k external inputs is:

$$\lambda = \frac{\mu(\overline{L}+1)}{k}. \tag{3.46}$$

Following the work in [51], we will model the multiplexer as a G/D/1 queue with n inputs of rates λ_i, $0 < i \leq n$ which gives the mean waiting time as:

$$w = \frac{\mathcal{V}}{2\mathcal{E}(1-\mathcal{E})} - \frac{1}{2}, \tag{3.47}$$

where the expectation ($\mathcal{E}$) and variance ($\mathcal{V}$) of the arrival process is given by:

$$\begin{aligned} \mathcal{E} &= \sum \lambda_i, \\ \mathcal{V} &= \sum \lambda_i(1-\lambda_i). \end{aligned} \tag{3.48}$$

An additive RSW graph is not regular, but has a degree distribution which has been analysed in section 3.3, and which is given as:

$$P(k = k_0 + \Delta) = \left(\frac{1}{N}\right)^{\Delta} \left(\frac{N-1}{N}\right)^{\epsilon-\Delta} \binom{\epsilon}{\Delta}. \tag{3.49}$$

The degree of the graph is the maximum expected degree of a node,Δ_M (which can be determined from $P(k = k_0 + \Delta_M) = 1/N$). A node with degree $k = k_0 + \Delta$ has k input (and output) links, which we assume to be of rate λ, and an input from the local PE of rate μ, so the expectation and variance of the arrival process is given as:

$$\begin{aligned} \mathcal{E}(k) &= k\lambda + \mu \\ \mathcal{V}(k) &= k\lambda(1-\lambda) + \mu(1-\mu), \end{aligned} \tag{3.50}$$

and the waiting time at the multiplexer is simply:

$$w(k) = \frac{\mathcal{V}(k)}{2\mathcal{E}(k)(1-\mathcal{E}(k))} - \frac{1}{2}, \tag{3.51}$$

The multiplexer will saturate when the expected arrival rate $\mathcal{E}(k) \rightarrow 1$, so the set of nodes of degree $k_0 + \Delta_M$ will form a bottleneck and cause saturation when the message probability is:

$$\mu_{sat} = \frac{k_0(1+\phi)}{k_0(2+\overline{L}+\phi) + \Delta_M(\overline{L}+1)}. \tag{3.52}$$

The average waiting time at a multiplexer can be calculated from the distribution of node degrees as follows:

$$\overline{T_{mux}} = \sum_{\Delta=0}^{\Delta_M} \left(P(k = k_0 + \Delta)\ w(k = k_0 + \Delta)\right). \tag{3.53}$$

This simple model can be extended to non unit packets by scaling the waiting time in the multiplexer $w(k)$ and the message generation probability by B, the message length (in phits) [51]:

$$\begin{aligned} \mu' &= \mu B \\ w(k)' &= Bw(k). \end{aligned} \tag{3.54}$$

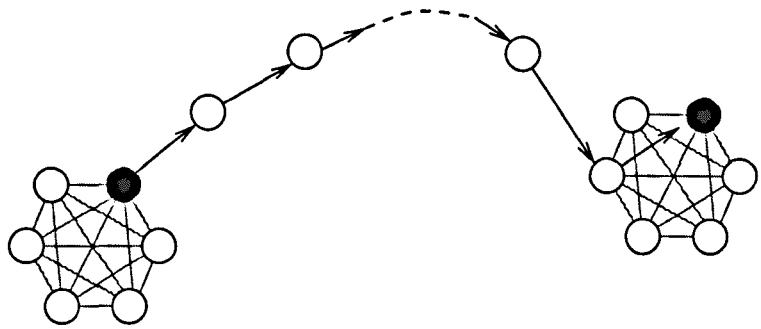


Figure 3.9.: Typical trajectory of a packet in the cave-world model. A typical packet will encounter $L - 1/C$ shortcuts in its path, and not more than one local edge.

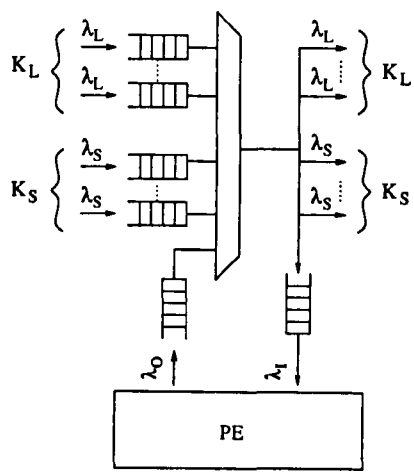


Figure 3.10.: Diagram of the network switch with non-uniform link utilisation. As it is shown there are k_L inputs of rate λ_L (which are the shortcut channels), as well as k_S inputs with rate λ_S, and the inputs and outputs of the local processor.

Since an average packet traverses $\overline{L}+1$ nodes (with an average delay of $\overline{T_{mux}}$ each) and the service time at the output queue where the message is ejected also causes a delay, the total latency is given as:

$$T = (\overline{L}+1)(T_{dec} + \overline{T_{mux}}) + B, \tag{3.55}$$

where T_{dec} is the decision time at each multiplexer.

3.8.1.1. Non-uniform Link Utilisation

The rewiring or addition of edges in a graph introduces a number of shortcuts which reduce the average length of the graph. If shortest routing is used, a large number

of packets will be attracted to these shortcuts causing an uneven link utilisation which can lead to saturation. A characterisation of the conditions which can lead into saturation is carried out in the next section. In the mean time we are interested in providing a simple extension to the latency model by taking the non-uniform link utilisation into account. We will do so based on the previous assumptions extended by the following:

1. The cave world model presented in [118] is an accurate representation for the graph (see figure 3.9).

2. A typical message will exit a node through a shortcut, and will continue to travel using only shortcuts until the cluster where the target resides is reached.

3. Once the cluster where the target node resides is reached the message will make one (and only one) additional jump (using normal edges, not shortcuts) if it still has not reached its destination.

4. Each node in the system has at least one shortcut connected to it.

5. There are only two classes of edges (shortcuts and local edges).

6. Packets are distributed evenly through all edges belonging to a given class in the network.

Note that assumptions 2 and 3 above, are a consequence of the first assumption which states that the cave world model is a correct representation of the graph.

We begin by noting that there are k_L inputs of rate λ_L (which are the shortcut channels), k_S inputs with rate λ_S, and the inputs and outputs of the local processor with rate $\lambda_I = \lambda_O = \mu$ into a multiplexer. The number of these edges in a node are given as:

$$\begin{aligned} k_L &= 1 + k_0\phi \\ k_S &= k_0 - 1. \end{aligned} \tag{3.56}$$

Note that due to assumption 4 there is at least one shortcut connected to the node when $\phi = 0$. This is necessary to guarantee that the cave-world model is connected, and to simplify the analysis. Furthermore note that the total node degree remains constant and equal to k_0.

An average packet will encounter $\overline{L}+1$ multiplexers and $\overline{L}$ edges in its path (the packet arrives at the first multiplexer via an internal link). Therefore according to the cave-world model depicted in figure 3.9, a packet will traverse $\overline{L}-1$ shortcuts. It follows that the probability that a packet will exit the switch through a shortcut is given as:

$$p_L = \frac{\overline{L}-1}{\overline{L}}, \tag{3.57}$$

and that the probability of a packet being switched to a local exit channel is:

$$p_S = \frac{1}{\overline{L}} - \frac{1}{\overline{L}+1}, \tag{3.58}$$

where the first term accounts for the probability of a packet not using a shortcut channel, and the second term is the probability of a packet having reached its final destination (and therefore not requiring an additional hop via a local channel). Finally the probability of a packet being ejected into the local processing element is simply:

$$p_O = \frac{1}{\overline{L}+1}. \tag{3.59}$$

Using these probabilities, one can determine the different traffic rates as follows:

$$\begin{aligned} \lambda_L &= p_L \frac{\lambda_T}{k_L} \\ \lambda_S &= P_S \frac{\lambda_T}{k_S} \\ \lambda_I &= P_O \lambda_T, \end{aligned} \tag{3.60}$$

where the total traffic at the multiplexer λ_T is the same as in the uniform link utilisation model (since it is a function of the total traffic in the network, which remains fixed). Alternatively it can be derived by noting that once equilibrium has been reached the injection and ejection rate need to be equal, $\lambda_O = \lambda_I = \mu$, which gives the total traffic as:

$$\begin{aligned} \lambda_T &= k\lambda \\ &= \mu(\overline{L}+1). \end{aligned} \tag{3.61}$$

We proceed as before, by noting that the degree distribution is given by eq. 3.49, and that the expected arrival rate and the expected variance in the arrival process for a node of degree $k_0 + \Delta$ (where the additional edges belong to the shortcut class) are given by:

$$\begin{aligned} \mathcal{E}(k_o + \Delta) &= (1 + \Delta)\lambda_L + (k_0 - 1)\lambda_S + \mu \\ \mathcal{V}(k_o + \Delta) &= (1 + \Delta)\lambda_L(1 - \lambda_L) + (k_0 - 1)\lambda_S(1 - \lambda_S) + \mu(1 - \mu). \end{aligned} \quad (3.62)$$

The average waiting time at a multiplexer can be calculated as follows:

$$\overline{T_{mux}} = \sum_{\Delta=0}^{\Delta_M} \left(P(k = k_0 + \Delta)\ w(k = k_0 + \Delta)\right). \quad (3.63)$$

As before, the largest degree (Δ_M) will cause saturation in the network when $\mathcal{E}(k_0 + \Delta_M) \rightarrow 1$, which gives the corresponding message probability as:

$$\mu_{sat} = \frac{\overline{L}(1 + k_o\phi)}{(\overline{L} + 1)\left(\Delta_M\left(\overline{L} - 1\right) + \overline{L} + k_0\phi\right)}. \quad (3.64)$$

The model can be extended to non unit packets by scaling the waiting time in the multiplexer $w(k)$ and the message generation probability by the message length (as before), giving the total latency as:

$$T = (\overline{L} + 1)(T_{dec} + \overline{T_{mux}}) + B. \quad (3.65)$$

3.8.1.2. Model Validation

A discrete event simulator has been created in order to validate the model. The simulator has been integrated into the INADS system described in appendix B. The simulator has been written in around 2000 lines of C++, and is compatible with the assumptions used to derive the model. In particular the following assumptions and specifications have been met:

- The switching method is virtual cut through. The switch can be either the multiplexer based switch shown in figure 3.8 or a non-blocking crossbar switch, which provides better performance than the multiplexer design.
- Messages are generated by an independent Poisson process. The destination for each packet is selected randomly from all of the possible destinations.

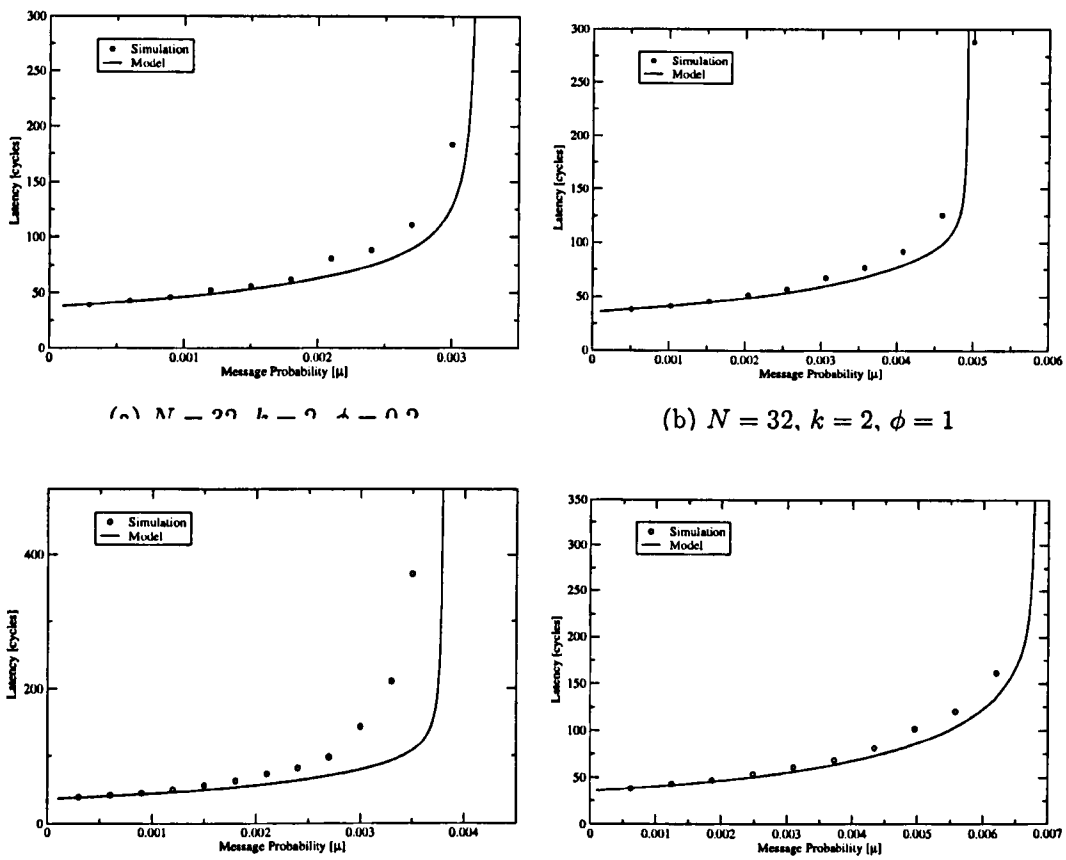
- All messages generated are of a fixed length.
- Messages are delayed by a decision time at each switch they visit. The time can be specified by the user.
- Routing is restricted and follows the shortest path between source and destination.
- Simulations are run until the collected statistics converge to a final value with a given accuracy.
- All queues are of infinite size.

To validate this model, simulations were carried out, and the results show a good fit of the data at low transmission probabilities (fig. 3.11). However near saturation this is not true. We believe this discrepancy arises from the approximations used to determine the characteristic length of the graphs. The model is very sensitive to changes in the characteristic length, especially close to the saturation region.

3.8.2. Discussion

It is interesting to investigate the effects of varying the rewiring parameter ϕ to the latency of the graphs. We will do so using the analytical model derived in section 3.8.1.1. This is shown in figure 3.12, where the average latency of a 128 node ring is shown as a function of the rewiring probability (ϕ). As it is apparent when the message probability is low enough ($\mu < 0.004$ in the graph) there is a reduction of latency as ϕ is increased. For the cases where the message probability is larger, the model predicts a saturation for small values of the rewiring probability, while for larger values the latency is reduced, as can be seen in the plot. This is due to the fact that when the rewiring probability is low, only a very small number of shortcuts are present, which can cause saturation if enough traffic is injected; however when ϕ is increased and a much larger number of shortcuts appear, latency becomes lower. This is experimentally confirmed by the plot of figure 3.13, where experimental measurements for the latency and average length of the same graph have been obtained, and a similar phenomenon can be observed.

These results have been derived for networks using the multiplexer switch shown in figures 3.8 and 3.10, which suffers from a poor performance when compared with other alternatives, such as a non-blocking switch; so it is natural to investigate if the same behaviour is observed when using such a switch. These work has been



(a) $N = 32$, $k = 2$, $\phi = 0.2$

(b) $N = 32$, $k = 2$, $\phi = 1$

(c) $N = 128$, $k = 6$, $\phi = 0.2$

(d) $N = 128$, $k = 6$, $\phi = 1$

Figure 3.11.: Latency in ring RSW networks. The figure shows experimental values obtained from the simulator against the non-uniform link utilisation model from equation 3.65. In all graphs $T_{dec} = 1$ and $B = 32$.

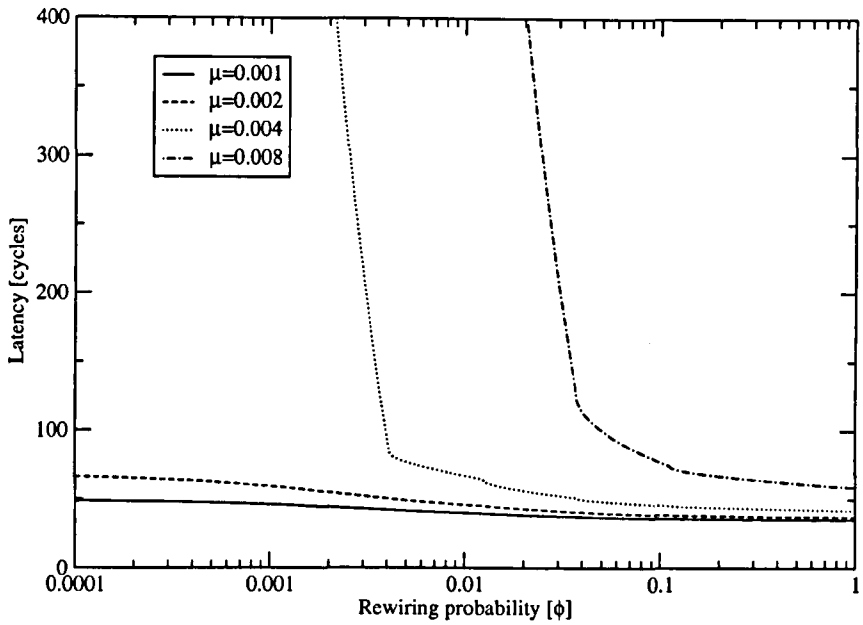


Figure 3.12.: Analytical Small-World latency plot of a 128 node ring (k=6, $T_{dec} = 1$, B=32). The figure shows the effect of varying the rewiring probability to the average latency of the graph using the analytical expressions for average latency (eq. 3.65), and the average length from the Newmann and Watts model (eq. 2.10).

carried out by using the discrete event simulator, and results are shown in figure 3.14. As shown in the figure, similar results are observed. When the communication probability μ is small, latency is mostly determined by the characteristic length. This means that latency is reduced by the rewiring process of the RSW network. However when traffic intensity rises, saturation starts to appear near the centre of the graph, and become larger as more traffic is present in the network.

As the simulations and models show, these "premature" saturations are generated when the graphs have only a few number of shortcuts (probably just one); when the number of shortcuts rises, there is a steep decrease in latency. By choosing a network outside of the saturation region, it is possible to have a network with lower latency than the original network; and as figure 3.14 shows, it is even possible to have a network with lower latency and a lower saturation threshold than the original network. However it is also desirable to choose a network generated with a small rewiring probability, as less shortcuts are needed and most of the original network structure is maintained.

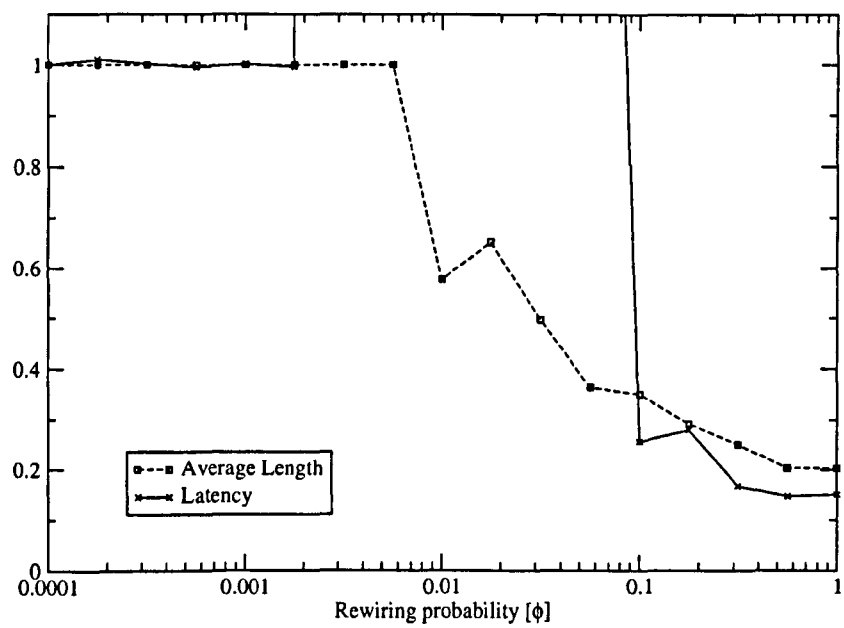
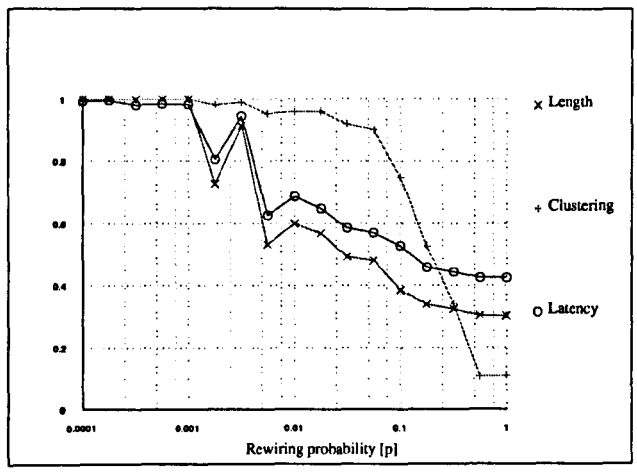
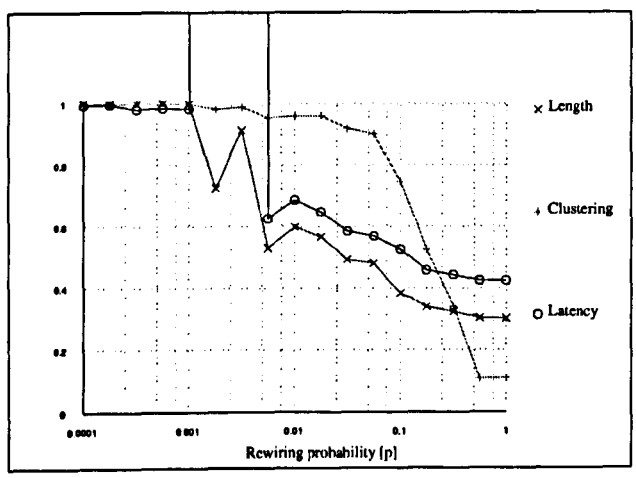


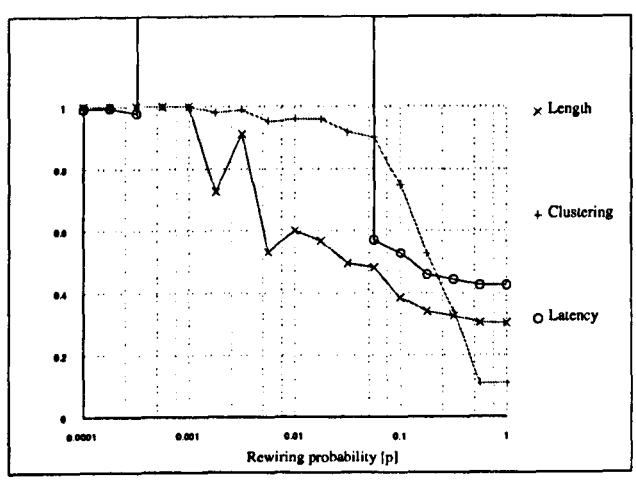
Figure 3.13.: Experimental Small-World latency plot for a 128 node ring (where k=6, $T_{dec} = 1$, B=32 and μ=0.008) using the multiplexer switch shown in figure 3.10. The figure shows experimental values for the Average length ($\overline{L}$) and for the latency of the graph, while varying the rewiring probability (ϕ). Note that all values have been normalised to unity in the origin.



(a) mu=0.02



(b) mu=0.04



(c) mu=0.06

Figure 3.14.: Small-World latency plot of a ring (N=128, T_{dec} = 1, k=10) using a non-blocking switch. Latency is reduced, and follows the characteristic length closely. Note the premature saturation occurring at m=0.04 and m=0.06; however at $\phi = p = 0.01$ latency is almost halved for all cases.

3.9. Congestion

In the previous section the problem of premature saturation, where traffic flowing through a very small number of shortcuts causes saturation, was identified. The determination of the exact conditions which lead into this problem are quantified in this section. With the aid of the results provided in this section it is possible to identify the critical point where premature saturation will occur, and design an interconnection network to avoid this problem.

The congestion of a graph is determined by the routing algorithm, the communication pattern and the interconnection network itself. An oblivious routing communication algorithm assigns a unique predetermined path for packets with a given source and destination. If the path $P_{S\to T}$ is the path followed by packets injected at S and consumed at T, then the routing graph for the source node S is simply the directed graph $R_S = \{P_{S\to T\in G}\}$. Note that R_S is in general not simple, since multiple edges are allowed. If the routing algorithm is minimal then the routing graph can be represented by the weighed spanning tree T_S where the weight of an edge is simply its multiplicity in R_S (there are no loops, since a loop would imply a packet is misrouted). The congestion of an edge is $K_e = |\{e = f,\ f \in E(R_{S\in G})\}|$, and the congestion of a graph G is the maximum congestion of any of its edges:

$$K_G = \max\{K_{e\in G}\}. \tag{3.66}$$

We are interested in providing a mathematical model of congestion for RSW graphs in order to fully understand these effects. We will do so based on the branching model proposed by Moukarzel as follows.

Define the volume $V_n(t)$ to be the number of nodes with n shortcuts in their shortest paths from the origin. The initial sphere's volume is $V_0(t)$ because paths to nodes in it do not traverse any shortcuts. The volume $V_n(t)$ is the sum of all spheres stemming from $V_{n-1}(t' < t)$ (see fig. 3.15). Suppose that we follow the development of the set of spheres forming V_n. For all $t < n$, $V_n = 0$ since any shortest path for a node in V_n needs to contain at least n shortcuts. The sphere starts to grow at time $t = n$, and every time a shortcut is reached, a new sphere is formed at a random location, provided the aforementioned location is not already infected (which happens with probability $\sum_{i\leq n} V_i(t)/N$). This probability is ignored in Moukarzel's model, and will also be ignored here in order to simplify the calculation. Therefore the volume can be expressed as:

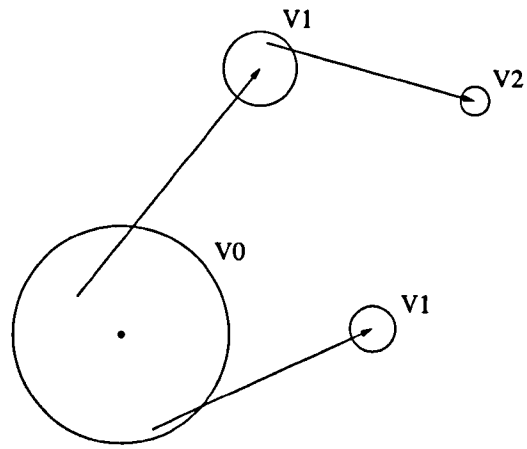


Figure 3.15.: Congestion model for a RSW graph. Suppose a disease starts spreading from the centre of the large sphere, as discussed previously (fig. 3.6). The volume $V_0(t)$ is simply the volume of the large sphere (V0), since shortest paths to nodes in V0 do not traverse any shortcuts. The volume $V_1(t)$ is formed by the spheres labelled V1 in the figure, since shortest paths between nodes in these spheres and the source node traverse one shortcut.

$$\begin{aligned} V_n(t) &= \Gamma_d \rho \int_n^t (t-\tau)^{d-1} V_{n-1}(\tau)\, d\tau \\ &\approx \Gamma_d \rho \int_0^t (t-\tau)^{d-1} V_{n-1}(\tau)\, d\tau, \end{aligned} \tag{3.67}$$

where the second expression will tend to over-estimate the volume. Noting that $V_0(t) = \Gamma_d t^d/d$ and performing the integral it follows that:

$$V_n(t) \approx \Gamma_d^{n+1} \rho^d \frac{(d-1)!^{n+1}}{((n+1)\,d)!}\, t^{(n+1)d}. \tag{3.68}$$

It is apparent that $V_n \to 0$ for large n and small ρ. To make this point more precise, note that for $t = t_{sat}$ the ratio V_{n+1}/V_n is given by:

$$\frac{V_{n+1}(t_{sat})}{V_n(t_{sat})} = \frac{((n+1)\,d)!}{((n+2)\,d)!} \ln (\rho \Gamma_d N)^d. \tag{3.69}$$

The traffic injected into the shortcut network is simply:

$$\mathcal{T}_s = \sum_{n=1}^{D \approx t_{sat}} n\, V_n(t_{sat}), \tag{3.70}$$

which is completely determined by the distribution of V_n, and the diameter of the network, t_{sat}. There are two problems for evaluating eq. 3.70. The first one is that the expressions for V_n and t_{sat} diverge at the critical point $\Gamma_d \rho = 1/N$. Congestion is likely to be at its largest near this point, since there are a very small number of shortcuts present in the network, which attract a large number of routing paths. The second reason against the direct evaluation of eqs. 3.68 and 3.70 is that the expression for V_n overestimates the volume as previously discussed such that $\sum V_n(t_{sat}) > N$. This divergence increases with increasing ρ, since contention between all growing spheres (which is ignored in the model) becomes larger.

These points will be illustrated more clearly for the one dimensional case. The volumes are given by:

$$V_n(t) = 4k\phi \int_n^t V_{n-1}(\tau)\, d\tau, \tag{3.71}$$

with $V_0(t) = 2kt$, which has the solution:

$$V_n(t) = 2k\,(4k\phi)^n \frac{(t+1)^n (t-n)}{(n+1)!}. \tag{3.72}$$

The ratio $V_{n+1}(t)/V_n(t)$ is an important characteristic of the system and is given by:

$$\begin{aligned} \frac{V_{n+1}(t)}{V_n(t)} &= \frac{4k\phi(t+1)(t-n-1)}{(t-n)(n+2)} \\ &\sim 4k\phi t/(n+2), \end{aligned} \tag{3.73}$$

where the last equation is the asymptotic limit for $t \gg n > 1$. Equation 3.73 implies that there is a critical point for any given n where $V_{n+1}(t) > V_n(t)$. For the final volumes $V_n(t = t_{sat})$ the critical point for each n is $4k\phi t_{sat}/(n+2) = 1$, and the fundamental critical point is $2k\phi t_{sat} = 1$, since at this point the volume $V_0 = V_1$. For values larger than this critical point there is an inversion of the distribution, such that $V_0 < V_1 < ... < V_{n'}$ for a given n'; in other words there is a transition from a large world network into a small world network at this point. The saturation time at this point is given as $t'_{sat} = 1/(2k\phi')$, where ϕ' is a function of N and k.

The total number of nodes whose shortest path to the destination involves n shortcuts is simply $V_n(t_{sat})$, where t_{sat} is the diameter of the network determined for instance in eq. 3.23. It is possible now to apply eq. 3.70 to obtain a rough approximation of the traffic through the shortcuts, subject to the divergences dis-

cussed when deriving equation 3.70. The direct evaluation of the sum is difficult, but by noting that the largest congestion is caused when a relatively small number of shortcuts are present in the network (i.e. $k\phi \approx 1/N$), and that at this point of operation $V_{n\gg 1} \rightarrow 0$ according to eq. 3.73, it follows that it is possible to approximate the traffic by taking the m first few terms of 3.70 (in fact the use of only the first term provides an acceptable solution, as will be shown later). Here we will try to accelerate the convergence of the approximation by accounting for the nodes contained in the rest of the series, by noting that:

$$\sum_{n=m+1}^{t_{sat}} V_n(t_{sat}) = N - \sum_{n=0}^{m} V_n(t_{sat}), \tag{3.74}$$

so the traffic caused by the source node can be approximated as:

$$\mathcal{T}_s^m = \sum_{n=0}^{m} n\, V_n(t_{sat}) + (m+1)\left(N - \sum_{n=0}^{m} V_n(t_{sat})\right). \tag{3.75}$$

The congestion can be estimated by considering that there are a total of N nodes in the network, and that each injects $\mathcal{T}_s$ paths into the shortcuts network, which consists of $k\phi N$ edges. The average congestion per link is therefore given as:

$$\mathcal{K}_m = \frac{\mathcal{T}_s^m}{k\phi}. \tag{3.76}$$

Due to the errors introduced in the derivation of the volumes V_n and in the saturation time, and in particular since no contention between growing spheres is included in the model, in general $\sum_{n=0}^{m} V_n(t_{sat}) > N$ and the congestion becomes negative.

Quite remarkably, a better approximation can be constructed by only considering the term V_0 to calculate the congestion. This is because the volume $V_0(t)$ is determined by an exact expression and there are no errors in its calculation. To obtain the traffic, however, $V_0(t_{sat})$ has to be evaluated, and any divergences in the approximation of t_{sat} will be incorporated into the traffic estimate.

$$\begin{aligned} \mathcal{T}_s^0 &= (N - V_0) \\ &= N - \frac{\rho\Gamma_d t_{sat}^d}{d}, \end{aligned} \tag{3.77}$$

which gives the congestion as:

$$\mathcal{K} = \frac{N - \frac{\rho \Gamma_d t_{sat}^d}{d}}{k\phi}. \tag{3.78}$$

The resulting model is shown in figure 3.16 , where different expressions for the saturation time are used. As it is apparent, the use of the diameter obtained in the mean-field approximation provides a well behaved model, although it does not provide a good approximation for small values of ϕ, since the model diverges at these points. The use of the saturation time in Moukarzel's model provides a better estimate for the critical region; however it fails to provide a good approximation for very small values of ϕ. This, however, is of no practical importance, as a designer would be interested in evaluating the point where the congestion is lowered to a given amount (such as the point of half congestion shown with a small red vertical line in the figure).

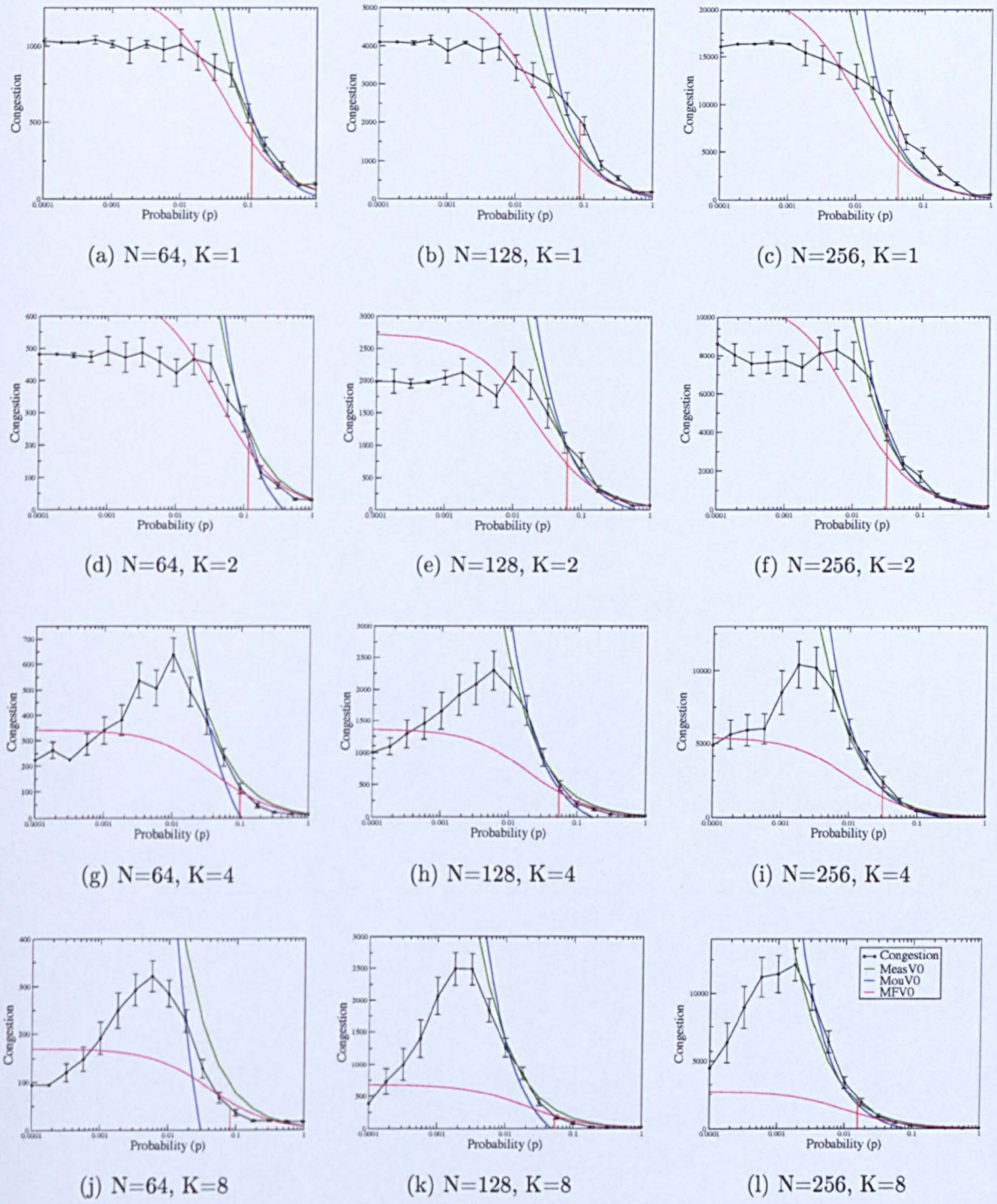
The models provide a good fit to the actual values near this point (in particular when using the saturation time derived in Moukarzel's model), although some deviations are observed in small systems with large degrees (subgraphs g, j and k). It is also shown that the best approximations are obtained when the actual diameter is used (as measured from the respective graph) as expected; although it is not necessary to resort to experimental data as purely analytical models provide adequate results.

3.10. Conclusions and Summary of Results

A new family of interconnection networks has been presented and analysed. The main advantage of these networks is that they posses a very short diameter and average length, while maintaining most of the structure of the original network (and therefore providing a deterministic and well organised local network) where locality of information can be exploited. These characteristics make them very suitable for use as the interconnection network of a multicomputer. Analytical expressions for some of the RSW graph characteristics have been derived, including diameter, degree distribution, switch complexity, congestion, bisection width, latency and routing.

It has been shown that the RSW networks are expandable, as for $\phi > 0$, their bisection width grows as $O(\phi N) = O(N)$, which is the same as the expected increase in traffic. However there is a constant penalty to be paid and to small values of ϕ will correspond small values of maximum throughput.

Through theoretical work and simulations, it has been determined that latency can be reduced by the use of RSW networks under certain conditions. This is accom-



(a) N=64, K=1 (b) N=128, K=1 (c) N=256, K=1

(d) N=64, K=2 (e) N=128, K=2 (f) N=256, K=2

(g) N=64, K=4 (h) N=128, K=4 (i) N=256, K=4

(j) N=64, K=8 (k) N=128, K=8 (l) N=256, K=8

Figure 3.16.: Congestion of RSW rings. Congestion in a family of rings $R_{N,K}$ is shown. The graph shows the actual congestion, against predicted values from eq.3.78. Different models for the saturation time have been used; including the actual diameter as measured from the graphs (MeasV0), the diameter obtained in the Mean-Field model using eq.3.20 (MFV0), and also the diameter derived by Moukarzel and shown in eq.3.23 (MouV0). The point where the measured traffic is half of the traffic in the underlying network is highlighted by a red vertical line.

plished by introducing a number of shortcuts via random rewiring on the underlying network. When the number of shortcuts is lower than a critical limit, congestion will occur on them as a large number of packets are diverted towards them. However, as the number of shortcuts is increased the traffic is distributed more evenly and since the average number of hops is reduced, so is latency. This crossover point has been identified, and analytical expressions have been provided. Simulations using random traffic have confirmed that premature saturation can occur in a RSW network when the rewiring probability is smaller than this critical value. Although it is always possible to provide a large enough number of shortcuts to ensure premature saturation does not occur, it is important to keep in mind that in the case of light or spare traffic, latency is reduced even with a very small number of shortcuts. Premature saturation is a problem if a moderate to large traffic with a general distribution is expected. On the other hand for communication patterns with a high degree of locality a very small number of shortcuts can prove sufficient to support the small inter-cluster traffic, even under large traffic conditions.

The routing algorithm used in the simulations is based on the shortest path between source and destination. Since randomness is introduced to the network, it is convenient to use a look-up table (LUT) to assign routes to packets. To simplify the router design, it is possible to use a pseudo-random number generator or other deterministic mechanism to randomise the network. In this way the rewiring is done in a deterministic way, and it is not necessary to use a LUT. Such a system is described in Section 4.2.

The main drawback of RSW networks, is their lack of regularity and an increased degree. In order to create regular networks, the rewiring process can be modified, as described in the next chapter. If an adequate rewiring process is used the main results derived in this chapter should still be applicable. It is also possible to generate regular graphs by modifying the rewiring process to do so, and indeed in section 6.3 this approach is used to construct regular clusters.

An analysis of the fault tolerance of the networks has not been carried out in this work. It is expected that the introduction of random shortcuts will significantly enhance the fault tolerance of the underlying graph; and that this enhancement will mimic the behaviour observed for other small-world characteristics, such as the diameter and average length, where the introduction of a very small number of shortcuts significantly modifies the characteristic, while gains beyond a certain point only provide modest improvements.

4. Deterministic Constructions by Superposition

4.1. Introduction

According to the multiple scales hypothesis in [10], randomness is not a fundamental requirement for the construction of small world networks, and it is possible to recreate the phenomena with a deterministic process. The construction of deterministic small world graphs provide substantial benefits over random implementations as regular graphs can be created, efficient routing algorithms can be designed, and the network becomes conceptually simpler.

Random regular graphs are attractive because for a given degree and order they provide diameters very close to the theoretical minimum [110,112]. However random graphs do not make very good interconnection networks for a number of reasons which have already been discussed i.e. they do not provide a way to exploit locality, they do not possess useful algorithmic properties and the memory required to store a description of the network grows as $O(N^2)$.

Deterministic small-world communication networks have already been proposed by Comellas et.al. [139] who proposed a two step construction mechanism. On the first step some nodes of the original graph are connected according to an additional deterministic graph, and in the second step edges are reconnected according to certain rules to recreate the regularity. In [140] Comellas et. al. propose a different approach in which nodes of the underlying graph are replaced by completely connected graphs, which are in essence product graphs of node symmetric networks with complete graphs.

In this chapter a new class of interconnection networks inspired by the small-world phenomenon, chordal graphs and the multiple scale hypothesis is introduced. Unlike [139, 140] the graphs are created by a direct one step process and are not product graphs, but still have a fixed degree and a low diameter, and can be scaled to large system sizes. The graphs are formed by the additive superposition of two

graphs ($G_S + G_L$). The underlying graph G_S provides a strong local clustering (which aids in exploiting the locality of information), while the long range graph G_L provide an efficient inter-cluster communication medium. G_L is constructed with the aid of one (or more) functions or algorithms $\mathcal{A}$, where node u is connected to node $\mathcal{A}(u)$. Note that using this definition there is one and only one shortcut end assigned to each node. Since the critical crossover point where the network changes from a large to a small world occurs near the region where there is only one shortcut in the network [8], the creation of N shortcuts should guarantee a small diameter. However it is important to keep in mind that enough shortcuts need to be introduced not only to reduce the diameter of the network, but also to avoid congestion, as described by eq. 3.76. In general the traffic increases as a function of the network size which is $O(N)$, and since the available bandwidth also grows as $O(N)$ congestion should be low (assuming traffic is distributed evenly through the network).

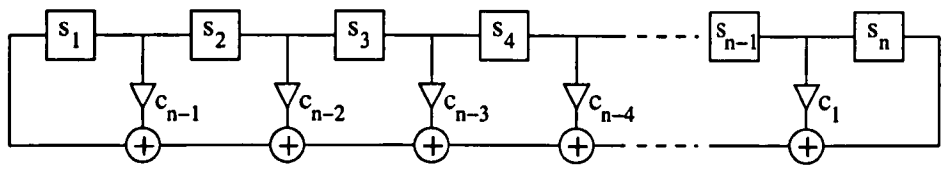
For $\mathcal{A}$ we are interested in functions with the following characteristics:

1. Multiple scales need to be generated
2. All networks constructed should be k_L regular
3. Only undirected graphs are considered, which implies that for any two nodes u, v the function needs to be symmetrical: $\mathcal{A}(v) = u \rightarrow \mathcal{A}(u) = v$.
4. $\mathcal{A}$ should define a closed group: $\forall v \in G$, $\mathcal{A}(v) \in G$, and because of (2), $\mathcal{A}$ needs to be bijective and cover the whole set.

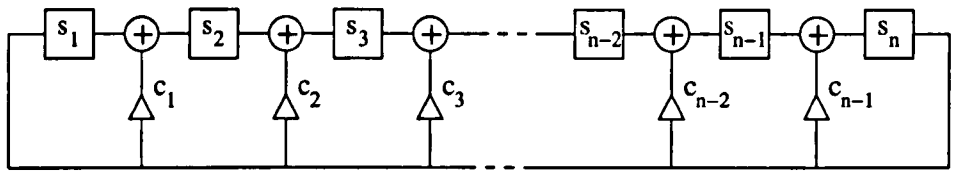
With the aid of $\mathcal{A}$ and G_S a very large family of deterministic small-world graphs can be defined, $\mathcal{G}_D(G_S, \mathcal{A})$, where the edge set is given as $E = E_S + \{(v, \mathcal{A}(v)) : v \in V_S\}$, and E_S and V_S are the edge and node set of G_S respectively.

A large amount of suitable functions can be constructed, and although it is impossible to analyse all of them, it is expected that graphs generated with any such function will have similar characteristics as that of a random small world interconnection network, with the additional benefit of being deterministic and regular. In the remainder of this chapter a suitable function is identified and the graphs generated are analysed, and shown to have all of the desirable properties of a random small-world network, while providing the additional benefits of a deterministic graph.

Finally it is important to note that since the graphs will be designed to incorporate a large number of scale lengths, long wires will be present in the system. The multiple length scale hypothesis implies that a collection of wires of all lengths are required



(a) Fibonacci



(b) Galois

Figure 4.1.: The figure shows the Fibonacci and Galois implementation of a linear feedback shift register (LFSR). The two implementations are equivalent and will produce the same output sequence, although at a different phase.

to reduce the average length of a graph; unfortunately these have the inconvenient of requiring more power to drive and introduce larger delays than short wires (which for simplicity have been left out of the models derived in this work).

4.2. Linear Feedback Shift Register Graphs

Functions based on linear feedback shift registers (LFSR) are interesting candidates for $\mathcal{A}$, as they are commonly used to produce pseudo random numbers. An n-bit LFSR is composed of an n-bit shift register and a feedback function. In the Fibonacci implementation the feedback function is the bit sum of a subset of the elements of the shift register. At each clock cycle the contents of the shift register is shifted one position, and the empty space is filled with the value produced by the feedback function. An alternative implementation is the Galois LFSR, in which at each element of a subset of elements of the shift register, and additional sum is carried out with the output of the shift register, as it is shown in figure 4.1.

Both implementations are equivalent, and will produce the same output sequence (at a different phase however). Formally these systems can be described by a state vector $S = (s_1, s_2, \ldots, s_n)$ and a transition matrix T, such that $S_{t+1} = S_t \cdot T$

(all operations are carried out over the Galois Field GF(2)). Associated with the transition matrix there is a generator polynomial $G(x) = \sum_{i=0}^{n} c_i x^i$ with $c_i \in G(2)$. The coefficients of this polynomial can be evaluated directly from the subset of elements in the feedback function (i.e. coefficient c_i is one if there is a "tap" at element i in the Galois LFSR, or at element $n - i$ in the Fibonacci LFSR; and $c_0 = c_n = 1$).

The output of a LFSR is a linear recursive sequence of length $l < 2^n$, which depends on both, the characteristic polynomial $\mathcal{P}$ and the initial state S^0. When the polynomial is *primitive* meaning it cannot be factored (i.e it is prime), and it evenly divides x^m+1 where $m = 2^n+1$ the LFSR produces maximal length sequences (sequences with $l = 2^n - 1$) [141,142]. (This is because consecutive products of any of its roots generate all the $2^n - 1$ non zero elements of $G(2^n)$).

The complexity of a binary sequence is defined by the amount of the sequence required to define the remainders; therefore a maximum length sequence from an n-stage LFSR has minimum complexity, however such sequences appear to be random [142]. For this reason we will restrict ourselves to maximum length sequences; since it is important to recognise that any given sequence can be produced by an appropriate linear generator [143–145].

4.2.1. Construction and Definition

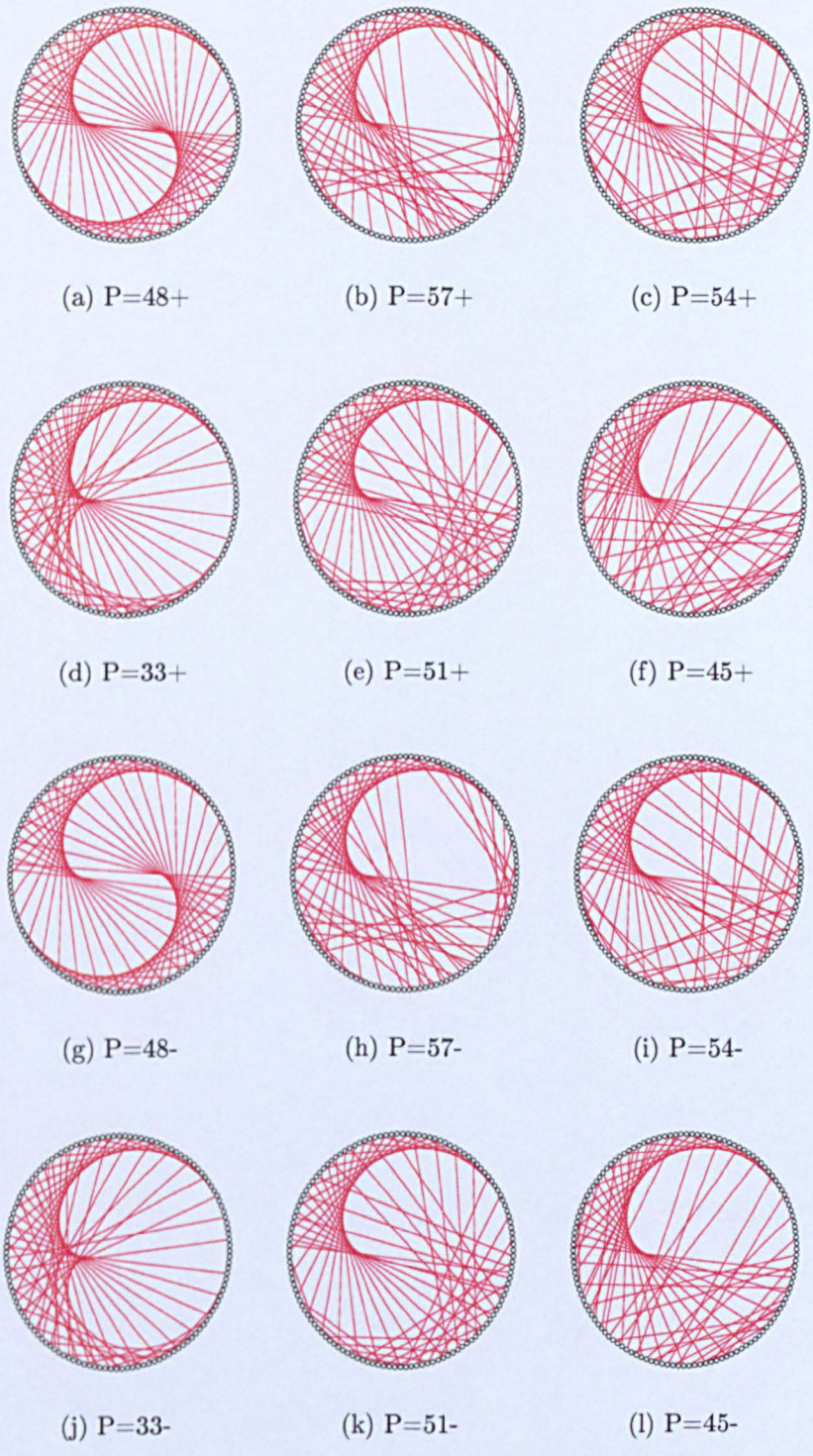
An LFSR graph $\mathcal{L}(\mathcal{G}, \mathcal{P})$ is formed by the additive superposition of the underlying labelled graph $\mathcal{G}$ (which is usually a regular, symmetric and highly clustered graph with a large diameter) with the state transition graph of the maximal sequence S defined by the LFSR with primitive characteristic polynomial $\mathcal{P} = x^m + \alpha_{m-1}x^{m-1} + \ldots + \alpha_1 x + 1$ where $2^{m+1} = |\mathcal{G}|$. The labels for the nodes of $\mathcal{G}$ will be denoted as $V_n = \{0 \ldots |\mathcal{G}|\}$. Two LFSR graph implementation are possible:

$$\mathcal{L}^+(\mathcal{G}, \mathcal{P}) = \mathcal{G} + \{e_{2S_i, 2S_{i+1}+1},\ 0 \leq i < 2^m - 1,\ S_{2^m} \equiv S_0\} + e_{0,1}. \qquad (4.1)$$

$$\mathcal{L}^-(\mathcal{G}, \mathcal{P}) = \mathcal{G} + \{e_{2S_i+1, 2S_{i+1}},\ 0 \leq i < 2^m - 1,\ S_{2^m} \equiv S_0\} + e_{0,1}. \qquad (4.2)$$

Note that both implementations are very similar, since the endpoints of the additional edges differ in one unit. Furthermore note that the starting vector S^0 is of limited importance, as its effect is only to shift the sequence. A complete family of LFSR graphs for $\mathcal{G} = \mathcal{R}_{2,128}$ is shown in figure 4.2.

The degree of $\mathcal{L}$ is $\Delta_{\mathcal{L}} = \Delta_{\mathcal{G}} + 1$, a direct consequence of the maximality of S



(a) P=48+ (b) P=57+ (c) P=54+

(d) P=33+ (e) P=51+ (f) P=45+

(g) P=48- (h) P=57- (i) P=54-

(j) P=33- (k) P=51- (l) P=45-

Figure 4.2.: The family of LFSR graphs with $\mathcal{G} = \mathcal{R}_{2,128}$.

(which of course is a result of the primality of $\mathcal{P}$), and if $\mathcal{G}$ is regular, $\mathcal{G}$ will also be regular with degree $\Delta_{\mathcal{L}}$. Note however that it is possible (and very probable) for $\mathcal{L}$ to have repeated edges which can be removed if desired (however then the graph will no longer be regular).

The diameter of $\mathcal{L}$ has the trivial lower Moore bound $D > \log_k N$. An upper bound can be constructed by noting that no regular connected graph can have a diameter lower than that of a k-ring of the same size. Using the upper bound derived for ring graphs in section 4.3 (eq. 4.7) it is obvious that $D < 4(\log_2 N - \log_2 2k - 3)$ for large enough N (this expression is congruent to the one given in [67]). As shown in figure 4.5 this is not a tight bound and it should be possible to provide a more accurate one.

4.3. LFSR Ring

A suitable choice for the underlying graph $\mathcal{G}$ is the ring $\mathcal{R}_{k,2^m}$, since it provides a strong locally connected graph, but it suffers from poor global connectivity. A family of such graphs has already been shown in figure 4.2. We will proceed to analyse such graphs; we will focus on the properties of the deterministic subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$, from which several properties can be extracted, and a routing algorithm can be designed.

4.3.1. The Subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$

It is obvious by looking at 4.2 that the graphs generated do not appear random. The most apparent deterministic feature is an anticlockwise spiral present in all graphs. This effect has a very simple explanation; when the most significant bit of the shift register is zero, the effect of the LFSR is simply to multiply by two; so for all elements $S_f = \{S_i < 2^m/2\}$ the next element is determined as $S_{f+1} = 2S_f$ and we have that:

$$\begin{aligned} \forall\, 0 < f < 2^m/2, \quad & (2f,\, 4f+1) \in E(\mathcal{L}^+) \\ & (2f+1,\, 4f) \in E(\mathcal{L}^-) \end{aligned} \tag{4.3}$$

Several properties can be derived from this observation. We start by defining the subgraph $\mathcal{L}(\mathcal{R}_{1,2^m}) = \mathcal{R}_{1,2^m} + \{(2f,\, 4f+1),\, 0 < f < 2^m/2\}$, which is formed by the ring and the edges defined in eq. (4.3). $\mathcal{L}(\mathcal{R}_{1,256})$ is shown in fig. 4.3.

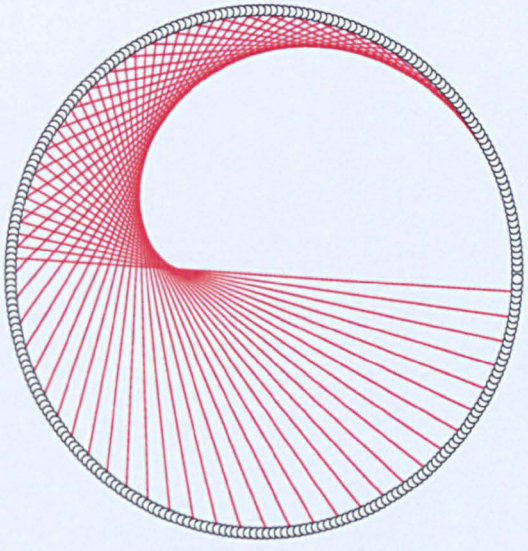
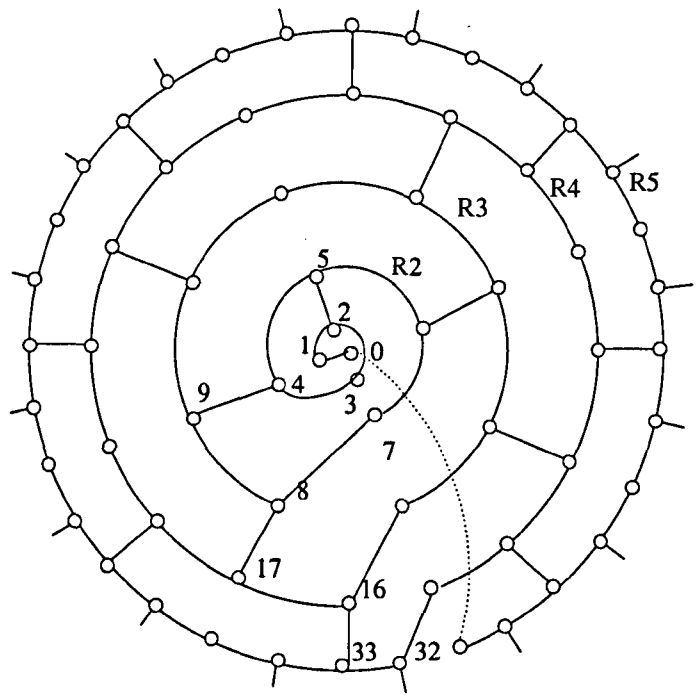


Figure 4.3.: The graph $\mathcal{L}^{+}(\mathcal{R}_{1,256})$.

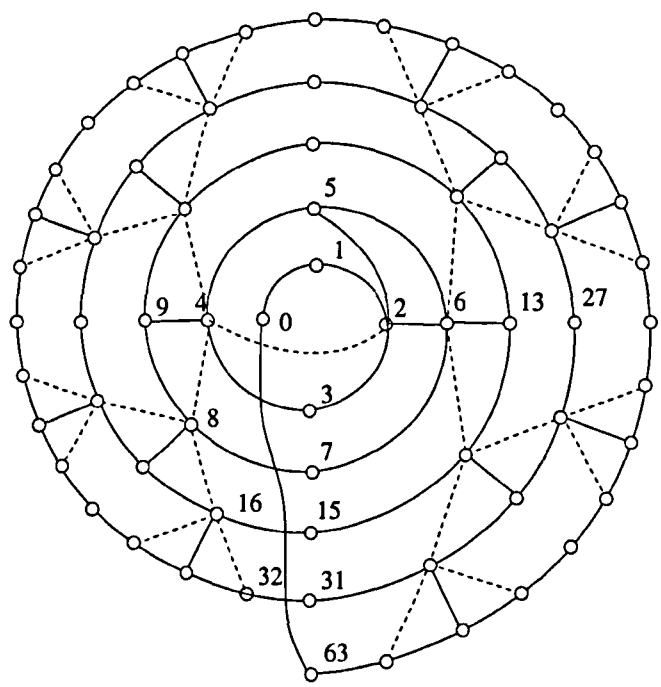
Now define the set of integer numbers $R_i = \{x : 2^i \leq x < 2^{i+1}, x \in \mathbb{Z}\}$. For the case of $\mathcal{L}^{+}$ we define the sequence $Q_i(t) = 2^i, 2^i + 2, 2^i + 4, \ldots, 2^{i+1} - 2$, with $1 < i + 1 < m - 1$. Note that these are the 2^{i-1} even numbers of R_i. For each of these numbers there exists an edge according to (4.3), whose endpoints are given by the sequence $Q_i'(t) = 2Q_i(t) + 1 = 2^{i+1} + 1, 2^{i+1} + 5, \ldots, 2^{i+2} - 3$. Note that $2^i \leq Q_i < 2^{i+1}$, and also that $2^{i+1} \leq Q_i' < 2^{i+2}$. This implies that each of these edges has an endpoint in R_i and the other in R_{i+1}. Furthermore, note that the elements of Q_x' are all odd and that $Q_x'(t+1) - Q_x'(t) = 4$, for all $2 \leq x < m$. With these observations, the isomorphisms shown in fig. 4.4 can be constructed. Note also that if neighbour nodes $a = (a_n, a_{n-1}, \ldots, a_1, 0)$ and $b = (b_n, b_{n-1}, \ldots, b_1, 1)$ are contracted into a single vertex, a spanning binary tree can be constructed (with a small appendix to the root node, node 2), as shown in the same figure.

Diameter

The diameter of the graph is obtained recursively with the aid of the spanning tree shown in the figure. First we define an associated function D_i', $2 \leq i < m$ to be the maximum distance between all pairs of nodes in $Q_i \subset R_i$ (these are the nodes that are connected to R_{i+1}). Since these are even, and all elements of Q_{i-1}' (which are the endpoint of edges from R_{i-1} to R_i) are odd (and separated by four edges), there is always a path of length not exceeding 2 between a node in Q_i and some node in



(a)



(b)

Figure 4.4.: Isomorphisms of $\mathcal{L}^+(\mathcal{R}_{1,64})$. (a) Shows the construction of the sets R_n. Note that the diameter of the subgraphs formed by these sets is $\Delta R \leq \Delta\mathcal{R}_n + 2$. In (b) the construction of a spanning tree is shown. Note that the dashed lines are not edges of the graph, but are used to show the structure of the tree.

Q_{i-1} (this path is the corresponding edge of the binary tree shown in the figure). Therefore:

$$D'_{i+1} \leq 4 + D'_i, \; i < m-1. \tag{4.4}$$

The initial condition of this recursion is taken as the largest D'_i such that $D'_i < 4 + D'_{i-1}$. This can only occur when the diameter of the ring R_i is $\Delta R_i < 4 + \Delta R_{i-1}$, which gives:

$$\frac{2^i}{2} < 4 + \frac{2^{i-1}}{2}, \tag{4.5}$$

and the initial condition as $i = 3$ and $D'_3 = 4$, which gives $D'_{i\geq 3} = 4(i-2)$. The diameter is obtained by noting that the largest distance between a node in R_{m-1} and the closest node in Q_{m-1} is 1 (since Q_{m-1} is formed by the even nodes of R_{m-1}), which would give the diameter $\Delta\mathcal{L}(R_{1,2^m}) = 4(m-3)+2$. However this reasoning does not take into consideration nodes in $\{0, 1, 2, 3\}$. The largest distance between these nodes and Q_2 is 4, so the diameter will be dominated by this distance for all $R_{j>2}$ such that $2(j-2)+1+4 > 4(j-2)+2$, which gives the smallest R not affected as R_4. However it is also true that the distance between node 6 and the furthest node in R_{m-1} should be larger or equal to the distance $l_{0,6}$, which gives the critical value $i_a$as the smallest solution of:

$$\frac{2^{i_a}}{2} \geq 2(i_a - 2) + 4, \tag{4.6}$$

which is a tighter bound, and the diameter is:

$$\Delta\mathcal{L}^+(\mathcal{R}_{1,2^m}) = \begin{cases} 2(m-3)+5 & m < i_a + 1 \\ 4m - 10 & m \geq i_a + 1 \end{cases}. \tag{4.7}$$

Some values of eq. (4.7) are tabulated in table 4.1.

This model can be extended for rings with $k > 1$. Two changes are required to determine the diameter of $\mathcal{L}^+(\mathcal{R}_{k>1,2^m})$. The first is a change in determining the initial condition of the recursion in eq.4.4. Again it is taken as the largest D'_i such that $D'_i < 4 + D'_{i-1}$; however the diameter of the ring R_i is now given as $\Delta\mathcal{R}_{k,N} = \lceil N/(2k) \rceil$. Therefore the initial condition is the smallest i which satisfies $\Delta R_i \geq 4 + \Delta R_{i-1}$, which gives:

$$\left\lceil \frac{2^{i_c}}{2k} \right\rceil \geq 4 + \left\lceil \frac{2^{i_c-1}}{2k} \right\rceil, \tag{4.8}$$

m	k=1		k=2		k=4		k=8	
	A	T	A	T	A	T	A	T
3	5	≤ 5	3	≤ 3	1	≤ 2	1	≤ 1
4	6	≤ 7	4	≤ 5	2	≤ 4	1	≤ 2
5	9	≤ 9	6	≤ 7	4	≤ 6	2	≤ 5
6	13	≤ 14	8	≤ 9	6	≤ 8	4	≤ 7
7	18	≤ 18	12	≤ 12	9	≤ 10	6	≤ 9
8	22	≤ 22	16	≤ 16	12	≤ 12	9	≤ 11
9	26	≤ 26	20	≤ 20	16	≤ 16	12	≤ 12
10	30	≤ 30	24	≤ 24	20	≤ 20	16	≤ 16

Table 4.1.: Diameter of $\mathcal{L}^+(\mathcal{R}_{k,2^m})$. Model predictions (T) are shown next to the actual values (A).

and for $k = 2^x$ the initial condition is found at i_c =4+$\log_2 k$, and $D'_{i_c} = 8$,which gives the diameter as:

$$\Delta\mathcal{L}^+(\mathcal{R}_{k,2^m}) = 4(m - \log_2 k - 3). \tag{4.9}$$

This calculation, again, does not take into account nodes $\{0,1,2,3\}$, which can be considered by noting that the largest distance between these nodes and Q_2 for $k > 1$ is the diameter of the ring $\mathcal{R}_{k,12} = \lceil 6/k \rceil$ (the largest distance between nodes in $\{0,1,2,3\}$ and nodes in Q_2 is $l_{0,6}$). The diameter will be dominated by this distance for all $R_{j>2}$ such that $D'_2 + 2(j_c - 2) < \lceil 6/k \rceil$, which gives the critical value j_c as the largest $j < (\lceil 6/k \rceil + 3)/2$. However it is also true that the distance between node 6 and the furthest node in R_{m-1} should be larger or equal to the distance $l_{0,6}$ for eq.4.9 to apply, which gives the critical value i_d as the smallest solution of:

$$\frac{2^{i_d}}{2k} \geq 2(i_d - 2) + \left\lceil \frac{6}{k} \right\rceil, \tag{4.10}$$

which is a tighter bound, and the diameter is:

$$\Delta\mathcal{L}^+(\mathcal{R}_{k,2^m}) = \begin{cases} 2(m-3) + \lceil \frac{6}{k} \rceil & m < i_d + 1 \\ 4(m - \log_2 k - 3) & m \geq i_d + 1 \end{cases}. \tag{4.11}$$

A table showing a number of experimental measurements and expected values is shown in table 4.1.

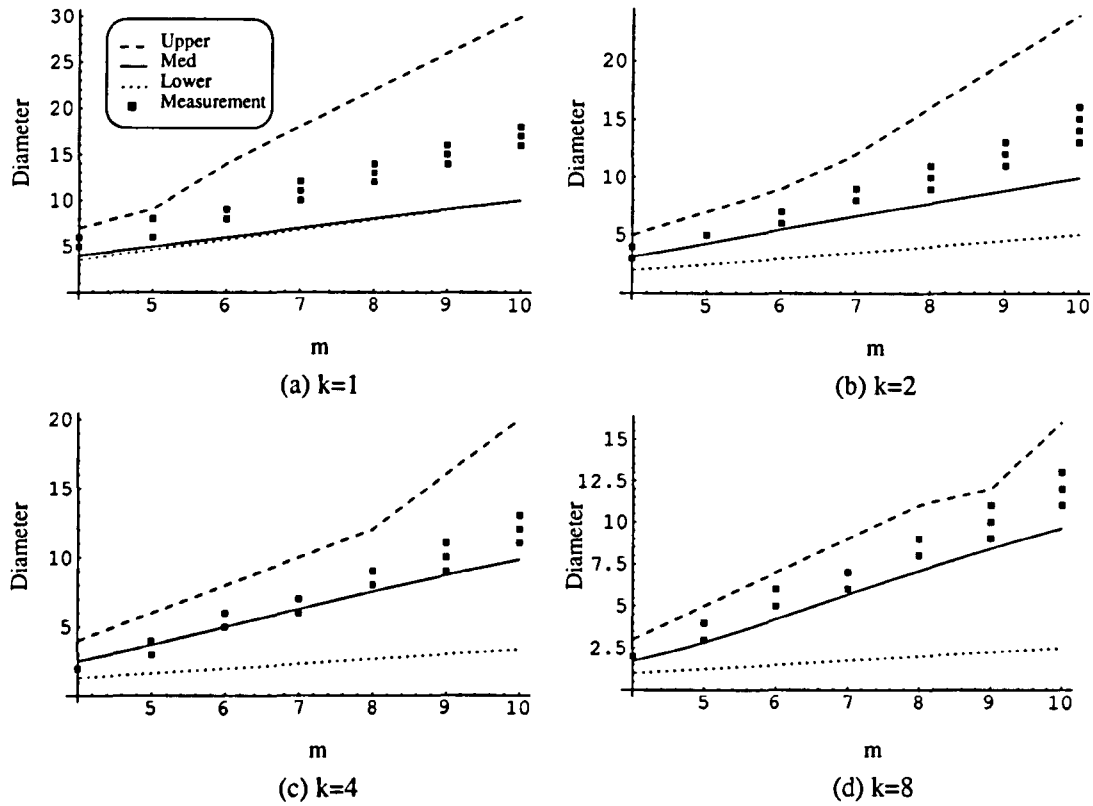


Figure 4.5.: Diameter of all the graphs in the family $\mathcal{L}(\mathcal{R}_{k,2^m}, \mathcal{P})$, (k={1, 2, 4, 8}, m={4,...,10}, and all primitive $\mathcal{P}$). The actual values (labelled "Measurement") are plotted against the diameter of the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$ from eq. (4.11) which is labelled "Upper", the Moore graph labelled "Lower", and the expected diameter from eq. 4.12, labelled "Med".

4.3.2. Diameter of $\mathcal{L}(\mathcal{R}_{k,2^m}, \mathcal{P})$

The diameter of the full LFSR graph $\mathcal{L}(\mathcal{R}_{k,2^m}, \mathcal{P})$, is bounded by the diameter of the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$ given in eq. (4.11), which is the upper bound, and by a Moore approximation providing a lower bound. We are interested in providing a better approximation. This is easily accomplished by use of eq. (3.12), which gives:

$$D_{\mathcal{L}(\mathcal{R}_{k,2^m},\mathcal{P})} = \frac{m}{\log_2\left(2^{km(2^{1-m})}+1\right)}. \tag{4.12}$$

The different bounds as well as eq. (4.12) are plotted against actual values in figure 4.5.

4.3.3. Routing

A simple greedy distributed routing algorithm can be constructed from the observations made in section 4.3.1 regarding the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$. We will construct a greedy routing algorithm based on computing the routing distance through the underlying ring graph, and through the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$. The algorithm selects the route based on the shortest distance. However, during this process a number of shortcuts are reached; at these points a routing decision is made, and if it is advantageous packets will be re-routed through the shortcut. Two functions for each graph are required to perform the routing task; one which determines the routing distance between two nodes $d(u,v)$ and one which determines the exit node of an incoming packet, called the continuation function $\epsilon(w,v)$, (provided an intermediate node w, and the given destination v are known).

The routing distance between two nodes $\delta^{\mathcal{R}}(s,d)$ in $\mathcal{R}_{k,2^m}$ is given as

$$\delta^{\mathcal{R}}(s,d) = \begin{cases} \left\lceil \frac{\min\{d-s,\, n-d+s\}}{k} \right\rceil & d \geq s \\ \left\lceil \frac{\min\{s-d,\, n-s+d\}}{k} \right\rceil & d < s \end{cases} . \tag{4.13}$$

The continuation function for the ring is simply:

$$\epsilon^{\mathcal{R}}(s,d) = \begin{cases} d & \delta^{\mathcal{R}}(s,d) \leq 1 \\ (n+s-k) \bmod n & d-s > n-d+s,\ \delta^{\mathcal{R}}(s,d) > 1,\ d \geq s \\ (n+s-k) \bmod n & s-d < n-s+d,\ \delta^{\mathcal{R}}(s,d) > 1,\ d < s \\ (s+k) \bmod n & \text{otherwise} \end{cases} . \tag{4.14}$$

For the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$ we begin by constructing a binary tree as shown in figure 4.6. Neighbouring nodes are merged and renamed by simply dropping the last bit; node $u = (u_n, u_{n-1}, \ldots, u_0)$ becomes $u' = (u_n, u_{n-1}, \ldots, u_1)$. Furthermore, nodes 0 and 1 are merged into node $u' = 0$ and attached as an extra appendix to node $u' = 1$ (which is the base of the binary tree). A simple optimal tree routing is used, based on routing packets through the root of the smallest subtree containing the current (i) and destination (d) nodes. This function is given as:

$$\epsilon^{\text{Tree}}(i,\, d) = \begin{cases} (i_n,\, i_{n-1},\, \cdots,\, i_0,\, d_{m-n-1}) & m > n,\, \forall 0 \leq j \leq n,\, i_{n-j} = d_{m-j} \\ \\ (i_n,\, i_{n-1},\, \cdots,\, i_1) & \text{otherwise} \end{cases}, \tag{4.15}$$

where $i_n = 1$ is the most significant bit of i which is equal to one ($n = \max\{n : i_n = 1\}$) and $d_m = 1$ with $m = \max\{m : d_m = 1\}$.

If we call r the base node of the subtree containing i and d. The node $r = (i_n,\, i_{n-1}, \ldots,\, i_j)$, where $j = \max\{j : \forall 0 \leq x \leq j,\; i_{n-x} = d_{m-x}\}$. The distance from r to i is simply $l_{r,i} = n - j$, and similarly $l_{r,d} = m - j$, which gives:

$$\delta^{\text{Tree}}(i,\, d) = m + n - 2j. \tag{4.16}$$

To obtain the corresponding expressions for the subgraph $\mathcal{L}^{+}(\mathcal{R}_{k,2^m})$ we refer to the explicit tree construction shown in figure 4.6. As the figure show, for routing purposes virtual nodes with two nodes are constructed. Note that some nodes are shown as smaller circles next to a virtual node. In particular, nodes $v = 4S + 3$ for all $0 \leq S < N/4$. are of this kind and do not form part of the subgraph $\mathcal{L}^{+}(\mathcal{R}_{k,2^m})$. In order to provide a routing path for these nodes we will choose to include them in the virtual node $2S + 1$, as shown in the figure. Note that other solutions are possible (i.e. assigning them to the other virtual node; or dynamically assigning them to the virtual node which provides the shortest path); however since the final version is a greedy routing algorithm which at each step looks for the best path, this decision is not important.

In addition to the tree structure, there is an appendix to the root node, formed by nodes zero and one. The virtual nodes are formed according to the following rules:

$$i' = \begin{cases} (i_n,\, i_{n-1}, \ldots, i_1) & i_0 = 0 \\ (i_n,\, i_{n-1}, \ldots, i_2) & i_1 = 0,\, i_0 = 1 \\ (i_n,\, i_{n-1}, \ldots, i_2,\, 1) & i_1 = 1,\, i_0 = 1 \end{cases}, \tag{4.17}$$

where the last rule is used to assign routes for nodes not in the subgraph $\mathcal{L}^{+}(\mathcal{R}_{k,2^m})$. Equation 4.16 can be used to obtain the routing distance in the graph. However we need to consider the effects introduced by the virtual nodes, and also of the additional nodes. If we proceed as before, and call r the base node of the subtree containing i' and d', then $r = (i'_n,\, i'_{n-1}, \ldots,\, i'_j)$, where $j' = \max\{j : \forall 0 \leq x \leq j,\; i'_{n-x} = d'_{m-x}\}$. The distance from r to i' is simply $l_{r,i'} = n' - j'$, and similarly $l_{r,d'} = m' - j'$. We are interested in obtaining the routing distance to get out of the virtual nodes i' and d'.

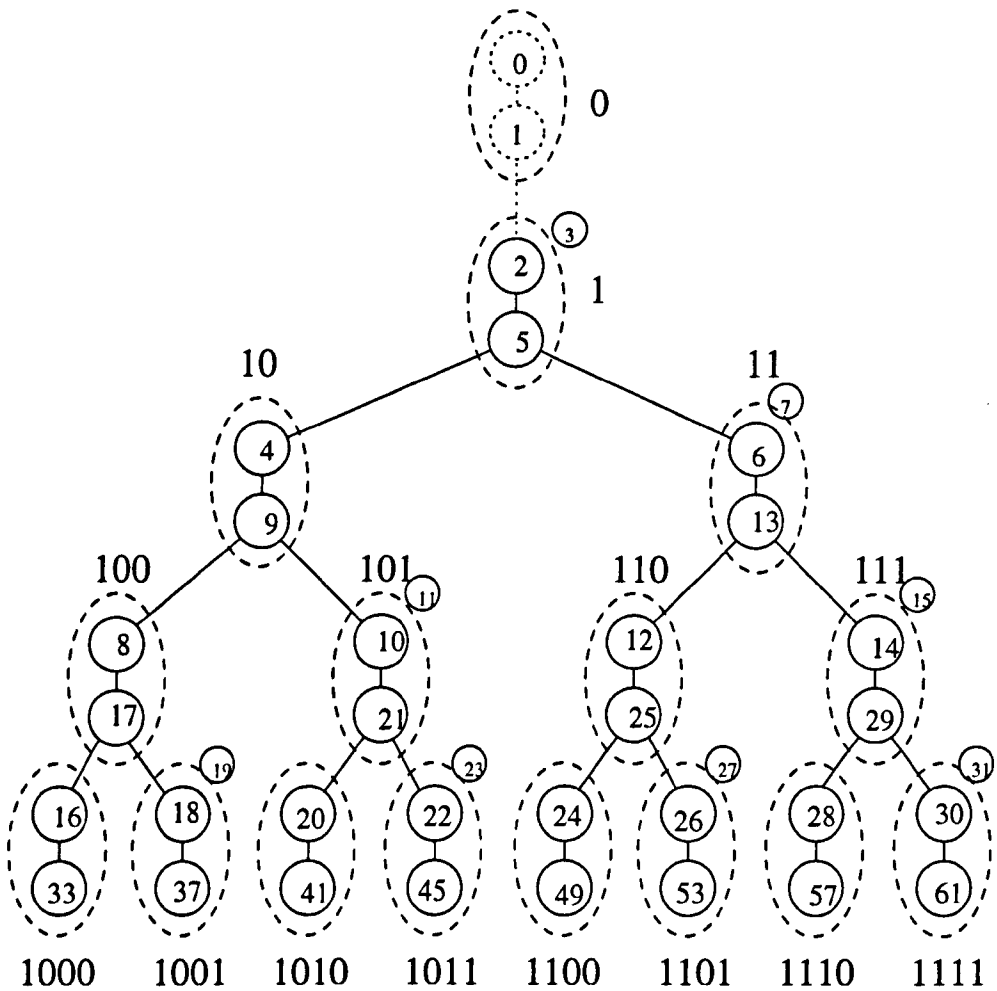


Figure 4.6.: Tree construction of the routing graph for the subgraph $\mathcal{L}^+(\mathcal{R}_{k,2^m})$. Virtual nodes corresponding to a binary balanced tree are shown as dashed ellipses, along their binary labels. Each virtual nodes consist of two nodes and are of the form $2S$, $4S + 1$(they are the edges from eq. 4.3).

This distance depends on whether we depart the intermediate virtual node from the top or bottom nodes, and on whether we arrive at the destination virtual node from the top or bottom. The only way for a packet to be routed in the down direction from the intermediate node is if the pattern matches; i.e. if $n' = j'$. Similarly, the only way for packets to arrive to the destination node from the bottom, is if $m' = j'$. The internal routing distance for packets departing from the top of the intermediate node, or arriving from the top of the destination node is given by:

$$I_{\text{top}}(x) = \begin{cases} 0 & x_0 = 0 \\ 1 & x_0 = 1 \end{cases} . \tag{4.18}$$

The internal routing distance for packets departing from the bottom of the intermediate node, or arriving through the bottom of the destination node is given as:

$$I_{\text{bot}}(x) = \begin{cases} 0 & x_{1\ldots0} = (0,\,1) \\ 1 & x_0 = 0 \\ 2 & x_{1\ldots0} = (1,\,1) \end{cases} . \tag{4.19}$$

The number of complete intermediate virtual nodes traversed by the path is $(n' - j') - 1 + (m' - j') - 1$, and in each one there is one (and only one) internal edge traversed. Further to this, there is an incomplete virtual node visited (the root node), in which no internal edges are traversed. If this node is either the source or destination then one additional edge is traversed; otherwise two additional edges are traversed. Therefore the total distance is:

$$\delta^{\mathcal{L}^+(\mathcal{R}_{k,2^m})}(i,\,d) = \begin{cases} 2(n' + m' - 2j' - 2) + 2 + I_{\text{top}}(i) + I_{\text{top}}(d) & n' \neq j' \wedge m' \neq j' \\ 2(n' + m' - 2j' - 2) + 1 + I_{\text{top}}(i) + I_{\text{bot}}(d) & n' = j' \wedge m' \neq j' \\ 2(n' + m' - 2j' - 2) + 1 + I_{\text{bot}}(i) + I_{\text{top}}(d) & n' \neq j' \wedge m' = j' \\ I_{\text{top}}(i) + I_{\text{top}}(d) & n = j \wedge m = j \end{cases} \tag{4.20}$$

The continuation function is based on pattern matching as in eq. 4.15, however it is required to account for the virtual nodes in the network, and of the additional nodes. For each kind of node within a virtual node, a specific rule is required. Packets will be routed towards the root of the routing tree if the patterns are distinct (up direction), and will be routed towards the bottom of the tree when patterns have matched (down direction); however packets can only depart from either the topmost (if going up), or down-most node (if going down) of a virtual node :

$$\epsilon^{\mathcal{L}}(i,\, d) = \begin{cases} d & i_{n\ldots1} = d_{m\ldots1} \\ \text{up} & i_0 = 0, \text{ unmatched} \\ \text{top/virtual} & i_{1\ldots0} = (0,1), \text{ unmatched} \\ \text{top/virtual} & i_{1\ldots0} = (1,1), \text{ unmatched} \\ \text{bottom/virtual} & i_0 = 0, \text{ matched} \\ \text{down} & i_{1\ldots0} = (0,1), \text{ matched} \\ \text{bottom/virtual} & i_{1\ldots0} = (1,1), \text{ matched} \end{cases}, \tag{4.21}$$

which can be rewritten as:

$$\epsilon^{\mathcal{L}}(i,\, d) = \begin{cases} d & i_{n\ldots1} = d_{m\ldots1} \\ (i_n,\, i_{n-1},\, \cdots,\, i_2,\, 0,\, 1) & i_0 = 0, \text{ unmatched} \\ (i_n,\, i_{n-1},\ldots,\, i_1) & i_{1\ldots0} = (0,1), \text{ unmatched} \\ (i_n,\, i_{n-1},\ldots,\, i_2,\, i_1,\, 0) & i_{1\ldots0} = (1,1), \text{ unmatched} \\ (i_n,\, i_{n-1},\, \cdots,\, i_0,\, 1) & i_0 = 0, \text{ matched} \\ (i_n,\, i_{n-1},\ldots,\, i_2,\, d_{n-m+1},\, 0) & i_{1\ldots0} = (0,1), \text{ matched} \\ (i_n,\, i_{n-1},\ldots,\, i_2,\, i_1,\, 0) & i_{1\ldots0} = (1,1), \text{ matched} \end{cases}, \tag{4.22}$$

where the strings are matched if $\forall 0 \leq x \leq n,\, i_{n-x} = d_{m-x}$, or if $i \in \{0,\, 1\}$, and unmatched otherwise. Using these expressions, the greedy routing algorithm 1 can be constructed, where at each intermediate node a routing decision is made based on finding the smallest routing distance from the current node, and from its neighbours, to the destination.

This routing algorithm can be extended to search for an optimal route not only within the immediate neighbourhood of the intermediate node, but with neighbours of neighbours and in general nodes at a depth (distance) h of the intermediate node as shown in algorithm 2. It is also possible to use minimal routing by solving the single source shortest path problem, which can be done using the Dijkstra or Bellman-Ford algorithms. Using the proposed algorithm with a large enough h yields the same results, however we are interested in determining if a smaller h can provide similar results saving computational effort and resources.

4.3.4. Discussion

The routing length and diameter of the the routing algorithms is shown in figure 4.7 (note that the values have been normalised to the actual diameter and average length

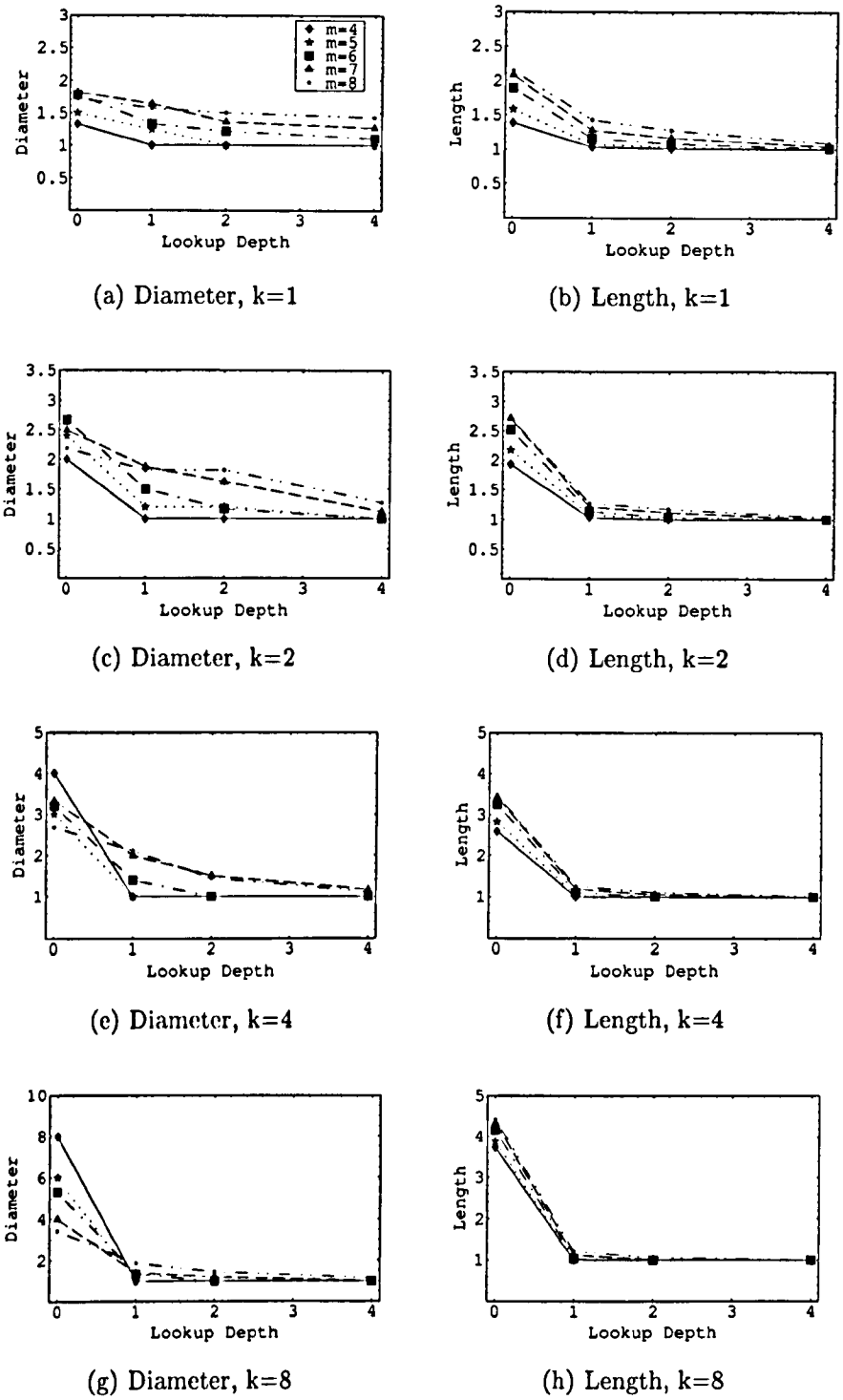
Algorithm 1 Greedy Routing in LFSR graphs.

```
Greedy LFSR Route (intermediate node i, final node d)
{
δ_Curr = ∞  //Current minimum distance
ϵ = {}     //Node to route to
N_x=Neighbours(N_{x-1})
forall n ∈ N_x
      {
         if (δ^R(n,d) < δ_Curr)
            then { δ_Curr = δ^R(n,d), ϵ = n }
         if (δ^L(n,d) < δ_Curr)
            then { δ_Curr = δ^L(n,d), ϵ = n }
      }

Route to ϵ
}
```

Algorithm 2 Greedy Depth Routing in LFSR graphs.

```
Greedy LFSR Depth Route (intermediate node i, final node d, depth h)
{
N_0 = i
δ_Curr = ∞  //Current minimum distance
ϵ = {}     //Node to route to
for x=1 to h
   {
      N_x=Neighbours(N_{x-1})
      forall n ∈ N_x
      {
         if (x + δ^R(n,d) < δ_Curr)
            then { δ_Curr = x + δ^R(n,d), ϵ = n }
         if (x + δ^L(n,d) < δ_Curr)
            then { δ_Curr = x + δ^L(n,d), ϵ = n }
      }
   }
Route to ϵ
}
```



(a) Diameter, k=1

(b) Length, k=1

(c) Diameter, k=2

(d) Length, k=2

(e) Diameter, k=4

(f) Length, k=4

(g) Diameter, k=8

(h) Length, k=8

Figure 4.7.: Routing diameter and length normalised to the actual diameter and length of the graphs $\mathcal{L}(\mathcal{R}_{k,2^m}, \mathcal{P})$, (k={1, 2, 4, 8} and $\mathcal{P}$={6, 12, 29, 54, 123} for m={4, 5, 6, 7, 8}(in that order) plotted against the lookup depth of the routing algorithm. When the lookup depth is zero, tree routing as shown in figure 4.6 and described in eq. 4.22 is used. When the lookup depth is one, the simple greedy routing algorithm (algorithm 1) is used.

of the networks). Although simulations were carried out only for modest network sizes (up to 256 nodes), some general conclusions can be reached. Networks with a relative small degree (compared to the network size) exhibit a slow convergence to the optimal value, as shown in (a). In this case there is a large reduction in diameter provided by the multiple shortcuts; however it is difficult for the routing algorithm to exploit the available shortcuts as only local information is considered. As a relative few number of choices are available at each routing step, short paths have a small probability of being found. Due to the low convergence rate a full shortest path algorithm (Dijkstra, Bellman-Ford) would most certainly be required to achieve minimal routing; however it is interesting to note that in the worst case the routing diameter is not larger than twice the optimal value.

These results seem to suggest that for large networks with small degrees there is little benefit in using greedy depth routing. However it is also important to note that although for these graphs the routing diameter convergence rate is slow, this is no necessarily true for the average routing length, where a fast decrease is observed, as shown in (b), at least for the modest network size of the graphs analysed.

For networks with large degrees, the results of figure 4.7 show that a fast decrease in routing diameter and average length are observed (sub-figures (e) to (h)), as the routing algorithm is able to find shortcuts more efficiently. This seems to suggest that a small lookup depth gives close to optimal performance. Larger lookup depths give marginal gains, except at the crossover length between large and small degrees, where some benefit can be expected, as is evident in sub-figure (c).

4.4. Conclusions and Summary of Results

A family of deterministic and regular small-world interconnection networks has been introduced. A specific sub-family of parametrised graphs which use a linear feedback register as the generator for the shortcuts in the network called the LFSR has been proposed. It has been shown that networks in the family have a small diameter. An upper bound has been constructed by embedding a binary tree in the graph, and shown to grow logarithmically with the size of the graph. However by taking measurements directly from ring LFSR graphs the diameter of graphs with a small degree in the family has been experimentally shown to be close to the Moore limit. Large degree LFSR graphs, on the other hand, posses larger diameters, since much of the edges forming part of the underlying graph do not expand as well as the additional edges created by the LFSR. Therefore it is expected that networks with

a small fixed degree such as the ring, mesh or torus are more suitable to be used as underlying graphs than networks with a larger degree, such as the hypercube. The LFSR family can be extended to accommodate networks with larger degrees by providing a mechanism to introduce a larger amount of additional edges, although more research is required to determine the merits of such approach.

Due to the deterministic nature of the graphs considered in this chapter, simple decentralised routing algorithms which do not require solving the shortest path problem can be constructed. For the LFSR such an algorithm is derived. Unfortunately the present algorithm does not provide minimal routing and makes use of a greedy approach in which each routing decision is made selecting the best path found at each visited node. The deterministic nature of the network is exploited to create a tree embedding which is used as part of the algorithm. The results suggest that for large degree networks a small lookup depth gives close to optimal results; whereas for networks with a relative small degree it is necessary to use a large depth to achieve close to minimal routing, and hence more computational effort and resources are required. These results seem to suggest that for the more suitable underlying graphs it is necessary to use a large depth, or possibly to solve the complete shortest path problem to obtain a small routing diameter. However it is important to note that although there is only a small reduction to the routing diameter, the average routing distance experiences a much faster reduction as the lookup depth is increased, and that therefore for small degree graphs a small lookup depth might provide acceptable results, especially if there is a large enough amount of locality in the communication pattern. It is important to note that even if the full single source shortest path problem needs to be solved, the memory requirements to do so are small since the topology of the network depends only on the polynomial $\mathcal{P}$ and on the choice of underlying network. The routing algorithm can very easily be extended to other members of the family. It is expected that better routing algorithms can be obtained by working directly in the Galois field representation; however it is also possible to solve the shortest paths problem using Dijkstra or other algorithms, and that the amount of information to describe the whole network is much smaller than in the random case. In particular for the LFSR it is only required to know the generator polynomial and the structure of the underlying network, while for a random small-world network information of the location of all additional edges is required (which grows as $O(\phi kN)$).

Future work will include the study of other important topological properties such as bisection bandwidth, fault tolerance and crossing number. It is anticipated that the fault tolerance of the networks is high, as well as the crossing number and the two

dimensional layout complexity. These last two undesirable characteristics might be offset by the small diameter and degree, the large scalability of the networks, and the expected large bandwidth and fault tolerance; in particular in applications requiring a fast, low cost and low latency interconnection network for large systems.

5. Deterministic Direct Constructions

5.1. Multi-Scale SW Interconnection Networks

5.1.1. Introduction

In order to fully exploit the small-world phenomenon, it is necessary to understand what is the fundamental mechanisms which cause the small-world behaviour. Watts and Strogatz used randomness to construct small-world networks in their original work. However it is not clear if and to what extent randomness is a fundamental requirement for the construction of small world networks. It is made clear that a number of shortcuts appear due to the random rewiring process; but is it possible to recreate the phenomena with a deterministic process?

Recently an alternative hypothesis has been put forward by R. Kasturiarangan [10]. The Multiple scale network hypothesis states that the fundamental mechanism behind the small-world phenomenon is not the interaction between order and randomness; but the addition of edges of a multitude of length scales to a graph. According to this hypothesis the distribution of the length scales of the new edges is more important than whether they are long range, or short range. The hypothesis is supported by experimental data and theoretical considerations.

The understanding of the fundamental mechanisms of the small-world phenomenon provides a framework for the construction of deterministic graphs which exhibit the small world phenomenon, and the desirable properties of small-world interconnection networks. Furthermore such graphs can provide better characteristics, such as a fixed degree and a more simple routing algorithm.

The design methodology of such graphs is simple and flexible. The most important step is to identify a suitable function or process which can generate a large amount of length scales, as measured in a proposed underlying network. This network is introduced into the design function or process, and is generated by it. Due to this reason the resulting graph is not an overlay of two different graphs, but a single

entity, generated by one mathematical process. Obviously a large number of such processes can be constructed, and due to the generality of the design methodology, it is not possible to derive characteristics common to all of them (of course the general characteristics of small-world interconnection networks should apply to them). For this reason this chapter is focused on a specific design example, where characteristics relevant to the design are analysed.

5.1.2. Multiple Scale Graphs

The exposition here follows the work in [10]. Let G be a graph, and $l_G(u, v)$ the distance between vertices u and v in G. Allow the set of edges H to be added to G forming a new graph G' ($V_H \in V_G$). For each new edge e_H the distance $l_G(e)$ is the distance between the endpoints of e in G.

Definition 5.1.1 *The graph G' obtained by adding edges to the graph G is multiple scale with respect to N (denoted $G' \preceq G$) if: $\exists r \gg 0$, and length scales λ_i, $i = 1, 2, \ldots r$ such that $0 < \lambda_1 \ll \lambda_2 \ldots \ll \lambda_r \ll n$ and $\forall i : i \leq r, \lambda_i \in \{l_G\}$.*

Definition 5.1.2 *A graph G is a multiple scale graph if it has a subgraph S with the same number of vertices, and $S \preceq G$.*

5.2. Hilbert Graph

The Hilbert graph was explicitly developed to provide a deterministic small-world graph by providing a multiple scale deterministic graph. Fractals are natural scale-free structures, and therefore make excellent candidates for the construction of scale-free graphs. For this graph, a well known fractal, the Hilbert curve has been used as a basis to develop the interconnection. As a result of its small-world scale-free nature the Hilbert graph has some remarkable characteristics: a fixed degree, modular expansion, ability to exploit the locality of information, an efficient two dimensional layout and a diameter that scales almost as well as that of a random graph.

The Hilbert curve is a Lindenmayer system (L-system) defined by the initial string L, with the replacement rules (L $\rightarrow$ +RF-LFL-FR+, R $\rightarrow$ -LF+RFR+FL-). In the limit of recursion the Hilbert Curve is a two dimensional space filling curve.

The Hilbert curve can also be defined recursively as follows. C_0 is a line segment. C_{n+1} is formed by the union of four copies of C_n arranged in a square, where the upper left copy is rotated 90°and the lower left copy is rotated -90°as shown in figure 5.1. The first few recursions are shown in figure 5.2.

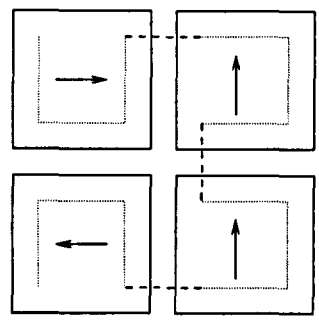


Figure 5.1.: Recursive construction of Hilbert graphs.

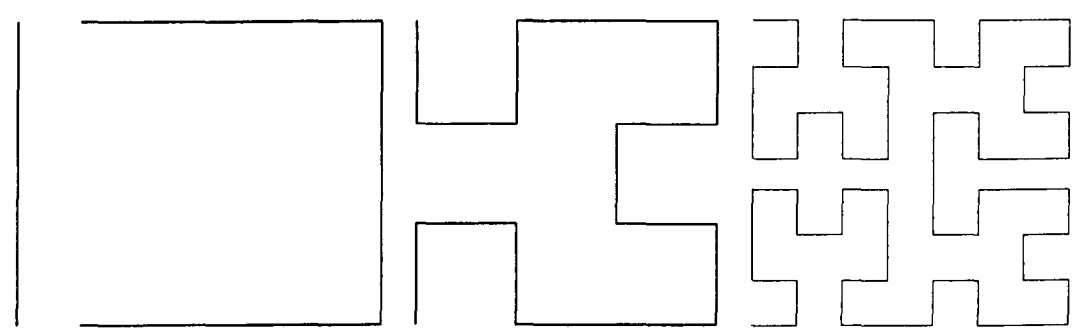


Figure 5.2.: Construction of Hilbert Curves.

5.2.1. Definition

The Hilbert graph is a 4-regular graph formed by the superposition of a Hilbert Curve with an extended mesh. Nodes are placed along the *middle* of a segment in a Hilbert curve, as shown in figure 5.4. The extended mesh is created by joining nodes with the same horizontal or vertical position, such that the additional connections are perpendicular to the Hilbert curve as they arrive at the nodes, and the degree of any node is fixed at four. The Hilbert graph can be in either of two states. It is in the closed state if edges at one side are allowed to loop back and connect with nodes on the opposite side, and an additional node. Otherwise it is in the open state. Usually it is not necessary to make an explicit distinction between them; when the Hilbert graph is referred on its own, as could be used in a real system, it is assumed to be a closed Hilbert graph. When a Hilbert graph is referred to in the context of a recursive construction, it will be assumed to be open. When necessary the status of the graph will be included in the description to avoid confusions.

Two equivalent constructions which will act as extended definitions are provided for the Hilbert graph:

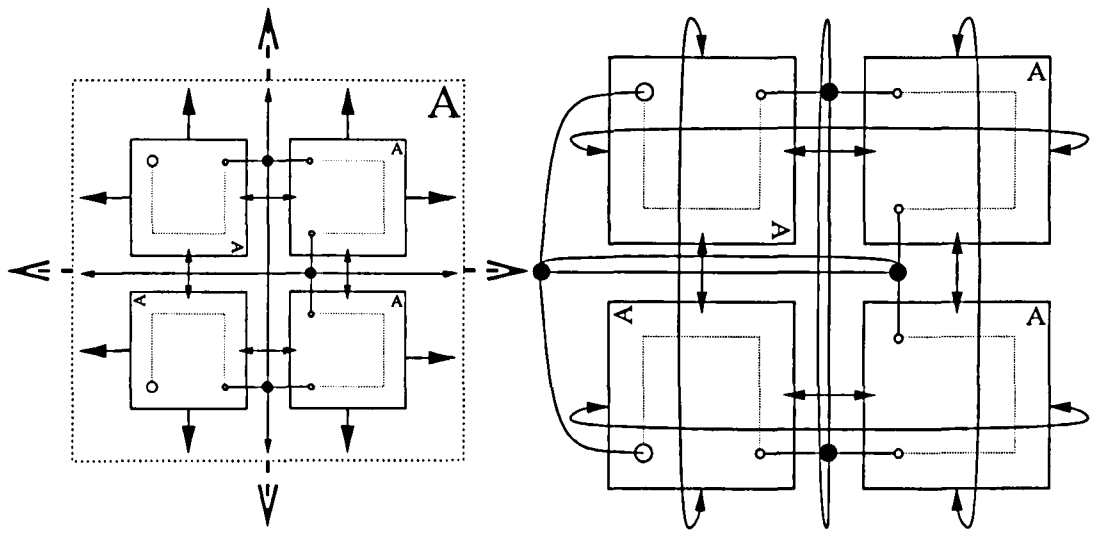
Geometric Definition: An open $H_{n>0}$ Hilbert graph is formed by placing $N = 4^n - 1$ equally spaced nodes across a C_n Hilbert curve, with each node in the middle of a line segment. For each node a new line perpendicular to the segment of the Hilbert curve containing it is drawn, and extended until it reaches another node (or until it loops back to the origin, in the case of the closed Hilbert graph). The graph whose edges are identified to the segments drawn is the H_n Hilbert graph. Furthermore, the closed Hilbert graph has an additional node joined to the start and end nodes of the graph (open circles in the figure),

Recursive Definition: The Hilbert graph is defined recursively (in a similar way as the Hilbert curve is) as follows. H_0 is a line segment with a node in the middle. H_{n+1} is formed by the union of four rotations of H_n as defined by the operator shown in figure 5.3. The top level graph from this recursion can be closed, as shown in the figure. The first few recursions are shown in figure 5.4.

5.2.2. Diameter

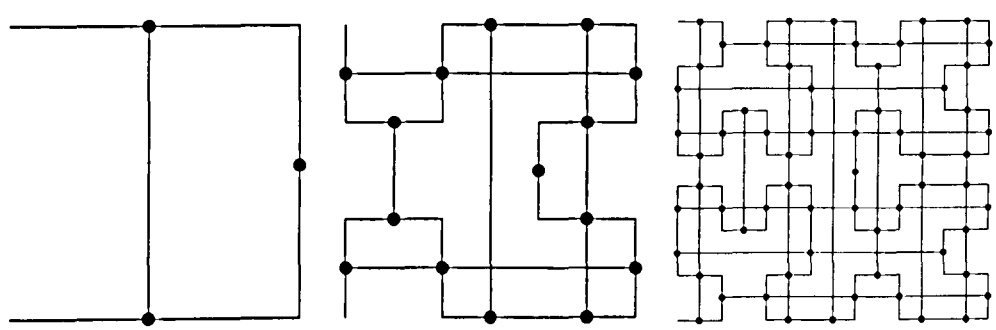
The length and diameter of a Hilbert graph have not been calculate exactly in this work. However some bounds and approximations have been established, which are discussed in this section for the diameter, and in the next section for the average length.

An upper bound for the open case can be constructed as follows. Define the



(a) Open (b) Closed

Figure 5.3.: Recursive construction of a Hilbert graph. The H_{n+1} Hilbert graph consists of four H_n graphs rotated as shown in (a). The letter 'A' is used as an orientation marker. Extensions are connected to adjacent graphs internally, and are grouped together externally to form the new graph. The top level graph can be closed by looping back the extensions as shown in (b).



(a) H1 (b) H2 (c) H3

Figure 5.4.: Construction of Hilbert Graphs.

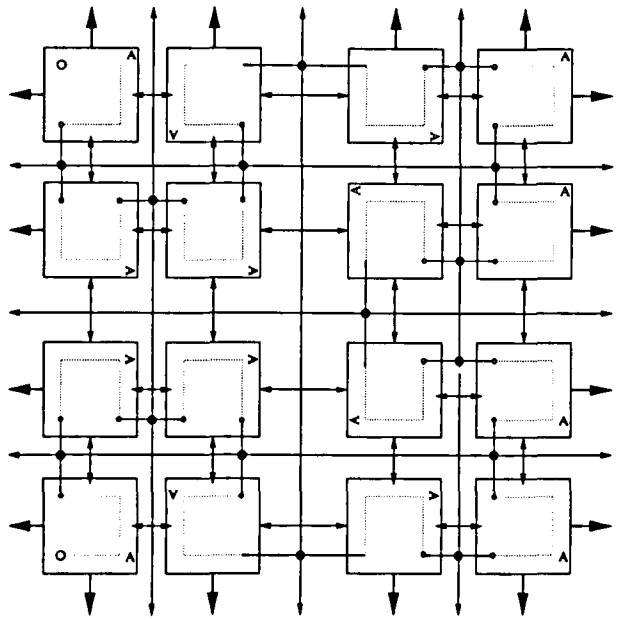


Figure 5.5.: Recursive construction of a Hilbert graph from sixteen H_{n-2} subgraphs.

diameter D_n^O as the largest minimum distance between any pair of nodes in the open Hilbert graph H_n. The largest minimum distance separates the graphs H_{n-1}^O in opposite quadrants. Since the graph H_n is horizontally symmetrical the results are equally valid for both quadrants. Since the asymptotic behaviour of distances within the H_{n-2}^O modules is not known at this stage, we will assume without any loss of generality that any distance traversed within the module is larger than a given constant. With this assumption the minimal route shown in figure 5.6 is constructed, and its length can be written as:

$$D_n^O \leq 2D_{n-2}^O + 8 + E_{n-2}^O, \tag{5.1}$$

where the minimum distance between the start and end nodes of a Hilbert graph is approximated following the path shown in the figure, and is given as $E_n^O \leq 2D_{n-2}+3$. These equations, with the appropriate set of initial conditions taken from table 5.1 give the diameter as:

$$D_{n\geq 4}^O \leq \begin{cases} \frac{1}{12}\left(7+13\sqrt{3}\right)\left(1+\sqrt{3}\right)^{\frac{n}{2}} - \frac{11}{3} + \\ \frac{1}{12}\left(7-13\sqrt{3}\right)\left(\sqrt{3}-1\right)^{\frac{n}{2}}(-1)^{\frac{n}{2}} & \frac{n}{2} \in \mathbb{N} \\ \\ \frac{1}{6}\left(5+3\sqrt{3}\right)\left(1+\sqrt{3}\right)^{\frac{n+1}{2}} - \frac{11}{3} - \\ \frac{1}{6}\left(3\sqrt{3}-5\right)\left(\sqrt{3}-1\right)^{\frac{n+1}{2}}(-1)^{\frac{n+1}{2}} & \frac{n+1}{2} \in \mathbb{N} \end{cases}, \tag{5.2}$$

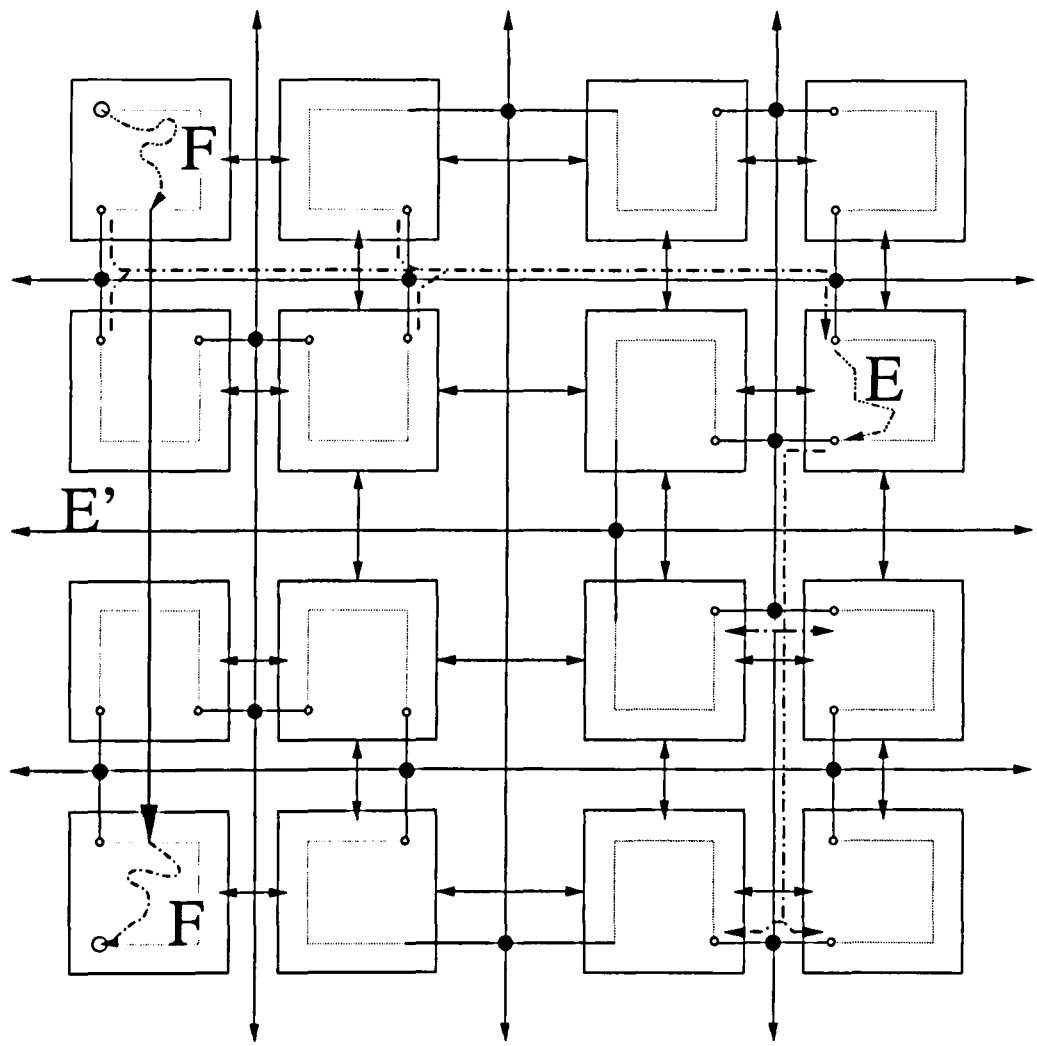


Figure 5.6.: Paths for calculating D_n^O. The path shown with a discontinuous line is not smaller than the diameter of the network. The minimum length of the path labelled E is analysed in the next level of recursion and shown in this diagram by the thick line (marked E'), and the two small paths F (which are approximated simply as the diameter of their respective blocks).

where the limit is valid for $n \geq 4$ since eq. 5.1 is only valid for such cases (it depends on the value of the diameter D^O_{n-4}). These can be rewritten as:

$$D^O_n = O\left[\left(1+\sqrt{3}\right)^{\frac{n}{2}}\right] \tag{5.3}$$

A less formal approach yields a good approximation from the recursive construction of the H_n Hilbert graph based on the H_{n-2}, as shown in figure 5.5. Since it is known that at least one edge connects each H_{n-2} graph to its nearest neighbour (effectively forming a two dimensional mesh), the diameter can be written as:

$$D^O_{n+2} = 6 + 2D^O_n, \quad n > 1, \tag{5.4}$$

where the condition $n > 1$ has been included because for $n = 1$ the diameter of the modules D^O_n has been effectively increased in the construction due to the introduction of the additional nodes, where for larger values of n this effect can be ignored. Note that this expression ignores the length of the path required to switch between the horizontal and vertical extensions. Solving eq. with the initial conditions $D^O_2 = 4$ and $D^O_3 = 9$ gives the diameter as:

$$D^O_{n>3} = 2^{\left(\frac{n-5}{2}\right)}\left(13+8\sqrt{2}+(-1)^n\left(-13+8\sqrt{2}\right)\right)-4, \tag{5.5}$$

which is shown in figure 5.7.

A good approximation for the diameter of the *open* Hilbert graph can be estimated using the same procedure. Due to the wrap around connections effectively a two dimensional torus is formed, so the diameter is given by:

$$D_{n+2} = 4 + 2D_n, \quad n > 1, \tag{5.6}$$

which can be solved as:

$$\begin{aligned} D_{n>3} &= 2^{\left(\frac{n-5}{2}\right)}\left(11+8\sqrt{2}+(-1)^n\left(-11+8\sqrt{2}\right)\right)-4 \\ &< 4\,2^{\frac{n}{2}}-4. \end{aligned} \tag{5.7}$$

As expected $O(D_n) = O(D^O_n) = O(2^{n/2})$, since only the uppermost level of recursion is different in both graphs; all other internal distances remain the same and should exhibit the same asymptotic behaviour.

A better approximation can be constructed numerically by assuming that the di-

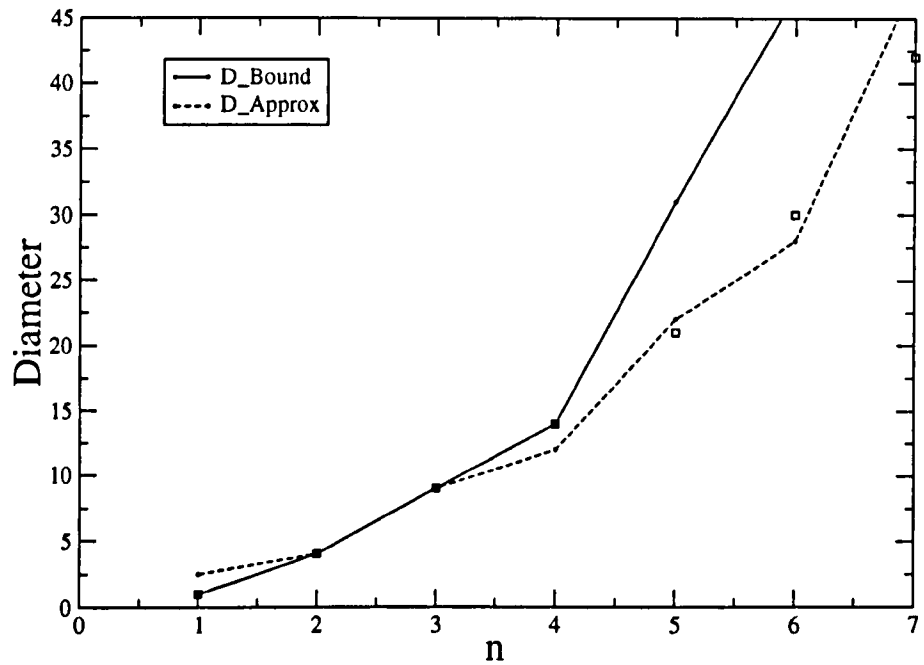


Figure 5.7.: Diameter of the open Hilbert graphs H_n^O. The figure shows experimental measurements taken directly from the graphs (squares), the upper bound (eq. 5.2), and the approximated diameter from equation 5.5. The figure shows results for graphs up to 16383 nodes (H_7^O), and as shown eq. 5.5 is a good approximation to the diameter.

ameter can be calculated recursively, and that therefore an x-order difference equation provides a good estimate. The first few of such equations are:

$$\begin{aligned} D_{n+1}^{(1)} &= \alpha D_n^{(1)} + \beta \\ D_{n+2}^{(2)} &= \alpha D_{n+1}^{(2)} + \beta D_n^{(2)} + \gamma \\ D_{n+3}^{(3)} &= \alpha D_{n+2}^{(3)} + \beta D_{n+1}^{(3)} + \gamma D_n^{(3)} + \delta \\ &\vdots \end{aligned} \tag{5.8}$$

We will use $D_2 = 4$, $D_3 = 7$ and $D_4 = 11$ as the initial conditions for solving these equations, which are direct measurements taken from the graphs (see table 5.1). The first equation has the solution:

$$D_n^{(1)} = \frac{D_2\alpha^{n+1} + (\beta - D_2)\alpha^n - \beta\alpha^2}{\alpha^2(\alpha - 1)} \tag{5.9}$$

$$= O\left(\alpha^{n-1}\right) \tag{5.10}$$

$$\approx D_2\alpha^{n-1} \tag{5.11}$$

Where the last approximation is valid for large α and n. The parameters α and β can be evaluated from $D_3 = 7$, $D_4 = 11$, which gives $\alpha = 4/3$, $\beta = 5/3$, and the diameter as:

$$D_n^{(1)} = 9\left(\frac{4}{3}\right)^{n-2} - 5. \tag{5.12}$$

$$= O\left[\left(\frac{4}{3}\right)^{n-1}\right] \tag{5.13}$$

Repeating this process for $D_n^{(2)}$ one obtains $\alpha = -4$, $\beta = 7$, $\gamma = 11$ and the diameter as:

$$\begin{aligned} D_n^{(2)} &= \frac{1}{539\,2^{n+2}}\left(\left(2\left(\sqrt{11}-2\right)\right)^n\left(5907 + 1781\sqrt{11}\right) + \right. \\ &\quad \left.\left(\left(\sqrt{11}+2\right)-2\right)^n\left(5907 - 1781\sqrt{11}\right) - 5929\,2^{n+1}\right) \\ &\approx \frac{11}{2}\left(3^{n/4} - 1\right). \end{aligned} \tag{5.14}$$

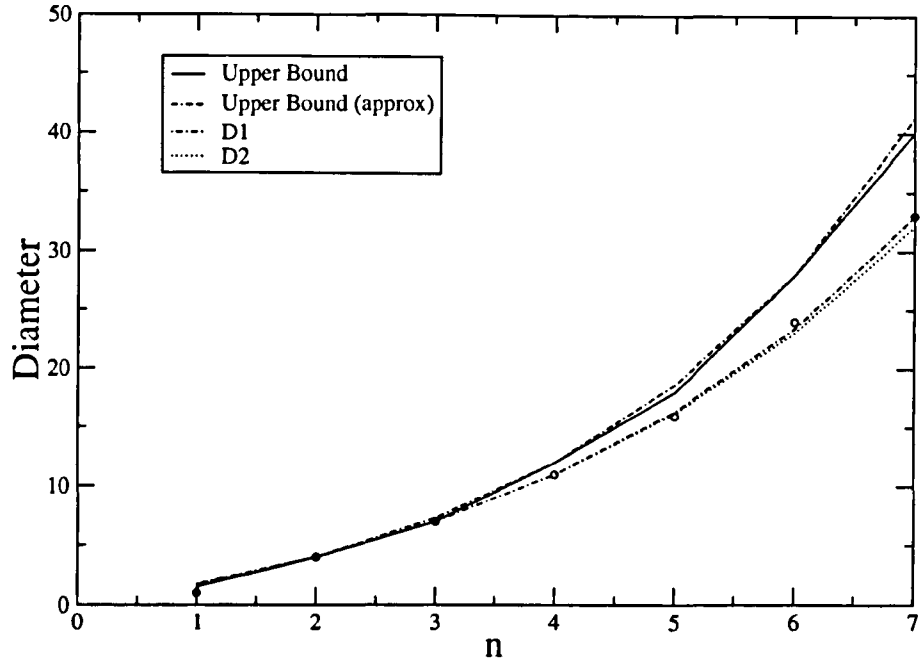


Figure 5.8.: Diameter of the closed Hilbert graphs H_n. The figure shows measurements taken directly from the graphs (circles), the upper bound (and its approximation) from eq. 5.7, and the numerical approximations $D_n^{(1)}$ (eq. 5.12) and $D_n^{(2)}$ (eq. 5.14, approximation). The figure shows results for graphs up to 16384 nodes (H_7), with a good fit of the data.

These are shown in figure 5.8, and it is clear that the actual diameter cannot be derived from any difference equation of order less than three.

It is possible to continue to determine ever more precise approximations, as long as enough measurements are accumulated. To determine the m-th approximation, $D_n^{(m)}$, $2m+1$ measurements are required: m to obtain the form of the function, and a further $m+1$ to estimate the corresponding parameters (α, β, γ, ...). Unfortunately to determine the diameter of a graph it is necessary to determine the shortest path between all pairs of nodes. The single source shortest path problem can be solved using the Dijkstra or Bellman-Ford shortest path algorithms. The worst running time for the most efficient implementation of the Dijkstra algorithm is $O(E+V\log V)$ [15,132], and it is required to run it V times (one for each source node), so the running time for the all-pairs shortest path algorithm is $O(VE\log V)$. The number of nodes in H_n is 4^n, so the complexity of the algorithm is $O(n4^{2n})$. Therefore to determine $D_n^{(m)}$ using this method, it is required to execute $O(m16^m)$ operations, which limits its usability to the first few m.

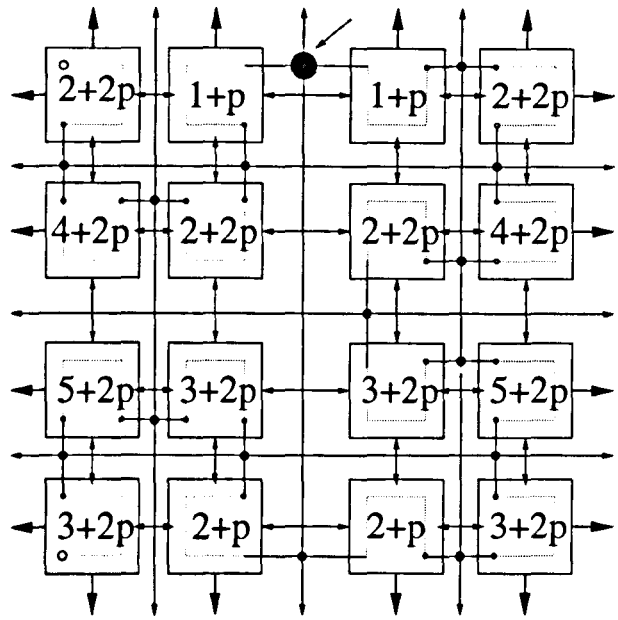


Figure 5.9.: Recursive construction used to obtain the average length of H_n.

5.2.3. Average Length

It is difficult to find an exact expression for the average length of a Hilbert Graph. Trivially the upper bound for the diameter (eq. 5.7) is also an upper bound for the average length, as the latter cannot be larger than the former.

We proceed to calculate the distance between the large node highlighted with a dashed arrow in figure 5.9, to all the nodes within the H_{n-2} open subgraphs in the recursive construction shown in the same figure. To do this, assume that $D^O_{n-2} \gg 1$, so effectively the goal is to minimise the number of H_{n-2} subgraphs for all paths.

The minimum distances are shown in the figure, and obtaining the average over all distances, it is possible to write:

$$L^O_n = \frac{7L^O_{n-2} + 11}{4}, \tag{5.15}$$

which has the solution:

$$L^O_n = \begin{cases} \frac{2552}{735}\left(\frac{\sqrt{7}}{2}\right)^n - \frac{11}{3} & \frac{n}{2} \in \mathbb{N} \\ \\ \frac{53}{6\sqrt{7}}\left(\frac{\sqrt{7}}{2}\right)^n - \frac{11}{3} & \frac{n+1}{2} \in \mathbb{N} \end{cases}. \tag{5.16}$$

This can be rewritten as:

n	N	D	$\sum_{H_n} l_{u,v}$	LH_n	D^O	$\sum_{H_n^O} l_{u,v}$	LH_n^O
1	4	1	12	1	1	12	1
2	16	4	522	2.175	4	506	2.4095
3	64	7	14762	3.6612	9	16380	4.1935
4	256	11	379458	5.8128	14	437066	6.7480
5	1024	16	9314098	8.8913	21	10639480	10.1764
6	4096	24	216854718	12.9287	30	244507642	14.5845
7	16384	33	4836402154	18.0181	42	5383471668	20.0587

Table 5.1.: Measured parameters for a H_n graph. The table shows the diameter, average length and sum of all lengths ($\sum_{H_n} l_{u,v}$) for graphs up to 16384 nodes (n=7).

$$L_n^O = \left(\frac{\sqrt{7}}{2}\right)^n \left(\frac{35728 + 13205\sqrt{7} + (-1)^n\left(35728 - 13205\sqrt{7}\right)}{20580}\right) - \frac{11}{3} \quad (5.17)$$

$$= \Theta\left[\left(\frac{\sqrt{7}}{2}\right)^n \left(\frac{35728 + 13205\sqrt{7}}{20580}\right) - \frac{11}{3}\right] \quad (5.18)$$

$$= O\left[\left(\frac{\sqrt{7}}{2}\right)^n\right] \quad (5.19)$$

The average length can also be approximated by using the same numerical method as described in section 5.2.2. The average length is given by $\frac{1}{N(N-1)}\sum_H l_{u,v}$, and it is obvious that $2\sum_H l_{u,v} \in \mathrm{N}$. The numerical data is shown in table 5.1 . For $L_n^{(1)}$ the solution has the same form as eq. 5.9 (we work with the sum and divide the result by $4^n(4^n-1)$ at the end of the process to determine the average length). For large n and α the solution can be approximated as $L_2\,\alpha^{n-2}$. Evaluating $\mathrm{L}_n^{(1)}$ one finds $\alpha = 45587/1780$, $\beta = 1239973/890$, and the average length is given as:

$$L_n^{(1)} = \frac{1}{4^n(4^n-1)}\left(\frac{2^{11}445^4}{43807\,45587^2}\left(\frac{45587}{1780}\right)^n - \frac{2479946}{43807}\right) \quad (5.20)$$

$$= O\left[\left(\frac{25.62}{16}\right)^n\right] \quad (5.21)$$

$$\approx \left(\frac{25}{16}\right)^n, \quad (5.22)$$

where the approximation is derived from $L_2\alpha^{n-1}/4^n$. For $L_n^{(2)}$ the following approximate solution can be found:

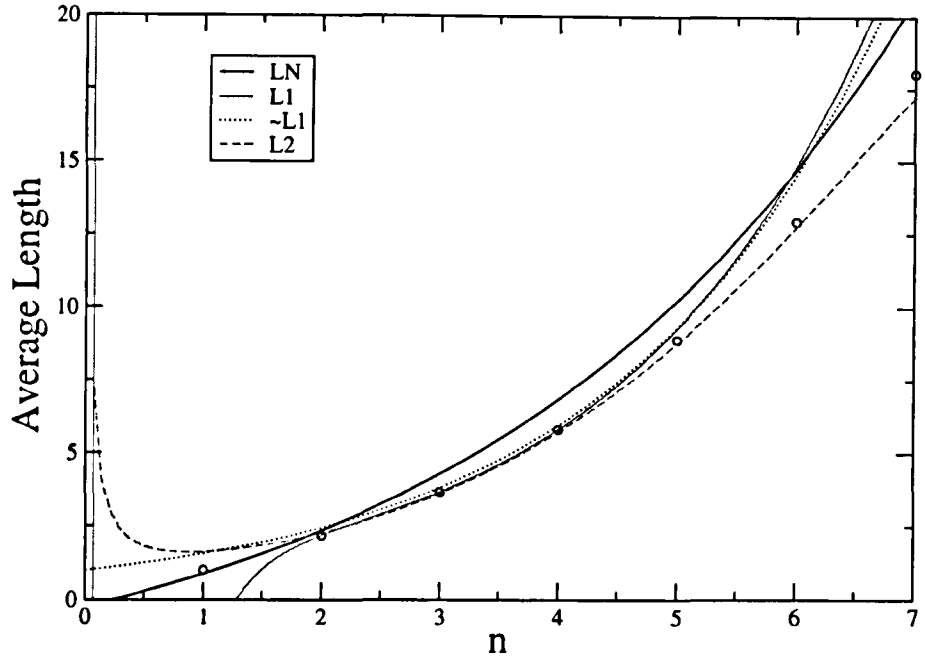


Figure 5.10.: Average length of Hilbert graphs. The figure shows experimental data (circles) against the average expressions obtained in the text where LN is the analytical average length determined in eq. 5.18, L1 and ~L1 are the numerical approximations of order 1 (and its approximation) shown in eq. 5.20 and eq. 5.22, and L2 is the second order approximation derived in eq. 5.23.

$$L_n^{(2)} \approx \frac{1}{4^n(4^n-1)}\left(\frac{2}{3}27^n\left[\cos\left(\frac{n}{5}\right)+\frac{1}{2}\sin\left(\frac{n}{5}\right)\right]\right) \tag{5.23}$$

$$= O\left[\left(\frac{27}{16}\right)^n\right]. \tag{5.24}$$

The previous expressions are plotted against measured values in figure 5.10.

5.3. Bisection Width

The bisection width of the open Hilbert graph can be established by noting that nodes in a Hilbert graph are placed in the middle of a section of a Hilbert curve (at $\lambda/2$, where λ is the length of the sections) and that additional horizontal or vertical extensions forming an extended mesh are connected to these nodes. Since every change of direction in the Hilbert curve introduces an offset of $\lambda/2$ to each coordinate, it follows that all nodes with horizontal extensions will have coordinates of the form

$(x\lambda + \lambda/2, y\lambda)$, and all nodes with vertical extensions will be at $(x\lambda, y\lambda + \lambda/2)$. Therefore all horizontal/vertical extensions will exist at evenly spaced λ units in the vertical/horizontal direction covering all the space. It follows that the bisection with the least number of extensions sectioned is obtained from a horizontal or a vertical cut, and since the minimum bisection width of the Hilbert curve is obtained when a horizontal cut through the middle of the graph is made, it follows that this is also the case for the Hilbert graph.

The number of edges through the bisection is given as:

$$BH_n^O = EV_{n-1} + EH_{n-1} + 2, \tag{5.25}$$

where EV_n is the number of external vertical extensions of a H_n graph, and EH_n is the same for the horizontal extensions which is given as:

$$\begin{aligned} EV_n &= 1 + 2EV_{n-1} \\ EH_n &= 1 + 2EH_{n-1} \\ EV_n = EH_n &= 2^n - 1, \end{aligned} \tag{5.26}$$

which gives the bisection width simply as:

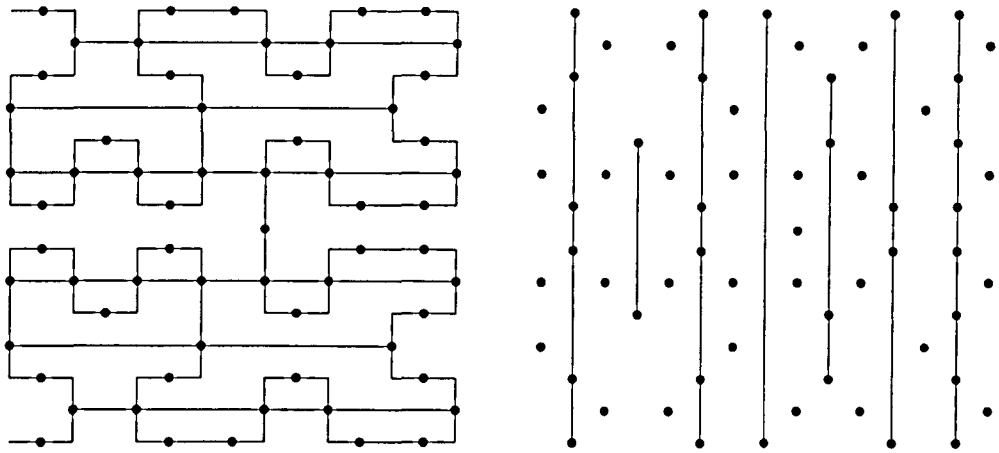
$$BH_n^O = 2^n. \tag{5.27}$$

In the closed Hilbert graph there are additionally EV_n extensions that loop back which need to be considered, as well as 3 new edges created by the insertion of the last node as shown in figure 5.3 which also needs to be considered, so the bisection width can be written as:

$$\begin{aligned} BH_n &= BH_n^O + EV_n + 3 \\ &= 2^{n+1} + 2. \end{aligned} \tag{5.28}$$

5.4. Scalability

The Hilbert graph provides an incrementally scalable network with a constant degree; and in principle there is no limit to the scalability of the network, as graphs of any size can be constructed (ideally the size of the graph should be of the form



(a) Layer 1 (b) Layer 2

Figure 5.11.: One of several possible 2D layered Hilbert graph implementations.

$N = 4^n$, but it is possible to construct incomplete Hilbert graphs which are not discussed in this text). Furthermore the Hilbert graph can be constructed in a simple two dimensional layered implementation as shown in figure 5.11, which makes it easy to construct them in a modular fashion. Locality of information can be exploited thanks to the presence of the ring interconnection backbone, and also because the hierarchical approach to its construction provides the means of constraining traffic to local clusters.

It is unavoidable to compare the Hilbert graph with the two dimensional torus and with the mesh, since they possess some of the same characteristics, such as the same fixed degree, a similar bisection bandwidth and a similar complexity for laying down in an essentially two dimensional medium, such as in VLSI. Although the torus and the mesh have the advantage of providing a simpler interconnection, the Hilbert graph makes use of the available wiring much more efficiently and exhibits a shorter average length as will be shown. Furthermore the Hilbert graph can exploit locality more efficiently since traffic can be confined at local clusters and the inter-cluster distance is small.

The torus network can be identified with the closed Hilbert graph, as both exhibit wrap-around connections; while the open Hilbert graph is similar to the mesh, since no wrap-around connections are present. To ease the comparison the two dimensional torus $\mathcal{T}_n$ and mesh $\mathcal{M}_n$ are defined, such that the number of nodes of both graphs is $N = 4^n$ (i.e. there are 2^n nodes per side). The bisection width is given as:

$$\begin{aligned} B\mathcal{T}_n &= 2^{n+1} \\ B\mathcal{M}_n &= 2^n, \end{aligned} \tag{5.29}$$

which are almost exactly the same as the corresponding expressions for the Hilbert graph. However the average length is given as:

$$\begin{aligned} L\mathcal{T}_n &= 2^n \\ L\mathcal{M} &= \frac{2}{3}\left(2^n - \frac{1}{2^n}\right), \end{aligned} \tag{5.30}$$

which grow as $O(2^n)$, which is larger than that of the Hilbert graph given as $O(2^{n/2})$. For this reason latency under light load and uniform traffic distribution will be reduced in a similar way in the Hilbert network. It is not clear however, that this will also be the case as traffic is increased, since the routing algorithm can introduce artificial bottlenecks in the network.

Although a routing algorithm has not been attempted in this work, we can proceed by providing a higher bound on the congestion of the network under uniform traffic by showing that there exists a non-minimal routing algorithm which exhibits the same contention (up to a constant factor) as any routing algorithm in a torus or mesh.

The recursive construction of the Hilbert graph implies that every H_0 subgraph has a direct connection with all of its horizontal and vertical neighbouring H_0 subgraphs (with wrap-around connections for the closed Hilbert graph), Figure 5.12 shows the connections between H_0 subgraphs in a Hilbert graph. If the four nodes of a H_0 subgraph are contracted into a single virtual node the result is a mesh (or a torus) network with an order equal to one quarter of the original Hilbert graph, where each node injects four time as much traffic into the network (again for a uniform traffic distribution). Several routing algorithms exist for the mesh and torus, which could be used in the contracted Hilbert network. Assuming that a routing algorithm is such that traffic is distributed evenly amongst the interconnections in the mesh (or torus), it is possible to estimate the average congestion in the networks. The congestion in a network with evenly distributed traffic is given by:

$$C = \frac{\mu N L}{E}. \tag{5.31}$$

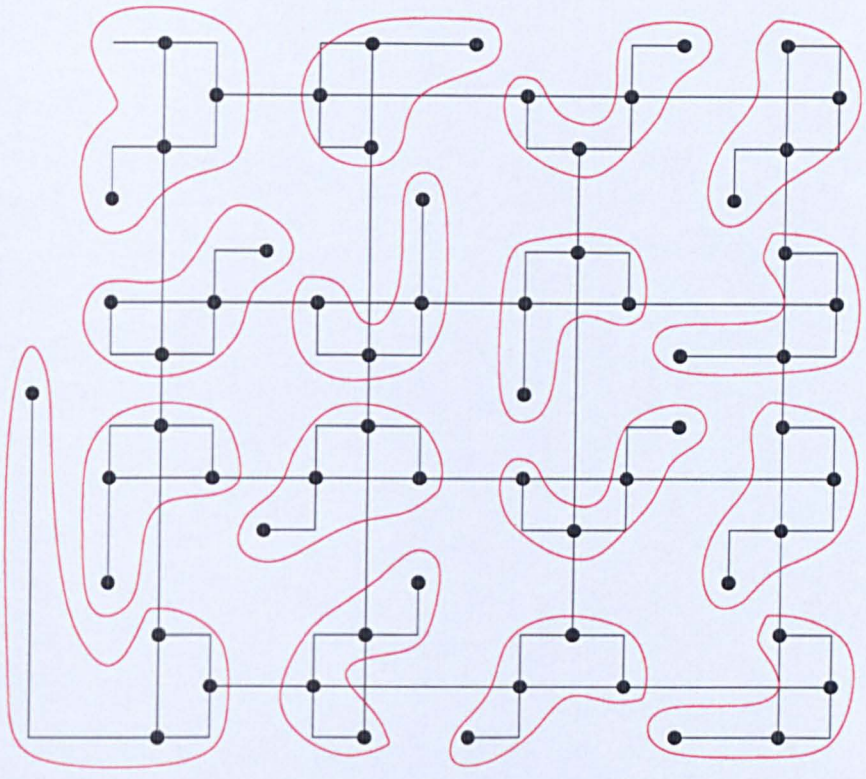


Figure 5.12.: Contracted Hilbert graph.

For a two dimensional mesh and torus the congestion can be found to be:

$$\begin{aligned} C_M &= \frac{4}{3}\mu N^{1/2} - O(1) \\ C_T &= \frac{\mu N^{1/2}}{2}. \end{aligned} \tag{5.32}$$

In both cases it follows that the congestion for the contracted Hilbert graph (where the number of nodes is one quarter of the original and traffic has increased four times) is exactly the double of these expressions and exhibits the same asymptotic behaviour. Note however that using this routing technique only a subset of edges is used; in particular since there are $N/4$ nodes in the contracted Hilbert graph, and the number of internal edges in each virtual node is four, it is easy to show that only half of the edges are used, and one would expect that the congestion figure would improve if all edges were used.

5.5. Conclusions and Summary of Results

The Hilbert graph has been proposed as an example of a fully deterministic interconnection network motivated by the small-world phenomenon and the multiple-scale hypothesis. These principles can be used to construct similar networks, where a variety of length scales are purposely introduced in order to provide a small diameter. At a local level the Hilbert graph is regular and simple, where locality of information can be exploited, as one can expect from a small-world graph; while providing a reduced world, where a small distance exists between different nodes.

The diameter and average length of the graph has been determined, and shown to be small (growing logarithmically with the number of nodes). Furthermore the graph is regular with a fixed and small degree. Since the degree is four, it is only natural to compare this graph against the torus and mesh, which also have a fixed degree of four. Furthermore the Hilbert graph is 2-planar (it can be laid down in two layers), which is similar to the torus and mesh. The bisection width of the Hilbert graph has been determined, and shown to be very similar to that of the mesh and torus. However the Hilbert interconnection makes a much better use of the available wiring, and since it provides a much reduced diameter, it exhibits a reduced latency (under light traffic) and it can exploit locality more efficiently. Due to its incremental expansion capabilities, the Hilbert graph provides the same ease of expansion as a mesh. We have shown that congestion exhibits (in the worst case) the same asymptotic behaviour as in a mesh or torus as the system is expanded; however we expect that the use of a purposely designed routing algorithm will provide a much better performance.

It is expected that other graphs constructed with the multiple-scale hypothesis, and the small-world phenomenon in mind will retain most of the good properties of the small-world interconnection networks analysed in this work, and will provide the same desirable properties as the Hilbert graph does.

6. Hypergraph Small-World Networks

6.1. Introduction

Hypermeshes have been shown to be good candidates for interconnection networks of parallel systems. Their main advantage is that they present a low diameter, high bandwidth, low latency network, which can naturally embed a wide range of communication patterns [46, 51, 146]. Furthermore, low dimensional hypermeshes lend themselves to natural implementation in space, where the wiring complexity is contained within the clusters running along each dimension.

Current technology seems to favour multi-package implementations where several computers are connected in a SMP/multiprocessor fashion in a single board or package. These packages are then linked through an interconnection network in a multicomputer fashion. For the remainder of this chapter we will consider the SMP/multiprocessor board as a single node, as we will concentrate solely on the multicomputer interconnection network design. Such multi-package implementations are pin-out rather than wire density limited [147], since the dominating cost is that of the I/O channels and connectors, and not of the interconnection complexity (this is especially true for the low dimensional hypermeshes, which contain all the wiring complexity along the clusters, which are naturally embedded in three dimensional space). Due to this reason networks in this chapter will be compared under the equivalence of a constant pin-out by noting that it is possible to increase the pin-out of a network by inserting additional channels running along the original channels (it makes no difference if the channels are serial or parallel). When the degree of both networks is made equal, the total bandwidth (the product of channel bandwidth and degree) of both networks will be equal. Therefore we will use total bandwidth as the equivalence between networks, comparing large pin-out structures with narrow channels, against low pin-out structures with wide channels.

Similar studies have been carried out between hypermeshes, meshes, torus, low

dimensional k-Ary n-Cubes (with and without bypass channels), and multi-stage interconnection networks, and showed that the hypermesh outperform the other networks when realistic decision times are accounted for [70,78,148,149].

In this chapter we introduce a number of small-world hypermeshes, which are incomplete hypermesh interconnections (i.e. not all nodes in a cluster are joined by a hyperedge); however other incomplete implementations are possible. We will study the networks under the assumption of restricted routing and wormhole switching. The structure chosen has a rich connectivity and low pin-out, in an attempt to enhance the performance of the network. As an additional benefit it provides reduced switch complexity and an increase in the maximum throughput.

6.1.1. Hypergraphs

Traditionally interconnection networks have been modelled as directed or undirected graphs, in which the vertices V are the nodes or processors, and the edges E are communication channels which connect the processors. Networks such as the k-ary n-cube, mesh, toroid, tree and ring are example of graph based networks. In a graph an edge always connects two vertices, and networks such as a simple bus cannot be constructed. This is a serious handicap, since a bus architecture can provide several benefits; including a small diameter and reduced costs. A hypergraph provides a more general model, in which several nodes can be connected together by a single edge (a *hyperedge*); and can therefore represent buses and any other interconnection structure.

Unfortunately hyperedges are susceptible to access contention, where transmissions from several nodes collide in the single channel. To alleviate this problem, the hyperedges can be organised in such a way that only one node is allowed to transmit in a given hyperedge. In this paper we are only interested in networks of this kind, which will be modelled as directed hypergraphs (we have assigned a direction to the links), and therefore connections based on multiple access buses will not be considered.

A directed hypergraph $\mathcal{H}(V, E)$ is formed by a set of vertices (nodes) V, and a set of hyperedges (wires, connections, or simply edges, when no confusion arises) E which link several nodes together [65]. Each hyperedge has a set of input $\iota(e)$, and output $\omega(e)$ nodes. We are only interested in networks with only one input per hyperedge, so we will fix $|\iota(e)| = 1$. The indegree of node v, $\delta_\iota(v)$ is the number of hyperedges whose set of output nodes include v; that is $\delta_\iota(v) = |\{e \in E : \omega(e) = v\}|$. The outdegree of a node is similarly defined as $\delta_\omega(v) = |\{e \in E : v \in \iota(e)\}|$. The

degree of a node, $\delta(v)$, is simply the sum of its indegree and outdegree, and the degree of a hypergraph is the maximum degree of any node; $\delta_H = \max\{\delta(v), v \in V\}$.

A path of length l is a list of hyperedges, $P_{uv} = \{e_1, e_2, \cdots, e_l\}$ such that $\forall 0 < i < l$, $\iota(e_{i+1}) \in \omega(e_i)$ and $\iota(e_1) = u$, $\omega(e_l) = v$ (which are called the source and the target respectively). The distance between two nodes is $d_{uv} = \min|P_{uv}|$. The diameter of a hypergraph is the length of the longest distance in the graph, $D = \max\{d_{uv}, u, v \in V\}$, and is an important measure since it is the largest number of hops a message will need to make to reach its destination. The average number of hops is given by the average length, which is defined as $L = \frac{1}{|V|(|V|-1)} \sum d_{u,v}$, where the summation is done over all pairs of nodes. The factor $|V| - 1$ accounts for the fact that nodes do not reference themselves.

Each channel entering a node has a bandwidth of W bits/sec. When comparing two different networks (say A and B), the constant pin-out argument states that the total bandwidth entering a node is the same for both networks, therefore:

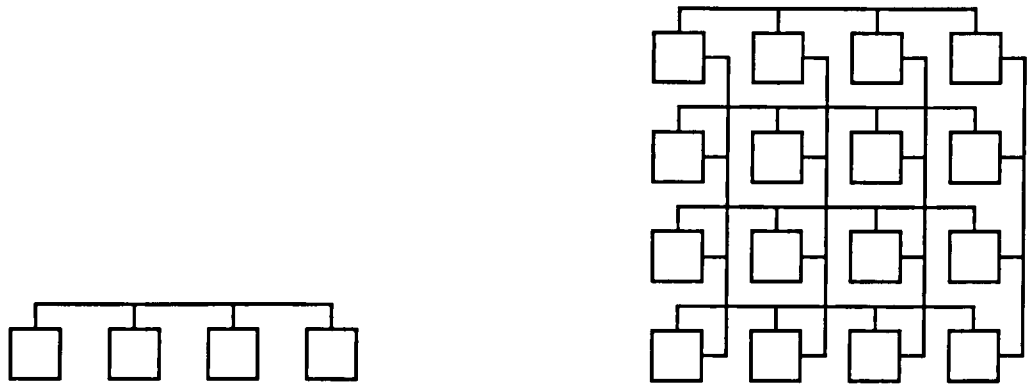
$$W_A \delta_A = W_B \delta_B. \tag{6.1}$$

In general W_A and W_B are technology dependent, so we are more interested in the quantity $\Delta W = W_A/W_B = \delta_B/\delta_A$. A packet of length B bits, will present an aspect ratio $\eta_B = B/W_B$ in network B; however the same packet will have an aspect ratio of $\eta_A = B/W_A = \eta_B/\Delta W$ in network A. If we assume $\Delta W > 1$, then it is clear that it will travel faster through network A than through network B, at least when no other traffic is present.

6.1.2. Hypermeshes

The hypermesh is a symmetric hypergraph network. Running along each dimension there are a number of *clusters*, which contain κ nodes in a linear array, along an interconnection network which links these nodes together. The Cartesian product of d clusters forms a d-dimensional hypermesh [46, 51].

Several implementations have been proposed which can be divided in two groups; those with centralised switching or shared buses (such as the crossbar switch hypermesh and the spanning bus hypercube) and those with distributed switching (such as distributed WDM [46] and DCSH [51]). Hypermeshes from the first group suffer from very poor performance under the constant pin-out constrain, because the centralised switch or bus forms a bottleneck which affects the rest of the system, and will not be considered any further (however they have the advantage of requiring shorter wires which can be driven at a higher speed than the longer and more com-



a) A Cluster

b) A 2-D Hypermesh

Figure 6.1.: Construction of a hypermesh. The 2-D hypermesh (b) is a 2-D Cartesian product of a cluster (a).

plex wires of the second group). The second group will be referred here as complete hypermeshes, since each node in a hyperedge is connected to every other node. We will compare these networks against the Hamming hypermesh.

Hypermeshes posses some very desirable characteristics. They have a low diameter, high bandwidth network, support for efficient broadcast operations, and since they can embed meshes, binary trees and hypercubes, applications that map naturally into these topologies will do so as well in the hypermesh [38,51,146].

In a hypermesh the total number of nodes is given as $N = \kappa^d$, and the average distance between any two nodes is:

$$L = \beta d \frac{\kappa - 1}{\kappa} \frac{N}{N - 1}, \tag{6.2}$$

where κ is the number of nodes in a cluster (hyperedge), and β is the inter-cluster distance ($\beta = 1$ for the complete hypermesh).

6.2. Hamming Hypermesh

The Hamming hypermesh is formed by the Cartesian product of the hamming cluster, which is constructed by joining nodes whose hamming distance is one. Formally, an α-hamming cluster is constructed by a set of $\kappa = 2^\alpha$ labelled nodes, $V = \{v_0, v_1, \cdots, v_{\kappa-1}\}$, and a set of κ labelled directed hyperedges (one for each node), $E = \{e_0, e_1, \cdots, e_{\kappa-1}\}$, such that $\iota(e_x) = \{v_x\}$, and $\omega(e_x) = \{v_y \in V : \mathcal{H}_{x,y} = 1 \vee \mathcal{H}_{\kappa-x,y} = 1\}$, where $\mathcal{H}_{x,y}$ is the hamming distance between x and y. In-

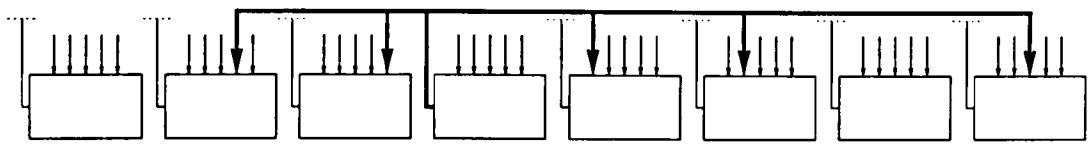


Figure 6.2.: A 3-Hamming cluster. For clarity, only one hyperedge is shown in the drawings. The rest of the hyperedges in the Hamming cluster are equivalent.

formally, we can say that the cluster is constructed by joining nodes whose modulus κ absolute distance is a power of 2 ($2^i = \{1, 2, 4, 8, \cdots, \kappa/2\}$). A hamming cluster is shown in fig. 6.2. The indegree of this hypergraph is given as $\delta_\iota = 2\alpha - 1$, and the total degree is $\delta = 2\alpha$.

The d dimensional Cartesian product of an α-Hamming cluster defines an $H_{d,\alpha}$ hamming hypermesh, with $N = 2^{\alpha d}$ nodes. The degree of such a graph is (considering all dimensions and output channels):

$$\delta_H = 2d\alpha = 2\log_2 N. \tag{6.3}$$

This is substantially less than the complete hypermesh, whose total degree is $\delta_C = d\sqrt[d]{N}$, and allows the hamming hypermesh to have much wider channels $\Delta W_{HC} = \frac{d\sqrt[d]{N}}{2\log_2 N} = O\left(\sqrt[d]{N}\right)$, and an equal static throughput increase (the maximum transfer rate under no load, see fig. 6.9(B)).

Of course the main drawback is a reduced connectivity; messages need to traverse several channels before reaching their destination within the same cluster. However as we will show, this deficiency is more than compensated by the wide channels.

6.2.1. Diameter and average length

To calculate the diameter and average length of this hypergraph we start by constructing an equivalent graph (Hamming graph) G such that $V_G = V_H$, and $E_G = \{e_{uv} : v \in \omega_H(f) \wedge u = \iota_H(f), f \in E_H\}$. Note that this construction does not require a directed graph, due to the symmetric nature of the Hamming cluster. An example of this graph is shown in figure 6.3. Next we proceed to label the edges going out of a given node, such that the edge labelled α_i spans 2^{α_i} nodes, as shown in the figure. Note that there are some repeated edges with such labelling (i.e there are two $\alpha - 2$ edges). When necessary they will be referred to as α_i Left and α_i Right to avoid confusions.

We are interested in computing the average number of links traversed by an aver-

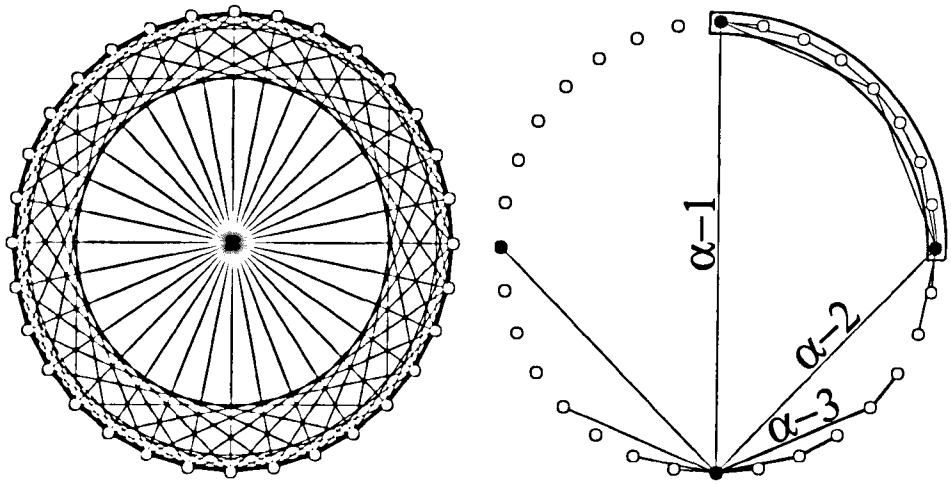


Figure 6.3.: Example of an equivalent Hamming graph (left), and edge labelling (right).

age packet. We group the different routing possibilities by allowing certain edges to be traversed only at some given time. Note however that this timing will most likely be different to the actual timing experienced by a packet. We start by supposing node v_0 sends a packet and considering the routing possibilities at the initial time, t_0. At this time the packet may wish to travel through edges $\alpha - 1$ or $\alpha - 2$. Of course, it could also travel through other edges, but we consider this at the next time step. Thus, at the beginning of t_1 the packet may be located at nodes $v_{2^{\alpha-1}}$, $v_{2^{\alpha-2}}$, $v_{\kappa-2^{\alpha-2}}$ or v_0. These nodes define four sets as shown in the figure. Each set has $1 + \kappa/4$ nodes, since each of the dividing nodes is in two sets. However for routing purposes the two dividing nodes are collapsed into a single node, since the routing decision was taken at a previous time step (and this was an optimal decision). Therefore each set is logically equivalent to the Hamming subgraph $G_{\alpha-2}$, and the sum of all distances can be written as:

$$S_{\alpha-1} = 2^{\alpha} - 1 - 2C_{\alpha-3} + 4S_{\alpha-3}, \tag{6.4}$$

where $2C_{\alpha-3}$ is the number of destination nodes that do not require to route at t_0 (that is the nodes that can be reached from v_0 by the edges $\alpha - 3$, $\alpha - 4$, etc.), and is given by:

$$C_{\alpha} = \begin{cases} \sum_{i=0}^{\alpha/2} 2^{\alpha-2i} & \alpha \rightarrow even \\ \\ \sum_{i=0}^{(\alpha-1)/2} 2^{\alpha-2i} & \alpha \rightarrow odd \end{cases} \tag{6.5}$$

$$= \frac{2^{\alpha+2}}{3} - \frac{3-(-1)^{\alpha}}{6}. \tag{6.6}$$

The solution of the difference equation 6.4, with its proper initial conditions ($S_0 = 1$, $S_1 = 3$) is:

$$S_{\alpha} = \frac{\alpha 2^{\alpha+1}}{3} + \frac{2^{\alpha+3}+(-1)^{\alpha}}{9}, \tag{6.7}$$

and the average length is:

$$\beta_H = \frac{S_{\alpha-1}}{\kappa-1} = \frac{\kappa}{\kappa-1}\left(\frac{\alpha}{3} + \frac{1-\left(-\frac{1}{2}\right)^{\alpha}}{9}\right). \tag{6.8}$$

From the preceding argument, it is obvious that the diameter is simply t, the number of recursive graphs traversed. At time t_0 we have α edges, and at each step two edges are removed. Therefore the diameter is given as:

$$D_H = \begin{cases} \frac{\alpha}{2} & \alpha \rightarrow even \\ \\ \frac{\alpha+1}{2} & \alpha \rightarrow odd \end{cases}. \tag{6.9}$$

6.2.2. Routing

Routing in a complete hypermesh is quite simple. Packets are routed in strict dimensional order to avoid deadlocks (also known as e-cube routing). Since the Hamming cluster is not fully connected, routing in a Hamming hypermesh is more complicated, as messages might need to experience multiple hops within a cluster before being routed to other dimensions. In this section we introduce an optimal deadlock and livelock free routing algorithm for the Hamming hypermesh.

It is well known that dimensional ordered routing in a hypermesh is deadlock free. Therefore we only require to provide a intra-cluster routing algorithm, since messages can switch between clusters in strict dimensional order in an optimal and deadlock-free fashion. Several intra-cluster routing algorithms are possible. Here we present a distributed routing algorithm in which each node only needs to know the current and the destination node to make a routing decision. This class of routing algorithm is known as $N \times N \rightarrow C$, and can be easily implemented in hardware (either as a lookup table or using logic circuits) [133,134].

The routing algorithm should provide deadlock (and livelock) avoidance. Deadlock occurs when messages are blocked from advancing because channel buffers are full, and a circular dependence exists between those packets. A simple approach to

avoid deadlocks consists on restricting routing so that there are no cyclic dependencies between channels [133]. Livelock occurs when new messages are forbidden to enter the network. This occurs when existing packets do not advance any further towards their destination, but are continually deflected. The routing algorithms considered here are livelock free since packets will always advance closer to their destination with each routing decision.

Packets switching within a Hamming cluster will do so following an acyclic path R. We will assume that R is a minimal path; that is the length of R is as small as possible. This supposition will turn out to be true for the routing algorithm chosen. The following theorems follow:

Theorem 6.2.1 *The path R has maximum length α and traverses an edge spanning 2^β nodes at most one time.*

Proof We will prove the second part of the theorem first. Suppose there are two or more edges spanning 2^β nodes in R. Two of these edges can be replaced by a single edge spanning $2^{\beta+1}$ nodes. However this contradicts the initial assumption that R is optimum, since the new path is shorter, and therefore an edge spanning 2^β nodes is traversed at most one time in R. Since the edges of the Hamming cluster span 2^i nodes where $i \in \{0, 1, ..., \alpha - 1\}$, and by the first part of this theorem, it follows that the maximal length of R is α. ∎

Theorem 6.2.2 *Any routing algorithm for a Hamming cluster which assigns edges to all R such that $R = \{e_1, e_2, ..., e_n\}$ with $|e_1| > |e_2| > ... > |e_n|$ or with $|e_1| < |e_2| < ... < |e_n|$, where $|e|$ is the number of nodes spanned by edge e, is deadlock free.*

Proof Construct an α dimensional space where all edges with $|e| = i$ are mapped at dimension i. In such space deadlock is not possible, since all paths traverse the dimensions in strict increasing or decreasing order. ∎

Corollary 6.2.3 *Any distributed routing algorithm for a Hamming cluster is deadlock free if for a packet arriving via channel e_j it assigns the outgoing channel e_i where $i < j$, and the local edge labelling is such that e_k spans 2^k nodes (from the current node).*

6.2.2.1. Routing Algorithm

From the previous set of theorems it is simple to construct a deadlock free distributed routing algorithm. Referring to fig. 6.3, it is clear that packets travelling in an

Target	Channel	Target	Channel
0	local	8	3
1	0R	9	3
2	1R	10	3
3	2R	11	2L
4	2R	12	2L
5	2R	13	2L
6	3	14	1L
7	3	15	0L

Table 6.1.: Routing table for a 16 node Hamming Cluster.

optimal path through edge $\alpha - n$ (right) are destined to a node within the interval (all labels are relative to the current node):

$$\begin{aligned} R'_{\alpha-n} &= [2^{\alpha-n} - 2^{\alpha-n-2} - 2^{\alpha-n-4} - ..., \\ &\quad 2^{\alpha-n} + 2^{\alpha-n-2} + 2^{\alpha-n-4} + ...], \end{aligned} \tag{6.10}$$

and packets travelling through edge $\alpha - n$ (left) are destined to nodes in the interval:

$$\begin{aligned} R''_{\alpha-n} &= [2^{\alpha} - 2^{\alpha-n} - 2^{\alpha-n-2} - 2^{\alpha-n-4} - ..., \\ &\quad 2^{\alpha} - 2^{\alpha-n} + 2^{\alpha-n-2} + 2^{\alpha-n-4} + ...]. \end{aligned} \tag{6.11}$$

With this intervals it is possible to construct a routing table for the switches. As an example a routing table for a 16 node Hamming cluster is shown in table 6.1. Since the table is constructed from an optimal path, and assigns routes as required by corollary 6.2.3, it provides deadlock free routing. A complete routing algorithm (which may be modified to use the lookup table) is as follows:

6.2.3. Hardware Complexity

We are interested in providing a means of comparing the intrinsic cost of different interconnection networks. Traditionally the bisection width and node degree have

Distributed routing algorithm.
D=Current dimension of router
C=Distance between destination and current node at dimension D
#First route within a cluster#
if (C $\neq$ 0) then #intra-cluster routing required#
  for (i=$\alpha - 1$ downto 0)
    if $C \in R'_{\alpha-n}$ route to $\alpha - n$ (right) of dim D
    else if $C \in R''_{\alpha-n}$ route to $\alpha - n$(left) of dim D
  endfor
else
if (D-1=0) Consume packet
else Send packet to router at dim D-1

been used for this objective. The bisection width metric is adequate when the network has to be constructed in an essentially two dimensional medium, such as VLSI. However for a network constructed in three dimensional space it is not a very appropriate metric. For networks constructed with discrete packages, the cost of wires and channels is a fraction of the cost of the transceivers, couplers, modulators, switches, packaging and integration, at least for current technologies. For this reason we argue that pin-out is a much better metric of complexity, since it reflects the costs of the internal switches, couplers and other expensive equipment.

Since we are maintaining constant the pin-out of the networks, under this metric all of them have the same cost (we are making equal the cost of the transceivers, couplers, modulators and other equipment associated with the input and output stages). However it can be argued that the switch complexity is not the same for all graphs, since the complexity of a switch is more a function of the number of channels and less dependent of the channel width. The number of internal routes grows as the square of the former (here we are only considering direct connection non-blocking switches, although it is possible to construct indirect switches with $O(\kappa \log^2 \kappa)$ internal routes, at an additional latency cost [110]). This observation leads to an alternative measure of complexity which is useful for cases where the cost of the switch is a significant factor in the total cost.

6.2.3.1. Switch Complexity

As we have stated, the complexity of a switch is proportional to the number of internal routes $X = O(\delta^2)$, where δ is the degree of the cluster. Since each route has W wires, the switch complexity will be measured as $S = d W \delta^2$. Figure 6.4 and table 6.2 shows the switch complexity, and the complexity reduction for the complete

Network	SW. Complexity	Reduction
Complete	$d\kappa^2$	-
Hamming	$2d\kappa \log_2 \kappa$	$O(\kappa)$

Table 6.2.: Switch complexity.

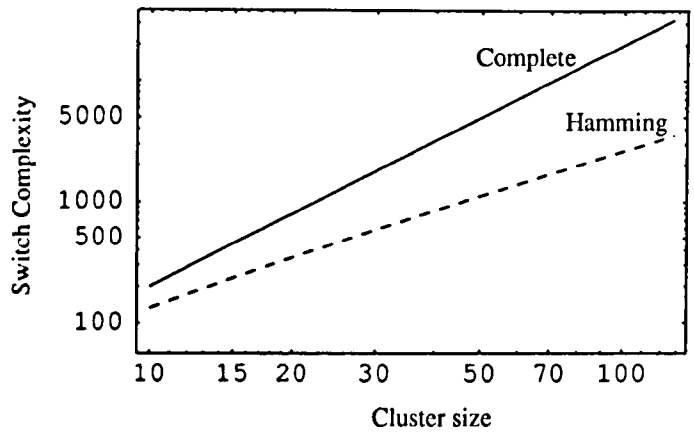


Figure 6.4.: Switch complexity for the complete and the Hamming cluster.

$\left(\delta = \kappa = \sqrt[d]{N}\right)$ and the Hamming ($\delta = 2\log_2 \kappa$) hypermeshes. Note the linear reduction in complexity for the incomplete network (with respect to the cluster size). If we were to change our constant pin-out argument for a constant switch complexity argument, the incomplete networks would exhibit an even better performance.

6.2.4. Stochastic Performance Model

6.2.4.1. Latency

Latency is a very important metric for interconnection networks, especially for networks designed for parallel computing, since it limits the speed of fine grain computations. Here we develop a model to study latency in wormhole switched hypermesh networks (using restricted routing). To simplify the analysis, the following assumptions are made (most of which have been used in similar studies [46, 78, 137, 138]):

1. Each node generates a message by a Poisson process of rate μ packets per cycle. The destination for each packet is chosen uniformly and independently amongst all possible recipients (uniform traffic).

2. Packets are distributed evenly through all edges belonging to a given class in the network.

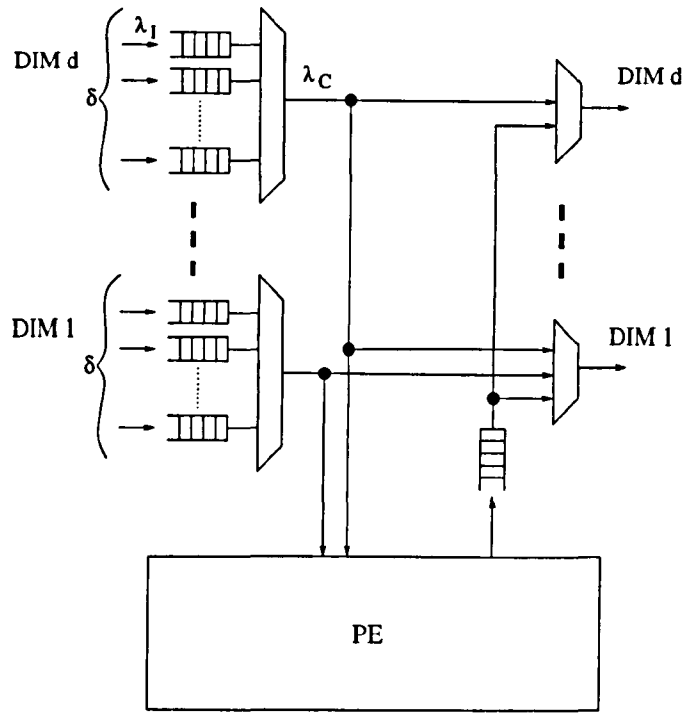


Figure 6.5.: Switch structure used for the complete, random and Hamming hypermesh.

3. Packets visit dimensions in order; i.e. once a packet switches from dimension $i+1$ to i, any further switching will be done only to dimensions $i, i-1, .., 0$ (the fictitious dimension 0 is the input to the local processing element).

4. All messages generated are of equal length of B phits.

5. One phit per dimension can be sinked in a cycle by the Processing Element (multiple acceptance model); however only one phit can be sourced. All information interchange amongst nodes is done synchronously at each clock cycle.

6. Queues are of finite size and can hold only a few number of flits.

7. Messages are routed along the shortest path.

8. The network uses wormhole switching and the switch shown in figure 6.5.

This analysis is similar to that in [20,51]; however unlike [51] we need to account for intra-dimensional routing, since packets can travel several nodes within a dimension before switching to a lower dimension; and although the model in [20] does take into account intra-dimensional routing, it needs to be extended to include the effects of contention from this traffic arriving at different network channels and competing for access at each successive node, and also to include the effects of the decision time.

We will develop a model for latency based on the intra-cluster distance β, the packet size B (measured in phits), the channel width W (which is equal to one phit), the number of nodes N, the dimension of the network d and the decision time T_{dec}. Where we compare networks with different channel widths, we will measure the packet size as well as the channel size in phits referred to the complete network, denoted as *phits[C]*. In this way we can draw general conclusions without requiring any further technological assumptions.

6.2.4.2. Static Latency

We start by computing static latency (i.e. latency under no load conditions). In this case latency is determined by two factors; the time required for a B phit packet to go through the switch-processor channel, and the time required for a phit to travel through the network. The former is given by:

$$T_{node} = \eta + T_{dec}, \tag{6.12}$$

(since packets are serviced as soon as they are received), and the latter is:

$$T_{net} = L\,T_{dec}, \tag{6.13}$$

where $\eta = B/W$ is the packet aspect ratio, and L is the average length between any two nodes, given by eq. 6.2. Therefore in this case latency is given by:

$$Latency_{(static)} = T_{node} + T_{net}. \tag{6.14}$$

This is an adequate approximation for the cases where the network traffic consists of short bursts of activity, with interleaved long periods of inactivity. However when the traffic in the network increases, queueing and contention will severely degrade its performance and this model is no longer valid.

6.2.4.3. Dynamic Latency

This is a more general model than static latency, since we make no assumptions on the traffic level μ. When $\mu \to 0$ latency is equal to the static case; however when the traffic is increased so is latency. To compute the latency in this case, we will begin at the output node and work our computation backwards to the point where the packet was injected into the network. Latency at the output node (dimension 0) is the same as in the previous case, and is given as:

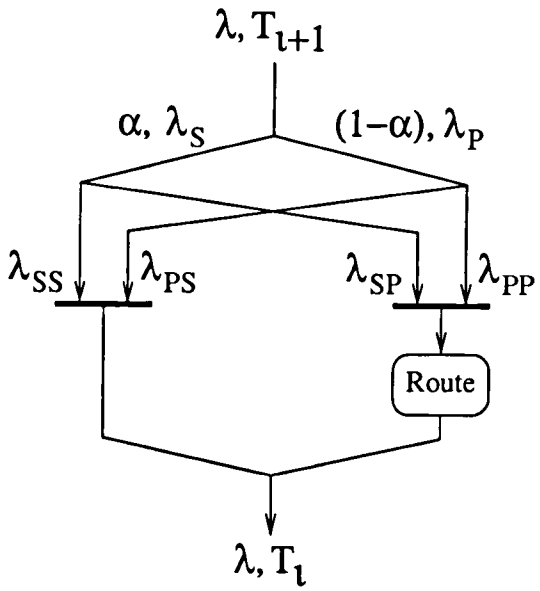


Figure 6.6.: Latency in a dimension.

$$T_0 = \eta + T_{dec}. \tag{6.15}$$

To find out the latency entering the previous dimension, we will consider a single dimension i as shown in fig. 6.6. Packets entering dimension i have two components; those who passed through the previous dimension to correct their spatial coordinates (λ_P), and a fraction $\alpha = N^{-1/d}$ which were already at the correct coordinate and skipped from routing in that dimension (λ_S). These streams are further divided according to packets that will route through or skip dimension i, as shown in the figure (the first letter in the subscript is for dimension $i+1$, while the second one is for dimension i).

Latency due to contention where different streams compete for access, can be computed for each stream, by multiplying the probability of a collision with the expected delay due to the collision. For example latency seen by component λ_{PS} is given by the multiplication of the collision probability $\lambda_{SS} T_i$, with the expected delay $T_i/2$ (the collision delay has a uniform distribution from 0 to T_i).

All messages increase their latency when switching to a lower dimension, because they have to compete for access to the new dimension. However packages needing to pass through the dimension will gain an additional overhead due to contention, T_{Ri}. Since the average number of nodes traversed through a dimension is β, packets will go through one entering channel, and $\sigma = \beta - 1$ continuing channels (see fig. 6.7).

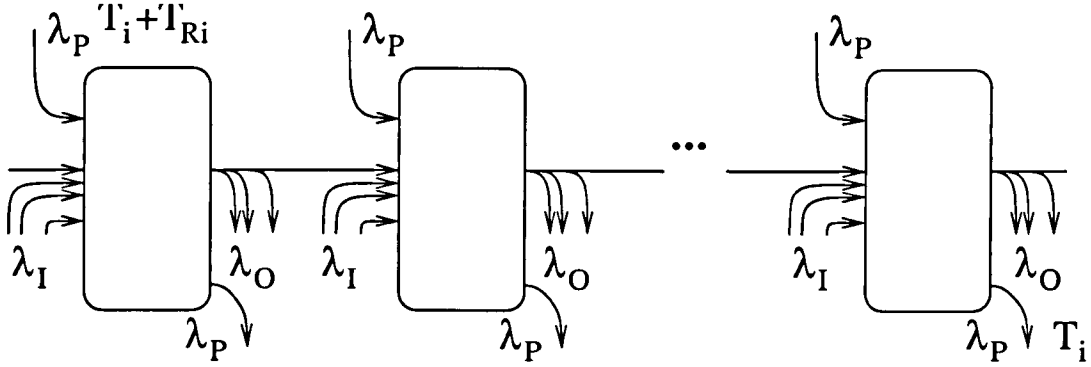


Figure 6.7.: Routing within a dimension.

The traffic rate at each continuing node is λ_c, therefore at each input a packet has to compete for access with $\delta - 1$ other inputs with traffic rate λ_c/δ (denoted in the figure as λ_I), and with one input of rate λ_P. Traffic λ_I and λ_O is traffic that is also routing within the dimension, but follows different routes. This traffic has to compete for access at each continuing node. Since at each node $1/\beta$ packets route to a lower dimension, the traffic at each continuing channel is $\lambda_c = \sigma\lambda_p$.

To compute T_{Ri} we will work out from the output and propagate the computation towards the input. The service time at the last continuing channel is T_i. The additional time due to contention for the rest of the channels is given as:

$$T_{i(j-1)} = T_{ij} + \frac{T_{i0}^2}{2}\left(\lambda_p + \frac{\lambda_c(\delta-1)}{\delta}\right) + T_{dec}, \tag{6.16}$$

where $T_{i0}^2/2$ is the average collision cost, and the term in parenthesis is the collision probability. Repeating this calculation for the $\beta - 1$ continuing channels gives:

$$T_{i0} = T_i + \sigma T_{dec} + \sigma\frac{T_{i0}^2}{2}\left(\lambda_p + \frac{\lambda_c(\delta-1)}{\delta}\right). \tag{6.17}$$

The solution to this equation is given as (ignoring the unrealistic larger solution):

$$T_{i0} = \begin{cases} T_i, & \beta = 1 \\ \frac{1-\sqrt{1-2\sigma\left(\lambda_p+\frac{\lambda_c(\delta-1)}{\delta}\right)(T_i+\sigma T_{dec})}}{\sigma\left(\lambda_p+\frac{\lambda_c(\delta-1)}{\delta}\right)}, & \beta > 1 \end{cases} \tag{6.18}$$

The possibility of a collision in the input channel also needs to be considered in the computation of T_{Ri}:

$$T_{Ri} = T_{i0}\left(1 + \frac{T_{i0}}{2}\left(\lambda_p + \frac{\lambda_c(\delta-1)}{\delta}\right)\right) - T_i. \tag{6.19}$$

Taking all these components into account, the latency seen entering dimension $i+1$ is given by:

$$T_{i+1} = T_i + T_{dec} + (1-\alpha)T_{Ri} + \alpha^3(1-\alpha)\lambda T_i^2 + \alpha(1-\alpha)^3\lambda(T_i + T_{Ri})^2. \tag{6.20}$$

If enough queueing is provided the service time remains constant, and the previous equation can be rewritten as [20]:

$$T_{i+1} = T_i + T_{dec} + (1-\alpha)T_{Ri} + (\alpha^3(1-\alpha) + \alpha(1-\alpha)^3)\lambda T_0. \tag{6.21}$$

Finally the queueing time at the source node has to be included, to give the total latency this can be modelled as an M/M/1 queue with an average service time T_d [51,137]:

$$Latency = \frac{T_d}{1-\mu T_d}. \tag{6.22}$$

A computer simulator was written and used to validate this model. The simulator has been integrated into the INADS system described in appendix B, and has been written in around 2000 lines of C++. The simulator is compatible with the assumptions used to derive the model. In particular the following assumptions and specifications have been met:

- The switching method is wormhole routing, and uses the switch shown in figure 6.5.
- A user selectable number of flits can be stored at the input queues. For the simulations a queue size of three flits has been used.
- Messages are generated by an independent Poisson process. The destination for each packet is selected randomly from all of the possible destinations.
- All messages generated are of a fixed length.
- Messages are delayed by a decision time at each switch they visit. The time can be specified by the user.

- Routing is restricted and follows the shortest path between source and destination.
- Simulations are run until the collected statistics converge to a final value with a given accuracy.

The results can be seen in fig. 6.8, for the complete, Hamming and random hypermesh (described in the next section).

6.2.4.4. Throughput

The throughput of a network is the average number of packets which reach their destination within a cycle. Since this measure is dependent on the the network size, perhaps a better measure is the throughput per node, which is simply the total throughput divided by the number of nodes.

When the network is not loaded, no contention occurs, and the throughput per node (or simply throughput from now on) is given by the traffic coming out of a node:

$$\Theta = \mu B. \tag{6.23}$$

This is also the case for all traffic levels below saturation. However, when the injection rate increases above the saturation point, throughput will remain constant, or fall, depending on the network design. The saturation point can be found by setting the injection node service time equal to the inverse of the injection rate, and solving the equation for m_{sat}:

$$\mu_{sat} = \frac{1}{T_d(\mu = \mu_{sat})}. \tag{6.24}$$

At this point of operation the network is ill behaved; latency becomes infinite and queues are marginally stable. However the maximum throughput is achieved at this point and is useful for theoretical considerations:

$$\Theta_{max} = \mu_{sat} B. \tag{6.25}$$

6.2.5. Results

We focused on the two and three dimensional hypermeshes because they map naturally into space [70]. Two different packet sizes were used; 32 and 128 phits (a phit is a word of size equal to the network channel). This packet sizes were chosen

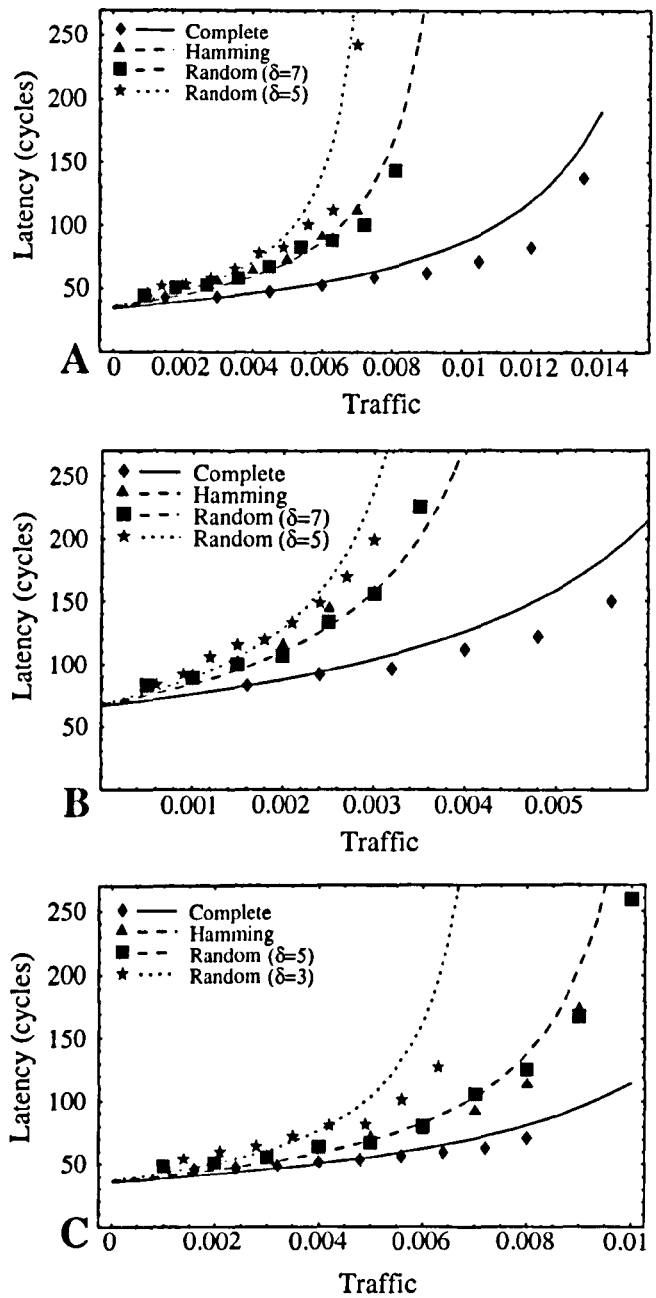


Figure 6.8.: Latency model validation for different networks. (A) N=256, d=2, B=32. (B) N=256, n=2, B=64. (C) N=512, d=3, B=32. The decision time is one cycle for all networks.

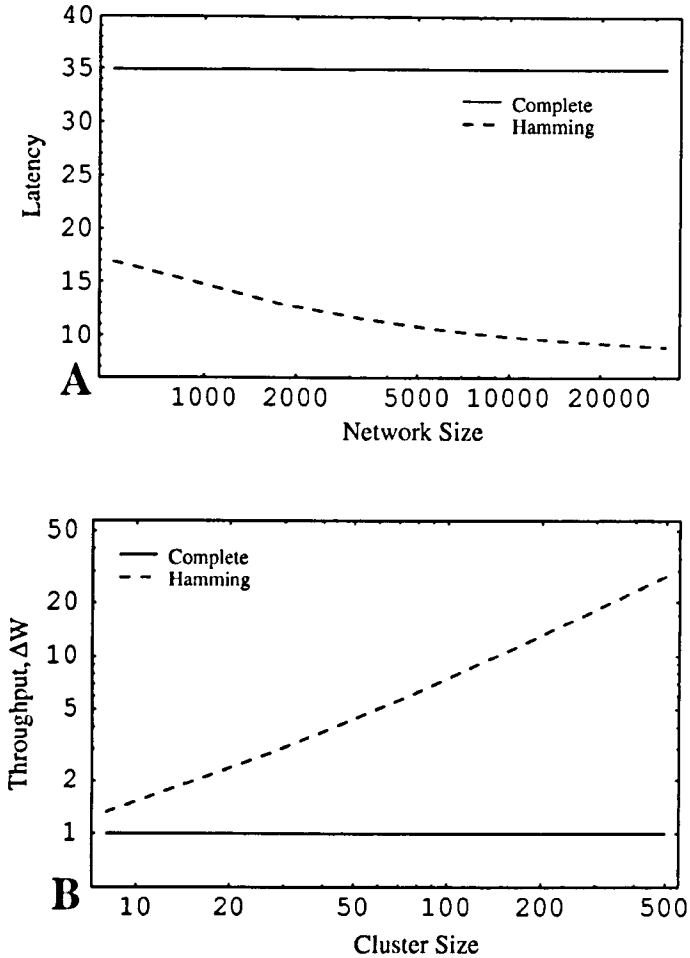


Figure 6.9.: Static performance for different networks with a 32 phit[C] packet size and a decision time of one cycle. In (A) a plot of static latency against network size for two dimensional networks is shown, and fig. (B) shows the normalised maximum static throughput (which is equal to the normalised channel width increase, ΔW).

to represent a short and a normal size packets respectively, and have been used in similar studies [20, 70].

6.2.5.1. Static Performance

The static performance of the network characterises it under a no load condition (no other traffic, or almost no other traffic exists in the network, except the one we are deliberately introducing). This is a useful metric for cases where the traffic is low or intermittent, or when a pair of nodes generate most of the traffic in the network (such as in *hot-spot* traffic).

The latency of a network under no load, is determined by eq. 6.14. Figure 6.9(A)

shows static latency for two dimensional networks with a 32 phit[C] packet size. Latency can also be seen in figure 6.10 when the traffic level approaches zero.

Note that latency in the incomplete networks actually lowers when the network size is increased. This is due to the increase in channel width ΔW for larger networks. As it is shown, the incomplete network exhibit a lower static latency. This is to be expected, since incomplete hypermeshes have wider channels and a smaller message aspect ratio. The situation is even better when the packet size is increased (not shown in the figure).

The maximum static throughput is simply the bandwidth coming out of a channel (W), as fig. 6.9(B) shows. As seen in the figure static throughput increases for incomplete networks with the network size. This is due to their wider channels which are a result of the constant pin-out argument. Note that the increase is almost linear for the Hamming hypermesh. However for the optimised random hypermesh the throughput increase is not linear. This is particularly true for large clusters, and is due to the fact that at this operation region the aspect ratio of the packets is very small, and the networks cannot exploit their wide networks to their advantage. For this reason the optimised random hypermesh reacts by trading channel width for a better connectivity.

6.2.5.2. Dynamic Performance

Latency The dynamic latency of different network configurations was evaluated using the model developed in section 6.2.4, and the results are shown in fig. 6.10.

Figure 6.10(A,B) shows the comparison between a small two dimensional complete hypermesh (N=256) and its incomplete counterpart. As it is shown the Hamming hypermesh outperform the complete one under all conditions, and have the additional benefit of providing much wider channels (which leads to a much higher throughput; see fig. 6.17).

A much larger two dimensional network is shown in figures 6.10(C,D), where the network size has increased to 4096 nodes; but perhaps more importantly, the cluster size is $\kappa = 64$. This large cluster size forces the Hamming hypermeshes to have very wide channels, since their pin-outs are much lower than δ_C. Unfortunately this increases the inter-cluster travelling distance, β. Packets need to make a large number of hops within a cluster, which can lead to a large latency and saturation, as they remain a long time in the network.

As it is shown in the figure, when the packet size is large incomplete hypermeshes outperform complete ones by a large margin. However when the packet size is small

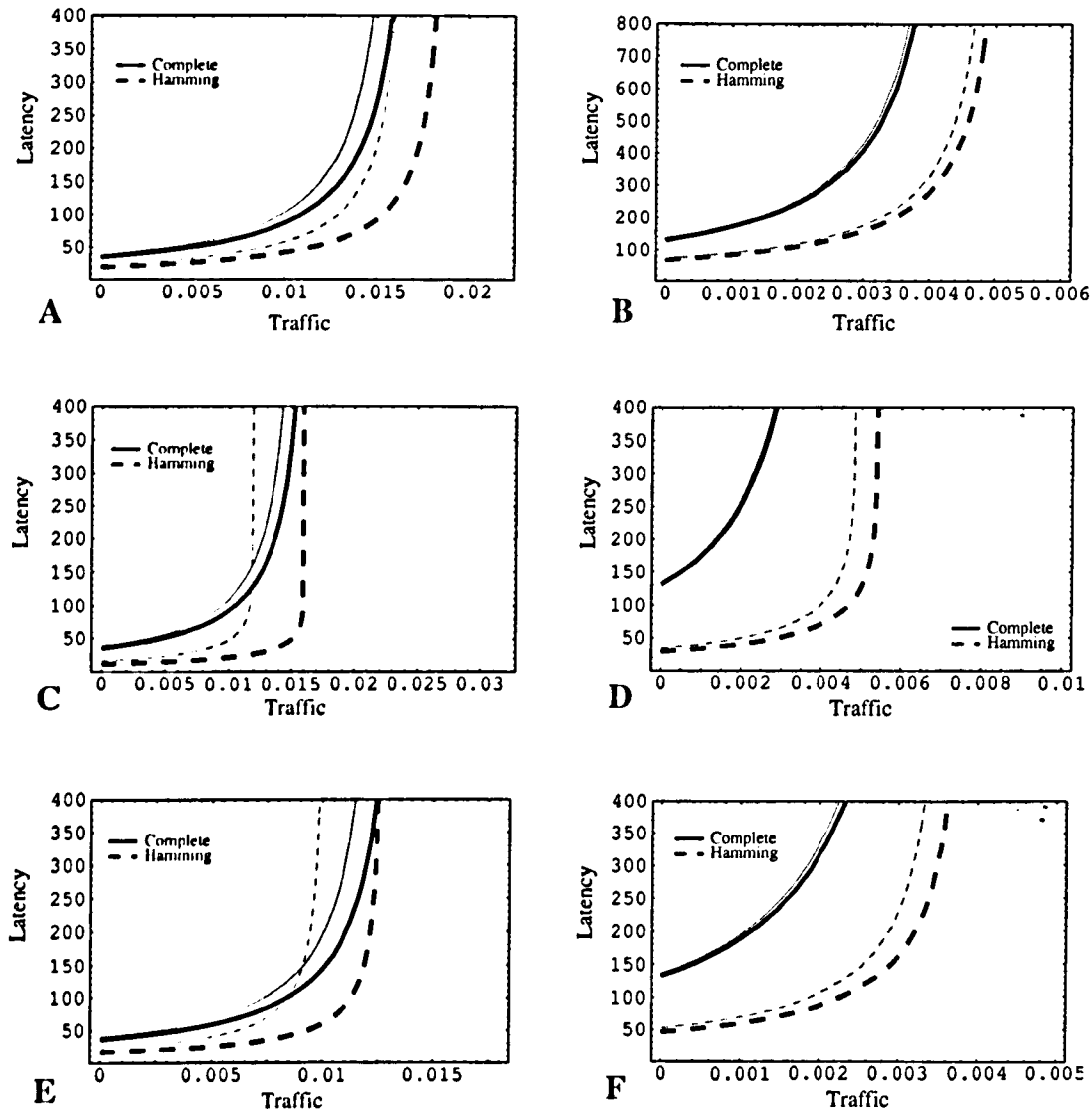


Figure 6.10.: Latency comparison for different networks with a decision time of one cycle (thick lines), and two cycles (thin, grey lines). A small network (N=256, d=2) with short (A) and large (B) packets, a large network (N=4096, d=2) with short (C) and large (D) packets, and a very large network (N=32K, d=3) with short (E) and large (F) packets are shown (short packets are 32 phits[C] long, and large packets are 128 phits[C] long).

and the decision time is large the margin is reduced, especially in networks with a large cluster size as can be seen in figs. 6.10(C,E).

This is due to the fact that smaller packets have short aspect ratios, and networks cannot benefit from their wide channels. Since the Hamming hypermesh has a fixed connectivity and cannot trade channel width for a shorter diameter, it will saturate sooner than the complete hypermesh under this conditions. It would be interesting to create a new hypermesh which could react by trading channel width for a short diameter. A similar result can be found for the networks shown in fig. 6.10(E,F).

It is worth noting that although Hamming hypermeshes saturate sooner than their complete counterparts when small packets, large decision times and large cluster sizes are used, for the networks shown, they provide a smaller latency for all operating conditions except when traffic is close to saturation. For example the two dimensional Hamming hypermesh shown figure 6.10(C) provides a lower latency even when the decision time is two cycles. Although this network saturates faster than the complete hypermesh, it would still provide a better performance, as it provides a lower latency for the majority of the packets, except when the traffic rate is close to saturation (which should be avoided anyway). However if the cluster size is increased further, we speculate that the saturation point of the Hamming network might prove to be too premature and the complete network would be a better choice.

Throughput The maximum dynamic throughput for different networks was evaluated using the model derived in section 6.2.4, and the results are shown in figure 6.17. The throughput is measured in phits per cycle referred to the complete network (see section 6.2.4.1).

The throughput of the Hamming hypermesh is shown plotted against the parameter α (the cluster size is $\kappa = 2^{\alpha}$). As it is shown, the Hamming network outperforms its complete counterpart when large messages are used (see figs. 6.17(B,D)). When small packets are used, and especially when the decision time is large, the opposite occurs. This is due to the fact that in the first case, the Hamming hypermesh can use its wide channels to its advantage; large packets present shorter aspect ratios in the Hamming hypermesh than in the complete one. However in the second case packets are already short, and the benefit of reducing their aspect ratio even further are marginal. The travel distance becomes the dominating factor, and the Hamming hypermesh with its larger diameter keeps more messages in-flight at a given time, which causes contention and the earlier saturation of the network.

The two dimensional Hamming hypermesh, if the decision time is kept short, will outperform its complete counterpart under all circumstances when the cluster size is

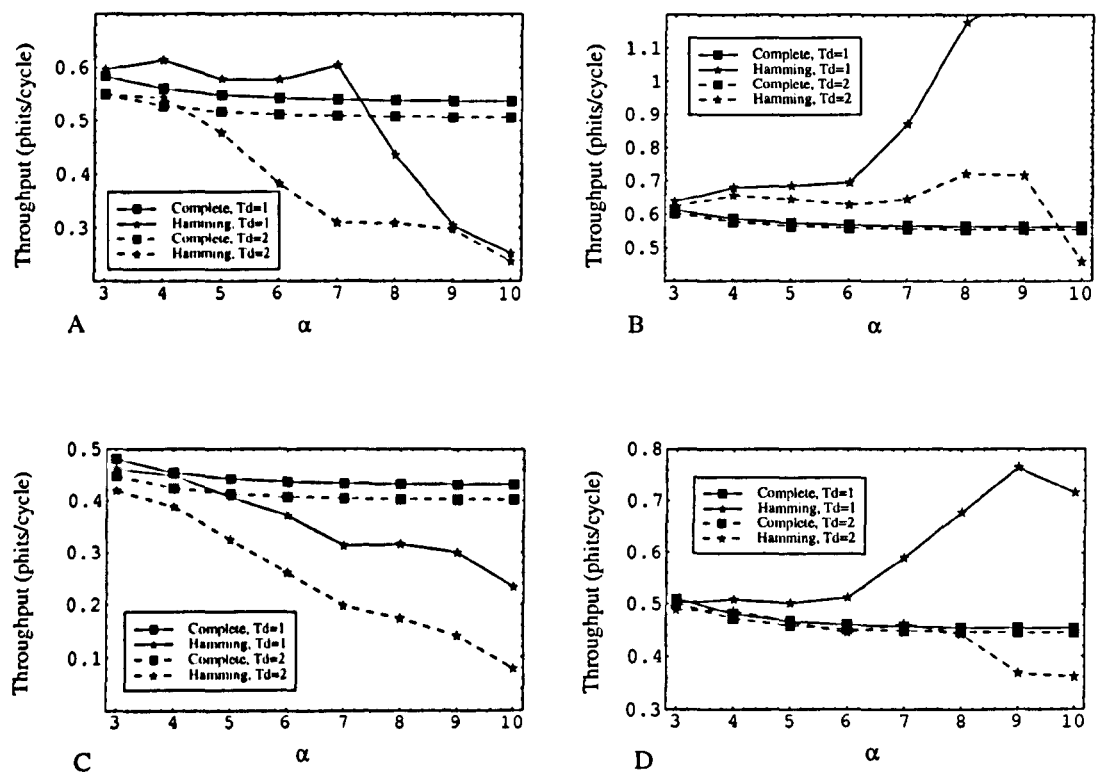


Figure 6.11.: Maximum throughput of different hypermeshes, with a decision time of one and two cycles. Two dimensional hypermeshes with short (A) and large (B) packets, and three dimensional hypermeshes with short (C) and large (D) packets are shown. Short packets are 32 and large packets are 128 phits[C] long.

equal to or less than 128 nodes (which corresponds to a network size of 16K nodes). If the decision time is larger, or if the cluster size is larger, then the maximum throughput of the hypermesh will be smaller than the maximum throughput of the complete network, when small packets are used. However it is important to note that for such cases the Hamming hypermesh will still provide a shorter latency for most packages (see figs. 6.10(C,E) with the two cycle decision time), unless the network is operated close to the saturation point (which should be avoided anyway). For extremely large clusters (or very short packets) and with large decision times this will cease to be valid, since the large diameter of the Hamming clusters will impose a large penalty.

Similar results are obtained for the three dimensional case, but the effects are more pronounced.

6.3. Random Hypermesh

6.3.1. Random Cluster

Random graphs are very efficient expanders as the number of new nodes reached in a given number of steps is very high. This characteristic makes them very attractive for pin-out limited networks, since they can achieve very rich connectivities with low pin-outs.

A random cluster $Q_{\kappa,\delta}$, is a directed hypergraph formed by $|V(Q)| = \kappa$ labelled nodes (v_l) and $|E(Q)| = \kappa$ labelled hyperedges (e_l), such that $\forall e_l,\ \iota(e_l) = v_l \wedge \delta_\omega(e_l) = \delta - 1$, and the output neighbourhood of a hyperedge is chosen randomly. A random cluster is shown in fig. 6.2(B). We have imposed a fixed degree δ for all nodes, since we are interested in exploiting the available pin-out of a node (as to exploit the available bandwidth); otherwise the node with the largest degree becomes a bottleneck which affects the rest of the system.

This hypergraph can be constructed from a random graph as follows. We start by randomly choosing a graph $F_{\kappa,\delta-1}$ from the set of graphs:

$$\left\{F \in G^U : |V(F)| = \kappa \wedge \forall v \in V(F),\ \delta(v) = \delta - 1\right\}. \tag{6.26}$$

Based on this graph we construct $Q_{\kappa,\delta}$ such that $V(Q) = V(F)$ and $E(Q)$ is formed by the hyperedges h_l, such that $h_u \in E(Q)$ iff $\iota(h_u) = v_u \wedge \omega(h_u) = N_F(v_u)$, where $N_F(v_u)$ is the neighbourhood of node v_u in F.

Note that this class of hypergraphs are symmetrical in the sense that $e_{uv} \in E$

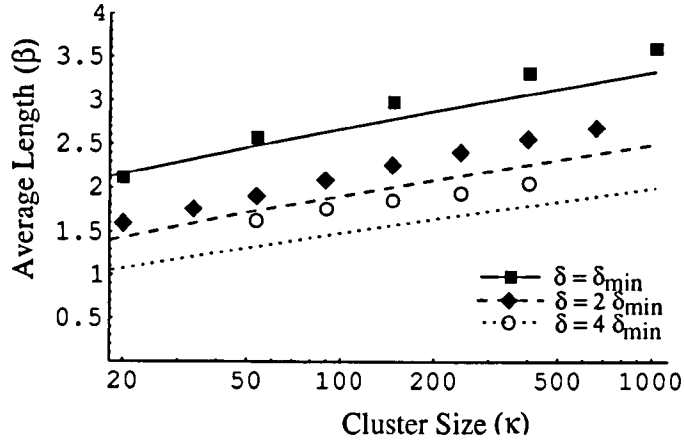


Figure 6.12.: Average length for different random clusters. The points are measurements taken from networks constructed with the previous algorithm. The lines are model predictions (using the simple model, $\beta \approx \ln\kappa / \ln\delta$).

implies $e_{vu} \in E$ (it would be interesting to investigate the effects of asymmetrical random hyperedges). We want this graph to be connected, so we impose the condition $\delta_Q - 1 > \ln\kappa$, since this is the threshold for connectedness in random graphs [87] and $\delta_{min} = 1 + \ln\kappa$.

A random hypermesh $R_{d,\kappa,\delta}$, is constructed by the d-dimensional Cartesian product of the random cluster $Q_{\kappa\delta}$. The total number of nodes in this hypermesh is $N = \kappa^d$, and the total degree is given as $\delta_R = d\delta_Q$, where $1 + \frac{\ln N}{d} \leq \delta_Q \leq \sqrt[d]{N}$. The random hypermesh is able to tune itself between a wide channel, $\Delta W_{RC} \leq \frac{d\sqrt[d]{N}}{d+\ln N} = O\left(\sqrt[d]{N}\right)$ large diameter network, and a narrow channel $\Delta W_{RC} = 1$, low diameter network ($\beta_R = 1$), by varying the parameter δ_Q.

The ability to trade channel width for smaller inter-cluster distances allows the random hypermesh to optimise its performance for a given operating regimen.

6.3.1.1. Diameter and average length

Messages will need to route through several nodes within the same cluster before routing to a lower dimension. This distance is given as the average length of a random graph, which has already been studied and can be written as [118]:

$$\beta_Q \approx D_Q - \frac{\delta_Q\left[(\delta_Q - 1)^{D_Q} - D_Q(\delta_Q - 2) - 1\right]}{(\kappa - 1)(\delta_Q - 2)^2}, \tag{6.27}$$

where the diameter of the cluster D_Q is given by [87,118]:

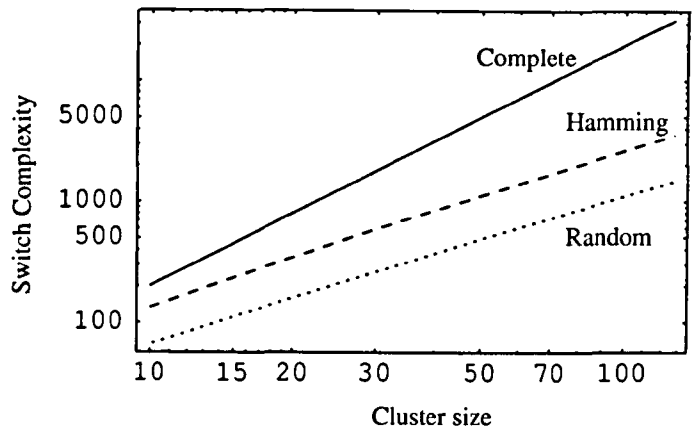


Figure 6.13.: Switch complexity for different cluster implementations. The random cluster is shown at its minimum complexity ($\delta = 1 + \ln \kappa$). The maximum complexity for the random cluster is equal to that of the complete hypergraph.

$$D_Q = \frac{\ln\left(\frac{\delta_Q - 2}{\delta_Q}(\kappa - 1) + 1\right)}{\ln(\delta_Q - 1)} + 1; \quad (\delta_Q > 2). \tag{6.28}$$

When $\kappa \gg \delta_Q > \ln(\kappa) \gg 1$, the following approximations can be made: $D_Q \approx \ln(\kappa)/\ln(\delta_Q)$, and $\beta_Q \approx D_Q \approx \ln(\kappa)/\ln(\delta_Q)$. These approximations will be generally used throughout this chapter.

6.3.2. Switch Complexity

As we have stated, the complexity of a switch is proportional to the number of internal routes $X = O(\delta^2)$, where δ is the degree of the cluster. Since each route has W wires, the switch complexity will be measured as $S = dW\delta^2$. Figure 6.4 and table 6.2 shows the switch complexity, and the complexity reduction for the complete $\left(\delta = \kappa = \sqrt[d]{N}\right)$, hamming $(\delta = 2\log_2 \kappa)$ and random $(1 + \ln \kappa \leq \delta \leq \kappa)$ hypermeshes. Note the linear reduction in complexity for the incomplete networks (with respect to the cluster size). If we were to change our constant pin-out argument for a constant switch complexity argument, the incomplete networks would exhibit an even better performance.

6.3.3. Random Cluster Optimisation

As we have stated when comparing different networks we will maintain constant the total pin-out. Since the random cluster is able to trade channel width for length

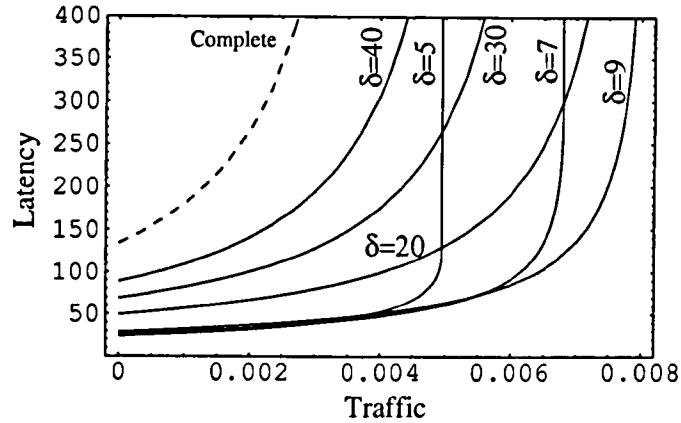


Figure 6.14.: Latency of random hypermeshes with different degrees (N=4096, d=2, B=128 phits[C], T_{dec}=2 cycles).

reduction, it is desirable to calculate at what connectivity level is the optimum performance achieved. It is obvious that under no load the maximum throughput is achieved when the network has the widest possible channels; while the smallest latency is realised when the diameter is small; that is when the channels are not very wide and the connectivity is high.

In a more realistic scenario, when the network is loaded the situation is different; narrow channels and large diameters lead to contention and eventually saturation. The effects of varying the connectivity of a random hypermesh can be seen in figure 6.14. As it is shown small connectivities result in early saturation. Large connectivities also suffer from poor performance due to large packet aspect ratios; packets spend more time in the network (despite the fact that they need to traverse short routes), increasing both, latency and contention, which leads to saturation. A similar situation is observed for the maximum throughput (see fig. 6.17).

We are interested in obtaining the optimum connectivity δ_{opt}, which is defined as the connectivity which provides the largest saturation traffic, μ_{sat}. The maximum throughput is also achieved with this connectivity. This is obtained from:

$$\delta_{opt} = \delta \; : \; \mu_{sat}(\delta) \; = \max\{\mu_{sat}(\delta)\}. \tag{6.29}$$

The optimum value of some networks is shown in figure 6.15.

6.3.4. Latency

The dynamic latency of different network configurations was evaluated using the model developed in section 6.2.4, and the results are shown in fig.6.16. The connec-

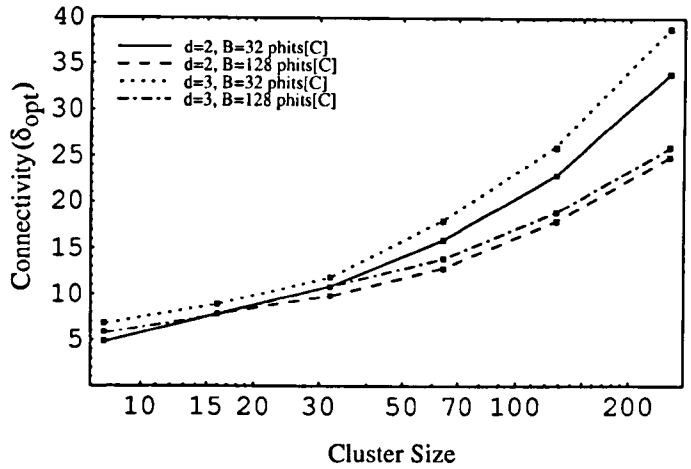


Figure 6.15.: Optimum connectivity for some random hypermeshes with a decision time of one cycle.

tivity of the random hypermeshes shown is the optimum ($\delta = \delta_{opt}$).

As it is apparent the random hypermesh outperforms the others under all conditions showed. We believe this is a consequence of the superior expander properties of random graphs, which allow them to use their available wiring very efficiently.

For small networks (A,B in the figure) similar results as before are found, where the incomplete implementations outperform their complete counterparts, providing the additional bonus of wider channels and therefore throughput.

For larger networks, and especially with large decision times, the random hypermesh shows a definite advantage over the other implementations, as the Hamming cluster can become too large and suffer from large latencies when small packets and large decision times are used. In this case the random hypermesh reacts by trading channel width for a shorter diameter (i.e. in fig. 6.10(C), for the two cycle decision time $\delta_{Hamm} = 12$ and $\delta_R = \delta_{opt} = 19$, while for the one cycle decision time $\delta_R = 16$), and manages to outperform both, the complete and the Hamming hypermesh (see figures 6.10(C,E)). A similar result can be found for the networks shown in fig. 6.10(E,F).

6.3.5. Throughput

The dynamic maximum throughput for different networks was evaluated using the model derived in section 6.2.4, and the results are shown in figure 6.17. The throughput is measured in phits per cycle referred to the complete network (see section 6.2.4.1).

The throughput of the random hypermesh is shown plotted against its connectiv-

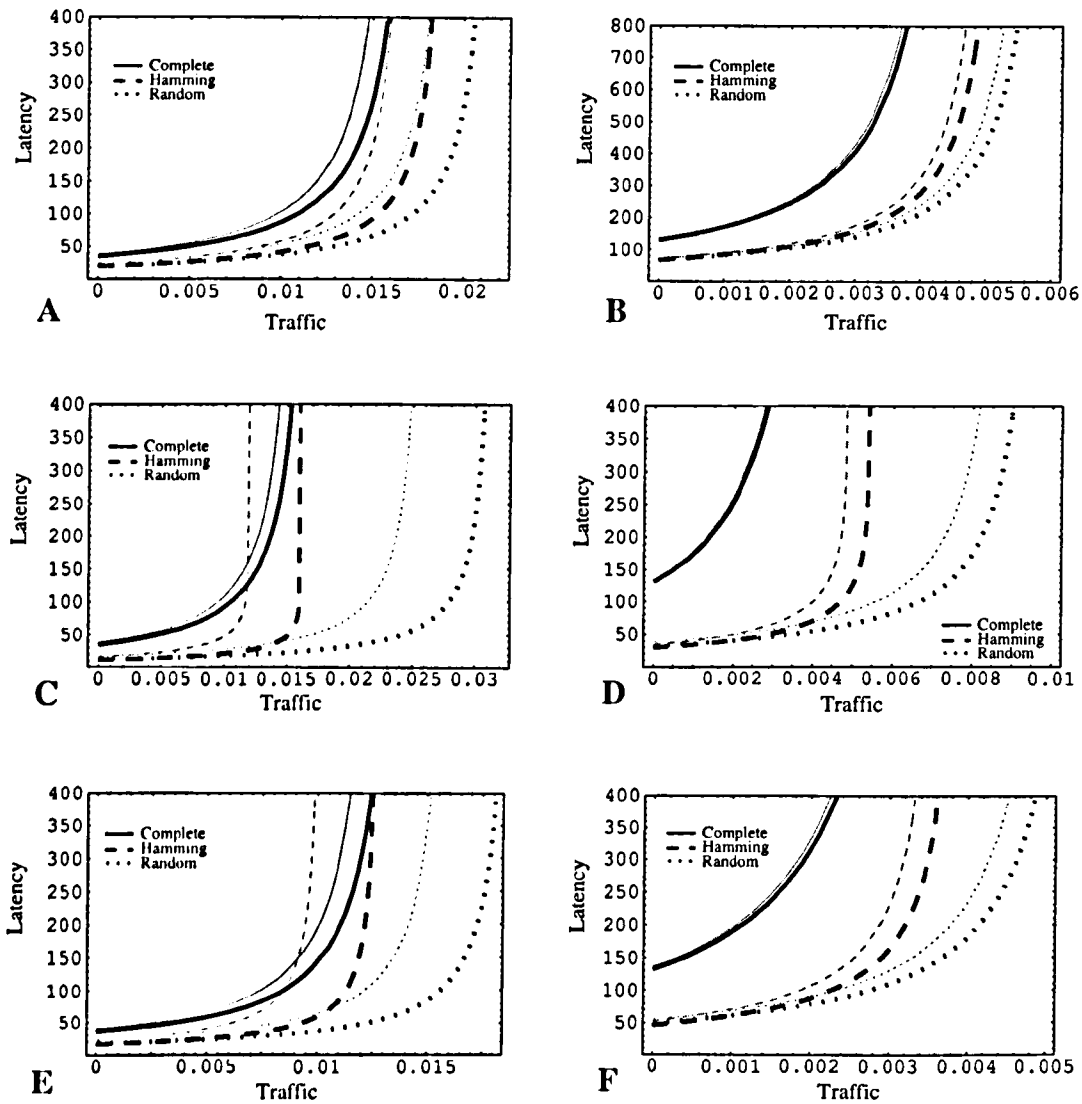


Figure 6.16.: Latency comparison of random incomplete hypermeshes against other implementations with a decision time of one cycle (thick lines), and two cycles (thin, grey lines). A small network (N=256, d=2) with short (A) and large (B) packets, a large network (N=4096, d=2) with short (C) and large (D) packets, and a very large network (N=32K, d=3) with short (E) and large (F) packets are shown (short packets are 32 phits[C] long, and large packets are 128 phits[C] long). For the random hypermeshes $\delta = \delta_{opt}$.

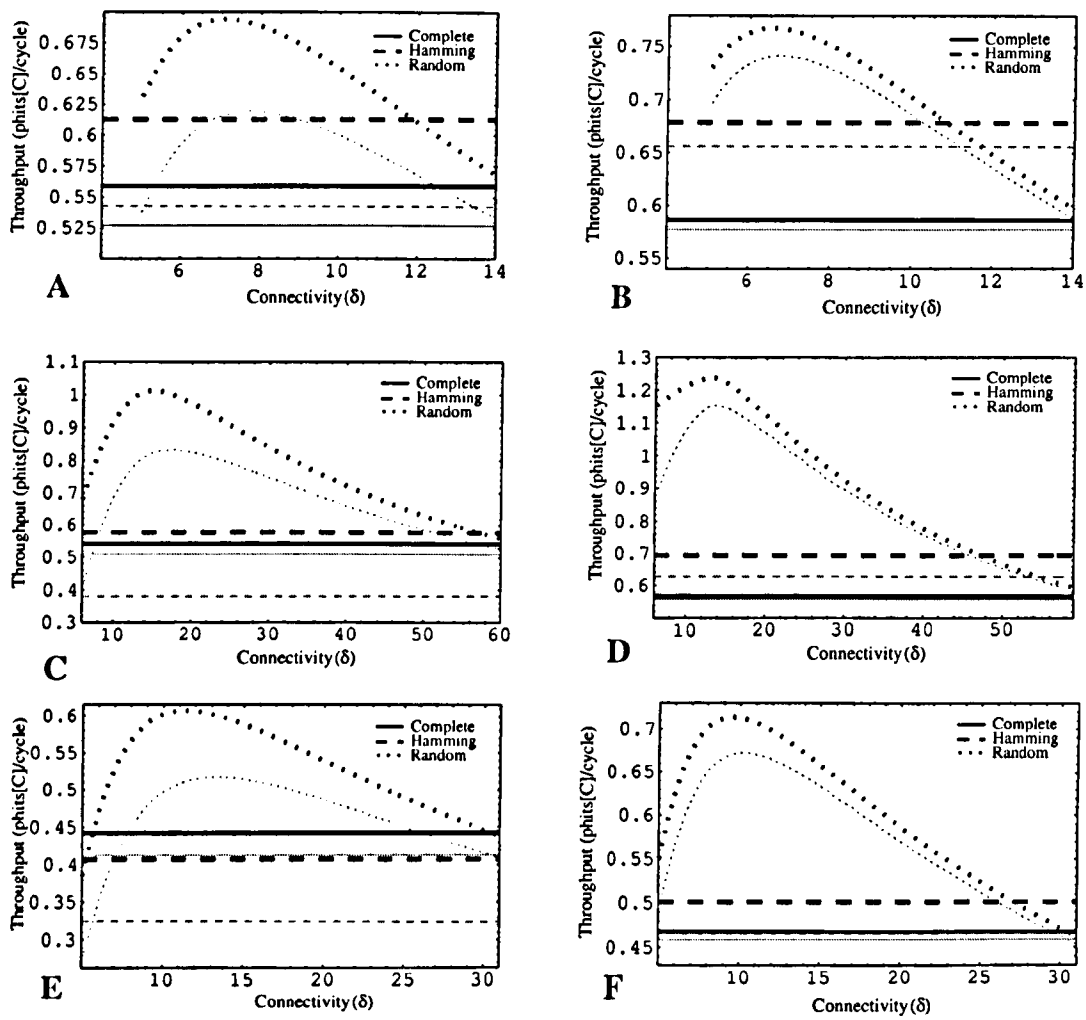


Figure 6.17.: Maximum throughput of different random hypermeshes against their connectivities (δ_R), with a decision time of one cycle (thick lines), and two cycles (thin, grey lines). The Hamming and complete hypermeshes are also include in the plot, although their connectivity is fixed. A small network (N=256, d=2) with short (A) and large (B) packets, a large network (N=4096, d=2) with short (C) and large (D) packets and a very large network (N=32K, d=3) with short (E) and large (F) packets are shown. Short packets are 32 and large packets are 128 phits[C] long.

ity. The peak of this curve is obtained when the connectivity is optimum $\delta_R = \delta_{opt}$ (see section 6.3.3). When the connectivity is less than the optimum ($\delta_R < \delta_{opt}$), packets travel large distances; and although they have a small aspect ratio (the network has wide channels), contention will cause early saturation (see fig. 6.14), and will limit the maximum throughput. On the other hand when the connectivity is larger than the optimum ($\delta_R > \delta_{opt}$), channels are narrow, and although packets traverse only short distances, the large packet aspect ratio will lead into early contention and saturation, and will limit the throughput.

In figure 6.17 we see the same effects as discussed in the previous section; incomplete hypermeshes outperform complete ones by a wide margin when large packets are used, and the hamming hypermesh is outperformed by its complete counterpart when large clusters, large decision times and small packets are used (see figs. 6.17(C,E)).

Random hypermeshes outperform the other implementations under all conditions, and in some cases provide twice the throughput of the complete hypermesh.

6.4. Conclusions and Summary of Results

The Hamming hypermesh has been introduced and some basic characteristics have been analysed such as the diameter and average length. Furthermore a routing algorithm has been provided, and a model describing the latency and throughput of the network has been constructed. It has been shown that the Hamming hypermesh outperforms complete hypermeshes under the constant pin-out argument, using wormhole switching and deterministic routing, except when used in large clusters with small packets and large decision times. When they operate away from this condition, they provide lower latency, higher throughput, lower cost and simpler switches. The reason for this is that due to its low pin-out, the inter-cluster travel distance becomes too large, leading into saturation when the traffic level is high. We speculate that an extended hamming connection which could trade channel width for a better connectivity would perform much better, when large clusters, small packets and large decision times are used.

With large packets or moderate cluster sizes, the Hamming hypermesh clearly outperforms their complete counterparts, and as they posses much higher bandwidths and their switch complexity is lower, they provide much larger throughputs and reduced costs.

Random hypermeshes outperform all other hypermeshes under the constant pin-

out argument, using wormhole switching and deterministic routing. They provide lower latency, higher throughput, lower cost and simpler switches. The same is true for the Hamming hypermesh, except when used in large clusters with small packets in which case it can be outperformed by the complete hypermesh.

Since it has been shown that complete hypermeshes outperform the mesh, the torus, low dimensional k-ary n-cubes (with and without bypass channels), and multistage interconnection networks (when realistic decision times are accounted for), it follows that incomplete hypermeshes outperform them as well.

The main disadvantage of using random interconnections is that a routing table or other sophisticated mechanisms are required for the router design, therefore it would be very beneficial to provide a deterministic network with the ability to tune its degree and diameter, so that the performance can be optimised according to the network variables (decision time, cluster size, etc.). To this effect the use of other clusters could be investigated, such as LFSR and extended Hamming

The use of the random network is also beneficial as a design tool. Once the optimum connectivity is established (according to the design criteria), a deterministic network with similar characteristics can be used in place of the random network.

Since it has been shown that complete hypermeshes outperform the mesh, the torus, low dimensional k-ary n-cubes (with and without bypass channels), and multistage interconnection networks (when realistic decision times are accounted for), it follows that incomplete hypermeshes outperform them as well. Furthermore incomplete hypermeshes posses much higher bandwidths and their switch complexity is lower, leading to reduced costs and better performance.

Although these results were derived for networks using wormhole switching, we speculate that the main results will still hold with different switching methods.

7. Conclusions

An efficient interconnection network should be capable of exploiting the locality of information present in the set of expected communication patterns, while at the same time accommodate other general communication patterns as efficiently as possible. Small-world graphs provide a communication model where traffic at a local level is confined at a local cluster of the underlying graph, and long range traffic is supported by a set of shortcuts present introduced in a network, and thus are a natural option for the construction of efficient interconnection networks.

The original construction methodology proposed by Strogatz and Watts where a subset of the edges are randomly rewired with a given probability has been superseded by an additive construction method in which edges are added with the same probability, and the structure of the underlying graph is preserved intact. This is important in order to exploit some of the useful properties of the deterministic underlying graph; for instance it is possible to construct deadlock free routing algorithms by exploiting this property.

Random Small-World (RSW) interconnection networks based on both construction methodologies have been introduced in the present work, and some of their basic properties have been analysed. It has been shown that RSW graphs are good candidates for the construction of interconnection networks, as they can exploit the locality of information, and due to their small diameter and high connectivity, can efficiently support random traffic if enough shortcuts are introduced.

A simple mathematical model for the diameter and average length of a RSW graph has been constructed; and it has become apparent that the most suitable underlying graphs for a RSW interconnection network have a strong local connectivity but suffer from a weak global level connectivity characterised by a large average length and diameter.

For the special case of an underlying array an expression for the diameter has been constructed by extending the average length Mean-Field like model proposed by Newman and Watts, and the forking process model proposed by Moukarzel. The models have been validated with the aid of a custom graph construction and analysis computer program, where a large number of RSW graphs have been constructed and

direct measurements have been taken from them. The models corroborate the fact that a fast transition from a large world to a small world is undertaken as soon as a large enough set of shortcuts is available, and that the diameter experiments this fast transition in a similar way as the average length does. Although models for the average length had already been constructed, a correction for systems of small size has been undertaken both for the average length and the diameter, and the results have been corroborated with experimental data. It follows that latency is reduced in the same way, at least for systems with low or sporadic traffic.

The degree distribution has also been studied, and a model has been constructed and verified with experimental data. The degree increase of a node follows a binomial distribution, from which the maximum expected degree has been calculated and verified against measurements taken directly from RSW graphs. As expected the degree of the rewired graphs is increased with the addition of shortcuts; however the maximum degree is only increased significantly for large rewiring probabilities, so there exists a large interval of probabilities where the maximum degree is kept small while the diameter has been reduced significantly.

As expected the bisection width is also increased and it is shown that this increase is linear with the number of edges introduced, and hence with the rewiring probability, and also with the number of nodes. Since the expected traffic increase with an increase in network size is proportional to the number of nodes, such an increase in bisection width is necessary to support traffic as the system is expanded.

A latency model for array RSW interconnection networks with random traffic and with shortest path routing has been presented and corroborated against simulations carried out with a purpose built discrete event simulator integrated into the graph construction and analysis program. It has been shown that latency is reduced in a similar fashion as the average length and diameter and throughput is increased if enough shortcuts are present. Otherwise the network can easily saturate if random traffic is present.

To provide a better characterisation of this "premature" saturation, congestion in RSW graphs has been analysed and an analytical model has been provided. The model corroborates the existence of the premature saturation, and provides a mechanism to calculate the number of shortcuts required to avoid it. Furthermore it is shown that a steep decrease in congestion occurs beyond a critical point, and therefore a large boost in performance can be expected beyond it.

Lack of regularity and a large degree are the most important drawbacks of RSW graphs. In order to improve both, deterministic small-world graphs have been introduced. Deterministic SW graphs constructed by superposition are a natural

extension of RSW graphs where the rewiring process is deterministic (and restricted to produce regular networks). An example of such a network is the family of Linear Feedback Shift Register graphs (LFSR). These graphs retain most of the properties of RSW which have already been analysed, while providing a deterministic network and a regular graph. The determinism in the graphs can be exploited to produce simple routing algorithms, as has been done for the ring LFSR graph. The amount of information required to create a representation of the network is reduced from $O(N)$ to $O(1)$, which aids in the exploitation of algorithmic properties of the network.

The family of ring LFSR networks has been analysed and shown to provide very desirable characteristics, such as regularity, small degree, diameter and average length. The reduction in length and diameter is achieved by the introduction of shortcuts, and since in LFSR graphs of small degree the ratio of shortcuts to normal edges is larger than for LFSR with a large degree, it follows that LFSR networks with a small degree are better expanders, and have a diameter and average length close to the Moore limit. On the other hand, large degree LFSR graphs can exploit locality more efficiently, and a natural tradeoff between cost and support for locality is created.

An n-level recursive greedy routing algorithm for the LFSR graphs has been provided using a tree embedding. It has been showed that for most graphs a small lookup depth gives close to optimal results, with the added advantage of requiring only local information to work. Large networks with a small degree require a large lookup depth in order to reduce the routing diameter significantly. However it is important to note that the diameter with a small lookup depth in these networks is already close to the optimal value. Furthermore, although the diameter experiences a slow decay in these networks, the average length is reduced much faster, which signifies that contention can be reduced very effectively by using such an algorithm. If on the other hand the communication patterns expected have a large amount of locality and make use of sporadic long range communications, a small lookup depth may be more efficient and provide better results (as the computational effort is reduced).

Deterministic SW graphs have also been constructed by a direct method, in which a suitable deterministic function generates a local structure, and shortcuts of a large variety of lengths, which according to the multiple scale hypothesis reduce the average length of the graph. A particular generating function based on the Hilbert curve has been used to construct the Hilbert graph, as an example of this design method. It has been shown that the Hilbert graph posses some remarkable characteristics; such as modular expandability, small fixed degree, two layered planar

implementation, simple recursive structure, small bisection bandwidth, it can exploit the locality of information in an application and it exhibits a small logarithmic diameter and average length. Due to its fixed degree of four, it is only natural to compare this graph against the torus and mesh, which also have a fixed degree of four. The bisection width of the Hilbert graph has been determined, and shown to be very similar to that of the mesh and torus. However the Hilbert interconnection makes a much better use of the available wiring, and since it provides a much reduced diameter, it can provide a reduced latency. It is expected that much larger systems with an equal diameter can be accommodated using a Hilbert graph than with a mesh or torus, provided an adequate routing algorithm can be designed.

The work has been extended to hypergraph interconnection networks, and in particular to the Hypermesh. While for graph networks the construction methodology has been based on adding or modifying the connections in them, for hypergraphs the construction goal is to reduce the degree of a node while maintaining a good connectivity. Traditional hypermeshes (such as the DCSH) already provide an excellent connectivity. However it has become evident that the degree can become too large making the network too expensive to implement. In this work incomplete clusters based on random connections and on a deterministic process have been introduced and analysed. These implementations have been compared using a constant pinout metric. Deterministic (Hamming) hypermeshes are shown to outperform the traditional (complete) implementation and provide a lower latency and higher throughput except when used in large clusters with small packets and large decision times. When they operate away from this condition, they provide lower latency, higher throughput, lower cost and simpler switches.

For the case of random hypermeshes it is shown that they outperform all other hypermeshes under the constant pin-out argument, and provide lower latency, higher throughput, lower cost and simpler switches. Since it has been shown that complete hypermeshes outperform the mesh, the torus, low dimensional m-ary D-cubes (with and without bypass channels), and multi-stage interconnection networks (when realistic decision times are accounted for), it follows that these incomplete hypermeshes outperform them as well.

It is expected that future work will include the fault tolerance analysis of the networks presented in this work, and that at least for the case of random small-world networks, the introduction of non-determinism will increase the fault tolerance significantly. Furthermore new routing algorithms need to be developed for the LFSR and Hilbert graph to provide better and simpler routing.

Since it is expected that for some of the graphs introduced in this work the

large bisection width will create difficulties for the physical layout of the networks, the use of Cartesian products of small-world arrays is proposed. In such a system the complexity of the interconnection can be hidden by constraining it along the vertical, horizontal or n-dimensional arrays. The resulting topology will resemble the incomplete hypermeshes proposed in this text; and it is expected to exhibit some of the same characteristics.

Fractals provide a natural scale free structure which can be used to generate small-world networks such as the Hilbert graph. It is expected that further work in this area can provide a more general design methodology for graphs which exhibit the same desirable characteristics as the Hilbert graph. Furthermore it is possible to fix the dimensionality of the network by this method and hence provide networks which are simple to construct.

The small-world model of graphs, where the structure is conceptually divided into a local level (with a strong local cluster) and a global level which provides inter-cluster communications, is of general applicability. As such, the work undertaken in this work can be extended into other areas of knowledge. In particular the model for the diameter of the network can be used instead of the average length in a large variety of settings, including disease spreading, neural networks, social networks and others. Previous work has used numerical simulations and mathematical models for the average length, where the reduction of average length is one of the most important characteristics under study. Although this is valid for some systems, the congestion model developed in this work shows that there is a fundamental limit in the rate of flow through the network. This work can be extended to provide more accurate model of complex systems where interchange of fluids or information can cause edge saturation.

A. Recent Parallel Systems

The following table is a list of some recent commercial parallel computers.

Name	Machine Type	Connection Structure	Processors	Year
CPP Gamma II	SIMD	2D Mesh Extended	4096	1995
Compaq GS	SMP	4 nodes/board - xbarup to 8 boards - xbar(hierarchical xbar)	32	1999
Compaq AlphaServer SC	SMP/NUMA	4 nodes/board - xbarBoards - Fat tree	512	1999
Cray SV1ex	SM Vector/NUMA	4 CPUs/board - xbar4-8 boards/cabinet - xbar4 Cabinets - Giga Ring	128	2000
Cray T3E (1350)	NUMA	3D Torus	2176	2000
Cray MTA-2	NUMA	3D Torus w/reduced connectivity	256	2001
Fujitsu AP3000	SMP/NUMA	2 CPUs/board - xbar SMPboard - 2D Torus	1024	1996
Fuj.Siem. PRIMEPOWER	SMP	xbar	128	2000
Fujitsu VPP5000	NUMA/Vector	Distributed xbar	128	1999
Hitachi SR8000	SMP/NUMA	8 CPUs/board - xbar SMP?Nodes- Multi -Dimensional xbar	512	1998
HP 9000 SuperDome	SMP	2 level xbar	64	2000
IBM RS/6000 SP	SMP/NUMA	4-16 CPUs/board - xbarboards - omega switch	2048	1999
NEC Cenju-4	NUMA	multistage xbar	1024	1998
NEC SX-5	NUMA / Vector	multistage xbar	512	1998
Quadrics Apemille	SIMD/Array	3D Torus	2048	1999
SGI Origin 3000	SMP/NUMA	2-4 CPUs/board - xbarboards - hypercube	512	2000
SUN E10000 Starfire	SMP	(64x64) xbar	64	1997

B. Integrated network analysis and design system (INADS)

A system for studying graphs and interconnection networks has been developed as part of this research. The program allows the user to interact, simulate and study the networks in an intuitive fashion. The program is written in approximately 20000 lines of C++ code, and is built around the LEDA library from the Max-Planck-Institut für Informatik in Saarbrücken [132] available from http://www.mpi-sb.mpg.de/LEDA. The following is a list of the system capabilities:

- Graphical editing of networks and graphs.
- Saving, loading, printing and other utilities.
- Automated graph creation and modification:
  - Creation of meshes, k-ary n-cubes, rings, complete graphs, random graphs (unconstrained, bipartite, triangulated, planar), Hilbert, LFSR Ring, de Bruijn, Kautz and others.
  - Random Rewiring, normal and true random distributions, as well as support for deterministic rewirings, such as LFSR.
- Basic graph algorithms:
  - Test if the graph is connected, simple, bipartite, planar, etc.
  - Diameter, degree and average length.
  - Shortest path algorithms (Dijkstra, Bellmon-Ford).
  - Cut-width measurement using the chacos library available at http://www.cs.sandia.gov/CRF/chac.html.
- Small-world graph utilities:

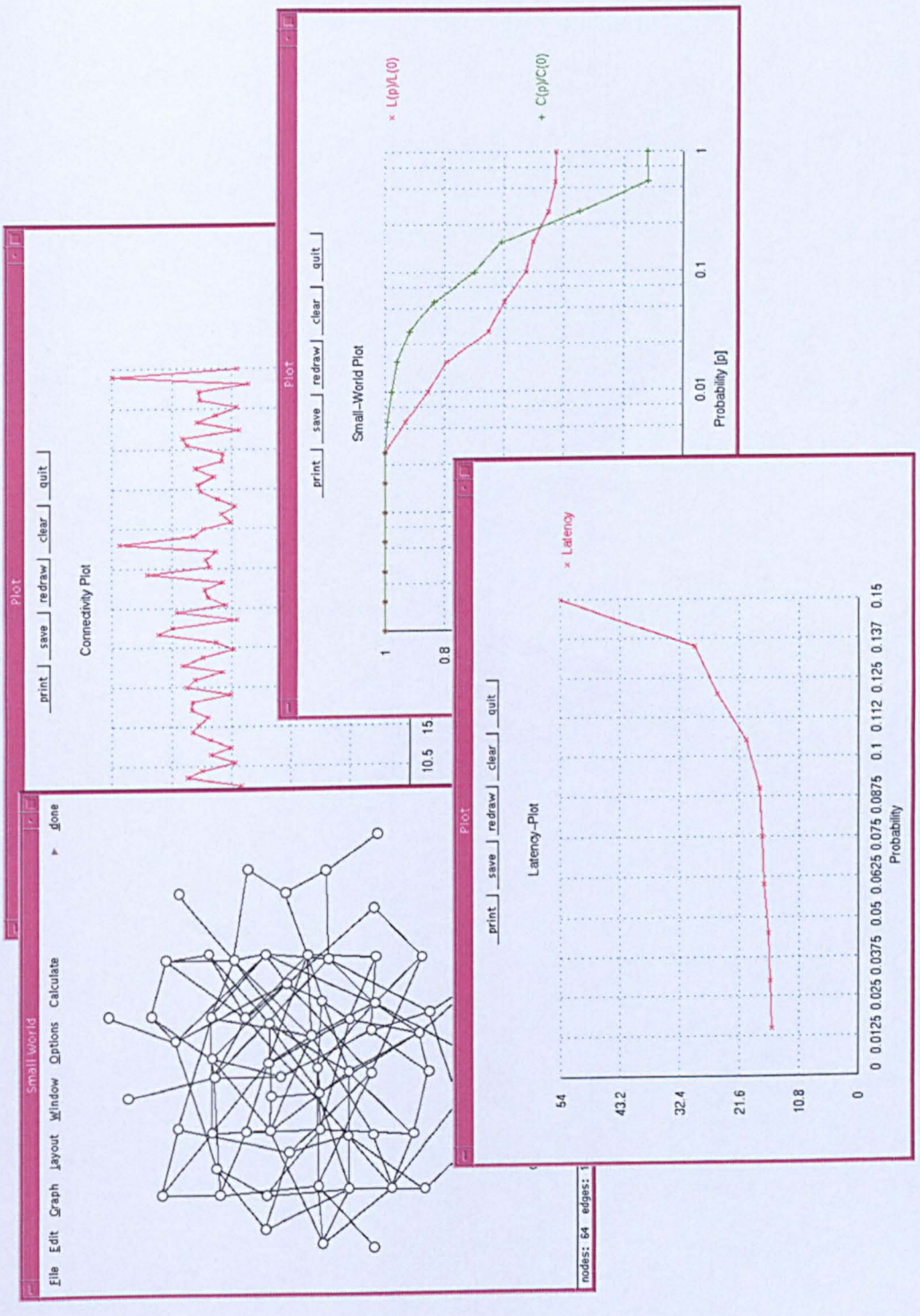


Figure B.1.: INADS Screenshot

  - Measurements (length, cliquishness, sampled length, sampled cliquishness)
  - Small-world plot (by automatically rewiring the graph and evaluating successive levels of randomness)

- Graph evolution using genetic algorithms (constraints of complexity, connectivity, wire density and connectivity matrix can be specified, and the evaluation parameters can be dynamically changed)
- Discrete event simulation to evaluate the performance of a given architecture. Automated measurements are available (latency vs. message probability plot). Support for different routing algorithms (including shortest path), as well as direct specification of a routing graph.
- Complex utilities, such as small-world latency plot (latency vs. rewiring probability), 3-D graph visualisation, spring embedding and others.

A screenshot of the system is shown in fig. B.1.

C. Fastrack System Integration

The starting point of this research project was to address the system integration of the ongoing Fastrack project, as described in [150]. The tasks involved in this work were to:

1. Provide operating system support for the computer to achieve the project goals as specified in [150], which are the experimental validation of the conclusions arrived at [51], [31] by Mackenzie et al and Ould-Khaoua.

2. Identify, quantify and address the problems that arise from the interaction between the network interface (NI), the computer and the software.

3. Determine the appropriate test algorithms and programs.

The aim of the fastrack project is to construct and test a DCSH architecture using PowerPC computers as the nodes. It was decided to construct a network interface card with connections to the PCI bus. This card would use packet switching, but could also support virtual cut-through and wormhole routing. This is shown in figure D.2. The high speed electrical links were designed using Motorola's Autobahn technology. However due to the complications encountered during the construction of the card it was never finished. The project was continued by Beeley in his *Design And Construction Of A Hypermesh Interconnection Parallel Computer* project. Due to the similarity of both projects they will be referred by the same name in this document (Fastrack). To simplify the design, it was proposed to modify the original design so that each SE has less inputs, and multiplexing is done externally to the SE [150], using a non-blocking switch. To further simplify the design Beeley proposed to partition it into two different network interface cards for each SE. Each card has an input and an output, which can be connected to different dimensions (i.e. one will be connected to the X cluster, and the other to the Y cluster). Only one of the cards is required to have routing capabilities to implement a restricted routing scheme, however both cards can share the same design, as shown in figure D.3, and other routing strategies could be tested.

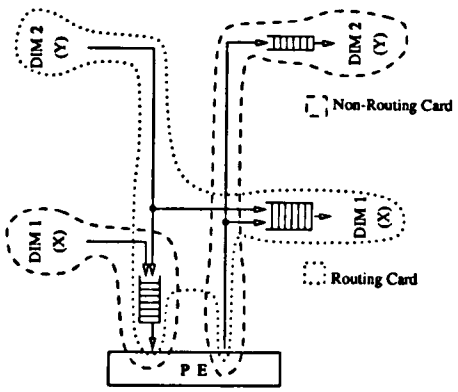


Figure C.1.: Current NIC partition of the Fastrack project.

C.0.1. Selection of an operating system

The selection of an operating system is fundamental in designing a computing system, since it will influence the power, usability, reliability and security of the whole system. In selecting the operating system, the following criteria was used.

- Availability. The computing nodes of this system will be Apple Power-PC computers, and the operating system has to be available for this platform.
- Maturity. It is desirable to have a mature and proven system, instead of an experimental or development one.
- Open system. It would prove advantageous to work with an open system, since we could modify or construct features on top of the operating system. Also it is very useful to work on an open system to evaluate different features of the interaction between the operating system, the applications and the network interface.
- Documentation. Since we will have to write drivers for the network interface card, documentation should be available to help during the development.
- Parallel systems support. We need tools to provide support for a multicomputer system. We need to have applications and libraries, so it is possible to use them.

A subjective evaluation of the currently available operating systems was performed and summarised as follows. MacOS is not used mainly because it is not an open system but also because its multicomputer applications support is not as good as in the other choices, as evidenced in [151]. BeOS is not a proven system and the

PowerPC version seems to be almost extinct. Mach lacks multicomputer software for this platform. All of the applications that run under Mach *do so under another* operating system running over Mach.

Linux on the other hand has very good software and support for multicomputer and is an open system. It has vast documentation, although the documentation specific to PowerPC systems is not as abundant and organised. Although the PowerPC implementations are not a completely mature product they work very well, and they are fast, reliable and generally compatible with UNIX systems.

MKLinux runs on top of the Mach micro-kernel, and therefore it is slower than LinuxPPC. The rest of the operating system is almost identical. The only important difference being that LinuxPPC does not run in non-PCI computers. However since the whole network interface design is based on the PCI bus, this is not important.

For these reasons the operating system chosen was LinuxPPC, which was installed in two computers to test and evaluate.

C.0.2. Defining the features of the operating system

It might be tempting at first to provide all of the functionality described in C.0.1. However most of it is unnecessary to achieve the goals of the project (C). Furthermore, some of these are very difficult to implement, and they are more research material than a physical reality.[1]

In deciding the required support that the OS should provide, it is important to keep a minimalistic approach, and provide only what is strictly required.

To provide a testing platform, the operating system is required to be able to:

1. Send and receive packages of information. To do this the first thing required is support for the PCI card (see C.0.3). It is also required that a communication library with basic communication primitives is constructed. It must also provide support for the link establishment, link maintenance and error detection and correction so as to provide the applications with an error-free link (for applications that require it).

2. Time stamp the packages in order to measure latency. There are two issues required to perform this service. First it is necessary to establish a global synchronised clock to be able to measure time differences in different places. To provide a global clock there are different techniques that can be used [153]. However this will depend on the error tolerance required. When a message

[1] None of the existing operating systems at the time of writing provides all of these features [152].

arrives it has to be time stamped. The error introduced in the time stamping will not only depend on the clock error, but also on the (variable) time taken for the operating system to respond to that event. The corresponding mechanism is discussed in C.0.3.

3. Provide a means of running "real" parallel software. Most of this software uses communication libraries that provide the communication primitives required to run in a particular machine. In some instances these libraries are embedded in the corresponding platform and they are invoked automatically by the compiler.[2] In other software, the libraries used are open and well documented (i.e. MPI and PVM) [154], [155].

4. It was decided to use PVM and MPI as the standard libraries to allow parallel software to run; since if these are used, many applications written for these libraries could run in the system.

The libraries were installed and tested. The source code was obtained and analysed. The modifications required are to transfer all the communication calls from the local network, into the new interface. However, it is also required to provide a reliable channel since these libraries do not provide these facilities. In fact, it is necessary to implement all of the OSI layers [156] up to the transport layer for the proposed network interface, and then link this layer with the corresponding calls of the libraries.

The PCI support is straightforward. It is only needed to develop a driver for the card. The development of this driver was planned until the network interface card was available to allow testing it. However it is necessary to understand how the NI works, and what mechanisms should be used to transfer information between the computer memory and the NI. The next section address these issues.

C.0.3. Analysis of the current network interface

The Network Interface (NI) is responsible for transmitting and receiving information between the network and the processing element (be it the memory, caches or registers inside the CPU). That is, it is responsible for transmitting information from the source processing element, through the network, and into the destination processing element via its own NI, and the other way around.

[2]These are closed libraries such as NX from Intel and Vertex from nCUBE.

In the current design the NI is connected to the PCI bus, which limits bandwidth and increases latency. Furthermore, all communications have to be mediated by the kernel which adds a significant overhead for every transfer (typically this consists of a system call and a memory copy for transmission; and an interrupt, a system call and a memory copy for reception. Each of these steps increase latency, and since they imply a context switch, second order effects such as cache pollution would arise). This overhead imposes a very high penalty for small transfers, and hence limits its usability to long messages (it is suitable only for coarse-grained communications). However, recent research in parallel computing suggests that for scientific applications, the average messages are small in size (average between 19-230 bytes) [99], and hence this kind of applications are severely penalised by the kernel involvement in the transfer, and by any extra latency added, i.e.. PCI bus arbitration, message copying, etc.

This makes the NI unsuitable for building high performance parallel systems. Even in a coarse-grained system, applications will transfer small messages, and if the NI is unsuitable for this purpose, latency will be added, with the result of a general slow down. This will also ill effect the experimental measurement of latency, since a received message can only be time stamped when the system has acknowledge the interrupt, adding uncertainty into the measurement. An alternative is to use polling to receive messages; however in this case it is difficult to test different algorithms running in the nodes, since the processor will be busy polling the NI.

Several alternatives have been proposed, such as the use of Coherent Network Interfaces [109], User Level DMA [100] and block transfer mechanisms. In all of them the kernel is removed from the critical path. The new architecture will be referred here as User Level Communication (ULC), since the messages are transfered directly from user space to the NI and vice-versa. In this architecture, the NI or the user process has to access each others memory directly. If the NI is able to access the user space directly, it needs to know the physical address of the user space. This poses some problems, since normally processes deal only with virtual addresses and physical addresses are only known by the operating system and are not made available to them. In all of the previous solutions either special software techniques and/or custom hardware have been developed to provide a mechanism for the translation of the virtual address to a physical address. However, I will argue that none of these mechanisms are strictly necessary at least for the DCSH Supercomputer.

Due to the problems explained in this section, a new network interface design was proposed. However since it requires additional hardware, and since it is more

complex than the current NIC, it was decided not to modify the current NIC. For these reasons it was decided to suspend the development of the system integration of the Fastrack project, since if the issues outlined here aren't solved, the test-bed provided by the Fastrack project would be useless for fine grain tests which are necessary for this research. Also it was shown that the Fastrack project would provide a very inflexible test-bed in which new architectures and routing schemes could not be analysed.

C.0.4. Design guidelines for a new network interface

The proposed architecture is based on the concept of virtual channel communication. The NI provides a number of virtual channels used to transfer data through these channels directly into user space. Each channel is assigned to a process through an operating system call. This call assigns physical memory for buffering incoming packages of this channel. The channel is assigned to the process by writing the physical address of this memory space into a table in the NI. Once the channel is assigned all interaction between the NI and the user is carried out without any further intervention of the OS. When a message form the Network arrives at a NI, its channel ID (Which is included in the header of the message) is read by a controller, the physical address assigned to that channel is looked up in the table (contained in the NI controller), and the NI transfers the package directly to the user memory space. After the transfer is finished, the NI will update the channel's associated physical address register to point to the next free byte. This effectively acts like a circular buffer in user space.

Once the package is in user space, interrupts or polling can be used to notify the process of its arrival. Interrupts will not be used since they have to be handled by the kernel, thus providing a large overhead [108], [101]. Furthermore, since interrupts are handled by the kernel, a change of context would be necessary, and second order effects such as cache pollution would arise [98], causing the system to slow down even more. For these reasons, the user process will use polling to determine when a new package arrives.

Depending on the application, some of these packages will need to be copied to another location, since if they are left in the circular buffer long enough they would be overwritten. However this copy can be issued by a block movement instruction and will use the high bandwidth main bus. These memory movement instructions can be minimised by adding system memory, or removed by assigning a separate buffer for every *non-erasable* data transfer. When the buffer is full, the user process

issues an OS call which will assign another memory area for that channel.[3] The old memory area would still be accessible to the process.

It is also important to note that the memory used for the receiving buffers will be "pinned" so it cannot be replaced by the operating system virtual memory management. This means that this physical memory will not be used by anything else, and should be considered part of the NI, and not part of the main memory. In the proposed implementation, the user process is also responsible of packetising the information, and assigning checksums and headers. However, this could be performed by a more advanced NI.

When a process wants to send information it has to packetise and copy it to the NI. To allow this, the operating system should provide the process with a virtual address which points to the network interface (this will require some modifications to the operating system). Normally these transfers are implemented with a DMA engine. However it is argued here that this isn't strictly necessary. All the data that will be transferred by an application can be either a result of a calculation, or the result of a block device access (i.e. hard-disk). If it is a result of a calculation, this information would have to be copied from the internal CPU registers to main memory before issuing the DMA cycle. This adds unnecessary latency. If the information will not be required afterwards, this copying can be eliminated by simply writing directly to the NI. It is only necessary to assure that data is sent to the PCI bus fast enough so as to originate burst transfers (this attains the maximum performance of the PCI bus). Since the PCI bus is much slower than the main bus, this is not difficult [157].

If the data that has to be sent comes from a block access device, then it has to be copied to main memory and later on, copied to the NI. However this can't be avoided until these devices are optimised to use a similar scheme in which data is delivered directly into user space, or NI's space (or another block device for that matter).

The only real benefit of using DMA would be that it would allow at least to some extent overlapping of the memory transfers with computation. However, since in a normal OS a DMA transfer has to be mediated by the kernel, and since this adds too much overhead, a mechanism to allow User Level DMA would have to be implemented, as in SHRIMP's UDMA [100]. However this is very costly and requires new hardware. Another option is to use a processor which allows efficient user block transfers like the SPARC does.

[3] It will also have to un-pin the previously used memory. The process of pinning and unpinning memory pages is a costly one. However, this will only have to be done when a *non-erasable* input buffer is full. If enough memory is provided, this would be a rare event.

Memory protection is enforced by this scheme, since the physical addresses of the receiving channels, and the output virtual addresses are assigned by the operating system. Any non-authorized access will be trapped by the memory management unit.

Ideally, the same concept of virtual channels should apply also for the output mechanisms. That is, each process asks for a unique output channel and writes into its own memory location. Unfortunately this is very difficult to implement in the NIC hardware, since it would require too much memory, and far too much logic (or a dedicated processor). In order to keep the design manageable, other mechanisms have to be developed to ensure the integrity of the information.

The problem is as follows; suppose a process is writing data into the NIC, and the operating system decides to preempt it to run another process. If the new process starts writing data into the NIC as well, the data would be mixed up. This is due to the fact that data transfers to the NIC are not done in an atomic fashion. To alleviate this situation, the operating system scheduler can be modified, so that data transfers are made atomically. Also, it is possible to enforce atomicity by using a software semaphore. Normally semaphores are mediated by the operating system. However it is theoretically possible to implement a semaphore without kernel involvement by assigning virtual addresses to all processes which point to the same physical address, and warrant atomicity by using a read-modify-write cycle. This would only be an issue if two or more processes are running, and so only to provide the capability of multiprogramming it would have to be solved. There are other issues to be solved for multiprogramming, such as synchronous scheduling algorithms.

D. DCSH Hardware Proposal

This document describes a new hardware proposal to implement the Network Interface (NI) of the DCSH Supercomputer. This design will allow for higher bandwidth and lower latency, and it supports User Level Communication. Other NIs to be used in the DCSH project have been proposed and constructed. The disadvantages of this designs have been exposed in other documents [150] [158] and will not be repeated here. This design will try to overcome any limitations and use the experience gained in the previous work.

D.1. Introduction

The goal of this project is to build and test a DCSH Supercomputer. This Supercomputer will be constructed with Macintosh PowerPC nodes running a version of the LINUX operating system (LINUXPPC). The nodes will be linked together by a DCSH network, formed by a central switch and NIs connected to the PCs via the PCI bus. In this paper I will concentrate on the design of the NI.

D.1.1. The Problem

In a modern computer the Network Interface (NI) is responsible for transmitting and receiving information between the network and the processing element (be it the memory, caches or registers inside the CPU). That is, it is responsible for transmitting information from the source processing element, through the network, and into the destination processing element via it's own NI, and the other way around. However, trivial as this might seem, most of the modern computers and supercomputers are severely penalised in their performance due to poor design of the NI. This is one of the most critical design issues for future computers. Traditionally a NI is connected to the low performance I/O bus (in our design this is the PCI bus), which limits bandwidth and increases latency. Furthermore, all communications are mediated by the kernel which adds a significant overhead for every transfer (typically this consists of a system call and a memory copy for transmission; and an interrupt,

a system call and a memory copy for reception). This overhead imposes a very high penalty for small transfers, and hence limits it's usability to long messages (it is suitable only for coarse-grained communications). However, recent research in parallel computing suggests that for scientific applications, the average messages are small in size (average between 19-230 bytes) [159], and hence this kind of applications are severely penalised by the kernel involvement in the transfer, and by any extra latency added in the transfer, i.e. PCI bus arbitration, message copying, etc.

This makes traditional NIs unsuitable for building high performance parallel systems. Even in a coarse-latency grained system, applications will transfer small messages, and if the NI is unsuitable for this purpose, latency will be added, with the result of a general slow down.

D.1.2. Proposals

Several alternatives have been proposed, such as the use of Coherent Network Interfaces [160], User Level DMA [100] and block transfer mechanisms. In all of them the kernel is removed from the critical path. The new architecture will be referred in this paper as User Level Communication (ULC), since the messages are transfered directly from user space to the NI and vice-versa. In this architecture, the NI or the user process has to access each others memory directly. If the NI is able to access the user space directly, it needs to know the physical address of the user space. This poses some problems, since normally processes deal only with virtual addresses and physical addresses are only known by the operating system and are not made available to them. In all of the previous solutions either special software techniques and/or custom hardware have been developed to provide a mechanism for the translation of the virtual address to a physical address. However, I will argue that none of these mechanisms are strictly necessary at least for the DCSH Supercomputer.

D.1.3. New Proposal

The proposed architecture is based on the concept of virtual channel communication. The NI provides a number of virtual channels used to transfer data through this channels directly into user space. Each channel is assigned to a process through an operating system call. This call assigns physical memory for buffering incoming packages of this channel. The channel is assigned to the process by writing the physical address of this memory space into a table in the NI. Once the channel is assigned all interaction between the NI and the user is carried out without any

further intervention of the OS. When a message form the Network arrives at a NI, it's channel ID (Which is included in the header of the message) is read by a controller, the physical address assigned to that channel is looked up in the table (contained in the NI controller), and the NI transfers the package directly to the user memory space. After the transfer is finished, the NI will update the channel's associated physical address register to point to the next free byte. This effectively acts like a circular buffer in user space.

Once the package is in user space, interrupts or polling can be used to notify the process of it's arrival. Interrupts will not be used since they have to be handled by the kernel, thus providing a very big overhead [108] [101]. Furthermore, since interrupts are handled by the kernel, a change of context would be necessary, and second order effects as cache pollution would arise [98], causing the system to slow down even more. For this reasons, the user process will use polling to determine when a new package arrives.

Depending on the application, some of these packages will need to be copied to another location, since if they are left in the circular buffer long enough they would be overwritten. However this copy can be issued by a block movement instruction (if available) and will use the high bandwidth main bus. These memory movement instructions can be minimised by adding system memory, or removed by assigning a separate buffer for every *non-erasable* data transfer. When the buffer is full, the user process issues an OS call which will assign another memory area for that channel[1]. The old memory area would still be accessible to the process.

It is also important to note that the memory used for the receiving buffers will be "pinned" so it cannot be replaced by the operating system virtual memory management. This means that this physical memory will not be used by anything else, and should be considered part of the NI, and not part of the main memory. In the proposed implementation, the user process is also responsible of packetising the information, and assigning checksums and headers. However, this could be performed by a more advanced NI.

When a process wants to send information it has to packetise and copy it to the NI. To allow this, the operating system should provide the process with a virtual address which points to the network interface (this will require some modifications to the operating system). Normally these transfers are implemented with a DMA engine. However I argue that this isn't strictly necessary. All the data that will be

[1] It will also have to un-pin the previously used memory. The process of pinning and unpining memory pages is a costly one. However, this will only have to be done when a *non-erasable* input buffer is full. If enough memory is provided, this would be a rare event.

transferred by an application can be either a result of a calculation, or the result of a block device access (i.e. hard-disk). If it is a result of a calculation, this information would have to be copied from the internal CPU registers to main memory before issuing the DMA cycle. This adds unnecessary latency. If the information will not be required afterwards, this copying can be eliminated by simply writing directly to the NI. We only have to assure that data is sent to the PCI bus fast enough so as to originate burst transfers (this attains the maximum performance of the PCI bus). Since the PCI bus is much slower than the main bus, this is not difficult [157].

If the data that has to be sent comes from a block access device, then it has to be copied to main memory and later on, copied to the NI. However this can't be avoided until these devices are optimised to use a similar scheme in which data is delivered directly into user space, or NI's space (or another block device for that matter).

The only real benefit of using DMA would be that it would allow at least to some extent overlapping of the memory transfers with computation. However, since in a normal OS a DMA transfer has to be mediated by the kernel, and since this adds too much overhead, a mechanism to allow User Level DMA would have to be implemented, as in SHRIMP's UDMA [100]. This is not an easy task, and I don't think it's justifiable for our project, since what we want is to evaluate the performance of the network. However this would have to be solved to implement a real high performance system. One option is to use a processor which allows efficient block transfers (i.e. SPARC). Another way of going around this is to use a mechanism as SHRIMP's UDMA or a similar one.

Memory protection is enforced by this scheme, since the physical addresses of the receiving channels, and the output virtual addresses are assigned by the operating system. Protection is achieved by the same mechanisms that enforce normal memory protection.

Ideally, the same concept of virtual channels should apply also for the output mechanisms. That is, each process asks for a unique output channel and writes into its own memory location. Unfortunately this is very difficult to implement in the NIC hardware, since it would require too much memory, and far too much logic (or a dedicated processor). In order to keep the design manageable, other mechanisms have to be developed to ensure the integrity of the information.

The problem is as follows; suppose a process is writing data into the NIC, and the operating system decides to preempt it to run another process. If the new process starts writing data into the NIC as well, the data would be mixed up. This is due to the fact that data transfers to the NIC are not done in an atomic fashion. To alleviate

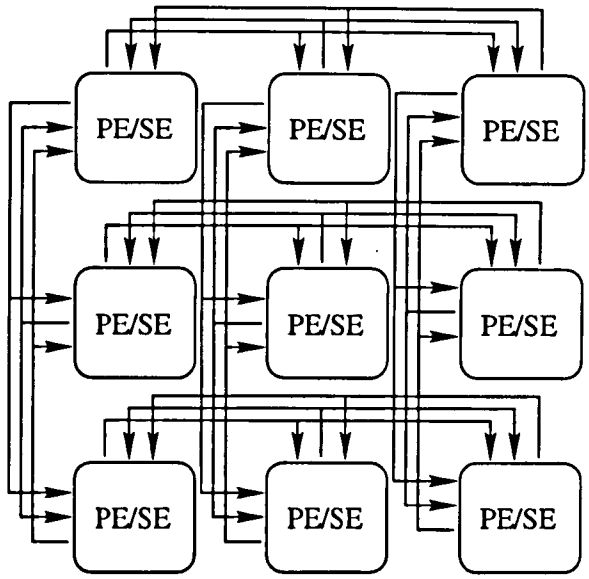


Figure D.1.: A 3 x 3 DCSH Network.

this situation, the operating system scheduler can be modified, so that data transfers are made atomically. Also, it is possible to enforce atomicity by using a software semaphore. Normally semaphores are mediated by the operating system. However it is theoretically possible to implement a semaphore without kernel involvement by assigning virtual addresses to all processes which point to the same physical address, and warrant atomicity by using a read-modify-write cycle. This would only be an issue if two or more processes are running, and so only to provide the capability of multiprogramming it would have to be solved. There are other issues to be solved for multiprogramming, such as synchronous scheduling algorithms.

D.2. NI Design

The general architecture of the DCSH [51] is shown in figure D.1. Each node in the network will be formed by a PowerPC and a Network Interface Card (NIC). The NIC will use packet switching, but could also support virtual cut-through and wormhole routing. This is shown in figure D.2. The main difference between this design and the one that uses wormhole routing is the size of the queues. This means that we can also test wormhole routing by varying their size. It is intended to implement most of this design using programmable logic, so it is easy to modify.

To simplify the design, it was proposed to modify the original design so that each SE has less inputs, and multiplexing is done externally to the SE [150], using a non-

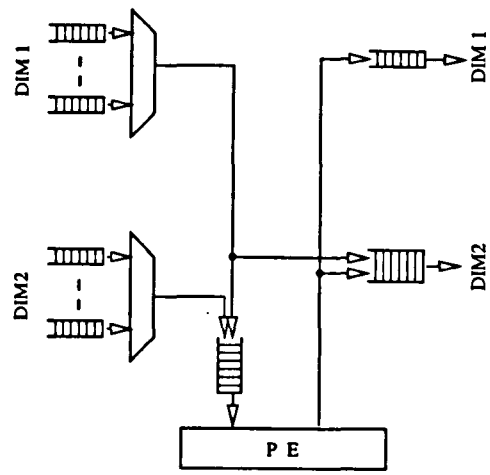


Figure D.2.: SE design using Packet or VC switching.

blocking switch. To further simplify the design James Beeley proposed to partition it into two different network interface cards for each SE. Each card has an input and an output, which will be connected to different dimensions (i.e. one will be connected to the X cluster, and the other to the Y cluster). Only one of the cards is required to have routing capabilities. However both cards will share the same design, the only difference being the code loaded into the programmable logic (and only if this proves to be strictly necessary). This is shown if figure D.3.

Since the limitation of our NI design is the PCI bus (it would be very interesting for further work to place the NI closer to the CPU, obtaining a much higher bandwidth with less latency), it is a good place to start the design.

The maximum throughput of the PC's PCI bus is 132 MB/s when used in burst transfer mode. However, it is not possible to maintain this transfer rate for long periods, since other devices are connected to the bus, and to warrant a fair access for every device the PCI specification doesn't allow long sustained transfers. A more realistic value would be somewhere around 100 MB/s, which is the goal of this design.

Since the PCI bus is 32 bits wide, and uses a clock of 33 MHz, it would be advantageous to maintain this wide data-path to minimise the speed requirements within the card. This will be another design goal.

A general diagram of the design is shown in figure D.4.

For a clearer explanation, the following design discussion will be divided into output, input, PCI interface and control sections.

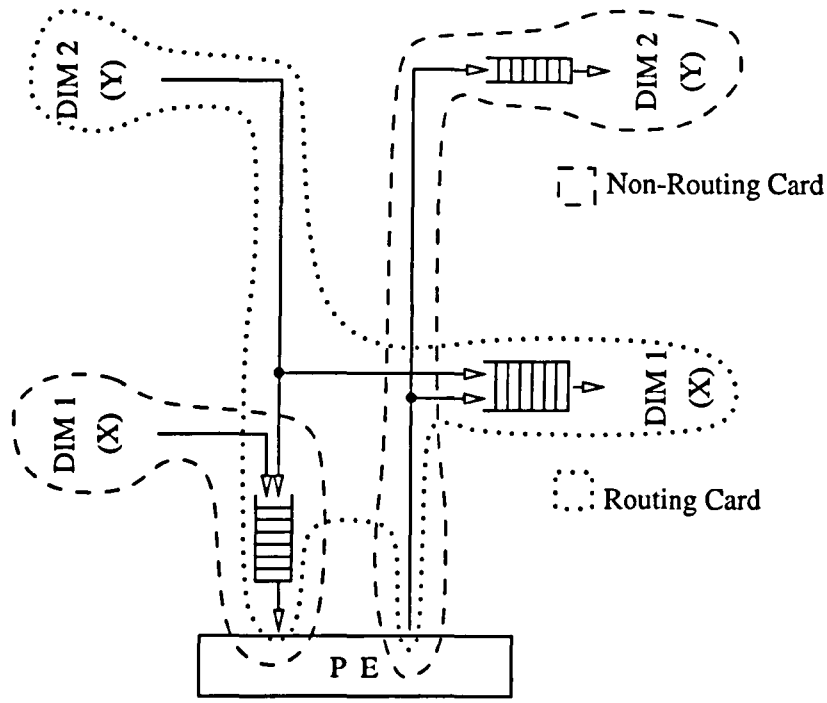


Figure D.3.: Beeley's proposed partition.

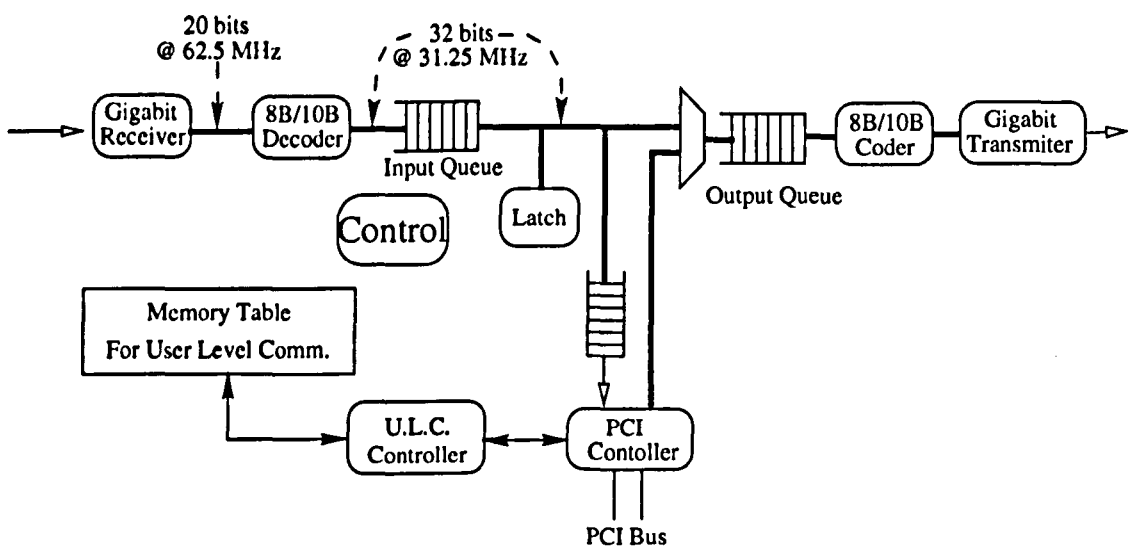


Figure D.4.: General diagram.

D.3. Output

The packets to be transmitted are queued at the output queue. They are either local packets which come from the PCI bus, or they are routed packages. The input to the queue is fitted with a multiplexer which selects the source, and restricts transit to complete packages only (so different flits/packages are not mixed). Achieving this is not a simple task, since the NIC must have enough intelligence to determine where a package starts, and where it ends. There are different alternatives:

1. The use of special "markers" to determine the header and the trailer. This special markers are supported by the hardware so they differ from normal information transmission. The problem of using this technique is that the hardware extensions do not map naturally into the PCI bus, and therefore special transmission techniques have to be developed (like writing into privileged registers before writing the header or trailer). This is a hardware intensive solution, and hence it is difficult to implement.

2. The use of logic and counters to determine the length of the packet, and hence where it starts and ends.

3. The use of certain patterns as identifiers. In this approach the naturally occurring patterns have to be specially coded to avoid misinterpretation. For this reason this scheme is more suitable for pure software solutions. This solution can make the hardware very simple (depending on what pattern is chosen).

4. The use of a fixed size packet. Using a fixed size, it is simple to determine where a packet starts and ends. This approach is a variant of (2), in which the size of the transfer is not present in the package itself, but is known beforehand by the hardware.

Any of the above could be used, however since one of the design goals is to make simple hardware, we will use initially the fixed packet approach. This is not the simplest approach, however it is still very simple, and will make software much simpler. Once the card is running smoothly, it would be desirable to change it to allow dynamic variable length packets. The only modification needed would be to change the logic of the PLDs to accommodate for this.

Since the output of the NI will be a high speed serial link, it is essential to maintain a zero DC level on it. To do this, the information will be coded using a balanced 8B/10B code. This means that the transceiver bit rate should be 10 times

the incoming byte rate (10 X 100 = 1000 Mbps). There are numerous transceivers designed to work at this speed, since they are used in gigabit ethernet cards.[2] The obvious advantage is that they are and will continue to be readily available. However a major disadvantage of this circuits is that they don't incorporate an 8B/10B coder/decoder, and hence this logic will have to be implemented in another device.

Alternatively 4 parallel connected TAXI chips could be used. However, they would use too much real estate (8 chips per card), and require 4 connectors for each link. Also the link would be slower (70 MB/s).

D.4. Input

The receiver chip will recover the clock and the data. The output is 20 bits of 8B/10B coded data, which must be decoded and de-multiplexed to form one long word (32 bits wide). The receiver will output this 20 bit words with a frequency of 62.5 MHz, and after they are decoded and assembled into long words, the frequency will decrease to 31.25 MHz. Since this logic must operate at a relative high frequency, it could prove useful to implement it with a single dedicated PLD. This logic is also in charge of providing the flow control signals which will be embedded in the communication (they will use some of the unassigned 8B/10B codes). By using this codes, flow control can be achieved without the use of an external control link.

Once data is recovered it is routed towards the proper queue (assuming it arrives in a *Routing* NIC). If the destination is to the other dimension cluster, the packet is stored in the proper queue, and when the output arbiter permits, it will be transmitted to its final destination.

On the other hand, if the destination is the local PE, the packet enters the queue, and at the output, the local controller reads the virtual channel number from the header of the packet, and checks its local table to find out the physical address of the destination. Then the packet is sent to the PCI interface chip, and finally transmitted to the appropriate physical address. After finishing the transfer, the table is updated.

The number of entries in the table will determine the maximum number of input channels available. This table and the controller will be implemented with a fast micro-controller. The technology of this implementation will determine the number of virtual channels (this is not a very critical factor, since the optimum value will be

[2]Gigabit ethernet uses a bit rate of 1.25 Gbps, to achieve a 1 GB/s throughput. By using this bit rate, the speed would be 125 MB/s; that is 25 % faster than the required 100 MB/s.

system dependent, and remains an open issue. It would be convenient to provide a reasonable number of channels, to allow further research).

D.5. PCI

The PCI controller provides the bridge between the local processor (PE) and the NIC. The PCI controller must support certain operation modes in order to be able to work in this design. First of all, it should be capable of acting as a master, since it will trigger a memory transfer when a data packet arrives at the NIC.

It is also necessary that it supports the write and invalidate transaction as an originator (the S5933 doesn't). There are two different commands for transferring data between the PCI bus and the main bus. They are the write transaction and the write and invalidate transaction. The main difference between them, is how the cache reacts to the transfer. In a normal system, when data is transferred from a master other than the CPU, the cache snoops the transaction to determine if any of the data it holds should be modified by the transaction. If the destination address is currently cached, it forces the current memory cycle to abort, writes the modified line into main memory, and resumes the previous bus cycle (if it didn't do this, data coherency could be lost). The problem is that this sequence of events slow down the transfers, and lower the data throughput.

For this reason, data transfers should be done using the write and invalidate command. In this way, although the cache might have a snoop hit during the transfer, it doesn't force an abort and save operation, since *all* the cache line is rewritten (as opposed to just a subset as in the previous case). This maximises throughput. However, all data transfers should consist of multiples of the cache line size (32 bytes) [157]. If a packet is smaller than this, it could be transferred with a normal write cycle, or using the 32 bytes cache line transfer wasting some of this bytes.

An alternative to the write and invalidate transaction would be to mark the receiving buffer memory locations as non-cachable. However this would cause a big slow down of the process when accessing the newly arrived data. If the received data is copied to a new cachable memory location to accelerate its access, the result is added latency and poor use of memory.

Not all PCI transactions will be destined to the network. Some of them will be control sequences to load the ULC table. The PCI controller must allow writing this configuration information, through special mailboxes or other transfer mechanisms.

Normally this is not a problem since the PCI bus differentiates between configuration and data space, and PCI controllers support this.

E. Constructing a large DCSH network.

The issues of designing and constructing a coarse grain DCSH network have been reviewed extensively in the Fastrack project. However no work has been done in a fine grain DCSH network. In this section those issues are analysed, and alternative implementations are proposed.

The main problem of designing a network for a parallel computer is to assign finite resources to an otherwise unrestricted problem. There is a wide range of design choices, which must be balanced to obtain the best network possible. However, it is not straightforward, if possible at all, to give a weight or marking to the different alternatives. What might prove very convenient for a specific topology, might be completely useless in another. In [51], Ould-Khaoua gives an example in which two authors arrive at opposite conclusions on the performance of a network, simply because they had different technological assumptions.

In constructing a large scale DCSH network a number of problems have to be addressed. All of the previous work in DCSH networks has been focused towards coarse grained systems, and some of the previous results cannot be directly applied into a large scale, fine grained system.

What follows is a discussion of some of these issues.

E.1. Pin-Out

The technology used to construct the DCSH will dictate various design parameters; such as maximum wiring density, and if the network is partitioned in different chips, also the pin-out of these chips. However, even in the case where the network is entirely implemented in a single chip, the number of channels connected to a node will be limited by technological factors. It is therefore important to minimize the number of channels arriving at a node, specially since it could be particularly high for certain combinations of cluster size and network dimensions.

In his work, Ould-Khaoua proposes a scheme in which pin-out and wiring density constrains are minimised by implementing the DCSH in a layered fashion. His method is based in using slices carrying the communication channels and a sub-partition of the SE called CIU. The processing layer is "sandwiched" between these clusters which run in different directions (one direction for each dimension). The pin-out is minimised because only a small number of channels have to emerge to connect the PE, and also because the cluster is formed from a number of slices; so each CIU chip can effectively be reduced in size since it doesn't have to deal with all the wires, but just a subset of them.

However this scheme is not well suited to be used in a large network. First of all, because this partitioning cannot be carried out in an essentially two dimensional medium, such as VLSI, and also because this scheme is only suitable for 2-D or 3-D systems, since these are the ones that map naturally into space.

In an n-dimensional regular DCSH architecture with k nodes per side, the network size is given by $N = k^n$. Thus, the number of channels arriving to a SE in the network is given by $l = nN^{1/n}$, which can be substantial for large values of N. It seems obvious that if a high number of nodes is to be linked, a high dimensionality network must be employed to minimize the number of channels arriving to the nodes.

The minimum number of channels arriving at a SE, is given when $n = \ln N$, and is[1]

$$l_{opt} = e \ln N \tag{E.1}$$

E.2. Switch Complexity

Since the network is going to be formed by a large number of nodes, it is important that their complexity be kept low. Part of the cost of the node has already been analysed and minimised (pin-out), but perhaps the switch complexity is a more important parameter in the network, since its size impact the total cost directly.

The switch complexity can be estimated using the number of internal routes through it, and for the restricted routing case it is given by

[1] In reality $k, n \in \aleph$, so they have to be approximated to an integer value. Also, since in general there are no $n, k \in \aleph$ that satisfy $k^n = N$, then

$$l' = \frac{1}{n} \sum_{x=1}^{n} k_x; \qquad \prod_{x=1}^{n} k_x = N$$

However $\min l \leq \min l'$ and therefore l' is *at least* equal to $e \ln N$.

$$C_{se} = n + nk + \sum_{i=1}^{n-1} ink \tag{E.2}$$

We want to minimise the total switch complexity, that is

$$C = NC_{se} = N[n + nk + \sum_{i=1}^{n-1} ink] \tag{E.3}$$

The minimum is obtained when $n = n_{opt}$ which can be approximated as

$$n_{opt} \approx \frac{2+3\ln N}{9} \tag{E.4}$$

and is given by[2]

$$C_{opt} = \frac{n_{opt}^3 - n_{opt}^2 + n_{opt}}{2} N^{\frac{1+n_{opt}}{n_{opt}}} \tag{E.6}$$

The switch complexity per node rises as the cube of the logarithm of the size of the network for large values of N. This means that the resulting network is expandable, although it might be difficult or costly to implement, specially if it is compared against simpler and less powerful networks. The resulting cluster size grows slowly and tends to $e^3 \approx 20$ when $N \rightarrow \infty$. This seems like a very reasonable size.[3]

The average distance between any pair of nodes, again for $N \rightarrow \infty$ is given by

$$d = \frac{\ln N}{3} \tag{E.7}$$

This is a very important metric, since it will affect directly the latency and the traffic in the network; and as it is shown, it only grows as the logarithm of the network size.

Although this network has very good characteristics, it has the drawback of a complex switching element. In some cases, to sacrifice some power in order to gain simplicity might prove to be advantageous, specially if that power is not strictly required to begin with.

For this situations, a new class of network, namely a recursive DCSH (RDCSH) is

[2]This equation can be approximated for large values of N as

$$C_{opt} = \frac{e^3 N(\ln N)^3}{54} \tag{E.5}$$

[3]It is important that the cluster size is limited, since it is necessary to send the address of the recipient in each message, and if the cluster size was allowed to grow, then it would increment the header size of the packages, and hence the traffic and complexity of the network.

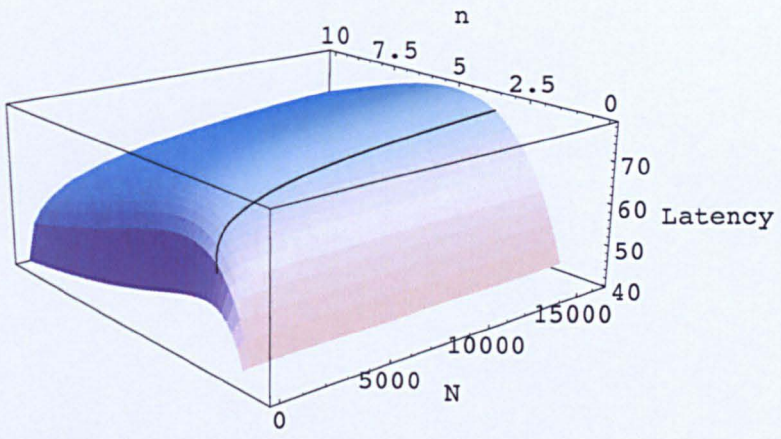


Figure E.1.: Latency in the network for a 32 flits message size, with m=0.01. The line shows the case where the switch complexity is minimum ($n = n_{opt}$).

proposed. The emphasis of this architecture is to allow for simpler DCSH-like networks which maintain some of the advantages of the DCSH network while providing the flexibility to choose between performance and cost.

E.3. Latency

The latency of the DCSH network has been extensively studied in [51]. The optimal routing algorithm is virtual cut-through (VCT), specially if the dimensionality of the network is high, so we will base our analysis on it. As fig. E.1 shows, the latency is minimum for very low values of n, but of course, the switch complexity is very large for this values.

As it is shown, the corresponding latency for the optimised network is almost at the peak of the graph. Nevertheless, the latency is almost the same as that of a 2-D or a 3-D hypermesh, since these choices lie almost at the peak of the graph as well ($n_{opt}(N = 15000) \approx 3.5$). Lower latency values can be obtained by migrating the network towards a higher dimensionality, at an increased system cost.

E.4. Recursive DCSH (RDCSH)

To allow for a flexible architecture that can trade off performance for simplicity as desired, this new architecture is proposed. The main motivation to propose this architecture is that although the DCSH provides excellent performance, its complexity can be too much for some applications. For example, in a two dimensional DCSH,

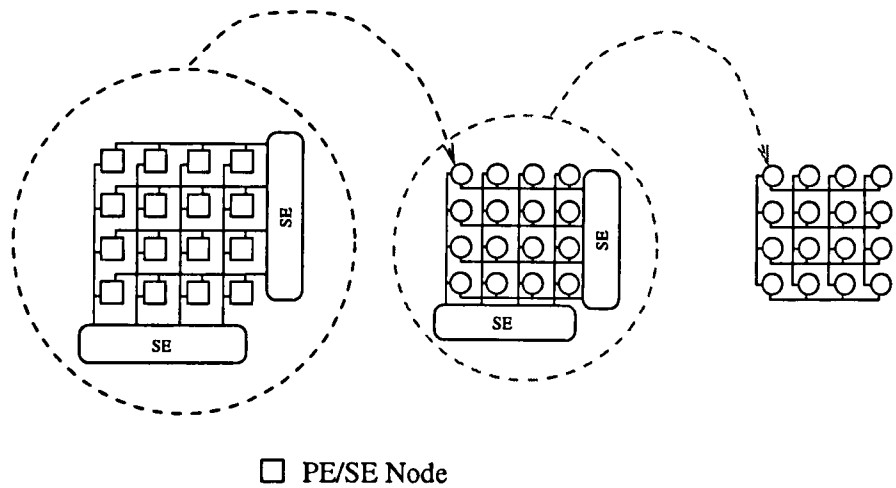


Figure E.2.: A 2D RDCSH.

$k = \sqrt{N}$ and the switch complexity is $C_{se} = 2\sqrt{N}$. The complexity is high due to the fact that all the k nodes in the adjacent clusters have a direct connection to the node. In E.2 it is proposed to rise the dimensionality of the network to provide a less complex switch, but the minimum obtained can still be substantial.

In the RDCSH one cluster is partitioned in a number of sets, each forming a DCSH. Then these sets are equipped with a SE, and are viewed as a PE/SE node for the next recursion level, and they are connected in a DCSH fashion. This process can be repeated several times, increasing the level of recursion (see fig.E.2).

In this architecture, a new variable is defined, r which is the number of recursive levels in the network. When $r = 1$ the normal DCSH is obtained.

E.4.1. Switch Complexity

There are two different types of switches in the RDCSH:

- Normal PE/SE switches (present only in the first level of recursion)
- Recursive switches which are those that permit communication between the different recursion levels.

The number of normal PE/SE switches in the network S_{se} is the same as N, the number of processors, and is given by

$$S_{se} = N = k^{rn} \tag{E.8}$$

Where n is the network dimension, and k is the cluster size, which are the same for every level of recursion. The complexity of these switches is given by

$$C_{se} = \begin{cases} nk & r = 1 \\ n(k+1) & r > 1 \end{cases} \tag{E.9}$$

Where $k+1$ accounts for the fact that there is an added channel which connects the nodes to the recursive switches.

The number of recursive switches is given as

$$S_{fs} = n \sum_{x=1}^{r-1} k^{xn} \tag{E.10}$$

The upper limit of eq. E.10 is $r-1$, since these switches are not present on the top recursion. The number of connections to the lower side of these switches is $(k+1)k^{n-1}$, since there are k^{n-1} clusters connected to it, and each cluster has $k+1$ channels. On the other side of the switch, there are $k+1$ channels (the whole lower-recursive layer can be viewed as a normal PE/SE in the next recursive layer). Therefore the complexity of the recursive switches is

$$C_{fs} = (k+1)^2 k^{n-1} \tag{E.11}$$

The total complexity can be written as

$$C_f = \begin{cases} nk^{n+1} & r = 1 \\ n(k+1)\left[k^{nr} + \frac{k^{n-1}(k+1)(k^{nr}-k^n)}{k^n-1}\right] & r > 1 \end{cases} \tag{E.12}$$

However, a more interesting measure is the complexity per PE, defined as the ratio between the total complexity and the number of PEs which yields[4]

$$\overline{C_f} = \begin{cases} nN^{1+\frac{1}{n}} & r = 1 \\ \frac{n(1+N^{\frac{1}{nr}})}{N}\left[N + \frac{N^{\frac{n-1}{nr}}(N-N^{\frac{1}{r}})(1+N^{\frac{1}{nr}})}{N^{\frac{1}{r}}-1}\right] & r > 1 \end{cases} \tag{E.13}$$

The minimum switch complexity is obtained when $n = 1$ and $r = r_{opt}$.[5] For this values, the switch complexity is approximated by

[4] Substituting $k \to N^{\frac{1}{nr}}$ from eq.E.8.

[5] There are no analytical solutions to $\frac{d\overline{C_f}|_{n=1}}{dr} = 0$. However r_{opt} can be approximated by $r_{opt} = \frac{\ln N}{\ln(1+\sqrt{2})}$. The exact solution is given by the most positive real root of $2N^{\frac{2}{r}} + 2N(N^{-\frac{1}{r}} - N^{\frac{1}{r}}) - N^{-\frac{1}{r}} - N^{\frac{1}{r}} + 4N - 4 = 0$. This approximations, and the values thereof are accurate for relatively large values of N (approx. $N > 15$).

$$\overline{C_{f_{opt}}} = 6 + 4\sqrt{2} \tag{E.14}$$

Compare this value to that of eq. E.5. The switch complexity has been made independent of the number of processors.

F. Characteristics of some Interconnection Networks

Graph	Size	Degree	Length	Diameter	Bisection Width
D-Mesh	k^D	$k = \sqrt[D]{N}$	$\frac{D}{3}\left(k - \frac{1}{k}\right)\frac{N}{N-1}$	$D(k-1)$	k^{D-1}
D-Torus	k^D	$k = \sqrt[D]{N}$	$\frac{Dk}{4}\frac{N}{N-1}$	$D\left\lfloor\frac{k}{D}\right\rfloor$	$2k^{D-1}$
k-Ring	N	$2k$	$\frac{N-1}{2k} + \frac{1}{2}$	$\left\lfloor\frac{N-1}{2k} + \frac{1}{2}\right\rfloor$	$k(k+1)$
D-Hypercube	2^D	D	$\frac{D}{2}\frac{N}{N-1}$	D	$2^{D-1} = \frac{N}{2}$
k-ary D-cube	k^D	$2D$	$\frac{D(k-1)}{2}\frac{N}{N-1}$	$\frac{Dk}{2}$	$k^{D/2-1} = \frac{2N}{k}$

Table F.1.: Characteristics of some interconnection networks.

Bibliography

[1] C.C. Mann. The end of Moore's law? *MIT Technology Review*, May 2000.

[2] Semiconductor Industry Association. *The National Technology Roadmap For Semiconductors*, 1997.

[3] P.W. Dowd and K. Jabbour. Spanning multiaccess channel hypercube computer interconnection. *IEEE Trans. Computers*, 37(9):1137–1142, September 1988.

[4] A. Louri, B. Weech, and C. Neocleous. A spanning multichannel linked hypercube: A gradually scalable optical interconnection network for massively parallel computing. *IEEE Transactions on Parallel and Distributed Systems*, 9(5):497–512, May 1998.

[5] W.J. Dally. Express cubes: Improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. Computers*, 40:1016–1023, 1991.

[6] Mark J. Clement, Bryan S. Morse, J. Kelly Flanagan, and Wei Wei. The chordal spoke atm interconnection network.

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

[8] M.E.J. Newman and D.J. Watts. Scaling and percolation in the small-world network model. *Physical Review E*, 60(6):7332–7342, December 1999.

[9] M.E.J. Newman. Models of the small world. *J.Stat.Phys.*, 101:819–841, 2000.

[10] Rajesh Kasturirangan. Multiple scales in small-world networks. Technical Report AIM-1663, MIT, 1999. http://portal.acm.org/citation.cfm?id=889200.

[11] *Computer Architecture and Parallel Processing.* McGraw-Hill, 1984.

[12] G.M. Amdahl. Validity of the single processor approach to achieving large-scale computing capabilities. In *AFIPS Conf. Proc.*, pages 483–485.

[13] L.N. Bhuyan, Q. Yang, and D.P. Agrawal. Performance of multiprocessor interconnection networks. *IEEE Computer*, 22(2):25–37, 1989.

[14] Junming Xu. *Topological Structure and Analysis of Interconnection Networks.* Kluwer Academic Publishers, 2001.

[15] Albert Y.H. Zomaya, editor. *Parallel and Distributed Computing Handbook.* McGraw-Hill, 1996.

[16] W.H. Kautz. Design of optimal interconnection networks for multiprocessors. In *Architecture and Design of Digital Computers*, pages 249–272. 1969.

[17] G.J. Lipovski and M. Malek. *Parallel Computing.* Wiley-Interscience, 1987.

[18] G. Ravindran and M. Stumm. Issues in the design of direct multiprocessor networks. Department of Electrical and Computer Engineering, University of Toronto. http://www.eecg.toronto.edu.

[19] L.D. Wittie. Communication structures for large networks of microcomputers. *IEEE Trans. Computers*, 30(4):264–273, April 1981.

[20] W.J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Computers*, 39(6):775–785, June 1990.

[21] A. Ahmed. A novel hypercube with lower latency. *1994 International Symposium on Parallel Architectures, Algorithms and Networks*, pages 413–420, 1994.

[22] S.F. Nugent. The iPSC/2 direct-connect communication technology. In *Proc. Conf. Hypercube Concurrent Computers and Applications*, pages 51–60, 1989.

[23] M. Noakes and W.J. Dally. System design of the J-machine. *Advanced Research in VLSI*, 1990.

[24] T. Boku, K. Itakura, H. Nakamura, and K. Nakazawa. CP-PACS: A massively parallel processor for large scale scientific calculations. In *Proceedings ACM Supercomputing 97*, pages 108–115, 1997.

[25] H. Fuji et al. Architecture and performance of the Hitachi SR2201 massively parallel processor system. In *Proceedings parallel processing symposium*, pages 233–241, 1997.

[26] U. Ramachandran, H. Venkateswaran, A. Sivasubramaniam, and A. Singla. Issues in understanding the scalability of parallel systems. In *Proceedings of the First International Workshop on Parallel Processing (IWPP-94)*, pages 399–404, Bangalore, India, December 1994 1995. IEEE Computer Society, McGraw-Hill.

[27] M.J. Flynn. Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909, 1966.

[28] Yakov Fet. *Parallel Processing in Cellular Arrays.* Research Studies Press LTD, 1995.

[29] W.D. Hillis. *The connection machine.* PhD thesis, Massachusetts Institute of Technology, 1985.

[30] S. Abraham and K. Padmanabhan. The twisted cube topology for multiprocesors: A study in network asymmetry. *Parallel and Distributed Computing*, 13:104–110, 1991.

[31] L.M. Mackenzie, M. Ould-Khaoua, and R.J. Sutherland. Cobra: A high-performance interconnection for large multicomputers. Technical report, Department of Computing Science, University of Glasgow, October 1991.

[32] D.A. Carlson. Modified mesh-connected parallel computers. *IEEE Trans. Computers*, 37(10):1315–1321, October 1988.

[33] Casavant et al. *Parallel Computers: Theory and Practice.* IEEE Computer Society Press, 1996.

[34] C.L. Seitz. The cosmic cube. *Comm. ACM*, 28(1):22–33, 1985.

[35] D. Lenoski, J. Laudon, K. Gharacholoo, A. Gupta, J. Henessey, M. Horowitz, and M. Lam. The stanford DASH multiprocessor. *IEEE Computer*, pages 63–79, March 1992.

[36] P.W. Dowd, K. Boginenei, K.A. Aly, and J.A. Perreault. Hierarchical scalable photonic architectures for high-performance processor interconnection. *IEEE Trans. Computers*, 42(9):1105–1120, September 1993.

[37] P.W. Dowd. Wavelength division multiple access channel hypercube processor interconnection. *IEEE Trans. Computers*, 41(10):1223–1241, October 1992.

[38] Z. Guo et al. Pipelined-communication in optically-interconnected arrays. *Journal of Parallel and Distributed Computing*, 12:269–282, 1991.

[39] S.L. Johnson. Data motion and high performance computing. *Proceedings Int'l. Workshop Parallel Processing Using Optical Interconnections*, pages 1–19, 1994.

[40] W.S. Lacy, J.L. Cruz-Rivera, and D.S. Wills. The offset cube: A three-dimensional multicomputer network topology using through-wafer optics. *IEEE Transactions on Parallel and Distributed Systems*, 9(9):893–908, September 1998.

[41] P. Cull and S.M. Larson. The Möbius cubes. *IEEE Transactions on Computers*, 44(5):647–659, 1995.

[42] H.J. Siegel, T. Schwederski, N.J. Davis, and J.T. Kuehn. PASM: A reconfigurable parallel system for image processing. *ACM SIGARCH Newsletter*, 12(4), September 1984.

[43] N. Tanabe et al. Base-m n-cube: High performance interconnection networks for highly parallel computers. PRODIGY. In *Proceedings of the International Conference on Parallel Processing*, pages 509–516, 1991.

[44] G.F. Pfister, W.C. Brantley, D.A. George, S.L. Harvey, W.J. Kleinfelder, K.P. MacAuliffe, E.A. Melton, V.A. Norton, and J. Weiss. The IBM research parallel processor prototype (RP3); introduction and architecture. In *Proceedings of the 1985 International Conference of Parallel Processing*, pages 764–761. IEEE, 1985.

[45] Y. Saad and M.H. Schultz. Topological properties of hypercubes. *IEEE Trans. Computers*, 37(7):867–872, July 1988.

[46] T. Szymanksi. "Hypermeshes": Optical networks for parallel computing. *Journal of Parallel and Distributed Computing*, 26:1–23, 1995.

[47] A. Gottlieb, R. Grishman, C. Kruskal, P. McAuliffe, L. Rudolph, and M. Snir. The NYU ultracomputer — designing an MIMD shared memory parallel computer. In *9th Annual International Conference on Computer Architecture*, pages 27–42, 1982.

[48] B. Webb and A. Louri. A class of highly scalable optical crossbar-connected interconnection networks (SOCNs) for parallel computing systems. *IEEE. Trans. Par. Dis. Syst.*, 11(5):444–458, May 2000.

[49] J. Wu. Extended Fibonacci cubes. *IEEE Trans. Par. Dist. Syst.*, 8(12):1203–1210, 1997.

[50] S.K. Yun and K.H. Park. Hierarchical hypercube networks (HHN) for massively parallel computers. *Journal of Par. and Dist. Comp.*, 37:194–199, 1996.

[51] M. Ould-Khaoua. *Hypergraph-based interconnection networks for large multicomputers.* PhD thesis, University of Glasgow, September 1994.

[52] Gramß, Bornholdt, Groß, Mitchell, and Pellizzari. *Non-Standard Computation.* Wiley-VCH, 1998.

[53] G. Ramanathan and J. Oren. Survey of commercial parallel machines. *ACM SIGARCH Computer Architecture News*, 21(3):13–33, 1993.

[54] T. Leighton, D. Lisinski, and B. Maggs. Empirical evaluation of randomly-wired multistage networks. In *Proceedings of the 1990 Int. Conf. on Computer Design (ICCD)*, pages 380–385, September 1990.

[55] W.D. Hillis. The connection machine: A computer architecture based on cellular automata. *Physica*, 10:213–228, 1984.

[56] P.W. Dowd. High performance interprocessor communication through optical wavelength division multiple access channels. In *Proc. Int. Symposium of Computer Architecture*, 1991.

[57] S.R. Ohring, M. Ibel, S.K. Das, and M.J. Kumar. On generalized fat trees. In *Proceedings of the 9th Internation Parallel Processing Symposium (IPPS'95)*, pages 37–44, 1995.

[58] Patrick W. Dowd, Kalyani Bogineni, Khaled A. Aly, and James A. Perreault. Hierarchical scalable photonic architectures for high-performance processor interconnection. *IEEE Transactions on Computers*, 42(9):1105–1120, 1993.

[59] S. Frank, J. Rothnie, and H. Burkhardt. The KSR1: Bridging the gap between shared memory and MPPS. In *Proceedings Compcon'93*, San Francisco, CA., February 1993.

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

[61] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Carl R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W. Daniel Hillis, Bradley C. Kuszmaul, Margaret A. St Pierre, David S. Wells, Monica C. Wong-Chan, Shaw-Wen Yang, and Robert Zak. The network architecture of the Connection Machine CM-5. *Journal of Parallel and Distributed Computing*, 33(2):145–158, 1996.

[62] Fabrizio Petrini and Marco Vanneschi. K-ary N-trees: High performance networks for massively parallel architectures. Technical Report TR-95-18, 1995.

[63] M. Valerio, L. Moser, and P. Melliar-Smith. Recursively scalable fat-trees as interconnection networks, 1994.

[64] K.J. Liszka, J.K. Antonio, and H.J. Siegel. Is an alligator better than an armadillo? *IEEE Concurrency*, pages 18–28, October 1997.

[65] C. Berge. *Graphs and Hypergraphs.* North-Holland, 1977.

[66] J.C. Bermond, R.W. Dawes, and F.O. Ergincan. De Bruijn and Kautz bus networks. *Networks*, 30(3):205–218, 1997.

[67] B. Bollobás and F.R.K. Chung. The diameter of a cycle plus a random mathcing. *SIAM J. Discrete Mathematics*, 1988.

[68] Qiaoliang Li and Qiao Li. Reliability analysis of circulant graphs. *Networks*, 31:61–65, 1998.

[69] F.P. Preparata and J. Vuilemin. The cube-connected cycles: A versatile network for parallel computation. *Communications of the ACM*, 24(5), May 1981.

[70] S. Loucif, M. Ould-Khaoua, and L.M. Mackenzie. On the performance merits of bypass channels in hypermeshes and k-ary n-cubes. *The Computer Journal*, 42(1):63–72, 1999.

[71] M.R. Samathan and D.K. Pradhan. The De Bruijn multiprocessor network: A versatile parallel processing and sorting network for vlsi. *IEEE Transactions on Computers*, 38(4):567–581, 1989.

[72] A. Cayley. The theory of graphs, graphical representation. *Mathematical Papers*, (10):26–28, 1895.

[73] W.R. Scott. *Group Theory.* Dover, 1987.

[74] F. Harary. *Graph Theory.* Addison-Wesley, 1994.

[75] Fabrizio Petrini, Eitan Frachtenberg, Adolfy Hoisie, and Salvador Coll. Performance evaluation of the quadrics interconnection network. *Journal of Cluster Computing*, 6(2):125–142, April 2003.

[76] Infiniband Trade Association. Infiniband Technology Overview.

[77] RapidIO Trade Association. Parts I,II,III, IV: I/O Logical, Messaging, Transport and Parallel Physical Layer Specifications.

[78] M. Ould-Khaoua and R. Sotudeh. Performance evaluation of hypermeshes and meshes with wormhole routing. *Journal of Systems Architecture*, 43:345–353, 1997.

[79] K. Itakura, M. Hattori, T. Boku, H. Nakamura, and K. Nakazawa. Preliminary evaluation of NAS parallel benchmarks on CP-PACS. Technical report, Institute of Information Sciences and Electronics, University of Tsukuba, 1995.

[80] J.C. Bermond, C. Delorme, and G. Fahri. Large graphs with given degree and diameter. *Journal of Combinatorial Theory*, 36:32–48, 1984.

[81] F. Comellas and J. Gómez. New large graphs with given degree and diameter. *JournaGraph Theory*, 1985.

[82] J.C. Bermond, C Delorme, and J.J. Quisquater. Strategies for interconnection networks: Some methods from graph theory. *Parallel and Distributed Processing*, 3:433–449, 1986.

[83] J.C. Bermond, C. Delorme, and D.F. Hsu. Distributed loop computer networks: a survey. *Journal of Par. and Dist. Comp.*, (24):2–10, 1995.

[84] F. Comellas and J. Gómez. New large graphs with given degree and diameter. In Y. Alavi and A. Schwenk, editors, *Graph Theory, Combinatorics and Algorithms*, volume 1, pages 221–233, 1995.

[85] F.R.K. Chung. Diameters of communication networks. In *AMS Proceedings of Symposia in Applied Mathematics*, volume 34, pages 1–18, 1986.

[86] N. Biggs. *Algebraic Graph Theory.* Cambridge University Press, 1974.

[87] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[88] E. Bannai and T. Ito. On finite Moore graphs. *Journal of the Faculty of Science, U. Tokyo*, 20:191–208, 1973.

[89] A.J Hoffman and R.R. Singleton. On Moore graphs with diameters 2 and 3. *IBM Journal of Research and Development*, (11):497–504, 1960.

[90] E. Bannai and T. Ito. Regular graphs with excess one. *Discrete Mathematics*, 37:147–158, 1981.

[91] K. Bolding, M. Fulgham, and L. Snyder. The case for chaotic adaptive routing. *IEEE Trans. on Computers*, 46(12):1281–1292, December 1997.

[92] M.D. Grammatikakis, D.F. Hsu, M. Kraetzl, and J.F. Sibeyn. Packet routing in fixed-connection networks: A survey. *IEEE Parallel and Dist. Computing*, 54:77–132, 1998.

[93] M.D Grammatikakis, D.F. Hsu, and M. Kraetzl. A journey into multicomputer routing algorithms. *Parallel Algorithms/Architecture Synthesis, 1995. Proceedings. First Aizu International Symposium on*, pages 19–27, 1995.

[94] S.A. Felperin, L. Gravano, G.D. Pifarré, and J.L.C. Sanz. Routing techniques for massively parallel communication. *Proceedings of the IEEE*, 79(4):488–503, April 1991.

[95] W. Liao and C. King. Valved routing: Efficient flow control for adaptive nonminimal routing in interconnection networks. *IEEE Trans. Computers*, 44(10):1181–1193, October 1995.

[96] J.T. Draper and J. Ghosh. A comprehensive analytical model for wormhole routing in multicomputer systems. *Journal of Parallel and Distributed Computing*, 23:202–214, 1994.

[97] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Comput. Networks*, pages 267–286, March 1979.

[98] I. Schoinas and M.D. Hill. Address translation mechanisms in network interfaces. In *Proceedings of the 1998 Fourth International Symposium on High-Performance Computer Architecture (Cat. No.98TB100224)*, pages 219–230. IEEE Computer Society, IEEE, 1998.

[99] S.S. Mukherjee and M.D. Hill. The impact of data transfer and buffering alternatives on network interface design. In *Proceedings of the 1998 Fourth International Symposium on High-Performance Computer Architecture (Cat. No.98TB100224)*, pages 207–218. IEEE Computer Society, IEEE, 1998.

[100] M.A. Blumrich, C. Dubnicki, E.W. Felten, and K. Li. Protected, user-level DMA for the SHRIMP network interface. In *Proceedings of the Second International Symposium on High-Performance Computer Architecture (Cat. No.96TB100017)*, pages 154–165. IEEE Computer Society, IEEE, February 1996.

[101] R. A.F. Bhoedjang, T. Rühl, and H.E. Bal. User-level network interface protocols. *Computer*, 31(11):53–60, November 1998.

[102] H. Tezuka, F. O'Carroll, A. Hori, and Y. Ishikawa. Pin-down cache: A virtual memory management technique for zero-copy communication. In *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (Cat.No.98TB100227)*, pages 308–314. IEEE Computer Society, IEEE, 1998.

[103] H. Nakajo and Y. Kaneda. High speed serial communication in a future parallel computer architecture. In *Proceedings of the International Workshop on Innovative Architecture (IWIA'97). Innovative Architecture for Future Generation High-Performance Processors and Systems (Cat. No.97TB100229).*, pages 125–132. IEEE Computer Society, IEEE, October 1998.

[104] E.P. Markatos and M. G.H. Katevenis. User-level DMA without operating system kernel modification. In *Proceedings of the Third International Symposium on High-Performance Computer Architecture (Cat. No.97TB100094)*, pages 322–331. IEEE Computer Society, IEEE, 1997.

[105] R.H. Arpaci-Dusseau, A.C. Arpaci-Dusseau, D.E. Culler, J.M Hellerstein, and D.A. Patterson. The architectural costs of streaming I/O: A comparison of workstations, clusters, and SMPs. In *Proceedings of the 1998 Fourth International Symposium on High-Performance Computer Architecture (Cat. No.98TB100224)*, pages 90–101. IEEE Computer Society, IEEE, 1998.

[106] G. Shah, A. Singla, and U. Ramachandran. The quest for a zero overhead shared memory parallel machine. In *Proceedings of the 1995 International*

Conference on Parallel Processing, volume I, pages 194–201. IEEE Computer Society, CRC Press, 1995.

[107] T. Kudoh, J. Yamamoto, F. Sudoh, H. Amano, Y. Ishikawa, and M. Sato. Memory based light weight communication architecture for local area distributed computing. In *Proceedings of the International Workshop on Innovative Architecture (IWIA '97). Innovative Architecture for Future Generation High-Performance Processors and Systems (Cat. No.97TB100229)*, pages 133–139. IEEE Computer Society, IEEE, October 1998.

[108] M.A. Blumrich et al. Design choices in the SHRIMP system: An empirical study. In *Proceedings of the 25th Annual International Symposium on Computer Architecture (Cat. No.98CB36235)*, pages 330–341. IEEE Computer Society, IEEE, 1998.

[109] S.S. Mukherjee, B. Falsafi, M.D. Hill, and D.A. Wood. Coherent network interfaces for fine-grain communication. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 247–258, May 1996.

[110] Bruce M. Maggs. *Probability and Algorithms*, chapter 11. National Research Council, 1992.

[111] S.E. Fahlman. The hashnet interconnection scheme. Technical report, Department of Computing Science, Carnegie-Mellon University, Pittsburgh PA, June 1980.

[112] B. Bollobas. The diameter of random graphs. *Transactions of the American Mathematical Society*, pages 41–52, 1981.

[113] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 171–180, 2000.

[114] A. Réka, H. Jeong, and A.L. Barabási. Diameter of the world wide web. *Nature*, (401):130–131, 1999.

[115] B. Bollobás. The diameter of random graphs. *IEEE Transactions of Information Theory*, 36(2):285–288, 1990.

[116] M.E.J. Newman, C. Moore, and D.J. Watts. Mean-field solution of the small-world network model. *Physical Review Letters*, 84:3201–3204, 2000.

[117] Christian F. Moukarzel. Spreading and shortest paths in systems with sparse long-range connections. *Physical Review E*, 60(6):6263–6266, 1999.

[118] Duncan J. Watts. *Small Worlds.* Princeton University Press, 1999.

[119] M.Barthélémy and L.A.N. Amaral. Small-world networks: Evidence for a crossover picture. *Physical Review Letters*, 82(15):3180–3183, April 1999.

[120] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[121] S.A. Pandit and R.E. Amritkar. Characterization and control of small-world networks. *Physical Review E*, 60(2), August 1999.

[122] J.J. Collins and C.C. Chow. It's a small world. *Nature*, 393:409–410, June 1998.

[123] H. Herzel. How to quantify 'small-world networks'? *Fractals*, 6(4):301–303, 1998.

[124] Jon Kleinberg. The small-world phenomenon: and algorithmic perspective. *Cornell Computer Science Technical Report 99-1776*, 1999. http://www.cs.cornell.edu/home/kleinber/swn.ps.

[125] Cristopher Moore and M.E.J. Newman. Exact solution of site and bond percolation on small-world networks. *Physical Review E*, 62:7059–7064, 2000.

[126] M.E.J. Newman, I.Jensen, and R.M. Ziff. Percolation and epidemics in a two-dimensional small world. *Physical Review E*, 65, 2002.

[127] S.N. Dorogovtsev and J.F.F Mendes. Exactly solvable small-world network. *Europhysics Letters*, 50(1):1–7, 2000.

[128] R.V. Kulkarni, E. Almaas, and D.Stroud. Exact results and scaling properties of small-world networks. *Physical Review E*, 61(4):4268–4271, 2000.

[129] T.F. Chan and Y. Saad. Multigrid algorithms on the hypercube multiprocessor. *IEEE Trans. Comput.*, 35:969–977, 1986.

[130] M.T. Jones and P.E. Plassmann. *Graph Theory and Sparse Matrix Computation*, chapter The efficient parallel iterative solution of large sparse linear systems, pages 229–245. Springer-Verlag, 1993.

[131] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Shculten. Namd2: Greater scalability for parallel mollecular dynamics. *J. Comp. Phys.*, 151:283–312, 1999.

[132] K. Melhorn, S. Näher, M. Seel, and C. Uhrig. *The LEDA User Manual.* Max-Planck-Institut für Informatik, Algorithmic Solutions Software GmbH, Saarbrücken, Germany, 4.0 edition.

[133] W.J.Dally and C.L.Seitz. Deadlock-free message routing in multiprocessors interconnection networks. *IEEE Trans. on Computers*, C-36(5):547–553, May 1987.

[134] J. Duato. A new theory of deadlock-free adaptive routing in wormhole routing networks. *IEEE Trans. Parallel and Distributed Systems*, 4(12):1320–1331, 1993.

[135] W.J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. Parallel and Distributed Systems*, 4(4):466–475, Apr 1993.

[136] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[137] Leonard Kleinrock. *Queueing Systems*, volume I Theory. John Wiley and Sons, 1975.

[138] Leonard Kleinrock. *Queueing Systems*, volume II. John Wiley and Sons, 1975.

[139] F. Comellas and M. Sampels. Deterministic small-world communication networks. *Information Processing Letters*, 76:83–90, 2000.

[140] F. Comellas and M. Sampels. Deterministic small-world networks. *Physica A*, 309:231–235, 2002.

[141] *Algorithmic Coding Theory*, chapter 7. McGraw-Hill, 1968.

[142] S.W. Golomb. *Shift Register Sequences.* Aegean Park Press, 1982.

[143] J. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. on Information Theory*, 15:122–127, 1969.

[144] E. Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Trans. on Information Theory*, 22:732–736, 1976.

[145] E. Groth. Generation of binary sequences with controllable complexity. *IEEE Trans. on Information Theory*, 17:288–296, 1971.

[146] S.Y. Kim and K.Y. Chwa. Optimal embedding of multiple graphs into a hypermesh. In *Proceedings International Conference on Parallel and Distributed Computing*, pages 436–443, 1997.

[147] S. Abraham and K. Padmanabhan. Performance of multicomputer networks under pin-out constraints. *Journal of Parallel and Distributed Systems*, pages 237–248, August 1992.

[148] M. Ould-Khaoua, L. M. Mackenzie, R.J. Sutherland, and R. Sotudeh. Constraint-based evaluation of hypergraph and graph networks. *Simulation Practice and Theory*, 4:119–140, 1996.

[149] M. Ould-Khaoua, L. M. Mackenzie, and R. Sotudeh. Comparative evaluation of hypermesh and multi-stage interconnection networks. *The Computer Journal*, 39(3):232–240, 1996.

[150] J. Beeley. Design and construction of a hypermesh interconnection parallel computer - 1st year PhD report. University of Glasgow, Advanced Computer Systems Group, Internal Report, August 1998.

[151] V.K. Decyk, D.E. Dauger, and P.R. Kokelaar. Appleseed: A parallel macintosh cluster for numerically intensive computing. See http://exodus.physics.ucla.edu/appleseed/appleseed.html.

[152] M.D. Schroeder. A state-of-the-art distributed system: Computing with BOB. In S. Mullender, editor, *Distributed Systems*, pages 1–16. ACM Press Books, 1998.

[153] H. Kopetz and V. Ochsenreiter. Clock synchronization in distributed realtime systems. *IEEE Transactions on Computers*, pages 933–940, 1987.

[154] M. Haines, D. Cronk, and P. Mehrotra. On the design of Chant: A talking threads package. In *Proceedings of supercomputing 94*. IEEE, November 1994.

[155] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine*. The MIT Press, 1994.

[156] ISO. ISO open systems interconnection basic reference model. *SIGCOMM*, 11(2):15–65, 1981.

[157] Apple Computer Inc. TECHNOTE: Understanding PCI bus performance. See: http://www.apple.com, October 1995.

[158] F. Rodríguez. DCSH report and proposals. University of Glasgow, Advanced Computer Systems Group, Internal Report, January 1999.

[159] Shubhendu S. Mukherjee and Mark. D. Hill. The impact of data transfer and buffering alternatives on network interface design. *HPCA*, 1998.

[160] Shubhendu S. Mukherjee, Babak Falsafi, Mark. D. Hill, and David A. Wood. Coherent network interfaces for fine-grain communication. In *proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 247–258, 1996.