PERFORMANCE EVALUATION OF NETWORK-ON-CHIP ARCHITECTURES

JIE CHEN

PERFORMANCE EVALUATION OF NETWORK-ON-CHIP ARCHITECTURES

by

©JIE CHEN, B.ENG

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

Memorial University of Newfoundland

May 2012

St. John's Newfoundland

Abstract

With the development of integrated circuit technology, System-on-Chip (SoC), which is composed of heterogeneous cores on a single chip, has entered the billion-transistor era. As the microprocessor industry moves from single-core to multi-core, and eventually to many-core architectures, providing tens to hundreds of similar cores on a single multiprocessor chip will be necessary. Efficient communication among different processors becomes critical. Therefore, a high-performance, flexible, scalable, and design-friendly interconnection architecture is highly desired for modern SoC and microprocessor designs.

How to provide efficient communication within a SoC architecture poses a challenge to both academia and industry. Before the advent of Network-on-Chip (NoC), interconnection architectures were usually based on dedicated wires or shared buses. However, they cannot be easily scaled up to meet the ever-increasing demand from the on-chip systems. NoC has been proposed as a highly structured and scalable solution to address the communication problems in on-chip systems. NoC has several advantages over dedicated wiring and buses, e.g., high bandwidth, low latency, low power consumption, and scalability. For NoCs, messages are transported back and forth via the interconnection networks. Thus, the interconnections among multiple cores on a chip have a significant impact on communication efficiency and the performance of a chip design in terms of end-to-end delay, throughput, and packets loss ratio. There-

fore, it is worthwhile studying the different characteristics of different interconnection network topologies. Another vital factor which can affect network performance is the particular communications requirements of the applications. Without targeting any specific applications, spatial and temporal distributions are explored to study the performance of various interconnection network architectures. It is clearly reflected through our study that networks of different architectures can perform differently under various traffic conditions. In this thesis, the most popular topologies and some recent topologies are reviewed and compared, and Three target architectures are chosen: torus (a representative topology of recent topologies), Metacube (a representative topology of recent topologies) and hypercube (a representative topology of popular topologies with relatively high cost). Their performance under different traffic models is studied. Three temporal distributions including Poisson, MMPP and Pareto, and three spatial distributions including bit-complement, random uniform and hot spot, are discussed in this thesis. Based on the simulation results, strengths and limitations of the torus, the Metacube and the hypercube are summarized.

Acknowledgements

I would like to take this time to thank my supervisors: Dr.Paul Gillard and Dr.Cheng Li. During my most difficult time, they offered me selfless help. I am impressed by their broad and profound knowledge. They provided me invaluable insights during my research. I feel deeply that I am so blessed to have them as my supervisors. They are not only my supervisors, but also my life mentors. What I have learnt from them is not limited to academic knowledge, but also professional spirit, research attitude and so much more.

I would like to thank all my colleagues in the CERL lab. They are a big family to me, sharing and supporting all the time. I also gained a lot academically through discussions with my colleagues, triggering me to think from a different perspective. I will always cherish my friendship with them. I was glad to have them around throughout my study at Memorial University.

I can never express my gratitude enough to my dearest parents. Without their continuous support and encouragement, all of this would not have happened. Their unconditional love makes me strong. I devote this thesis to them. I love you.

Table of Contents

List of Tables

List of Figures

List of Acronyms

SoC	System-on-Chip
NoC	Network-on-chip
IC	Integrated Circuit
IP	Intellectual property
MC	Metacube
DSP	Digital System Processor
CPU	Central Processing Unit
GPU	Graphics Processing Unit
UART	Universal Asynchronous Receiver/Transmitter
DDR RAM	Double Data Rate Read Only Memory
QoS	Quality of Service
GALS	Global Asynchronous Local Synchronous
FFT	Fast Fourier Transform
RH	Reduced-hypercube
MSB	Most Significant Bit
LSB	Least Significant Bit
HCN	Hierarchical cubic network
CCC	Cube Connect Circles
MMPP	Markov Modulated Poisson Process

BDP	Bounded Pareto distribution
TCP	Transmission Control Protcol
UDP	User Datagram Protocol
FIFO	First-in-First-out
FTP	File Transfer Protocol
CBR	Constant Bit Rate
CI	Confidence Interval
cdf	Cumulative Distribution Function
pdf	Probability Density Function
HC	Hop Count
LRC	long-range correlation
df	Degree of freedom
OTcl	Tcl script language with Objective-oriented extensions
VoIP	Voice over IP
IDC	Index of dispersion for counts
PMR	peak-to-mean ratio
CV	Coefficient of variation

Chapter 1

Introduction

1.1 Background of Network-on-Chip (NoC)

Moore's law [1] states that the number of transistors on a chip doubles about every two years. In other words, the number of processing elements which can be placed on a single chip doubles about every two years. Transistor size shrinks over time, as shown in Figure 1.1. At the present time, the transistor size is approximately as small as 32 nm [2]. With the development of integrated circuit (IC) technology, System-on-Chip (SoC), composed of heterogeneous cores on a single chip, has entered the billion-transistor era. The most distinguishing feature of SoC over traditional IC is the high communication complexity. As the microprocessor industry moves from single-core to multi-core and eventually to many-core architectures, it is likely that a chip will contain tens to hundreds of identical cores as parallel on-chip processors, and efficient communications is critical for such architectures. Both SoC and the microprocessor market call for a high-performance, flexible, scalable, and design-friendly interconnection network architectures[3]. How to provide efficient communication poses a challenge to researchers from academia and industry.

Before the advent of network-on-chip (NoC), interconnection architectures were usually based on dedicated wires or shared buses. Dedicated wires provide point-to-point connection between pairs of nodes, which is effective for small systems of a few cores. But as the number of cores increases, the number of wires using in the point-to-point connections grows quadratically. Hence, the architecture is not scalable. Compared to dedicated wires, a shared bus, which is a set of wires shared by multiple cores, is more scalable and reusable. However, due to the inherent disadvantage of buses, only one communication is allowed at a time, while the rest of cores wait their turn. The disadvantages of shared bus architectures include long delay, high energy consumption, increasing complexity in decoding/arbitration, low bandwidth [4][5]. It would be extremely inefficient if hundreds of nodes are connected via shared buses. Thus, the use of shared buses is limited to the connection of only a few dozens of intellectual property (IP) cores. To deal with the problems in shared buses, a hierarchical architecture, which segments the bus into shorter buses, has been introduced. Hierarchical bus architectures may relax some of the constraints faced by dedicated wires and shared buses, because different buses may account for different bandwidth needs, protocols and also improve communication parallelism. Nonetheless, scalability remains as a problem for hierarchical bus architectures. In order to meet the communication requirements, accelerate time-to-market and cut down the communication energy consumption of large scale SoCs, there is a great push to find new design alternatives to the conventional point-to-point and bus based architectures.

NoC has been proposed as a highly structured and scalable solution to address the communication problems in SoC. On-chip interconnection networks have several advantages over dedicated wiring and buses, e.g., high bandwidth, low latency, low

power consumption and scalability. NoC architectures can guarantee communication pipelining with a pre-specified clock rate regardless of the network size, which is infeasible for bus-based architectures. For SoCs, cores can be Digital System Processors (DSPs), embedded memory blocks, Central Processing Units (CPUs), video processors, etc. Figure 1.2 shows a typical Nvidia Tegra 600-series SoC [6], which consists of 14 components, including CPU, Graphics Processing Unit (GPU), image processor, video processor, Universal Asynchronous Receiver/Transmitter (UART), Double Data Rate Read Only Memory (DDR RAM). A single chip can be partitioned into functional tiles and the interconnection network. A partition of a single chip is shown in Figure 1.3 [7]. Figure 1.4 provides a detailed pictorial view of a grid network-on-chip architecture [8]. Since its inception, NoC has drawn great attention from researchers all over the world. In order to fully explore the benefits of NoC, numerous challenges and open problems remain to be addressed. Open problems can be classified into four main categories: (1) application modeling and optimization; (2) NoC communication architecture analysis and optimization; (3) NoC Communication Architecture Evaluation, and (4) NoC Design Validation and Synthesis. The main problems under each category are listed below.

(1) **Application modeling and optimization**

–Traffic modeling and benchmarking

–Application mapping

–Application scheduling

(2) **NoC communication architecture analysis and optimization**

–Routing scheme

–Switching technique

–QoS and congestion control

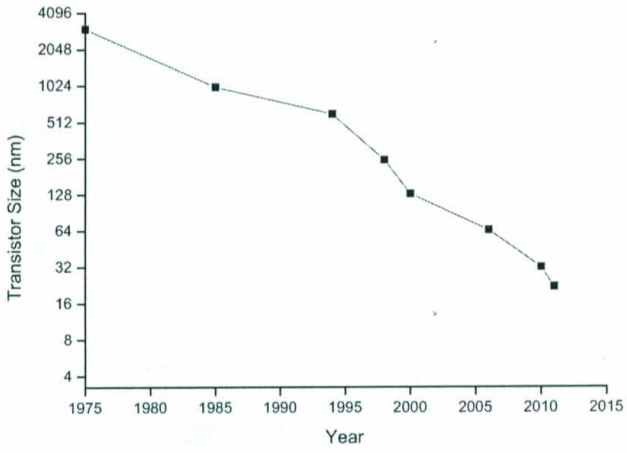

Figure 1.1: Transistor Size vs Time

–Power and thermal management

–Reliability and fault tolerance

(3) **NoC Communication Architecture Evaluation**

–Topology design

–Router design

–Network channel design

–Floor planning and layout design

–Clocking and power distribution

(4) **NoC Design Validation and Synthesis**

–Analysis and simulation

–Prototyping, testing and verification

To better evaluate a network, a thorough analysis of the topology and traffic pattern is required. For selection of a network topology, there are some prior examples; e.g.,

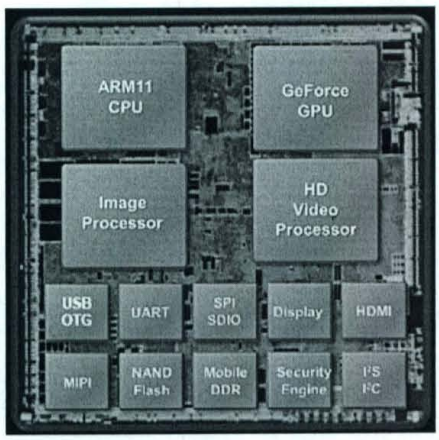

Figure 1.2: Nvidia Tegra 600-series SoCs (www.nvidia.com)

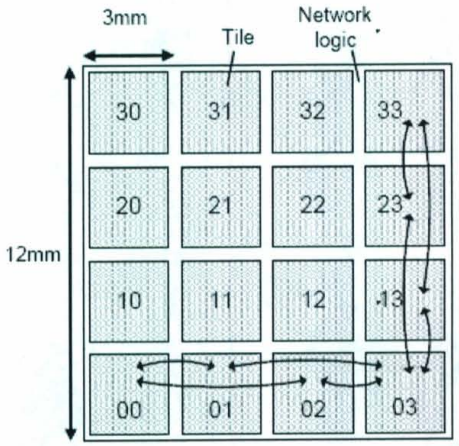

Figure 1.3: Partition of a Die Into Module Tiles and Network Logic

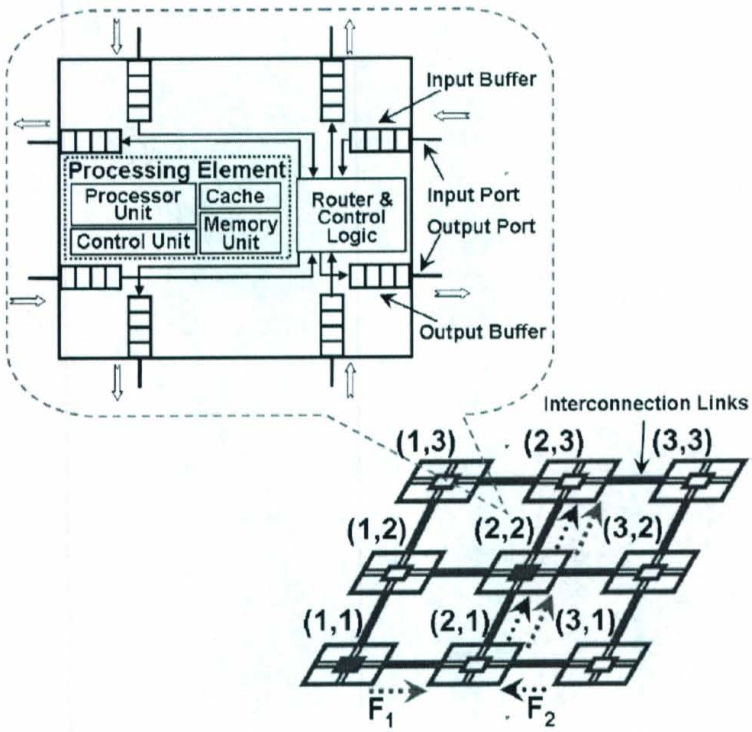

Figure 1.4: On-chip Network Architecture

high-bandwidth mesh networks connecting dozens of components, and ring and star networks for modest bandwidth communication between nearby IP blocks. NoC provides higher bandwidth with moderate area overhead. Compared with point-to-point and bus-based implementations, NoC has smaller energy budgets [9] and better scalability. NoC also enables globally asynchronous, locally synchronous (GALS) design.

1.2 Related Work

As technology scales down to the deep-submicron domain, the potential for SOC to have more types of errors is increasing. For example, the high transistor density makes cross-talk noise and high field effects harder to control. To cope with this, the authors in [10] applied stochastic communication to the routing scheme, to provide reliable communication. One node's confidant is defined as its selected neighbor. Initially, the sender passes the data randomly to its confidant. In the second round, both the sender and its confidant pass the data to their select confidants. The process continues until the data reaches its destination. The routing algorithm itself is not destination aware. The destination node can get the data from more than one path. Reliability is achieved by adding redundancy. Thus data retransmission is avoided. But the overhead involved with this gossip algorithm is non-negligible. Even for delivery of a single packet, many nodes assist in broadcasting the data but do not necessarily need the data in themselves. The network can handle the traffic if there are a few nodes sending data. Otherwise, if many nodes are sending data at the same time, the limited bandwidth will be easily used up by redundant transmission. It would be too pricey to implement this gossip algorithm for NoC. To verify the efficiency of the gossip algorithm, they tested it on a fully connected network of 1000 nodes. First of

all, building a fully connected network of 1000 nodes is itself a challenge. Secondly, for a fully connected network, broadcast is easy to implement, because every node is only one hop away from the source node. For the gossip algorithm, it takes 20 broadcast rounds to reach every node in the network. Later, the authors applied the gossip algorithm to a mesh-based network. Manhattan Distance is the distance between the source and the destination $|x_{src} - x_{dst}| + |y_{src} - y_{dst}|$. Regardless of what routing algorithm is applied, the length of path can not smaller than Manhattan Distance. Hence, the gossip algorithm does not deliver data faster to their destinations than other routing algorithms. As the size of the network increases, the gossip algorithm takes more resources because of the wide spread of data. The target application in their work is an MP3 encoder, which only consists of six function modules. There is no proof showing that gossip algorithm is efficient for larger gird-based networks. It is just impractical to implement in SoCs of very large sizes.

A NoC cliche is proposed in [11]. A grid structure is the most popular topology for NoC implementations. Authors in [12] [13] [14] used an MPEG-2 video decoder as their target application, but the size of networks under consideration is limited to a small number of components in MPEG-2. Therefore, the conclusions made are not guaranteed to still hold for larger network sizes. In [15], the authors applied five well-known load balancing strategies to different topologies, and compared their performance, however, their research is also limited to small size of networks of 4×4 nodes. The authors in [16] compare mesh and torus, however, the size of network limited to 8×8. A model, which captures both temporal and spatial traffic characteristics, was proposed in [17]. This model is described by three statistical parameters, which capture hop count, burstiness, and packet injection distribution, respectively. Injection rate has to be decreased in their study as the network grows. In [18], the

authors calculated the average channel load, compared 2D mesh, 3D mesh and 3D bus architectures. In their study, the average channel load is obtained under uniform traffic, with localized traffic condition considered. The grid structure is very easy to implement and so is its routing algorithm. But due to the lack of fast paths between two remotely located nodes, the network may suffer from long packet latency. In addition, long paths increase the chance of blocking at the intermediate links or nodes. In contrast, fully customized topologies are highly application-oriented. Such architectures can improve the overall system performance at the expense of altering the regularity of the network, which leads to widely varying lengths of wires. Another issue with fully customized topologies is the routing algorithm. The irregularity of the customized network makes it difficult to implement a corresponding routing algorithm and this specific routing algorithm can rarely be reused. To avoid these two extreme situations, the authors in [19] tried to find a sweet point between a regular grid structure and customized topologies by adding long range links to regular grid architecture. Long-range links are inserted to connect remotely located nodes [20]. The communication delay can be greatly reduced between the nodes connected by long-range links. But long-range links are not that necessary for localized traffic, since the chance of node communicates with a distant node is slim. The insertion of long links makes the routing algorithm much more complicated. Long-range links are used under certain conditions, where they can lead to shorter paths without causing deadlocks. To avoid dead-lock, the algorithm has to check the status of the desired range links to decide whether or not to use the long-range links. If it may cause a dead-lock, the long-range link is bypassed and default the XY routing is used. XY routing first route packets along the x dimension, until packets reach the same column as destination, then route packets along the y dimension. In every step, the algorithm follows the same procedure [19]. Long-range links are segmented into regular fixed-

length links connected by repeaters. Repeaters can be modeled as routers with two ports, one input port, and one output port. The number of long-range links is constrained to 1 for any node. Its architecture resembles torus in that way. Its routing algorithm, however, is more complicated than that of torus.

1.3 Motivation

To better understand the network, a thorough understanding of the topologies and the traffic pattern is necessary. On a chip, messages travel back and forth via its interconnections. The topological characteristics of each set of interconnections determine its application. Given a problem and algorithm for its solution, it is important to select a suitable topology. Different topologies will require different algorithms to route data from one node to another. A good routing algorithm on one topology may not be efficient on another topology. The interconnection architecture has a significant impact on the performance of networks in terms of end-to-end delay, throughput, loss rate, etc. Consequently, it is worthwhile to study different topologies. As the size of the network increases, the importance of scalability stands out. The performance of a network depends heavily on the traffic pattern. Traffic patterns have two aspects, their temporal distribution (when to send packets) and their spatial distribution (where packets are sent to), characterizing the temporal and spatial correlations of communications among processing cores. The temporal distribution governs such the frequency of communication and the burstiness of the traffic. The spatial distribution governs the spatial correlation of the communicating pair and determines whether they are strongly coupled or loosely related. Different traffic patterns are applied to the same network to see the impact of traffic patterns on network per-

formance. Here traffic patterns are examined in two ways: first, the same temporal distribution but different spatial distributions; second, the same spatial distribution but different temporal distributions. Each time, a traffic pattern is applied to networks of different architectures and their performances are compared.

1.4 Challenges

Although researchers have done vast amounts of work on traffic modeling, there is no complete traffic model for on-chip networks. On the other hand, the performance of a network relies heavily on the kind of traffic applied to it. Realistic traffic traces are hard to obtain and it is difficult to project what future on-chip network traffic will look like [17]. Instead of implementing specific applications, synthetic traffic patterns are often used as benchmarks. In order to make the generated traffic patterns cover a reasonable range of real traffic patterns, different traffic models are employed to simulate different classes of applications sharing similar properties. The benefit of synthesized traffic is twofold. First, traffic generation is easy to implement regardless of the network size. Second, only focusing on certain types of applications may bias the evaluation of one network against another. In the literature, some researchers adopted real applications, usually multimedia applications, as benchmarks. However, the number of components in these targeted multimedia applications is often small. The size of networks should be close to the number of the components in the target applications. When scalability is of interest, it refers to networks of hundreds of nodes. For now, we do not have the knowledge of how real applications will be in the future. Therefore, evaluation of different architectures is conducted on a general application basis.

To cover a reasonable range of traffic patterns, a few representative traffic models are chosen. A traffic model consists of two aspects: one is its temporal model, the other is its spatial model. Challenges in this thesis include three aspects: architecture exploration, temporal distribution modeling, and spatial distribution modeling. The first challenge is architecture exploration. There are many topologies available now, and potentially more in the future. It is impossible to compare the performance of all of the existing architectures. Hence, some popular and representative topologies are considered as the baseline. Besides, it is desirable to discuss some new topologies as the trend develops. The second challenge is the temporal distribution modeling. The Poisson distribution is simple; however, it cannot characterize the bursty nature of network traffic. How to build a reliable model to simulate on-chip traffic is one of the issues to be addressed in the thesis. The third challenge is the choice of spatial distribution. In this thesis, several representative spatial distributions are investigated.

1.5 Objective and Scope of This Thesis

NoC is emerging as a solution to multi-core communication problems. NoC is such a broad topic that it is impossible to cover all the open issues in this thesis. Among all the problems which need to be addressed, this research is limited to two of them, traffic modeling and architecture exploration. On-chip cores can be homogeneous or heterogeneous. They can have either the same or different sizes. In this thesis, IP modules are assumed to be the same size. Instead of targeting a specific application, the objective of this thesis is to discuss the impact of topologies and traffic patterns on the design of Network-on-Chip systems in general. Different topologies are evaluated using performance metrics. As the size of NoCs keeps growing in the foreseeable

future, hundreds of nodes will be placed on a chip. In this thesis, the network size has been limited to 1024 nodes. This is a practical limit for the number of cores available for the next few years, and should also give some insight into how the different topologies scale under different traffic patterns. Our main interest is to study how performance scales along with the size of networks. By comparing performance of different architectures under different injection rates, each topology under consideration has an applicable scope, when the size of network or average channel load, or both are given. To make a fair comparison, different topologies are evaluated under different traffic models.

1.6 Thesis Organization

In Chapter 1, a brief introduction to Network-on-Chip is provided. Since point-to-point and bus-based architectures cannot satisfy the ever-increasing communication demand, NoC emerges as a solution to on-chip communication problems. In this chapter, we conduct a high-level review and summarize related works on topology exploration, traffic modeling, and their results. Following the motivation of this work, we describe the challenges and the objective and the scope of this work.

Chapter 2 reviews different topologies, including classic topologies such as ring, star, mesh, binary tree, fat tree, butterfly, torus and hypercube, and some recent topologies, mainly mutations of hypercube, including Reduced Hypercube, Cross Hypercube, Dual Cube, and Metacube. Torus is a popular topology with small cost and decent connectivity. Hypercube offers a reasonable compromise between the network cost and performance. Metacube is a mutation of hypercube. Metacube is a powerful

topology to connect huge number of nodes with only a small node degree. Though node degree is a constraining factor for hypercube, it is not a problem for Metacube. Because of their desirable topological properties, torus, hypercube and Metacube are chosen as the targeted topologies.

Chapter 3 first provides an introduction to the study of traffic modeling, followed by an overview of traffic generation. Traffic generation is discussed in two domains: spatial and temporal. Spatial distribution governs the spatial correlations among communicating pairs, i.e., transmitters and receivers. Temporal distribution describes the time correlations of packet generations, i.e., when to send. Three spatial distributions are considered: bit complement, random uniform and hot spot. Three temporal distributions are applied: Poisson, Markov Modulated Poisson Process (MMPP) and Bounded Pareto Distribution (BPD). Poisson is very popular traffic model because of its simplicity. Packet arrivals follow a temporally independent and memory-less process. MMPP is used to model a short-range dependency process where the sending probability of the current interval only depends on the last interval. Different sending probabilities correspond to whether a packet is sent or not during the last interval. For the normal Pareto distribution, it is hard to control its generation range. BPD has more controllable upper bounds than the normal Pareto distribution. Thus, it is used to model self-similar/long-range correlated traffic. Finally, it gives the justification of the chosen traffic patterns.

Chapter 4 explains why NS-2 is chosen as the network simulator, then introduces the network simulator NS2. A brief introduction describing how to model, simulate and analyze in NS-2 is provided. How to visualize a simulation result via the animation panel is also shown. Then a simple example is presented to explain a typical TCL file

and its functional components. Modifications to NS-2 system files are implemented to meet simulation requirements for our research, where new applications (Poisson, MMPP and BPD traffic generation) are added. Three target topologies in NS-2 are constructed for network sizes of 32, 64, 128, 512 and 1024 nodes, where minimal deterministic routing algorithms for target topologies are implemented. Three spatial traffic models are implemented in NS-2, including random uniform, bit complement, and hot-spot.

Chapter 5 analyzes the performance of different topologies in terms of average end-to-end delay, loss rate and average throughput. The simulation results of different setups of the same architecture are compared, and so are the different architectures of the same setups. The impact of topological properties on network performance is investigated from the perspective of node degree, average hop count and so on. The impact of burstiness on network performance is also discussed with respect to the adjust burstiness for MMPP and BPD traffic. How performance scales with network size is one of the main interests of this research, where the simulation covers networks of 32, 64, 128, 512 and 1024 nodes, respectively, are considered.

Chapter 6 summarizes the thesis on architecture modeling, traffic modeling, wire modeling and implementation of routing algorithms. It concludes the simulation results of the torus, the Metacube and the hypercube under different traffic patterns, including temporal and spatial distribution. Suggestions on the topological choices are given based on the analysis and simulation results. Possible future research directions are provided and elaborated.

Chapter 2

NoC Architecture

In static networks, topology determines the arrangement of links and nodes. In the context of NoC, topology determines the connectivity among on-chip nodes. A few characteristics are useful to describe a certain topology: how many direct neighbors a node has; what is the distance a packet can travel in the worst case (for minimal routing); what is the average distance between nodes; how many links are there in the topology; how many links need to be removed to get two approximately equal subnetworks. Some of these characteristics greatly affect network performance. The others are related to implementation cost. Topologies in macro-networks can be easily utilized in NoC. Classic topologies, for example, ring, star and binary tree, have been very well studied and widely applied to real systems. Hypercube and torus are two of the most popular topologies adopted by commercial applications. As well, researchers have continuously endeavored to explore new topologies. Some topologies have been recently developed to improve certain properties of existing topologies.

2.1 Topology Parameters

In general, networks can be classified into two categories, direct networks and indirect networks. A direct network consists of a set of nodes, each one being directly connected to a subset of other nodes via links, such as ring and mesh. Instead of providing direct connections among nodes, an indirect network provides connections through switches. In this thesis, links are considered as bidirectional unless otherwise specified. In a network, the topology specifies the arrangement of nodes and links [21]. It determines the interconnection of nodes and can usually be modeled as a graph. A topology is characterized by several parameters such as node degree, diameter, link complexity and bisection width.

Node degree: The number of links connected to a node. A network is considered to be regular if all nodes have the same degree; otherwise, it is irregular. Node degree describes the I/O complexity of a node. Node degree can be constant or vary with the size of the network. For instance, a ring has a fixed node degree of 2; the node degree of an N-node hypercube is $log_2 N$. Small and fixed node degree is a preferred topological characteristic. A smaller node degree leads to a lower link cost. Fixed node degree reduces the necessity to add new nodes to the existing network. In most cases, there is a constraint on node degree, which is the number of direct neighbors of a node. Limitation of node degree arises from two considerations: one is the hardware limitation, i.e., the number of ports a node can support; the other is the communication protocol limitation, e.g., BlueTooth networks can support a node degree up to eight. Finally, performance considerations, such as the space complexity and network scalability, may limit the number of nodes with which each node may communicate directly.

Diameter: The maximum shortest path between all pairs of nodes. If there is no direct connection between two nodes, the message has to travel through some intermediate nodes which will introduce multiple hop delay. Since the message delay is proportional to the number of hops, the length of the maximum shortest path becomes an important metric. In minimal routing, the diameter determines the worst case distance between nodes. One example is broadcasting. The node farthest from the sender determines the worst case distance for broadcasting. While in non-minimum routing, the length of path can be longer than the diameter, this depends on the status of the network and the specific routing algorithms. Even for non-minimal routing, small diameter can help to provide low and predictable latency, predict routing paths and traffic flow, and make troubleshooting easier.

Average hop count: Since the diameter is the maximum distance, it overestimates the path length in many cases. A better measure is average hop count, which is the average number of hops between source and destination. Its quantity is obtained by averaging the total length of paths over the number of paths. From a performance perspective, shorter average hop count is preferred. Average hop count can be reduced by increasing the number of links.

Link complexity: it is the total number of links in the topology. As the network scales up, the link complexity increases. Adding more links to a network can reduce its diameter and the average hop count, and provide path diversity among nodes. However, links are expensive. Higher link complexity may incur higher hardware complexity and area overhead. Among all topologies, the fully connected network has the highest link complexity.

Bisection width: The number of links needed to be removed to divide the topology into two networks with approximately equal size. A large bisection width is preferable, because it provides more paths between two sub-networks, and thus improves overall performance. Large bisection width provides greater bisection bandwidth. Equation 2.1 gives the calculation of bisection bandwidth. Among the topologies list in Table 2.7, only ring, star and tree topologies have fixed bisection width, regardless of the number of nodes.

$$Bisection\_bandwidth = bisection\_width \times channel\_bandwidth \quad (2.1)$$

2.2 Review of Different Topologies

2.2.1 Quick Review of the Most Popular Topologies

In this section, the most popular topologies will be reviewed, including ring, star, mesh, tree, fat tree, butterfly, and torus, etc. Later their strengths and limitations will be summarized. Note that, among these topologies to be reviewed, only the fat tree and butterfly are indirect networks. The remaining topologies are all direct networks.

In a ring architecture, all nodes are connected in a ring fashion [21], as shown in Figure 2.1. Every node has two neighbors regardless of the size of the ring. Its small degree is preferable, but its diameter increases linearly with the number of nodes. A ring architecture is susceptible to failure. A failure in one link in the connection can disrupt the entire network. Its strengths and limitations are listed in Table 2.1.

In a star architecture of N nodes, $N - 1$ nodes are connected to a center node [21],

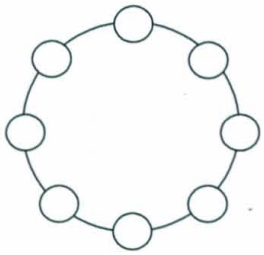

Figure 2.1: Ring

Table 2.1: Ring Architecture

Main Advantage	Main Disadvantage
Interconnection faults are easily located, making troubleshooting easier.	Expansion to the network can cause network disruption.
Ring networks are moderately easy to install.	A single break in the interconnection can disrupt the entire network.

as shown in Figure 2.2. Only the center node has a degree of $N - 1$. Other nodes have degree of 1. The diameter of a star architecture is 2, regardless of its size. A small diameter means a small average hop count, which is a favorable characteristic. The node in the center plays an important role in the network. If the central node is down, the entire network is down as well. Its strengths and limitations are listed in Table 2.2.

Table 2.2: Star Architecture

Main Advantage	Main Disadvantage
Simplicity to operate	Failure of the central node fails the entire network
Each node is isolated free of impact from failed nodes	Central node is the bottleneck

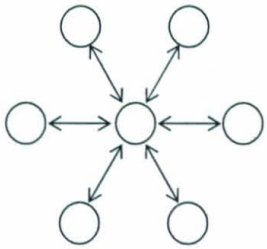

Figure 2.2: Star

In a mesh [21], nodes are connected as a grid, as shown in Figure 2.3. Expansion is easy for meshes. Little effort is needed when adding more nodes to the existing architecture. Nodes have different degrees according to their locations within the mesh. Corner nodes have a degree of 2. Edge nodes have a degree of 3. Inner nodes have a degree of 4. A mesh architecture is more robust with respect to link or node failure compared to star or ring. Its strengths and limitations are listed in Table 2.3.

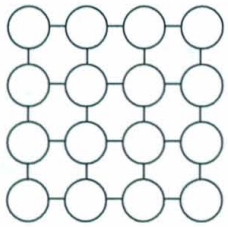

Figure 2.3: Mesh

Table 2.3: Mesh Architecture

Main Advantage	**Main Disadvantage**
Multiple paths between a pair of nodes, tolerant to link failure	Diameter can be very large
Easy to expand	Irregularity, less bandwidth for nodes at corners and edges

In a binary tree [21], the top node is the root and the bottom nodes are the leaves, as shown in Figure 2.4. Every node except the root node has two child nodes which are the nodes that appear immediately beneath the node. A node's parent is the node immediately above it. The root of a tree is the single node that has no parent. Its strength and limitation are listed in Table 2.4.

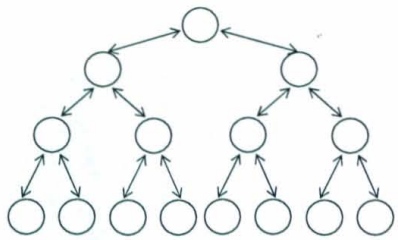

Figure 2.4: Binary Tree

Table 2.4: Tree Architecture

Main Advantage	**Main Disadvantage**
Supported by many network vendors and even hardware vendors	Bottleneck on the root node
All the nodes have access to the larger and their immediate networks, best for branched out networks	When the tree is big, it is difficult to configure and can get complicated after a certain point.

In a fat binary tree [21], only leaves are IPs, as shown in Figure 2.5. Other nodes are switches. When moving towards the root node, there are more links between a parent node and a child node. The number of inter-node links doubles at each level.

In a butterfly architecture [21], as shown in Figure 2.6. Basic butterfly networks have two main disadvantages. First, it lacks of path diversity. There is only one path from

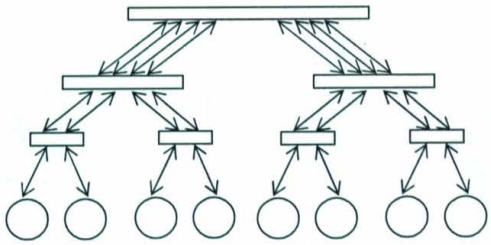

Figure 2.5: Fat Binary Tree

a source node to a destination node. Second, long wires are inevitable. Long wires must transverse half of the diameter of the network.

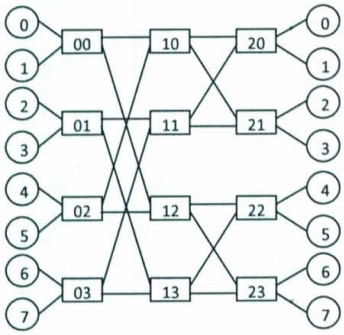

Figure 2.6: Butterfly

A torus architecture [21] is obtained by adding direct connections to two end nodes in the same row or column in a mesh architecture. A 16-node torus is shown in Figure 2.7. Compared with mesh, its diameter is reduced. A regular torus has long wraparound wires. By folding a torus, long wires can be avoided.

A 16-node hypercube architecture is shown in Figure 2.8. In a hypercube, only nodes whose addresses differ by exactly one bit are connected by links [21]. Node degree grows linearly with the size of networks. Long wires are unavoidable when projecting

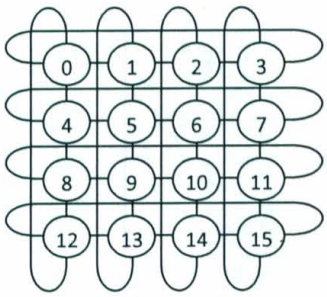

Figure 2.7: Torus

a hypercube to a low dimensional layout.

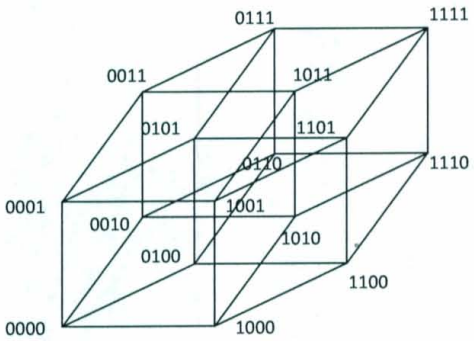

Figure 2.8: Hypercube

From the perspective of performance, low diameter and high bisection width are preferred. Clearly, a fully connected topology satisfies these two requirements. But technically, considering the cost and effort of implementation, low degree, short wires, small area and regular structures are of great interest. The fully-connected architecture fails to scale. There is a necessity to make a trade-off between the two points above. In Tables 2.1-2.2, the pros and cons of ring, mesh, tree and star architectures are listed.

The topological characteristics of each topology determine their application. Given a problem and communication algorithm, it is important to select a suitable topology. A good algorithm for one topology may not be efficient for another topology. A tree architecture can be used efficiently in: (1) select and rotate operation; (2) broadcasting; (3) Census functions; (4) Selection of an arbitrary number from among those offered by a subset of leaves; and (5) Depth computation. The butterfly organization is efficient for Fast Fourier Transform (FFT). The mesh structure is found good for odd-even merge sorting, but not efficient for operations like divide-and-conquer, ascend-descend and parallel merge. But binary trees, meshes of trees, shuffle-exchange, and de Bruijn networks are good for those operations. The strength of the hypercube is its ability to emulate all of the above.

Some topologies can be simulated using other topologies. Any normal algorithm that runs in $T(n)$ time on a butterfly network can be made to run in $O(T(n))$ time on a shuffle-exchange network [22]. Any normal algorithm that runs in $T(n)$ time on a butterfly network can be made to run in $T(n)$ time on a hypercube [22]. Therefore, there is no significant time efficiency difference among these three topologies.

The hypercube is a very popular interconnection network topology due to its small diameter and large bisection bandwidth. The number of nodes of a k-dimensional hypercube is 2^k with node degree k. A hypercube can be treated as an n-dimensional mesh in which each dimension is 2, which is also known as 2-ary n-cube or binary n-cube. In a hypercube, two nodes are connected if and only if their binary representations differ in exactly one bit. This property is crucial for efficient routing and communication. But the size of hypercube is limited by the degree constraints. If the node degree is constricted up to 8, then the size of hypercube is limited to $2^8 = 256$

nodes.

Among all the listed topologies in Table 2.7, mesh and torus are the most popularly used in practice due to their simplicity and regularity. Although there are many topologies, classic or new, commercially only a few have been applied to real network systems. BlueGene/L is a scalable system constructed as 64*32*32 three-dimensional torus, in which the maximum number of computing nodes assigned to a single parallel job is $2^{16} = 65{,}536$ [23].

2.2.2 Some Non-conventional Topologies

Due to the limitations of existing topologies, researchers have been working on exploring the possibility of new topologies for years. In the literature, new topologies can be categorized into two classes, based on the way how they are constructed. One class is hierarchical topologies. The other class is a combination of several existing topologies. Hierarchical topologies have several levels from the highest to the lowest. On different levels, usually different interconnections are utilized. A node address consists of a number of binary strings, each of which represents a level. Nodes within the same level share the same address part representing the corresponding level. The cross product of two graphs can be employed to construct new interconnection networks [24]. Two or more existing topologies are used as the base topologies to construct new topologies.

The hypercube is considered a very useful topology. But the network size is restricted due to its degree limitation. To overcome this major disadvantage, one possible solution is to reduce the node degree. Some variations have been proposed including the folded hypercube [25], the crossed cube [26], the dual cube [27], the reduced-

hypercube (RH)[28], the hierarchical cubic network [29], cube-connected cycles [30] and the metacube [31], all with the goal of minimizing the network diameter or node degree. Table 2.5 gives several mutations of hypercube and their improvements compared to hypercube topology.

Table 2.5: Variations of Hypercube

Topology	Reduce Diameter	Reduce Node Degree
Folded Hypercube	√	
Crossed Hypercube	√	
Dual-cube	√	√
Reduced Hypercube		√
Hierarchical Cubic Network	√	√
Cube-connected Cycles		√
Metacube	√	√

The folded hypercube [25] is a variation of hypercube obtained by connecting each node to the unique node farthest from it. Its diameter is reduced to about half of hypercube at the cost of more links.

The crossed cube [26] is constructed by repositioning some edges in a hypercube to reduce the diameter by about half for the hypercube without increasing the link complexity. The crossed cube can profitably emulate a hypercube. In order to facilitate explanation, the authors defined [26]: two binary strings x and y that are pair-related are denoted as $x \sim y$, if and only if $(x, y) \in \{(00,00),(10,10),(01,11),(11,01)\}$. In a hypercube, the presence of a edge is simply defined: nodes differing in one bit and only bit are connected. To obtain a similar characteristic for edges, a lemma is given [26]:

for all $n>1$,$(u_{n-1}, \cdots, u_0, v_{n-1}, \cdots, v_0)$ is an edge if and only if there exist an l which

satisfies 1) $u_{n-1}, \cdots, u_l = v_{n-1}, \cdots, v_l$,

2) $u_{l-1} \neq v_{l-1}$,

3) $u_{l-2} = v_{l-2}$ if l is even, and

4) for $0 \leq i < \lfloor (l-1)/2 \rfloor$, $u_{2i+1}u_{2i} \sim v_{2i+1}v_{2i}$

The dual cube [27] follows a hierarchical hypercube structure, which has two classes. Each class consists of 2^m clusters, and each cluster has 2^m nodes, where m is the dimension of each cluster. The dual cube is self explanatory. It has two classes, class 0 and class 1. The binary address of each node in an m-dual cube is $1 + 2m$ bit long. The address format of class 0 is as follows : the most significant bit (MSB) is its class ID, the middle m bits represent the cluster ID, and the m least significant bits (LSB) show its node ID. To provide similar properties as the hypercube, the authors revised the address format of class 1 for which its cluster ID and node ID are swapped. There is an edge between two nodes if and only if: (1) their binary addresses differ in only one bit; (2) for class 0, the different bit must be in the m MSB; (3) for class 1, the different bits must be in the middle m bits; (4) their addresses only differ in class ID.

The reduced-hypercube (RH)[28] is obtained by reducing edges from an n-dimensional hypercube to reduce node degree.

The hierarchical cubic network (HCN) [29] : a (n, n) HCN has n clusters and each cluster is an n-cube.

The cube-connected cycles [30] is a virtual node hypercube. Each virtual node in the hypercube is a ring of nodes instead of a single node. Each node has three ports: F,B, and E, which stand for forward, backward, and external. It can emulate a Benes

permutation network. It is remarkably suitable for algorithms such as FFT (Fast Fourier Transform), sorting algorithms, odd-even merge, matrix multiplication, etc.

The Metacube (MC) [31] is a two-level hypercube structure, a symmetric network with a small node degree and a short diameter. MC is a multi-class topology with two parameters $MC(k, m)$, where k is the dimension of the high-level hypercubes (classes) and m is the dimension of the low-level hypercubes (clusters). It is an extended version of the dual cube in terms of structure. A $MC(k, m)$ has 2^k classes; each class consists of $2^{(2^k-1)m}$ clusters; a cluster has 2^m nodes. The total number of nodes is $2^{(2^k m)+k}$. In a node address, the MSB k bits form its class ID. If the other bits are partitioned in a group of m bits, then the $(c+1)th$ m bits from the right represents the node ID, and the remaining $(2^k-1)m$ bits represent its cluster ID. The position of its node ID in a node address varies with its class ID. For the Metacube, there is a link between two nodes if and only if their addresses differ in only one bit in class ID or in node ID. The hypercube can be considered as a special case of MC. When k is 0, MC degrades to a hypercube. Figures 2.9 and 2.10 give the 2D layout of Metacube of 32 nodes and 1024 nodes, respectively.

The Metacube has three advantages: 1) it is symmetric network: the network is the same viewed from every node; 2) it has small node degree : $k+m$; 3) it has small diameter: $2k(m+1)$.

The mathematical definition of Cross-product is

The cross product of two graphs $G = (U, E)$ and $H = (V, F)$ is the graph $G \oplus H$ whose vertex set is $U \times V$, and edge set is defined as follows: Assume $\{u, u'\} \in U$ and $\{v, v'\} \in V$, then $(uv, u'v')$ is an edge in $G \oplus H$ if and only if either $u = u'$ and

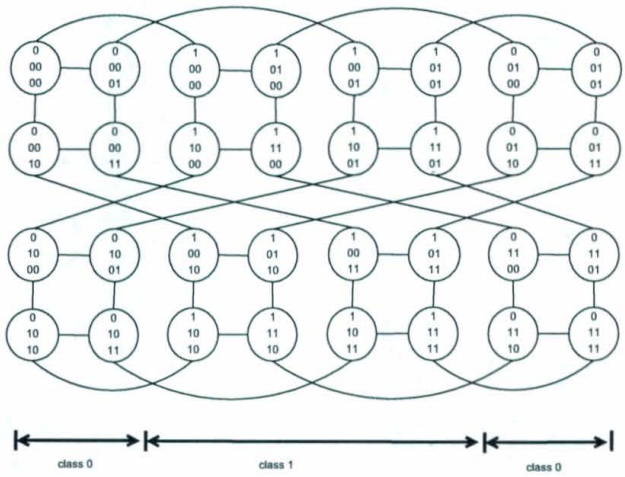

Figure 2.9: 32-node Metacube

Table 2.6: Total number of nodes vs node degree

Links/node	3	4	5	6	7	8
MC(0,m)(Hypercube)	8	16	32	64	128	256
MC(1,m)(Dual-cube)	32	128	512	2048	8192	32,768
MC(2,m)(Quad-cube)	64	1024	16,384	2^{18}	2^{22}	2^{26}
MC(3,m)(Oct-cube)	-	2048	2^{19}	2^{27}	2^{35}	2^{43}
MC(4,m)(Hex-cube)	-	-	2^{20}	2^{36}	2^{52}	2^{68}

$(v, v') \in F$, or $v = v'$ and $(u, u') \in E$. The cross product obeys the commutative law shown in Equation 2.2 and the associative law given in Equation 2.3. Because of these two properties for the cross product, the order to do the cross product does not matter.

$$G \oplus H = H \oplus G \tag{2.2}$$

$$(G \oplus H) \oplus L = G \oplus (H \oplus L) \tag{2.3}$$

For the three base topologies above, there are twelve possible orders in which the cross

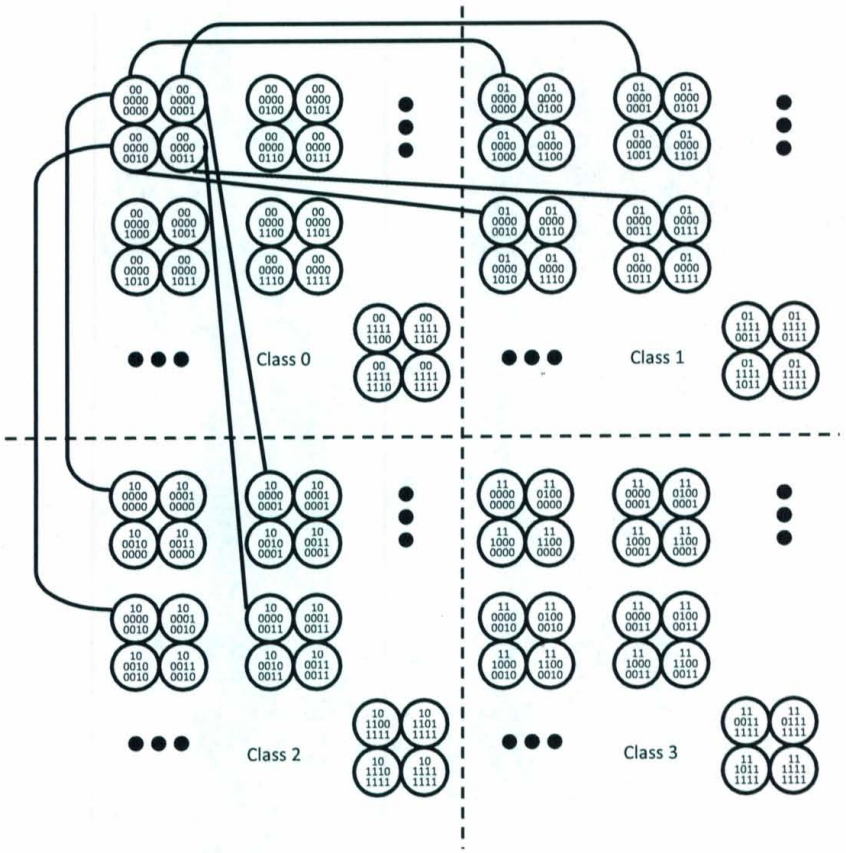

Figure 2.10: 1024-node Metacube

product can be performed. All of them yield the same final graph. To compute the cross product of the n different base topologies, there are $2(n!)$ different combinations. Due to the commutative and associative characteristic of the cross product operator, it is not difficult to prove that once the base topologies (e.g. G and H) are determined, the final cross product graph is determined, regardless of the order in which the cross product is done. This is a very useful property which enables the calculation of the cross product in any order without affecting the final graph. In the cross product graph, there is an edge between two nodes if and only if their representations differ in one symbol and the two different symbols define an edge in the original graphs. The

regularity of the product graph is determined by the regularity of its base topologies. i.e., if the base topologies are all regular, then their cross product is also regular.

The hypercube has a very favorable characteristic, as mentioned earlier, that only nodes whose addresses differ in only one bit are connected by links. Routing in a hypercube is very simple. From node A to node B, compare their addresses, and flip the different bits one by one from the most significant bit to the least significant bit or the other way. If minimal routing is adopted, the number of hops is equal to the number of different bits. Both the dual cube and the Metacube are hierarchical topologies using the n-cube as basic block. To obtain a similar characteristic, the node address is coded differently in the Dual cube and the Metacube. Because the dual cube and the Metacube have smaller node degree than the hypercube, the addresses of the two nodes which are connected must differ in one bit position in a certain portion in the node addresses. The limitation on the locations of the differing bit makes routing not as straightforward as in the hypercube.

2.2.3 Justification of Choice of Topologies

As discussed before, due to the poor scalability, point-to-point and bus-based architectures are not appropriate for large-sized SoCs. A 2D Mesh suits well for VHDL technology because of its simplicity and regularity. Adding new nodes to a mesh-based network needs little effort. Routing in a mesh is also easy to implement. But the diameter and average HC of a mesh grow linearly with its dimension. The torus is another very popular topology. For example, BlueGene/L is constructed as a three-dimensional torus. A torus is obtained by adding wraparound links to a mesh to connect nodes at two ends in a row or a column. A torus is similar to a mesh but

with smaller diameter (approximate half of mesh of the same size) and uniform node degree. Between mesh and torus, torus has similar cost as mesh but better topological properties, hence, is chosen as one of the targeted architectures.

Hypercube is a very popular topology as well. It has higher node degree but smaller diameter. As long as the node degree is allowed in the technology, hypercube can provide small diameter and high throughput. The routing algorithm for a hypercube can take advantage of its uniquely useful property, that is only nodes whose binary addresses differ in one and only one bit are connected. To route in hypercube, we first compare the source and destination addresses, then flip bits differing one by one.

We chose the Metacube in this thesis as the representative architecture to represent recent topologies for the following two considerations: simple routing and scalability. Simple routing algorithms are preferred for NoC implementation. The Metacube has similar routing algorithm to that of hypercube. What is more important is that the Metacube is powerful in establishing a huge network with relatively small cost, i.e., small node degree and small diameter.

2.2.4 Summary

In this chapter, first, some classic topologies were reviewed, e.g., star, ring, binary tree, mesh, and torus. Then, the strength and limitation of the classic topologies are summarized. Ring, mesh and folded torus have uniform length of wires. But for tree, fat tree, hypercube and butterfly, long wires are inevitable. Variations of the hypercube all have long wires, which increase link delay. Network design is application-oriented. It is difficult to conclude that one network design is superior to another design with-

out prior knowledge of the application requirements. Topologies like ring and star can provide modest bandwidth between nearby nodes, but can hardly meet higher bandwidth requirements. The limiting factor for ring topology is its rapidly increasing diameter, which makes it unable to scale. Star topology fails when the center node becomes congested. A high bandwidth mesh network can provide decent performance to dozens of nodes, but similarly, has poor scalability due to its large diameter. Diameter of a network determines the maximum shortest distance the data can travel. In minimum path routing, the diameter defines the longest path. In a contention-free network, this diameter determines the worst case delay. For applications with limited bandwidth requirement, a bus or segmented bus can also be a feasible solution. With increasing wire density, the application range of concentrated lower bandwidth links can be extended. Increased wire density also encourages the exploration of topologies with higher node degree.

In summary, the torus is a representative topology of low cost conventional architectures; the Metacube is a representative topology of recent topologies; and the hypercube is a representative topology of popular topologies with relatively high cost. Torus and hypercube are conventional topologies which have drawn considerable research. They serve as a fair reference to evaluate other recent topologies such as the Metacube. In this paper, the focus is on planar layout. Higher dimensional topologies are projected onto a 2D layout in our studies.

Table 2.7: Characteristics of Some Popular Topologies

Topology	Degree	Diameter	Link Complexity	Bisection	Uniform Length
Ring	2	$N/2$	N	2	Yes
Star	1,$N-1$	2	$N-1$	1	Yes
2D-Mesh	2,3,4	$2(\sqrt{N}-1)$	$2(N-\sqrt{N})$	$\sqrt{N}$	Yes
3D-Mesh	3,4,5,6	$3(\sqrt[3]{N}-1)$	$3(\sqrt[3]{N}-2)^3+15(\sqrt[3]{N}-2)^2+24(\sqrt[3]{N}-2)+12$	$(\sqrt[3]{N})^2$	Yes
Hypercube	$\log_2 N$	$\log_2 N$	$N\log_2 N/2$	$N/2$	No
2-ary tree	1,2,3	$log_2((N+1)/2)$	$(log_2((N+1)/2))^2+log_2((N+1)/2)$	1	No
Fat 2-ary Tree	2^0, 2^1, 2^2,$\cdots$, N	$2\log_2(N)$	$N\log_2 N$	$N/2$	No
Fully Connected	$N-1$	1	$N(N-1)/2$	$N^2/4$	No
Dual-cube	$(log_2N+1)/2$	log_2N+1	$N(log_2N+1)/2$	$N/3$	No
2D-torus	4	$2\lfloor\sqrt{N}/2\rfloor$	$2N$	$2\sqrt{N}$	No
folded 2D-Torus	4	$2\lfloor\sqrt{N}/2\rfloor$	$2N$	$2\sqrt{N}$	Yes
3D-Torus	6	$3\lfloor\sqrt{N}/2\rfloor$	$3N$	$2\sqrt{N}$	No
Folded 3D-Torus	6	$3\lfloor\sqrt{N}/2\rfloor$	$3N$	$2\sqrt{N}$	Yes
Butterfly(k,m)	$2k$	$m+1$	$k^m(m+1)$	k^m	No
Meta-cube(k,m)	$m+k$	$(2^k)(m+1)$	$(m+k)2^{2^k m+k-1}$	$2^{2^k m-1}$	No
CCC	3	$n+\lfloor n/2\rfloor-2$	$3n2^{n-1}$	2^{n-1}	No

Chapter 3

NoC Traffic Generation

Traffic modeling is one of the most active and important topics for NoC. Although extensive work has been done in this area, there is no consensus on what constitutes an appropriate traffic model. Realistic traffic traces are hard to obtain and it is difficult to project what future on-chip network traffic will look like. In the literature, real applications (usually multimedia applications, e.g., MPEG encoder/decoder) are often adopted in the context of NoC, however, they are often too small to apply to large systems. On the other hand, synthesized traffic generated by certain algorithms is more controllable in term of problem size and easy to implement.

3.1 Introduction

Traffic patterns have both a temporal distribution and a spatial distribution. Traffic modeling contains two aspects, temporal distribution modeling and spatial distribution modeling. The temporal distribution determines the time characteristics of the traffic, while the spatial distribution captures the spatial characteristics. The temporal distribution determines when the packets are sent and the spatial distribution

determines the transmitters and the receivers. The two distributions together provide a more comprehensive description of the complete traffic pattern characteristics.

3.2 Spatial Distributions

The spatial distribution describes the spatial correlations of the communicating pairs. Generally speaking, spatial distribution patterns are synthesized from particular applications. In this subsection, some of the most common spatial distributions used to evaluate interconnect networks are reviewed. For a network of N nodes, by convention, an N-bit binary string indexed with the value 0 to $N-1$ is used to represent a node. Bit 0 represents LSB, and bit $N-1$ represents MSB (assuming that the source node address is $\{s_{N-1}, s_{N-2}, s_{N-3}, \cdots, s_0\}$). Different spatial distributions will be represented under the correlation between the destination address and source address.

3.2.1 Uniform Random

In this case, the probability of sending a packet from one node to another node in the network is $1/(N-1)$, which means each source has an equal probability of sending packets to a destination. It is the most popular traffic pattern in network evaluation and a benign traffic pattern because of its uniformly distributed traffic. Random traffic does not favor any topologies. Thus, it provides fair comparison between topologies. However, evaluation with only random traffic does not sufficiently indicate the efficiency of certain topologies or algorithms [21]. Nevertheless, it is often studied and can be used as a benchmark.

3.2.2 Bit Permutation

Bit permutation is a class of permutations, including bit complement, bit reverse, bit rotation, shuffle and transpose. Following bit complement, bit reverse and shuffle permutation are explained.

3.2.2.1 Bit complement

This scenario creates load for source-destination pairs, its destination address is $\{\neg s_{N-1}, \neg s_{N-2}, \neg s_{N-3}, \cdots, \neg s_0\}$. For example, if the source is 0000, then the destination is 1111. Every source node corresponds to one and only one destination node. Unlike other patterns of bit permutation, the source address and the destination address under bit complement traffic cannot be the same. This avoids the effort to prevent one node from sending packets to itself.

3.2.2.2 Bit Reverse

It is quite self-explanatory. The binary representation of the destination address is obtained by reversing the source address, i.e., $\{s_0, s_1, \cdots, s_{N-2}, s_{N-1}\}$. Similar to bit complement, all communicating pairs are determined before simulation. One possible problem with bit reverse traffic pattern is the sender could also be its receiver. For example, if the sender's address is 0000, then the address of the receiver is still 0000 under bit reverse assumption or shuffle permutation. It generally does not make sense to allow to a node to send packets to itself. Thus, to avoid this from happening, the addresses of the communicating pairs should be compared. If and only if the source

address and destination address are different, the communication is valid. Nodes with symmetric binary representations are supposed to pair with themselves if they strictly follow this bit reverse rule. Assuming that there are 2^N nodes in the network and a source node's address is represented as $\{s_{N-1}, s_{N-2}, s_{N-3}, \cdots, s_0\}$. There are $\frac{1}{2^{\lfloor N/2 \rfloor}}$ nodes sending packets to themselves.

3.2.2.3 Shuffle

The source address and destination address satisfy the property that $d_i = s_{(i-1) mod N}$ [21], where d_i and s_i are the destination bit index and the source bit index, respectively. For example, if the address of the source node is $\{s_{N-1}, s_{N-2}, s_{N-3}, \cdots, s_0\}$, then the address of the destination node is $\{s_0, s_{N-1}, s_{N-2}, \cdots, s_1\}$.

3.2.3 Hot Spot

In hot spot distribution, a number of nodes are randomly selected as potential hot spots. At a given time, only one node is selected from these nodes as the hot spot [32]. As the name implies, hot spot nodes send or receive packets with a greater probability than the rest of the nodes. This phenomenon was first observed in shared memory interconnects. When multiple processors continually request accessibility to a shared data structure, the node where this shared data structure is located becomes a hotspot. It is obvious that hot spot nodes are loaded with heavier traffic, thus they will very likely become a bottleneck for the network.

Hotspot nodes are requested more frequently than the ordinary nodes, hence can be overloaded. This spatial traffic model resembles the behavior of multiple requests towards the same destination. This traffic model was proposed by Pfister and Norton

and has been adopted by many researchers [33] :

1. Every node generates data independently of each other.

2. Each generated message has a finite probability α of being directed to the hot spot node and the probability of $(1-\alpha)$ being directed to non-hotspot nodes. α is defined as the hotspot proportion. The hotspot is random selected. At any time, there is only one hotspot in the network.

3. When the generated message is directed to non-hotspot nodes, the probability of being directed to any non-hot-spot node is equal.

3.3 Temporal Traffic

Temporal distribution governs when to send the packets, which can be used to describe the burstiness of the traffic. Three temporal distributions, Poisson, MMPP and Pareto, are discussed in this thesis. To better understand temporal traffic, the first need is to clarify what burstiness is. Intuitively, temporal distributions can be classified into two classes, bursty traffic and non-bursty traffic. Commonly used measurements of burstiness include peak-to-mean ratio, index of dispersion, and coefficient of variation. Peak-to-mean ratio (PMR) is the ratio of the peak rate to the mean rate. Index of dispersion for counts (IDC) is given by the variance of the number of arrivals during the interval of length L divided by the expected value of the number of arrivals. Coefficient of variation (CV) is the ratio of the standard deviation of the interarrival time to the expected number of the interarrival time.

Poisson is the simplest temporal distribution model, having no burstiness and no arrival correlations. Usually, a Poisson assumption is too close to ideal for realistic network exploration. For the MMPP model, by setting different burst rates in the

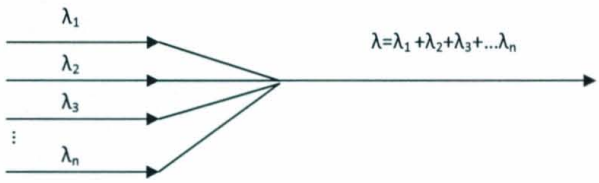

Figure 3.1: Aggregation of Poisson Processes

MMPP model, the burstiness of the generated traffic is changed. Pareto is self-similar traffic, which demonstrates long range correlation among arrivals. Two Pareto distributions will be considered in our work; one is the ON/OFF Pareto distribution, the other is the Bounded Pareto distribution (BPD). Earlier research has shown that compared with the ON/OFF Pareto distribution, the BPD is more practical to generate network traffic for SoC simulations.

3.3.1 Poisson Process

A Poisson process is the easiest one used in queuing theory [34]. It is a memoryless process. The history does not influence the current state; every time slot is independent of any other. The time interval between two adjacent arrivals follows an exponential distribution. The superposition of several Poisson processes is still a Poisson process. The equivalent injection rate seen from the receiving side is equal to the sum of all injection rates, as shown in Figure 3.1. The mean inter-arrival time is given by Equation (3.1), wherein t is the inter-arrival time and λ is the injection rate.

$$E(t) = \frac{1}{\lambda} \tag{3.1}$$

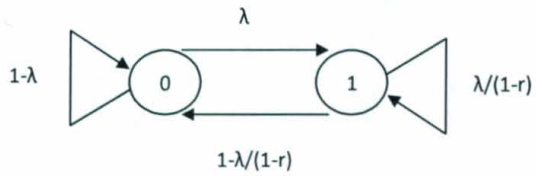

Figure 3.2: Two-state Markov chain

3.3.1.1 Markov Modulated Poisson Process

MMPP is not a normal Poisson process, thus the time between two adjacent arrivals is not exponential. The distribution of time between the $(k-1)-th$ and the $k-th$ arrivals depends on the state at $(k-1)-th$ and $k-th$ arrivals.

For discrete MMPP, there are two sending probabilities. The current sending probability depends on the sending history in the previous time slot only. r is defined as the burst rate and $0 < r < 1$. If a packet is sent during the previous time slot, the sending probability is $\lambda/(1-r)$. If no packet is sent during the previous time slot, the sending probability is λ. This process is described by using a two state Markov chain model, as shown in Figure 3.2. If the source sends a packet during the previous time slot, the current state is state 1. Otherwise, the current state is state 0. Since $\lambda/(1-r)$ is greater than λ, thus state 0 corresponds to the state with low injection rate and state 1 corresponds to the state with high injection state. When the burst rate r approaches 1, the difference between the high injection rate and the low injection rate increases. The larger r is, the longer the sojourn time in state 1 will be. Note that none of the transition rates is greater than 1. MMPP is a short memory process, because only the behavior during the last time slot influences the sending probability in the current time slot. Any behavior before the last time slot does not impact the current behavior. When the burst rate r is 0, the MMPP becomes a Poisson process. Therefore, the

Poisson process is a special case of MMPP. The values labeled in Figure 3.2 denote chain transition probabilities. The corresponding transition matrix is as shown below.

$$A = \begin{bmatrix} 1-\lambda & \lambda \\ 1-\frac{\lambda}{1-r} & \frac{\lambda}{1-r} \end{bmatrix}. \tag{3.2}$$

The probability vector P is given by Equation 3.4. P_0 is the probability of being in state 0; and P_1 is the probability of being in state 1. Notice that A is not invertible, to calculate the probability vector P, Equation 3.3 needs to be used jointly with Equation 3.5. P_0 and P_1 are expressed in Equations 3.6 and 3.7 respectively.

$$P \cdot A = P. \tag{3.3}$$

$$P = \begin{bmatrix} P_0 & P_1 \end{bmatrix}. \tag{3.4}$$

$$P_0 + P_1 = 1. \tag{3.5}$$

$$P_0 = \frac{1-\frac{\lambda}{1-r}}{1-\frac{\lambda}{1-r}+\lambda}. \tag{3.6}$$

$$P_1 = \frac{\lambda}{1-\frac{\lambda}{1-r}+\lambda}. \tag{3.7}$$

Based on the state probabilities P_0 and P_1, the average sending rate is given by

$$\begin{aligned} \lambda_0 &= \lambda P_0 + \frac{\lambda}{1-r} \cdot P_1 \\ &= \frac{\lambda}{1 - \frac{\lambda}{1-r} + \lambda}. \end{aligned} \tag{3.8}$$

To model MMPP, the transition probability of the two-state Markov chain in Figure 3.2 is required. Under different burst rates, the average sending rate should be kept the same. Correspondingly, for a given burst rate and average sending rate, λ is calculated as in

$$\lambda = \frac{1}{\frac{1}{\lambda_0} + \frac{r}{1-r}}. \tag{3.9}$$

In practice, the buffer size is usually fixed. In the following analysis, buffer size is considered as a fixed number. Simulation results under different burst rates are compared. When the burst rate is 0, MMPP becomes a memoryless Poisson process. The purpose of this comparison is to show how the burstiness of the traffic will influence the performance index of the same network. When the burst rate increases, the traffic becomes more uneven. Sometimes it is heavily-loaded, while other times it is lightly-loaded. In reality, if the buffer size is limited, when the traffic is heavy, the packet arrival exceeds the capacity and the excessive packets will be dropped. If more packets are dropped, the loss rate increases. When the loss rate increases, the throughput decreases. When the loss rate is high, the number of packets in the queue decreases, i.e., the average queue length decreases. Since latency is proportional to average queue length, when the average queue length decreases, the latency decreases. When the average length deceases, the average latency decreases. When the traffic is light, the buffer is not fully utilized. Due to the increasing packet loss, the utilization

rate of buffers in bursty traffic is smaller than smooth traffic with the same mean. The MMPP is a short-range-dependent process, which is unlike self-similar processes that manifest long-range correlation (LRC) properties. Applying the MMPP model to an application with self-similar characteristics underestimates the buffer overflow probabilities which may cause significant performance degradation [35].

A long Markov chain is formed as transitions continue. It is useful to know the probability of each state in the Markov chain. To facilitate explanation, $y(x, s_0)$ is the probability of being in the state of (x, s_0) that there are x packets in the buffer and it is at low injection state s_0; $y(x, s_1)$ represents the probability of being in the state of (x, s_0) that there are x packets in the buffer and it is at high injection rate s_1. The previous state of (x, s_0) must be one of (x, s_0), (x, s_1), $(x+1, s_0)$ and $(x+1, s_1)$. The transition probabilities are labeled as shown in Figure 3.3. The probability of $y(x, s_0)$ is given in Equation 3.10. The previous state of $(x, 1)$ must be one of $(x-1, 0)$, $(x-1, 1)$, $(x, 0)$ and $(x, 1)$. The transition probabilities are labeled as shown in Figure 3.4. The probability of $y(x, 1)$ is given in Equation 3.11. But there are four special states: (N, s_0), (N, s_1), $(0, s_0)$, and $(0, s_1)$. These four states correspond to when the buffer is full or empty. Equations 3.12 - 3.16 give the expression for $y(0, s_0)$, $y(0, s_1)$, $y(N, s_0)$ and $y(N, s_1)$, respectively.

$$y(x, s_0) = (1-\lambda)(1-\mu)y(x, s_0) + (1 - \frac{\lambda}{1-r})(1-\mu)y(x, s_1) + \mu(1-\lambda)y(x+1, s_0) + \mu(1 - \frac{\lambda}{1-r})y(x+1, s_1). \tag{3.10}$$

$$y(x, 1) = \lambda(1-\mu)y(x-1, s_0) + \frac{\lambda}{1-r}(1-\mu)y(N, s_1) + \lambda\mu y(x, s_0) + \mu\frac{\lambda}{1-r}y(x, s_1). \tag{3.11}$$

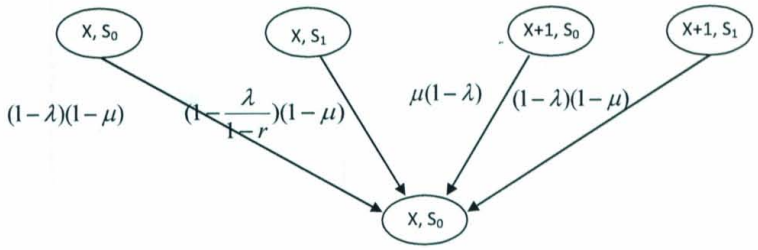

Figure 3.3: Transitions to (x,s_0)

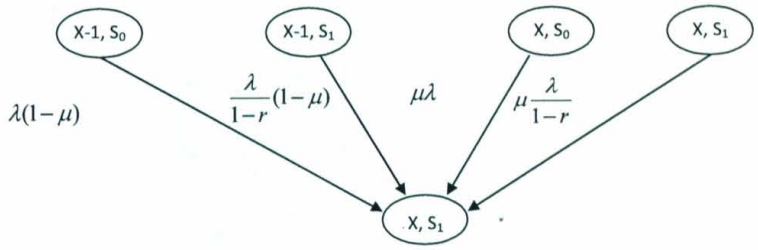

Figure 3.4: Transitions to (x,s_1)

$$y(0,0) = (1-\lambda)y(0,s_0)+(1-\frac{\lambda}{1-r})y(0,s_1)+\mu(1-\lambda)y(x,s_0)+\mu\frac{\lambda}{1-r}y(1,s_1). \quad (3.12)$$

$$y(0,1) = \lambda\mu y(0,s_0) + \mu\frac{\lambda}{1-r}y(0,s_1). \quad (3.13)$$

$$y(N,0) = (1-\lambda)(1-\mu)y(N,s_0) + (1-\frac{\lambda}{1-r})(1-\mu)y(N,s_1). \quad (3.14)$$

$$y(N,0) = (1-\lambda)(1-\mu)y(N,s_0) + (1-\frac{\lambda}{1-r})(1-\mu)y(N,s_1). \quad (3.15)$$

$$y(N,1) = \lambda(1-\mu)y(N-1,s_0)+\frac{\lambda}{1-r}y(N-1,s_1)+\lambda y(N,s_0)+\frac{\lambda}{1-r}y(N,s_1). \quad (3.16)$$

3.3.1.2 On/Off Pareto

Self-similarity is a characteristic that the shape of a feature is independent of its time scale [36]. In this case, the aggregation of streams will not smoothen, but intensify the process. Assuming a fixed-sized packet, the intervals between two arrivals will

vary. During the ON Periods, packets are sent at its peak rate; while during the OFF periods, no packets are sent. Both ON and OFF Periods are obtained from the Pareto distribution.

The major parameters for the Pareto ON/OFF distribution include:

packet_size: the size of the packets (assuming fixed packet size is considered);

burst_time: the average time for the ON period;

idle_time : the average time for the OFF period;

rate: the sending rate during the ON period;

shape: the shape parameter.

The formula used to generate the Pareto distribution is given by Equation 3.17. U is uniformly distributed between (0,1). Its cumulative distribution function (cdf) and probability density function (pdf) are given in Equations 3.18 and 3.19 respectively. x_m is the minimum possible value of x.

$$T = \frac{x_m}{U^{\frac{1}{\alpha}}} \tag{3.17}$$

$$F_X(x) = \begin{cases} 1 - (\frac{x_m}{x})^\alpha & \text{if } x \geq x_m \\ 0 & \text{else} \end{cases} \tag{3.18}$$

$$f_X(x) = \begin{cases} \alpha \frac{{x_m}^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m \\ 0 & \text{else} \end{cases} \tag{3.19}$$

Theoretically, the average sending rate is as in Equation 3.20.

$$\lambda = \frac{rate \times E[ON]}{E[ON] + E[OFF]} \tag{3.20}$$

However, when U is very small, the value of $\frac{x_m}{U^{\frac{1}{\alpha}}}$ is very extremely large. In simulations,

the computer cannot deal with a true Pareto distribution of sufficiently large value, if there is no upper bound for the value generated. In NS-2, there is a traffic generator named POO_Traffic, which generates traffic according to a Pareto ON/OFF distribution. By analyzing the results generated by NS-2, it is observed that there is a great discrepancy between the theoretical mean values and the measured mean. To obtain a desirable load, the scale parameter for the OFF periods needs to be decided. The largest Pareto distribution value is given as

$$x_{max} = \frac{x_m}{(U_{min})^{1/\alpha}}. \tag{3.21}$$

If $rate >> \lambda$, the traffic is considered as bursty traffic, i.e., $\frac{E[ON]}{E[ON]+E[OFF]} << 1$. The shape parameter α indicates the self-similarity and is related to the Hurst parameter H which gives the degree of long-range dependence. The relationship between H and α is $H = (3 - \alpha)/2$. On one hand, the minimum number of packets sent over one ON period cannot be smaller than one. On the other hand, the maximum number of packets the traffic generator can generate should be limited. In other words, there is an upper bound and a lower bound for the Pareto Distribution. This is called the Bounded Pareto Distribution (BPD), which has three parameters, the lower bound L, the upper bound H and the shape parameter α. Its probability density function is given by Equation 3.22. Its cdf is given by Equation 3.23.

$$f_X(x) = \frac{\alpha L^\alpha x^{\alpha-1}}{(1 - \frac{L}{H})^\alpha} \quad (L << x << H). \tag{3.22}$$

$$F_X(x) = \frac{1 - L^\alpha x^{-\alpha}}{(1 - \frac{L}{H})^\alpha} \quad (L << x << H). \tag{3.23}$$

The formula to generate BPD is given by Equation 3.24, where L is the minimum

value, H is the maximum value, U is a uniformly distributed number on (0,1) and α is the shape parameter [37]. With all these parameters, a random number which follows BPD is generated. The mean value of the BPD distribution is given by Equation 3.25. For the Pareto ON/OFF distribution, the length of ON periods and length of the OFF periods are all determined by the Bounded Pareto Distribution. The mean value is a tunable parameter which can be selected based on the application. When the shape parameter increases, the traffic becomes less bursty and the loss rate decreases.

$$(-(\frac{UH^{\alpha} - UL^{\alpha} - H^{\alpha}}{H^{\alpha}L^{\alpha}}))^{-\frac{1}{\alpha}}. \tag{3.24}$$

$$MeanValue = \frac{L^{\alpha}}{1-(\frac{L}{H})^{\alpha}} \cdot \frac{\alpha}{\alpha - 1} \cdot (\frac{1}{L^{\alpha-1}} - \frac{1}{H^{\alpha-1}}). \tag{3.25}$$

3.4 Justification of Choice of Traffic Patterns

Usually, simulated and synthetic traffic are used to evaluate different architectures. Real applications are implemented using NoC and measured using real data clips for the first time in [9]. Real data traces are used when comparisons are done, which gives more convincing results. But real traces are not always the best choice. Sometimes, they are not even feasible. When studying the scalability of a certain architecture, for example, a 1000-node network, it is hard to apply a real traffic to it. In contrast, it is easy to implement a synthetic traffic. Synthetic traffic patterns are abstracted from certain applications and are representative classes of similar kinds. Synthetic traffic is not always inferior to real traces in terms of extensibility. The goal is to compare the performance of the architectures of interest in general. Thus, instead of targeting a certain type of traffic, some typical traffic patterns are used as the benchmark. Applications which perform well on one architecture do not necessarily show similar

performance on another architecture. The performance of a certain architecture depends heavily on the application applied to it.

To make a fair comparison, a set of traffic patterns are covered. Each traffic pattern represent a certain typical class of similar applications. The uniform random is one of the most commonly used traffic. It represents applications which have evenly spread traffic and can be used as an unbiased traffic pattern. The communicating pairs can be close or far from each other with the same probability under random uniform traffic.

As stated earlier, uniform random traffic provides fairness for all topologies. Each source has an equal probability of sending packets to a destination in the network. It provides the maximum randomness. However, sometimes, the destination is determined. To include this scenario in our simulation, permutation traffic is used. Permutation traffic includes a few subsets. There is a common problem for bit rotate, bit shuffle and bit transpose: the sender could also be its receiver. To avoid this from happening, checking the communicating pairs is necessary. This triggers another problem which is that a portion of nodes can no longer be source nor destination if the traffic requirement is strictly followed. The bit complement is the only permutation traffic free of this problem. A receiver and its destination are always different. The bit complement traffic is biased compared with random uniform. Traffic loads on different links can be significantly uneven. Under bit complement traffic, the addresses of the two communicating nodes differ in every bit, which means the source and destination will be either far from or close to each other depending on the arrangement of nodes. Communicating pairs are far from each other for the hypercube and the Metacube, but not necessary for the torus.

Another popular spatial traffic pattern is the hot spot traffic, which represents applications having a small portion of nodes which are more frequently requested than the rest of the nodes in the network. Hot spot traffic imposes intensive load to those hot spot nodes. In the performance study, we will demonstrate how a spot node can become overwhelmed with requests and overloaded.

The Poisson distribution is often assumed due to its simplicity. It can serve as a good example of random memoryless temporal distribution. MMPP represents short-term arrival correlation, an appropriate model for two-state Markov process. Some on-chip applications, for example, multimedia applications, are characterized as self-similar, which manifests long range correlation. To better control the generated numbers, in stead of the normal Pareto, the BPD is used to model self-similar traffic.

The chosen spatial distributions include uniform random, bit complement and hot spot traffic. Each represents a different type of spatial distribution. Study of these three spatial traffic patterns will give a relatively well-rounded comparison of three topologies under consideration. As to the temporal distribution, Poisson, MMPP and BPD are chosen as representatives of traffic with similar characteristics.

3.5 Summary

Traffic modeling is the first step towards a network design. Poisson is the simplest and most popular temporal traffic model. However, it cannot capture the inherently bursty nature of the on-chip network traffic. In contrast, MMPP with adjustable burst rate is applicable to model bursty traffic. A higher burst rate will lead to a burstier

traffic. Poisson can be considered as a special case of MMPP. When the burst rate is 0, MMPP becomes a Poisson process, in which there is no distinction between high injection rate/state or low injection rate/state. Similar to normal Pareto, BPD is a ON/OFF model. During the ON-period, packets are sent continuously; during the OFF-period, no packet is sent. In other words, during the ON-period, the sending probability is 1; during the OFF-period, the sending probability is 0. Both the ON-period and the OFF-period are modeled by BPD. Network performance is greatly influenced by traffic patterns. If the real traffic is self-similar but Poisson is assumed during design, the design made under poisson traffic assumption cannot meet the requirement in real traffic.

Chapter 4

Simulation Implementation

In this chapter, we study the simulation platform (NS-2) for performance analysis of different architectures. First of all, a brief introduction to the simulation tool NS-2 is presented, however, some changes need to be made to NS-2 to meet our simulation requirements. Topology formation is the first step towards simulation. Because NS-2 does not provide topology generation, all connections need to be defined by users. After the topology has been defined, different traffic models are used to generate network traffic. To project multidimensional architectures on to a 2D-grid, wires of different lengths are considered.

4.1 Introduction to Network Simulator- NS-2

4.1.1 Justification

Existing network simulators include OPNET, Orion, NOCSim, MIT's NETSIM, NIST, NS-2, *etc.* To facilitate this study, NS-2 was chosen as the platform for simulation. Because NS-2 is an open source network simulator and is very useful for NoC designers and researchers who are interested in the analysis and evaluation of a large

set of performance metrics, such as delay, throughput and loss rate. Unlike other commercial software, users can modify the core components in NS-2 to meet their specific application needs. It is developed in C++ and OTcl (Tcl script language with objective-oriented extensions), where the main body of the program is written in C++ and OTcl provides the interface between command and configuration. Users write their Tcl file in which the topology, buffer size, packet size, bandwidth, and traffic type are defined. NS-2 generates two trace files as the simulation results. One is named `out.tr`, and the other is named `out.nam`. The `out.tr`file records all the events produced by the simulation. By analyzing the `out.tr` file, throughout, delay, and loss rate performance are obtained. The `out.nam` is file for the network animator `nam` to visually display the simulation results.

4.1.2 Modeling in NS-2

Simulation modeling includes four parts: traffic scenario generation, simulation control, topology and recorder monitor. The basic primitive to create a node is:

```
set ns[new Simulator]
$ns node
```

Every node in the network has an ID, labeled from 0 to $N-1$ (assuming that N is the size of the network). It also has a list of agents attached to it. By default, all nodes are constructed for unicast.

Agent:

TCP is too complex for on-chip communication. For simplicity, UDP is adopted in this thesis.

Traffic:

Poisson, MMPP, and BPD are implemented in NS-2.

Links:

There are unidirectional links and bidirectional links available depending on the requirements. The bandwidth of the link is specified and so is the link delay.

Queue:

A drop tail queue is adopted to simulate First-In-First-Out (FIFO) queues.

Simulation time:

The simulation is terminated by calling the `pro finish {}` function.

4.1.3 The Network Animator(NAM)

`nam` is the Tcl/Tk based animation tool, which visualizes the simulation process. To use `nam`, the first step is to produce a `nam` trace file, which contains topological information. A `nam` trace file is produced during simulation. Once a `nam` trace file is generated, it is ready to be animated by `nam`. The `nam` console window is shown in Figure 4.1. By opening the corresponding `out.nam` file under the File menu, the `nam` trace file will be loaded to `nam`. As shown in Figure 4.2, on the menu bar, there are three menu options, File, Views and Analysis. Under the menu bar, there are five control buttons, rewind, backward play, stop, forward play, and fast forward, shown from left to right. By clicking the play button, the animation is played forward with time increasing. On the right hand side of the control bar, the time label shows the current animation time. The user can change the animation speed by adjusting the time step on the right side of the time label. In the middle of the console window, the main view panel shows the layout of the network and animation. Figure 4.2 is the screen snap of a network of four nodes: node 0, node 1, node 2 and node 3. The solid squares besides the links are packets carried by the links.

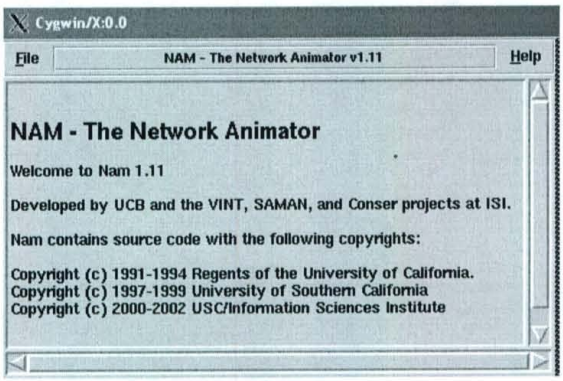

Figure 4.1: NS-2 Animator

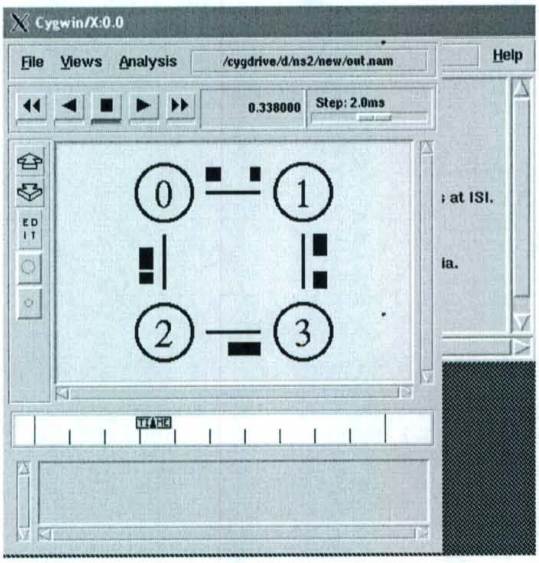

Figure 4.2: NS-2 Simulator View

4.1.4 Trace File Analysis

The `Awk` files are used to analyze `out.tr` to obtain the performance metrics of interest. `out.tr` records all events during the simulation. The format of the `out.tr` file is shown as below.

+	0.0396	1	3	poisson	200	-------	0	0.0	3.0	12	24
−	0.0396	1	3	poisson	200	-------	0	0.0	3.0	12	24
+	0.04	0	1	poisson	200	-------	0	0.0	3.0	19	33
d	0.04	0	1	poisson	200	-------	0	0.0	3.0	19	33
−	0.0404	0	1	poisson	200	-------	0	0.0	3.0	17	31
+	0.0408	3	2	poisson	200	-------	3	3.1	0.1	14	34
−	0.0408	3	2	poisson	200	-------	3	3.1	0.1	14	34
r	0.0412	1	3	poisson	200	-------	0	0.0	3.0	7	13

From left to right, there are twelve columns. The first column is the event: `+`, `-`, `r`, or `d`, where `+` denotes the event that the packet is entering the queue; `-` denotes the event that the packet is departing from the queue; `r` denotes the event that the packet is received; `d` denotes the event that the packet is dropped. The second column shows the time when the event happens. The third column and fourth column together give the location where the event happens, from the node in the third column to the node in the fourth column. The fifth column is the type of the packet, i.e., FPT, CBR, etc. In this example, the packet type is Poisson. The sixth column is the size of the packet in bytes. The seventh column is the `flag`. The eighth column is the flow ID of the packet. The ninth column is the source port ID, in the format of `node.port`. The tenth column is the destination port ID, in the format of `node.port`. The eleventh column is the sequence number of the packet. The twelfth column is the unique ID of the packet.

Trace file `out.tr` gives details about all data transactions, which can be used to con-

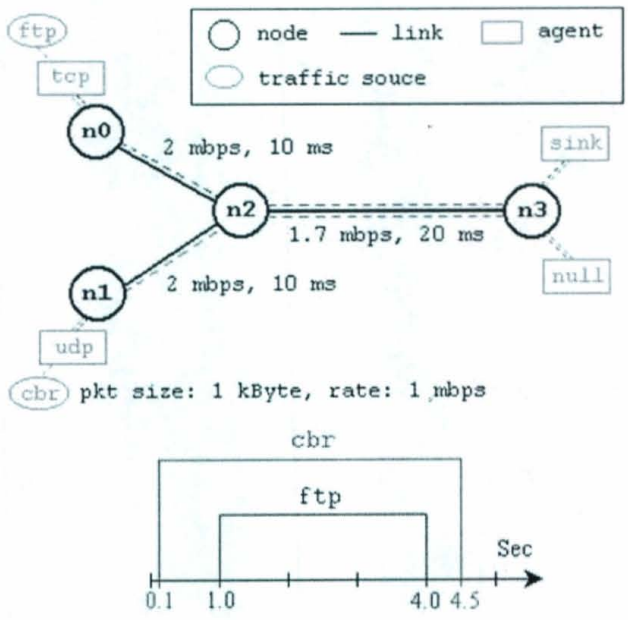

Figure 4.3: A Simple Example

duct the performance analysis. Take the first line of the previous trace file as an example, where packet type is poisson. It means that a packet enters node 3's queue from node 1 at time 0.0396s. Poisson is its packet type with a size of 200 bytes. It is from flow 0. Its source port is port 0 of node 0 and its destination port is port 0 of node 3, its sequence number is 12 and the packet ID is 24.

4.1.5 A Simple Simulation Example

To better understanding the simulation, a simple example of a general network is shown in Figure 4.3. There are four nodes in the network, n_0, n_1, n_2, n_3. Each node has a DropTail queue. The queue on the link between n_2 and n_3 is of size for 10 packets. n_0 and n_1 are sources. n_3 is the sink for both n_0 and n_1. n_2 serves as the

relay node, helping with the transmission. An FTP traffic generator is attached to a TCP agent which is attached to node n_0. Note that, TCP is too complex for on-chip communication. Here this example is a generic network, which only uses TCP to show that different agents can be used in NS-2. A CBR traffic generator is attached to a UDP agent, which is attached to n_1. Three bidirectional links are used to connect the four nodes together. The link between n_0 to n_2 and the link between n_1 and n_2 have the same bandwidth of 2 Mb/s and the same link delay of 10 ms. The link between n_2 and n_3 has a bandwidth of 1.7 Mb/s with a delay of 20 ms. Two traffic flows are shown as well. CBR traffic starts at 0.1 s, and ends at 4.5 s. FTP traffic starts at 1.0 s and stops at 4.0 s.

```
set ns [new Simulator]
#define different colors for data flows (for nam)
$ns color 1 Blue
$ns color 2 Red
#open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
set nd [open out.tr w]
$ns trace-all $nd
#define a finish procedure
proc finish {} {
        global ns nf nd
        $ns flush-trace
        close $nf
        close $nd
```

```
}
        exec nam out.nam &
        exit 0
#create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#set queue size on link between $n_{2}$ and $n_{3}$ to be 10
$ns queue-limit $n2 $n3 10
#set node position
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
#monitor the queue on the link between $n_{2}$ and $n_{3}$
$ns duplex-link-op $n2 $n3 queuePos 0.5
#setup a tcp agent
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
#setup a TCPsink
```

```
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#setup a FTP application
set ftp [new Application/FTP]
#attach FTP traffic generator to TCP agent
$ftp attach-agent $tcp
$ftp set type_ FTP
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate 1mb
$cbr set random_ false
#schedule event for CBR and FTP
$ns at 0.1 "$cbr_start"
$ns at 1.0 "$ftp_start"
$ns at 4.0 "$ftp_stop"
```

```
$ns at 4.5 "$cbr_stop"
$ns at 4.5 "$ns detach-agent $n0 $tcp ;
$ns detach-agent $n3 $sink"
#call the finish procedure at 5.0s
$ns at 5.0 "finish"
$ns run
```

4.2 Implementation of Architecture

Three architectures will be discussed: torus, Metacube and hypercube. The topology decides the arrangements of nodes and links. In NS-2, a link is set by

$ns duplex-link-op <n1> <n2> <op> <args>

A torus is obtained by adding wraparound links to a mesh. In each row or column, end nodes are connected by wraping around links. Other nodes are connected in a grid manner. Every node in a torus has a node degree of 4 regardless of its size. In a Metacube, there is a link if and only if the addresses of two nodes differ in one bit in class ID or node ID. The number of links attached to a node is decided by the node degree. In a hypercube, there is a link if and only if the addresses of two nodes differ in one bit. For the Metacube and the hypercube, the addresses of two nodes are checked and compared to decide if they are connected.

4.3 Implementation of Traffic Generation

4.3.1 Spatial Distribution

Uniform random and bit complement traffic are applied to test different topologies. Under bit complement traffic, nodes are paired to communicate with each other. Under uniform random traffic, a node can communicate with any other nodes with the same probability. Uniform random traffic provides a fair evaluation of the performance of a network, compared with bit complement traffic, which may favor some nodes but have bias against the other nodes.

4.3.2 Temporal Distribution

In the first setup, traffic burstiness is not taken into account, so a simple Poisson process is considered. To incorporate burstiness with traffic generation, a burst rate is defined. The traffic is modeled by MMPP. Under different burst rates, the mean injection rate is kept constant. Low burst rate means even traffic. High burst rate means spiky traffic. There is no correlation among data flows. Each source sends packets independently with the same probability, which defines the sending rate. All setups are the same except that the burst rate is varied.

4.3.2.1 Injection Rate Calculation for Different Sizes of Networks

The calculated injection rates provide some rough ideas about when the network becomes saturated. As the network scales up, the load grows faster than the network capacity. Thus, the injection rate needs to decrease to maintain stability of the network. The total load over all links is equal to the traffic generated by all the nodes

[18]. Hop count is the number of hops that a message travels in a connected network on the shortest path from the sender to its destination [18].

The average hop count for a $2n \times n$ torus is given by Equation 4.1.

$$HC_{torus_{2n\times n}} = \frac{3}{4}n \cdot \frac{2n^2}{2n^2 - 1} \tag{4.1}$$

The average hop count for an $n \times n$ torus is given by Equation 4.2.

$$HC_{torus_{n\times n}} = \frac{n}{2} \tag{4.2}$$

The average hop count for an N-node hypercube is given by Equation 4.3.

$$HC_{hypercube_N} = \frac{\log_2 N}{2} \cdot \frac{N}{N-1} \tag{4.3}$$

The upper bound of the average hop count for an MC(k,m) is given by Equation 4.4.

$$HC_{MC(k,m)} = 2^k m/2 + 2^k. \tag{4.4}$$

The average hop count for an MC(1,m) is given by Equation 4.5.

$$HC_{MC(1,m)} = \frac{n}{2} + 1 - \frac{1}{2^{(n-1)/2}} \tag{4.5}$$

The average hop count for an MC(2,m) is given by Equation 4.6.

$$HC_{MC(2,m)} = \frac{n}{2} + 3 \tag{4.6}$$

The formula to calculate injection rate is given by Equation 4.7 [18].

$$\lambda = \frac{L_N \times \gamma_{chnl}}{HC \times N}, \tag{4.7}$$

where λ is the injection rate per node;

HC is the average hop in the network;

N is the number of nodes in network;

L_N is the link complexity of the network;

γ_{chnl} is the load per channel.

If constant load per channel is desired, when the size of the network grows, the injection rate must be lowered. HC is a characteristic of the topology itself. The calculation above is made under the uniform traffic assumption. However, in most cases, traffic is not uniform, nor even close. That means if the average traveling distance is greater than the average hop count used in the calculation, the actual average channel load is greater than what is calculated. Further, even if the uniform traffic assumption holds, loads may vary on different links. To the best of our knowledge, no routing algorithm can guarantee spreading traffic evenly over links. Therefore, the injection rate obtained by Equation 4.7 may not be precise and simulation based on it may give poorer performance than it assumes. The simulation results verified this conclusion. The estimate of injection rates should be application-specific. In [18], the average traveling distance in the localized traffic model is smaller than the average hop count of the topology. Thus, the injection rate based on the average hop distance is greater than the calculation result.

When the injection rate is 10%, the average channel load is calculated in each of the topologies under the considerations shown in Table 4.1. The performance of networks can be explained from the perspective of average channel load. The Metacube and

the hypercube have very close average channel load for 512-node networks. Their performance should be similar too.

Table 4.1: Average Channel Load for Injection Rate=0.1

Network Size	Torus	Metacube	Hypercube
32	7.5%	10.8%	6.25%
64	10%	20%	7.5%
128	15%	10.94%	8.75%
512	30%	10.88%	11.25%
1024	40%	20%	12.5%

4.3.2.2 Injection Rates for Bursty Traffic

To model burstiness in traffic generation, a burst rate is used. The traffic is modeled by MMPP. Under different burst rates, the mean injection rate is kept the same. A low burst rate means smooth traffic, whereas high burst rate implies spiky traffic. With this model, there is no correlation among different data flows. Each source sends packets independently with the same probability, which is defined as the sending rate. Assuming that the equivalent mean injection rate are 10%, 20% and 30%, respectively, we list the accordingly injection rates under different burst rates in Table 4.2. The numbers in the first row are mean injection rates: 0.1, 0.2, and 0.3, respectively. The numbers in the first column are burst rates: 0, 0.2, 0.4, and 0.8, respectively.

Table 4.2: Injection Rates under Different Burst Rate

Burst Rate	0.1	0.2	0.3
0	0.1	0.2	0.3
0.2	0.09756	0.19048	0.27907
0.4	0.09375	0.17647	0.25
0.6	0.08696	0.15385	0.20690
0.8	0.07143	0.11111	0.13636

4.4 Routing

Routing involves selecting a path from a source node to a destination node in a particular topology. A few popular routing algorithms include XY, west-first, north-last and negative-first routing algorithms. A routing algorithm can be either deterministic or adaptive. XY routing is a deterministic routing algorithm, in which routing paths are fixed for communicating pairs. West-first, north-last and negative-first routing algorithms are partially adaptive. Due to the limited resource on chip, the cost of implementing fully adaptive routing algorithms seems prohibitively high for NoC. In contrast, a deterministic routing algorithm is simple and needs less resource to implement [5]. Therefore, deterministic routing algorithms are preferred in this context. XY routing is adopted as a representative deterministic routing algorithm, and is simple to implement. XY routing is a minimal routing algorithm, which is guaranteed to be live-lock free. Under XY routing algorithm, data is first routed in the X direction, until reaching the Y coordinate and then in the Y direction. XY routing only allows four turns which are west-south, west-south, east-north and east-south.

Figure 4.4 shows an example of routing paths under XY routing algorithm. If node(0,1) sends a packet to node (2,2), it takes the path (0,1), (0,2), (1,2), (2,2). If node (3,3) sends a packet to node (0,0), it takes the path (3,3), (3,2), (3,1), (3,0), (2,0), (1,0), (0,0).

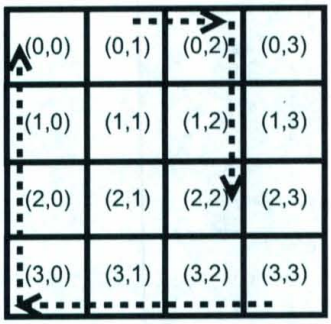

Figure 4.4: XY Routing

XY routing is popular due to the above preferred properties. The XY routing outperforms odd-even and DyAD-OE routing [38] under uniform traffic load because the XY algorithm always maintains evenness [5]. Adaptive routing algorithms make decisions based on short-term information which means the overall balance may not be maintained in the long run. However, for non-uniform traffic, adaptive routing may perform better than XY routing.

4.5 Parameter Setup

4.5.1 Random Seed Generation

Required by statistical analysis, the simulation must be repeated for a large number of times, without changing the setup of parameters. If each run yields the same results because of the inappropriate choice of the parameters such as the random seed, it would be meaningless for statistical analysis. In NS-2, the random number generator basically generates pseudo random numbers. The key to improve randomness in a simulation is to use a different seed for each simulation trial. To get a different seed, the default RNG is set to 0, so that the seed is based on the current time of day

and a counter. It is extremely unlikely that two random seed will overlap by doing this.

4.5.2 Sample Phase

Simulations starting with empty buffers could lead to a more favorable result. To avoid this, we want that the simulation starts with the condition that the length of queues stabilized at their mean lengths, hence, a warm-up period that lasts sufficiently long is considered. By observation, after a certain number of time slots, queues begin to converge to their mean length. This period is called the warm-up stage. During the warm-up stage, data is not collected for analysis. At the end of simulation, source nodes no longer generate packets, but there are still packets transmitting in the network heading to their destinations. During this period of time, the queue length is shorter than normal, and throughput and loss rate are both lower than normal. To eliminate its impact, data is not collected during this period either.

4.6 Modifications to NS-2

Modifications were made to NS-2 to facilitate simulation. The changes include adding new traffic types, wire modeling, and routing algorithm implementation.

4.6.1 Adding New Traffic Types

It takes 7 steps to add a new traffic generator to NS-2 as follows:

1. Add c++ file, which describes the behavior of the traffic generator;
2. Add the default values to /tcl/lib/ns-default.tcl;

3. Modify /trace/cmu-trace.cc;

4. Modify /trace/trace.cc;

5. Modify common/ packet.h;

6. Makefile.in;

7. Configure;

Three traffic types were added to NS-2: Poisson, MMPP and BPD. (1) Poisson traffic Packet generation is a stochastic process. In a certain time slot, whether a packet is sent or not is decided by comparing the sending probability determined by the distribution and a randomly generated number on (0, 1). If this random number is smaller than the sending probability, a packet will be sent. Otherwise, no packet will be sent. A traffic generator called poisson is instantiated by

set poisson [new Application/Traffic/Poisson].

(2) MMPP traffic

Instead of using an ON/OFF model, a 2-state Markov chain is used. Each transition takes place with a transition probability determined from the state machine. The Poisson distribution can be treated as a special case of MMPP. A traffic generator called mmpp is instantiated by

set mmpp [new Application/Traffic/MMPP].

(3) Bounded Pareto Distribution

BPD essentially follows an ON/OFF model. During the ON-period, packets are sent continuously; during the OFF-period, no packet is sent. Both ON and OFF periods follow the Bounded Pareto Distribution. A traffic generator called bpd is instantiated by

set bpd [new Application/Traffic/BPD].

4.6.2 Model Wires

For the initial setting in our simulation, wire length was not taken into consideration, i.e., all wires are assumed to be of the same length. Because of its smaller diameter and more connections, there is no surprise that hypercube outperforms the rest of the topologies in every performance metric. Hypercube has the shortest delay, highest throughput, and lowest loss rate. When projecting high dimensional topologies on a 2D plane (such as the case for the Metacube and the hypercube), long wires cannot be avoided. To better model wires, different wire lengths need to be considered. We assume that the network nodes are arranged in grid fashion. Link delay on long wires is considered to be proportional to the wire length. In a torus, long wires can be avoided by folding the architecture at the cost of doubling its wire length [7]. As networks becomes bigger, longer wires are needed. For Metacube and hypercube in a grid embedding, the number of wires of different lengths are given in Tables 4.3 and 4.4, respectively. In Table 4.3, the numbers in the first row represent wire length in unit length. Unit length is the length of the wire connecting two adjacent nodes in a smallest grid. Take a 64-node Metacube network for example, it has 32 wires of length 2 and 64 wires of length 4.

Table 4.3: Metacube Different Wire Lengths

Size	1	2	4	8	16
32	32	0	16	0	0
64	32	0	64	0	0
128	128	64	0	64	0
512	512	512	0	0	256
1024	1024	0	0	0	1024

To incorporate the impact of wire length on network performance, different link delays

Table 4.4: Hypercube Different Wire Lengths

Size	1	2	4	8	16
32	32	32	16	0	0
64	64	64	64	0	0
128	128	128	128	64	0
512	512	512	512	512	256
1024	1024	1024	1024	1024	512

are assigned according to their link length. Assume the length of the link connecting node n(1) and node n(2) is twice of the length of the link connecting node n(0) and node n(1). If the link connecting n(0) and node n(1) has a link delay of 0.2 ms, then the link delay of the link connecting node n(1) and node n(2) will be 0.4 ms, as shown below.

/tt $ns duplex-link $n(0)) $n(1) $bandwidth 0.2ms DropTail

/tt $ns duplex-link $n(1)) $n(2) $bandwidth 0.4ms DropTail

4.6.3 Routing Algorithms Implementation

Minimal routing algorithms are adopted in this thesis. Minimal routing always chooses the shortest paths for all packets [21]. Minimal routing algorithms are live-lock free and have a better control of communication cost. Furthermore, static routing algorithms are considered in our research that is paths are fixed between the same source and destination. To balance data flows, the routing algorithms developed for torus, Metacube and hypercube adopt dimension order routing that is the packet is routed one dimension at a time [21]. The default static routing in NS-2 uses the same path from node 1 to node 2 as from node 2 to node 1. The problem with this routing scheme is that there is only one fixed path allowed between any pair, which creates

extremely uneven loads among channels. In this thesis, all implemented routing algorithms guarantee a different path from node 1 to node 2 and from node 2 to node 1. A deterministic routing path is determined by explicitly declaring the next-hop node from a source to its destination. For example, `"addExplicitRoute $n(0) $n(2) $n(1)"` determines the next-hop node from node $n(0)$ to $n(2)$ is $n(1)$. By doing this, the next-hop node is updated as a message is transferred towards its destination.

4.6.3.1 Routing in Torus

Routing in a torus is similar to XY routing in a mesh. The difference is whether there exists a shorter path due to the wrap-around links. If routing via wrap-around links does not lead to a shorter path, routing in a torus will be the same as in a mesh: packets are routed along the x dimension first, then along the y dimension. Otherwise, wrap-around links are utilized to create shorter paths, still first route data along the x dimension, then along the y dimension, when the column of the destination is reached. For example, to route from node(0) to node(15), it takes 6 hops in a 4×4 mesh and only 2 hops in a 4×4 torus as shown in Figures 4.5 and 4.6. Assume that C_{src} and C_{dst} are the column ID of the source node and the column ID of the destination node and R_{src} and R_{dst} are the row ID of the source node and the row id of the destination node, respectively. C is the number of columns; R is the number of rows. When implementing the routing algorithm for the torus, whether wrap around links will be used or not is determined by the following conditions:

1) If $\mid C_{src} - C_{dst} \mid \leq C/2$, horizontal wrap around links are not used for routing;
2) If $\mid C_{src} - C_{dst} \mid > C/2$, horizontal wrap around links are used for routing;
3) If $\mid R_{src} - R_{dst} \mid \leq R/2$, vertical wrap around links are not used for routing;

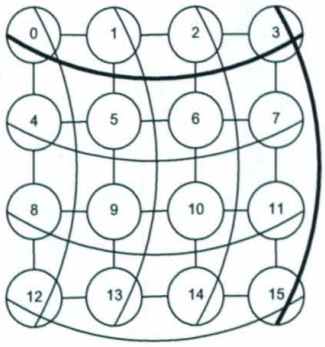

Figure 4.5: Routing in a torus

4) If $\mid R_{src} - R_{dst} \mid > R/2$, vertical wrap around links are used for routing.

For routing in a torus, there are only four possible movements: left, right, up and down. Assume that C_{cur} and R_{cur} represent the column and row ID of the current node, the current node ID is updated with every movement. Movement decision is based on the comparison between current node and destination node in terms of row id and column ID, which is given by

1) If $0 < C_{cur} - C_{dst} \leq C/2$ or $C_{dst} - C_{cur} > C/2$, next movement is left;

2) If $C_{cur} - C_{dst} > C/2$, or $0 < C_{dst} - C_{cur} \leq C/2$ next movement is right;

3) If $0 < R_{cur} - R_{dst} \leq R/2$, or $R_{dst} - R_{cur} > R/2$ next movement is up;

4) If $R_{cur} - R_{dst} > R/2$, or $0 < R_{dst} - R_{cur} \leq R/2$ next movement is down.

4.6.3.2 Routing in Hypercube

To facilitate the explanation, edges connecting different cubes are referred as external edges and edges connecting nodes within the cube are referred as internal edges. From the source node, a packet is first routed to the destination cube via external edges, and then it is routed within the destination cube until they reach the destination node. In the worst case, a packet travels through at most $log_2 N - 3$ external edges

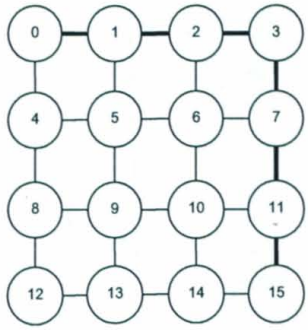

Figure 4.6: Routing in a mesh

and 3 internal edges on to its destination. Totally, from one node to any node in the hypercube, it takes at most $log_2 N$, the diameter of an n-node hypercube. In binary coding, the lowest three bits of a node's address represent its node address within a cube, and the rest bits represent its cube address. Routing in a hypercube is to flip a different bit each time.

For example, from s (00,000,000,000,000) to d (00,001,110,101,011), it takes 7 hops. The path is s(00,000,000,000,000)

→ (00,000,000,000,001)

→ (00,000,000,000,011)

→ (00,000,000,001,011)

→ (00,000,000,101,011)

→ (00,000,010,101,011)

→ (00,000,110,101,011)

→ d(00,001,110,101,011)

4.6.3.3 Routing in Metacube

Routing in the Metacube is not like routing in the hypercube. In the Metacube, a node only connects to nodes which differ in one bit in class id or node id. Fewer connections make the routing in the Metacube no longer as simple as that of the hypercube. The diameter of a Metacube is bigger than the diameter of a hypercube of the same size but smaller than that of a torus of the same size. The location of the node id in the node address varies according to the value of its class id. The hop count between two nodes in a hypercube can be determined by the number of different bits in their addresses. But in a Metacube, the hop count is not decided by the number of different bits. Always, the routing path in a Metacube network is equal or longer than in a hypercube as shown in Figures 4.7 and 4.8. From node (0,0,0,0,0) to node (1,0,0,0,0), it takes only one hop in a 32-node hypercube but 3 hops in a Metacube of the same size. Another example, from node s (00,000,000,000,000) to node d (00,001,110,101,011), it takes 7 hops for minimal routing in a hypercube to reach the destination, flipping a bit at one time. But in a MC, it takes 10 hops.

(00,000,000,000,000) → (00,000,000,000,001)→ (00,000,000,000,011)

→ (01,000,000,000,011)→ (01,000,000,001,011) → (01,000,000,101,011)

→ (11,000,000,101,011)→ (11,001,000,101,011)

→ (10,001,010,101,011) → (10,001,110,101,011)

→ (00,001,110,101,011)

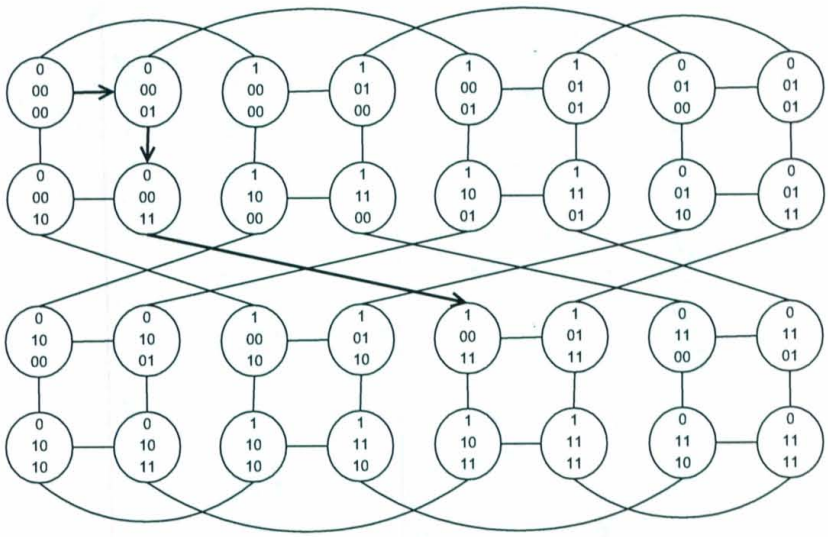

Figure 4.7: Routing in a 32-node Metacube

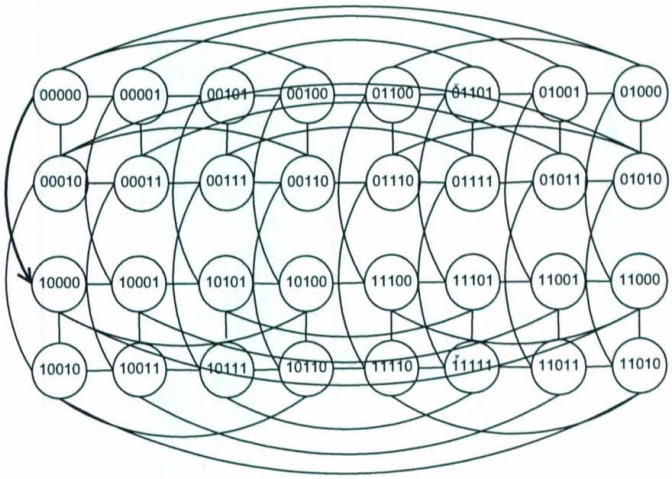

Figure 4.8: Routing in a 32-node Hypercube

4.7 Summary

NS-2 is an open source network simulator which allows designers to evaluate network performance and to modify source code to meet specific research requirements. A few modifications to NS-2 have been made to accommodate our simulation. When projecting the three architectures to 2D layout, different lengths of wires need to be taken into account. For regularity, a grid arrangement of nodes is assumed. Minimal deterministic routing has small overhead and is both dead-lock and live-lock free. Therefore, minimal deterministic routing is adopted for all three topologies under consideration.

Chapter 5

Performance Analysis

In this chapter, we study the performance of the three target architectures under different traffic conditions. Aside from different traffic patterns, the channel load has a great impact on network performance. High channel load can incur excessive delay and packet loss. We change the channel load by varying the injection rate. Three injection rates are considered: 10%, 20% and 30%, which represents a light load, a moderate load and a heavy load, respectively. The corresponding channel load for 10% injection rate can be found in Table 4.1. Because of the linear relationship, the corresponding channel load for 30% injection rate can be obtained by tripling the values in Table 4.1. Network size is another factor needed to be taken into account. How network performance scales up as the network size increases reflects the scalability of the network. In our study, we consider network sizes of 32, 64, 128, 512 and 1024 nodes, respectively. A fixed packet size is used throughout our research.

5.1 Performance Metrics

Performance metrics quantify the comparisons among different network topologies. High performance is the most important goal of a network design [39][40]. Latency is an important metric to evaluate a network. If the latency requirement is not met, a network may cause significant delay to the communication between components. Excessive latency can degrade system performance, and sometimes may even render some real-time applications unusable. Throughput defines the amount of data delivered per time unit, which is closely related to the loss rate performance. Given the same generation rate, a network with higher throughput has lower loss rate, and vice versa. Loss rate is a key performance metric to achieve the desired level of Quality of Service(QoS). Like in other literature [5][8][21][41][42], we focus on the three critical performance metrics: latency, loss rate and throughput in our performance evaluation. In general, under the same traffic, networks with short delay, low loss rate and high throughput are highly desired.

5.1.1 Average End-to-End Delay

End-to-end delay is the time elapsed for a packet to traverse the network to reach its destination. The delay is the sum of three components: processing time at the transmitter, transport delay and processing time at the receiver. It is obtained by averaging the end-to-end delay of all successfully delivered packets. Note that lost packets are not included in the calculation as their delay will be infinitely large. The average end-to-end delay performance reflects how fast the network can deliver packets to their destinations. A smaller value indicates a better network efficiency. TCP/IP is a reliable stream delivery service, where failure to meet the delay requirement will

cause the retransmission of the delayed message.

5.1.2 Loss Rate

Loss rate is defined as the ratio of the number of lost packets to the total number of packets generated. In the network context, it is calculated by dividing the number of dropped packets by the total number of packets generated by the source nodes. Low loss rate is preferred to ensure QoS. For different applications, there are corresponding maximum acceptable loss rates. Often, different loss rates are required for different applications. For example, higher than 5% loss rate will affect the service quality in Voice over IP (VoIP) applications and higher than 0.1% loss rate is unacceptable for TCP/IP [43].

5.1.3 Throughput

Throughput is the data rate in bits-per-second (b/s) for packets successfully delivered to their destinations. Compared to a bandwidth requirement, throughput is a more appropriate metric to evaluate network performance. Throughput provides the maximum accepted traffic and it is related to the peak sustainable data rate for the system [42].

5.2 Confidence Analysis Basics

Statistical analysis quantifies the reliability of the obtained results. Due to the stochastic property of network simulation, each run may yield a different result. It

is possible that the result from a single trial is not accurate. Confidence analysis is based on samples, which are a subset of elements taken from a population. To achieve a given confidence level, corresponding sampling strategies must be determined. From the perspective of data analysis, the true mean is of greater importance than that of the sampled data. Although the confidence interval(CI) does not determine the true mean, it does give the probability of the true mean lying within the CI. CI quantifies the confidence level. For example, 95% confidence interval means that the probability of the true mean lies with the interval is 95%. Unless otherwise specified, 95% confidence is adopted in our study.

When a new RNG object is created, it is automatically seeded to the beginning of the next independent stream of random numbers. Sometimes, multiple independent replications of a simulation are needed. For each replication, a different sub-stream should be used to ensure that the random number sub-streams are independent [44].

Suppose a sample set is $\{X_1, X_2, X_3, \cdots, X_n\}$. The mean of this sample set is given by

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}, \tag{5.1}$$

where n is the size of the sample set, a.k.a., the *degree of freedom* in CI.
The variance σ^2 is defined as

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu)^2, \tag{5.2}$$

where μ is the true mean of the population. However, the true mean μ is unknown, only the sample mean $\overline{X}$ is available. By replacing μ with $\overline{X}$, the sample variance

${S_x}^2$ is obtained as

$$ {S_x}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2. \tag{5.3} $$

The sample mean $\overline{X}$ has a normal distribution with the Probability Density Function (pdf) given by

$$ P = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{X-\mu}{2\sigma^2}). \tag{5.4} $$

Again, neither μ nor σ is known. By replacing μ with $\overline{X}$, σ^2 with ${S_x}^2$, Equation (5.4) becomes

$$ \hat{P} = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{X-\mu}{2\sigma^2}). \tag{5.5} $$

The T distribution is a continuous probability distribution that arises when estimating the mean of a normally distributed population in situations where the sample size is small and the population standard deviation is unknown. The tail probability T is obtained from the t-table based on the degree of freedom and confidence interval. The true mean is computed with a certain level of confidence. The estimate of the true mean μ is given by

$$ \mu = \overline{X} \pm T \cdot \frac{S_x}{\sqrt{n}}. \tag{5.6} $$

As the sample size increases, the sample variance approaches the true variance. As a result, the distribution becomes close to the normal distribution. In Table 5.1, *df* stands for the degree of freedom and p is the confidence interval. For example, under 95% confidence interval, when *df* is 5, the t value is 2.015048.

Table 5.1: T-table with Right-tail Probability

$df \backslash p$	0.4	0.25	0.10	0.05	0.025	0.01	0.005	0.0005
1	0.324920	1.000000	3.077684	6.313752	12.70620	31.82052	63.65674	636.6192
2	0.288675	0.816497	1.885618	2.919986	4.30265	6.96456	9.92484	31.5991
3	0.276671	0.764892	1.637744	2.353363	3.18245	4.54070	5.84091	12.9240
4	0.270722	0.740697	1.533206	2.131847	2.77645	3.74695	4.60409	8.6103
5	0.267181	0.726687	1.475884	2.015048	2.57058	3.36493	4.03214	6.8688

The shape of the t-distribution changes with the degree of freedom. Additionally, the confidence interval can also be used to compare two sets of results to quantify how different they are, as shown in Equations (5.7) and (5.8).

$$P_r(-c < T < c) = 1 - \alpha, \tag{5.7}$$

$$P_r(\overline{X} - \frac{cS_x}{\sqrt{n}} < \mu < \overline{X} + \frac{cS_x}{\sqrt{n}}) = 1 - \alpha. \tag{5.8}$$

The t statistic can also be used to compare two sets of results, as shown in Equation (5.9). With a certain confidence, we can determine whether two sets of results are considered to be significantly different or not.

$$t = \frac{(\overline{X_1} - \overline{X_2}) - (\mu_1 - \mu_2)}{\sqrt{\frac{{S_1}^2}{n_1} + \frac{{S_2}^2}{n_2}}}. \tag{5.9}$$

The confidence interval for the difference in means $\mu_1 - \mu_2$ is given by

$$(\overline{X_1} - \overline{X_2}) \pm t^* \sqrt{\frac{{S_1}^2}{n_1} + \frac{{S_2}^2}{n_2}}, \tag{5.10}$$

where t^* is the upper $(1-\alpha)/2$ critical value for the t-distribution with n degrees of freedom, and n is the smaller value of $n_1 - 1$ and $n_2 - 1$. If the confidence interval includes 0, with a given level of confidence, it is safe to say that there is no significant difference between the means of the two populations. If two statistics have non-overlapping confidence intervals, they are necessarily significantly different but if they have overlapping confidence intervals, it is not necessarily true that they are not significantly different. The reason is that *CI overlap space* overlaps with both *not significant space* and *significant space*, making it hard to tell if the difference is significant or not within the *CI overlap space*. But, it is very clear that there is no confusion within *No CI overlap space*, as Table 5.2 shows.

Table 5.2: The Relation of Not-significant space, Significant Space, CI Overlap Space and No CI Overlap Space

Not Significant	Significant	
	CI Overlap	No CI Overlap

5.3 Confidence Analysis

The purpose of conducting confidence analysis is to examine if results from different trials are consistent. When repeating the same simulation, if there is an obvious discrepancy among different single trials, the corresponding confidence analysis is necessary. Otherwise, a single trial is sufficient for the performance evaluation.

Networks of different sizes, from 32 nodes to 1024 nodes, are studied for the confidence analysis. In the confidence analysis, all of the traffic distributions that will be adopted in the performance analysis are discussed including three spatial distributions: bit-complement, uniform random and hot spot; and three temporal distributions: Poisson

distribution, MMPP and BPD. All the results reported here are based on 100 trials. CI min and CI max represent the lower bound and the upper bound of the CI interval, respectively, with 95% confidence. Because the results are similar, we only present the confidence analysis for Metacube networks in this chapter. The other analysis results can be found in Appendix A.

5.3.1 Confidence Analysis of Network Performance Based on Bit-complement and Poisson Distribution

In this subsection, a series of simulations based on bit-complement and Poisson distributions are conducted. A statistical analysis of the performance of a 128-node Metacube architecture network is conducted and the results are shown in Table 5.3. It is observed that the individuals in the sample set fluctuate slightly around their mean, which is exactly what is expected. With 95% confidence, the true mean for the loss rate falls within (0.004998539, 0.005604786). To test if there is consistency for a small network, a statistical analysis of a 32-node Metacube network is conducted and the results are shown in Table 5.4.

Based on Table 4.1, networks are lightly loaded and heavily-loaded under 10% and 30% injection rate, respectively (the average channel load under 30% injection rate can be calculated by tripling the value under 10% injection rate). To see if there exists good consistency as channel load increases, a 30% injection rate is also considered. Table 5.5 and 5.6 shows the confidence analysis of 32-node and 128-node Metacube networks under 30% injection rate, respectively. It can be seen from the results that after increasing the injection rate to 30%, there is a slight increase for point-to-point delay, almost linear increase for the throughput, and significant increase for the loss rate.

Table 5.3: Confidence Analysis of 128-node Metacube, 10% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00462098	101.826290	0.00530166
Standard Deviation	1E-07	0.276475	0.000144145
CI min	0.00462062	100.962779	0.00499854
CI max	0.00462133	102.6898001	0.00560479

Table 5.4: Confidence Analysis of 32-node Metacube, 10% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00262302	25.457202	0.00526019
Standard Deviation	2E-07	0.0916171	0.000181257
CI min	0.00262240	25.172950	0.00469782
CI max	0.00262364	25.741455	0.00582256

In general, all the confidence analysis results reported in this subsection show a high consistency among a single trial and a 100-trial sample set. Therefore, it gives us the confidence that the result from a single trial can be used for the performance analysis. For average delay, throughput and loss rate performance, the standard deviation is only a small percentage of the mean.

Table 5.5: Confidence Analysis of 32-node Metacube, 30% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00271252	72.647805	0.0523918
Standard Deviation	6.43224E-07	0.123269776	0.00040937
CI min	0.00271052	72.265347	0.0511217
CI max	0.00271452	73.030264	0.0536619

Table 5.6: Confidence Analysis of 128-node Metacube, 30% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.0047102	291.138341	0.0527042
Standard Deviation	4.49467E-07	0.366484	0.000293204
CI min	0.00470881	290.001280	0.0517945
CI max	0.00471160	292.275401	0.0536139

5.3.2 Confidence Analysis of Network Performance Based on Bit-complement and MMPP Distribution

In the previous subsection, the simplest temporal distribution, i.e., the poisson distribution, was applied to Metacube networks. In this subsection, burstiness is incorporated in the traffic generation. The spatial distribution follows the bit-complement distribution; and the temporal distribution follows MMPP distribution. Three network sizes, 128, 512, and 1024, are considered under 10% injection rate and 0.8 burst rate. The confidence analysis results are shown in Tables 5.7 - 5.9. Because of the relatively high node degree, the 512-node Metacube network has a similar loss rate as the 128-node Metacube. For the 1024-node Metacube network, the loss rate is much higher than that of 512-node Metacube as shown in Table 5.9. Regardless of the size difference, for all the simulations in this subsection, a good consistency is observed when compared to that from single trial using the same setup.

Table 5.7: Confidence Analysis of 128-node Metacube, 10% Injection Rate, 0.8 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00463296	100.609023	0.0182861
Standard Deviation	4.69687E-07	0.359456	0.000313386
CI min	0.00463150	99.493769	0.0173138
CI max	0.00463442	101.724278	0.0192585

Table 5.8: Confidence Analysis of 512-node Metacube, 10% Injection Rate, 0.8 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00740513	401.937271	0.0181410
Standard Deviation	1.06983E-06	1.43499233	0.00030523
CI min	0.00740181	397.485040	0.0171940
CI max	0.00740845	406.389502	0.0190880

Table 5.9: Confidence Analysis of 1024-node Metacube, 10% Injection Rate, 0.8 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.0167180	726.847666	0.113355
Standard Deviation	5.33048E-06	2.953631	0.000837548
CI min	0.0167015	717.6836807	0.110756
CI max	0.0167346	736.011652	0.115953

5.3.3 Confidence Analysis of Network Performance Based on Uniform Random and Poisson Distribution

In this subsection, the spatial distribution follows the uniform random distribution and the temporal distribution follows the Poisson distribution. Two sizes of networks are studied in this subsection: 32-node and 128-node Metacube networks. 10% and 30% injection rates are applied. Results are shown in Table 5.10 - 5.13. At first, we study a 32-node Metacube network under two injection rates: 10% and 30%. Table 5.10 shows the confidence analysis of the 32-node Metacube with a 10% injection rate. We increase the injection rate to 30% as shown in Table 5.11. When increasing the injection rate to 30%, there is a slight increase in point-to-point delay and dramatic increase in the loss rate. Because of the increased loss rate, the throughput does not increase linearly with the injection rate.

Then, the 128-node Metacube network is studied with two injection rates. Table 5.12 and Table 5.13 give the confidence analysis of the 128-node Metacube with a 10% injection rate and a 30% injection rate, respectively. Comparing the 32-node and the 128-node Metacube network under 10% injection rate, there is a nearly linear relationship between their throughput. This is because ideally, if there is no packet loss, throughput should be nearly proportional to network size. A good consistency is observed when compared to that from single trial using the same setup.

Table 5.10: Confidence Analysis of 32-node Metacube, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00214272	25.589248	0.00047304
Standard Deviation	1.75252E-05	0.102001	6.67887E-05
CI min	0.00208835	25.272777	0.00026582
CI max	0.00219709	25.905718	0.00068026

Table 5.11: Confidence Analysis of 32-node Metacube, 30% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00271252	72.64780544	0.0523918
Standard Deviation	6.43224E-07	0.123270	0.00040937
CI min	0.00271052	72.265347	0.0511217
CI max	0.00271452	73.030264	0.0536619

Table 5.12: Confidence Analysis of 128-node Metacube, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00393163	100.522779	0.0179471
Standard Deviation	1.80037E-05	0.243803	0.000379015
CI min	0.00387577	99.766352	0.0167711
CI max	0.00398749	101.279205	0.0191230

Table 5.13: Confidence Analysis of 128-node Metacube, 30% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00434034	272.584013	0.112987
Standard Deviation	3.08552E-05	0.654403	0.00133938
CI min	0.00424461	270.553651	0.108832
CI max	0.00443607	274.614375	0.117143

Table 5.14: Confidence Analysis of 128-node Metacube, 30% Injection Rate with one buffer (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00449381	292.447320	0.0481428
Standard Deviation	3.18122E-05	0.574319	0.00118329
CI min	0.00439511	290.665428	0.0444715
CI max	0.00459251	294.229213	0.0518141

Table 5.15: Confidence Analysis of 64-node Metacube, 10% Injection Rate (Hotspot, Pareto)

	Average Delay	Throughput	Loss Rate
Mean	0.00376033	50.583158	0.0849934
Standard Deviation	2.89985E-05	0.294799	0.00229589
CI min	0.00367036	49.668510	0.0778701
CI max	0.00385030	51.497806	0.0921166

5.3.4 Confidence Analysis of Network Performance Based on Hotspot and Pareto Distribution

In this subsection, the performance of a 64-node Metacube network under 10% injection rate is analyzed. The spatial distribution follows a hotspot distribution; and the temporal distribution follows a BPD. From Table 5.15, a good consistency is observed for average delay, throughput and loss rate. Although 10% injection rate is moderate, the 64-node Metacube network has relatively high loss rate. The reason is two-fold. One is that the low node degree of a 64-node Metacube makes it perform poorly. The other is that the traffic pattern is bursty and unbalanced. BPD is a bursty traffic which makes traffic uneven in terms of time. Hotspot traffic causes uneven traffic spatially, which overloads the hotspot nodes.

5.3.5 Summary on Confidence Analysis

The confidence analysis covers a wide range of simulations for Metacube networks. To cover both moderately loaded networks and heavily loaded networks, both 10% and 30% injection rates were considered. In addition, different traffic distributions were applied to Metacube networks of different sizes. For all the simulations for the confidence analysis, after comparing a single trial and a sample containing 100 trials, a good consistency is observed. The individuals in a sample set fluctuate slightly around their mean. Although each run of the same simulation yields a different result, results from a single trial are reliable, regardless of the network size, traffic distributions or traffic loads. Hence, we have demonstrated that we do not need to conduct further confidence analysis for the following simulations in this chapter. Results from a single trial can confidently be utilized to evaluate network performance, and will be used for the rest of the thesis.

5.4 Impact of Topology on Network Performance

In this section, the performance of networks will be compared and analyzed from the perspective of their architecture properties. When projecting the hypercube and the Metacube to a 2D layout, wires of different lengths are needed to connect all nodes in the networks. Link delay is assumed to be proportional to wire length. After long wires are taken into consideration, all three networks experience longer average delay induced by the longer link delay. The torus has doubled its link delay. For the Metacube and the hypercube, the added link delay depends on how many long wires of each length the packets travel through during simulation. The more long wires a packet travels through, the longer the transmission delay becomes.

(1) Node Degree

Table 5.16 shows the node degree of the torus, the Metacube, and the hypercube network as the size of the networks grows, ranging from 32 nodes to 1024 nodes. The hypercube has the largest node degree among the three network topologies under study. The torus has a fixed node degree of 4, regardless of the size of the network. The Metacube has the lowest node degree and the largest diameter for 32 and 64-node networks; therefore, the Metacube has the poorest performance for those cases as the simulation results have shown.

Table 5.16: Node Degree

Size	Torus	Metacube	Hypercube
32	4	3	5
64	4	3	6
128	4	4	7
512	4	5	9
1024	4	4	10

(2) Link Complexity

Table 5.17 shows the link complexity of torus, Metacube and hypercube for different network sizes. Link complexity is the number of links in a topology. For regular topologies, link complexity is proportional to node degree, i.e., higher node degree leads to higher link complexity, and vice versa. Hypercube has the highest link complexity among the three topologies under consideration. For the network size of 32 and 64, the torus has higher link complexity than the Metacube. For the size of 128 and 1024 nodes, the torus has the same link complexity as Metacube. For 512 nodes, the Metacube has higher link complexity than torus.

Table 5.17: Link Complexity

Size	Torus	Metacube	Hypercube
32	64	48	80
64	128	96	192
128	256	256	448
512	1024	1280	2304
1024	2048	2048	5120

(3) Diameter

Table 5.18 plots the diameter of the torus, the Metacube and the hypercube as the size of the networks grows. In general, network diameter increases along with the network size for all topologies. For 128-node and 1024-node networks, with the same node degree but a smaller diameter, the Metacube has much better performance than torus.

Table 5.18: Diameter

Size	Torus	Metacube	Hypercube
32	4	6	5
64	8	8	6
128	10	8	7
512	22	10	9
1024	32	12	10

(4) Average Hop Count

Average hop count gives a better indicator of path lengths than diameter for the network. It shows the average travel distance, which can be used to estimate communication cost. Table 5.19 shows the average hop count for the torus, the Metacube and the hypercube for different network sizes. Due to the low node degree, the 64-

node Metacube network has the largest average hop count among the three network topologies. The hypercube networks have the lowest average hop count, regardless of size. For 128, 512 and 1024-node networks, the average hop count of the Metacube lies in between the other two.

Table 5.19: Average Hop Count

Size	Torus	Metacube	Hypercube
32	3	3.25	2.58
64	4	6	3
128	6	4.375	3.53
512	12	5.4375	4.5
1024	16	8	5

5.5 Performance Comparison

The three target topological networks are evaluated using three performance metrics: point-to-point delay, throughput and loss rate. Small delay, high throughput and small loss rate are preferred for NoC.

5.5.1 Network Performance under Bit-complement and Poisson Distribution

The sender and the receiver are pre-determined at the beginning of each simulation. The sum of decimal representations of two communicating node IDs is always equal to the size of the network minus one. The requirement for the hypercube is different, because all the communicating peers are a full diameter away from each other. Under

bit complement traffic, the hypercube network is unable to take full advantage of its short diameter. Before wire length is considered, all wires are assumed to be the same length. The situation is worse for the Metacube because it takes more hops to route packets than in the hypercube.

When the effect of long wires is considered in the hypercube and Metacube topologies, most packets have to travel via long wires to arrive at their destinations. Therefore, the torus always has shorter delay than the Metacube and the hypercube, regardless of the network size.

5.5.1.1 Average Point-to-Point Delay

We start with an injection rate of 10%. When uniform length of wires is assumed, the average point-to-point delay of the torus, the Metacube and the hypercube of different sizes is shown in Figure 5.1. The hypercube network has the lowest delay for all sizes among the three network topologies. The result for the injection rate of 20% and 30% are shown in Figures 5.2 and 5.3, respectively. Under bit-complement distribution, communicating pairs are a full diameter away from each other for the hypercube. It is worse for the Metacube, which has lower node degree and larger diameter than the hypercube. Due to the low node degree and the spatial characteristic of the bit-complement distribution, the average point-to-point delay for the 64-node Metacube is much higher than that of the torus and the hypercube. If wire length is not differentiated, the average delay is proportion to the average travel distance. For networks of size 32, 64 and 128, the average travel distance in a torus is smaller than in a hypercube and a Metacube, thus the torus has smaller delay than the hypercube and the Metacube. The situation changes for networks size of 512 and 1024, where

the hypercube and the Metacube have smaller delay than the torus. The reason is that the torus begins to have larger average travel distance than the hypercube and the Metacube. Especially under 30% injection rate, due to its large diameter, the 1024-node torus network has a dramatically high point-to-point delay, which is much higher than the Metacube and the hypercube.

To incorporate the effect of wire length on network performance evaluation, we need to project multidimensional topologies onto a 2D plane. Wires of different lengths are utilized to construct the networks as shown in Tables 4.3 and 4.4. Figure 5.4 shows the average point-to-point delay of three types of networks under 10% injection rate with consideration of different wire lengths. The torus has the lowest delay because of its uniform length short wires. Due to the use of long wires, both the hypercube and the Metacube have larger delay than the results shown in Figure 5.1. Note that only the successfully delivered packets are considered in the calculation of point-to-point delay.

5.5.1.2 Throughput

Figures 5.5 - 5.7 compare the throughput of the torus, the Metacube and the hypercube networks of different sizes without distinguishing different lengths of wires for the injection rate of 10%, 20% and 30%, respectively. Under 10% injection rate, the differences among the three curves are not significant. The curves begin to diverge when the network size increases. Figure 5.8 shows the throughput of the three networks under 10% injection rate when the difference in wire lengths is taken into account. Figures 5.5 and Figure 5.8 have similar curves. Due to longer link delay, long wires lead to a lower throughput.

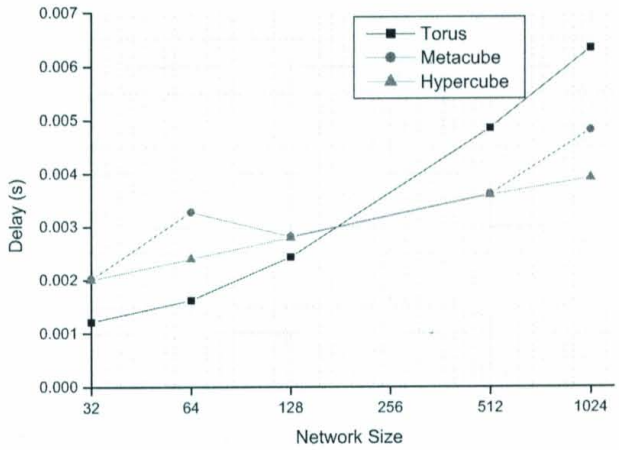

Figure 5.1: Delay Performance (Bit-complement, Poisson, λ=0.1, uniform wire length)

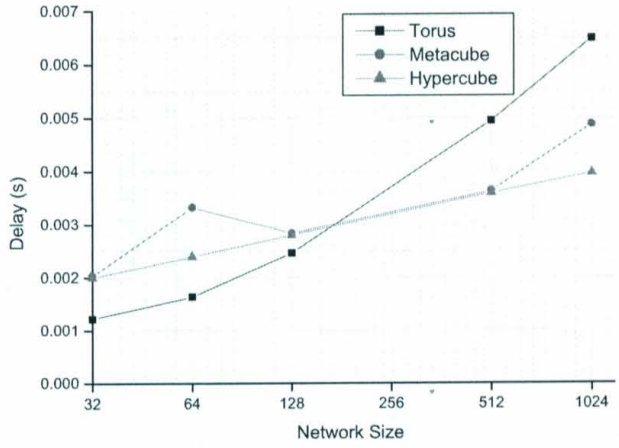

Figure 5.2: Delay Performance (Bit-complement, Poisson, λ=0.2, uniform wire length)

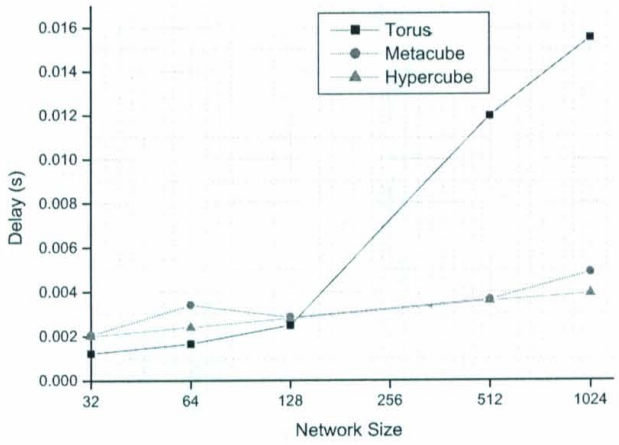

Figure 5.3: Delay Performance (Bit-complement, Poisson, λ=0.3, uniform wire length)

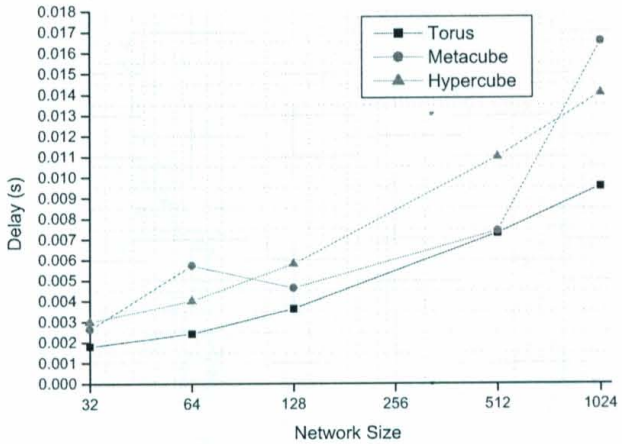

Figure 5.4: Delay Performance (Bit-complement, Poisson, λ=0.1, different wire length)

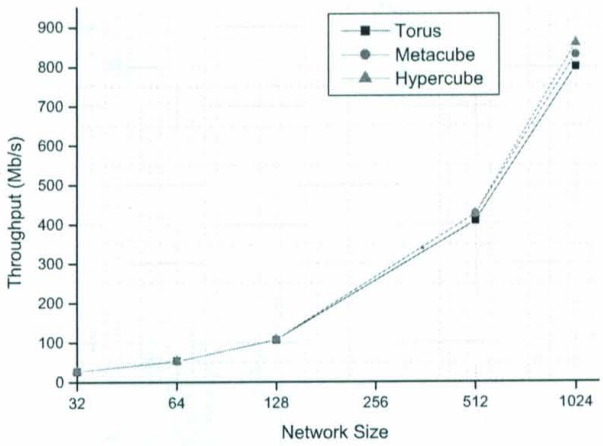

Figure 5.5: Throughput Performance (Bit-complement, Poisson, λ=0.1, uniform wire length)

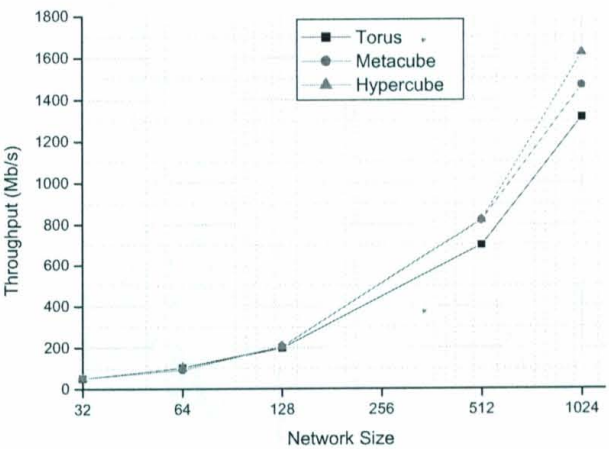

Figure 5.6: Throughput Performance (Bit-complement, Poisson, λ=0.2, uniform wire length)

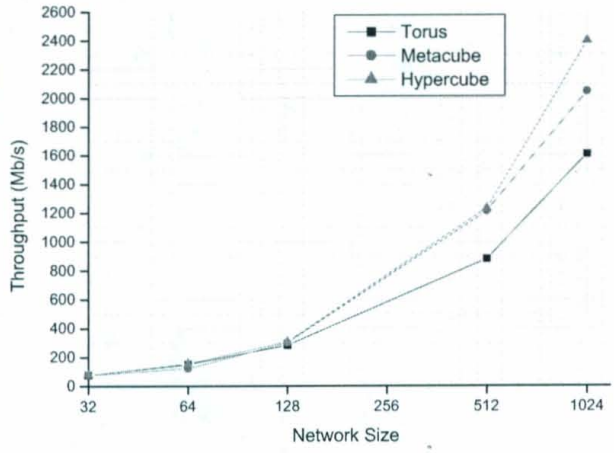

Figure 5.7: Throughput Performance (Bit-complement, Poisson, λ=0.3, uniform wire length)

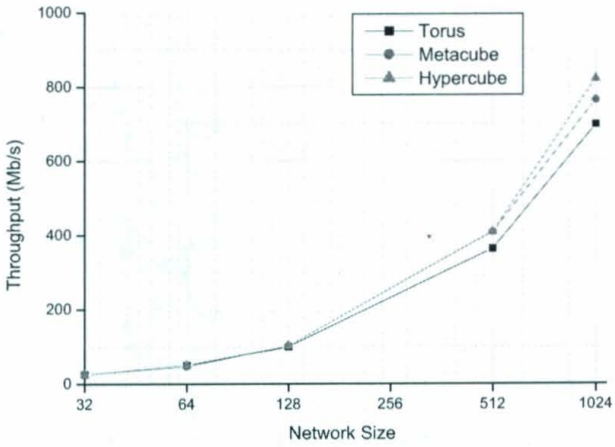

Figure 5.8: Throughput Performance (Bit-complement, Poisson, λ=0.1, different wire length)

5.5.1.3 Loss Rate

Figure 5.9 shows the loss rate of the torus, the Metacube and the hypercube of different sizes under uniform wire length conditions. Figures 5.10 and 5.11 show the loss rate when injection rate increases to 20% and 30% respectively. Assuming there is no buffer for the nodes, due to contention, loss rate increases with the injection rate. Again, for a 64-node network, due to the low node degree, the Metacube has extraordinary loss rate. Figure 5.12 shows the loss rate of the three topologies under consideration after taking into account wire length differences under 10% injection rate. Comparing Figure 5.9 and Figure 5.12, both the torus and the Metacube have noticeable increase in the loss rate when the difference in wire lengths is taken into account. The Metacube has much lower loss rate than the torus for sizes of 128-node, 512-node and 1024-node. Especially, for 128-node and 512-node networks, the Metacube has a loss rate which is close to that of the hypercube.

5.5.2 Network Performance Analysis Based on Bit-complement and MMPP Distribution

Unless otherwise specified, different wire lengths are considered in our analysis for the remainder of the thesis. We study the network performance when the network traffic becomes burstier. The simulation results show that the loss rate increases along with the burst rate. The average channel load of the hypercube network is the smallest among the three, and the hypercube network is more resilient to burstiness than the torus and the Metacube due to high node degree. Hence, the loss rate for the hypercube is not affected as much as those for the torus and the Metacube.

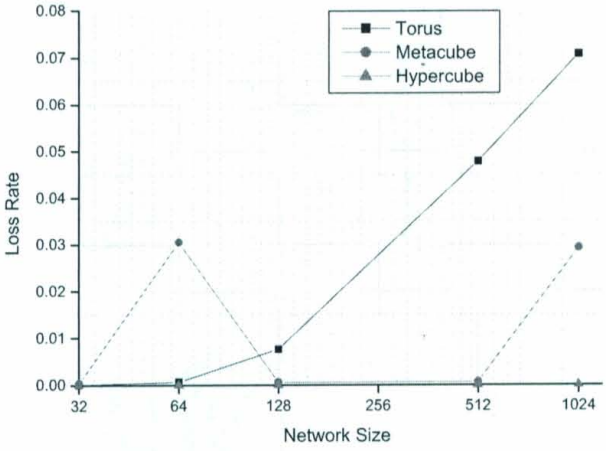

Figure 5.9: Loss rate Performance (Bit-complement, Poisson, λ=0.1, uniform wire length)

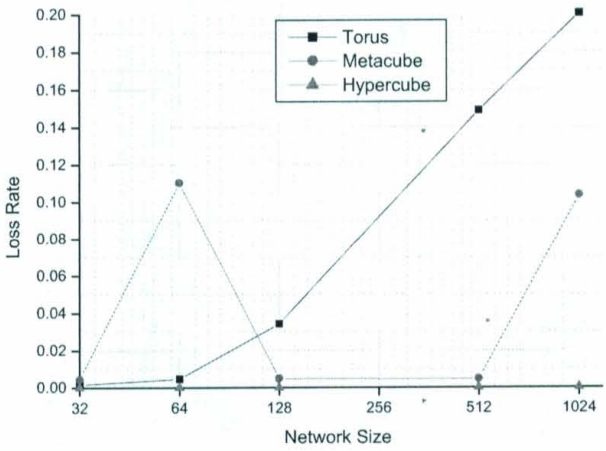

Figure 5.10: Loss rate Performance (Bit-complement, Poisson, λ=0.2, uniform wire length)

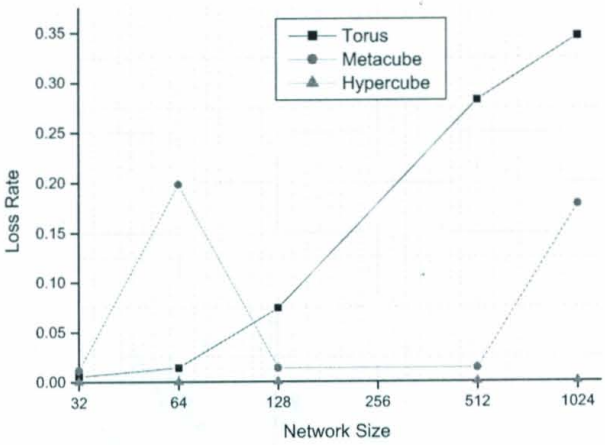

Figure 5.11: Loss rate Performance (Bit-complement, Poisson, λ=0.3, uniform wire length)

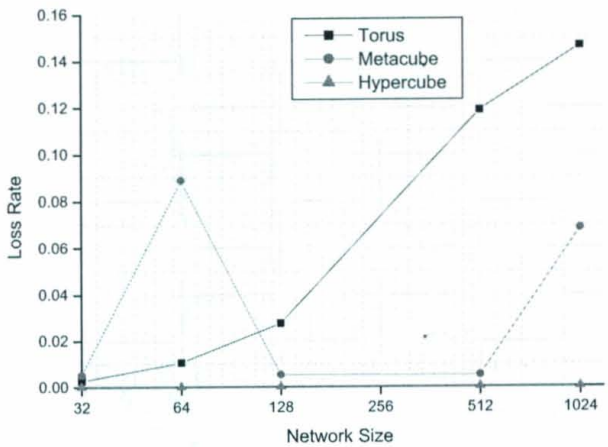

Figure 5.12: Loss rate Performance (Bit-complement, Poisson, λ=0.1, different wire length)

MMPP has short range correlation between packet arrivals. Injection rate is set as 10%, 20% and 30%. 30% is quite high for injection rate especially when the size of networks is large. For bit complement distribution, the hypercube does not lose packets even when the injection rate is up to 30%. To show the influence of burstiness, burst rates of 0, 0.4 and 0.8 are applied to the networks. When the burst rate is 0, MMPP degrades to a Poisson process. Then the burst rate is set to 0.4, and simulation results are obtained. Finally, simulation is conducted with a burst rate of 0.8, which is burstier than when the burst rate is 0.4. But for hypercube (injection rate 10 %), when further increasing the burst rate to 0.8, the hypercube networks perform almost the same as under Poisson distribution. In contrast, the torus and the Metacube networks perform quite differently under different burst rates.

5.5.2.1 Average Point-to-point Delay

Figures 5.13 and 5.14 show the average point-to-point delay of the three topologies under 10% injection rate when the burst rate of the traffic is 0.4 and 0.8, respectively. These two plots are very consistent, because for bufferless networks, average delay is largely determined by spatial distribution. Therefore, burstiness does not produce any significant delay difference for the networks. Not surprisingly, due to the low node degree, the 64-node Metacube has the largest delay which makes it an odd point. From Table 4.3, it can be seen that for 128-node and 512-node networks, unlike the hypercube, the Metacube does not have a significant number of the longest wires, which makes it have similar delay as the torus. However, the 1024-node network, due to relatively low node degree and large number of the longest wires, the Metacube has the largest delay exceeding the hypercube.

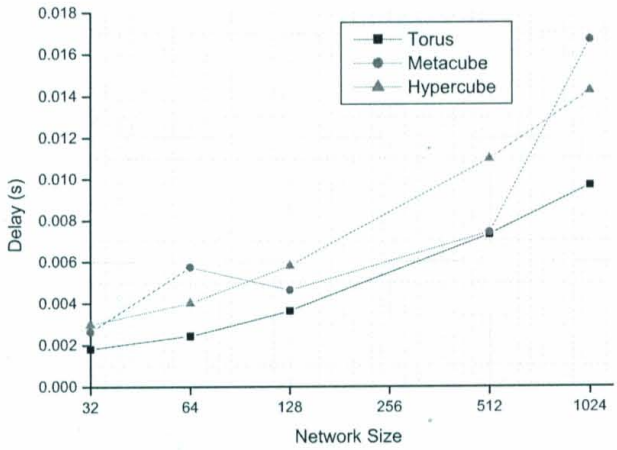

Figure 5.13: Delay Performance (Bit-complement, MMPP, λ=0.1, r=0.4,)

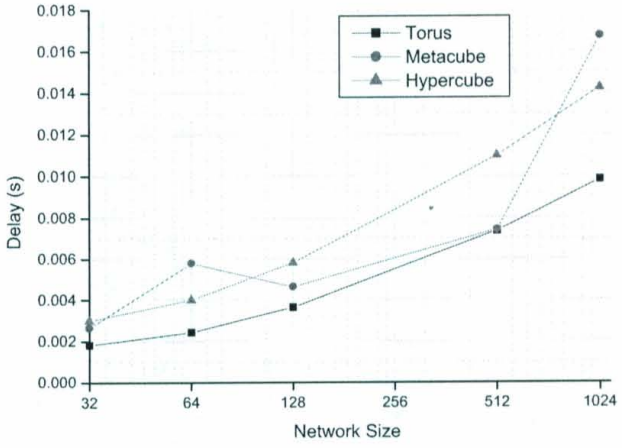

Figure 5.14: Delay Performance (Bit-complement, MMPP, λ=0.1, r=0.8)

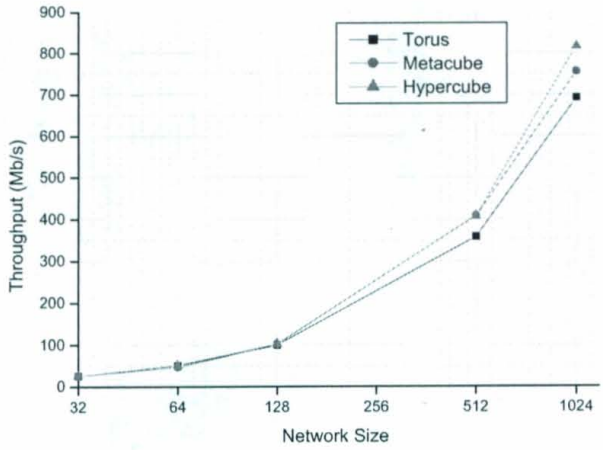

Figure 5.15: Throughput Performance (Bit-complement, MMPP, λ=0.1, r=0.4)

5.5.2.2 Throughput

Under 10% injection rate, the throughput performance for traffic with burst rate of 0.4 and 0.8 is shown in Figures 5.15 and 5.16, respectively. It is observed that, due to increased traffic burstiness, the throughput under 0.4 burst rate is higher than that of under 0.8 burst rate. In general, the hypercube has the highest throughput, followed by the Metacube, then the torus. It can be expected that throughput degradation caused by traffic burstiness will become more dramatic as the injection rate increases.

5.5.2.3 Loss Rate

Figures 5.17 and 5.18 show the loss rate under 10% injection rate for a burst rate of 0.4 and 0.8, respectively. The loss rate of the hypercube remains at 0, even when the burst rate goes as high as 0.8. But for the torus and the Metacube networks, the increased burst rate will cause more packet loss. For a 128-node Metacube, the loss rate

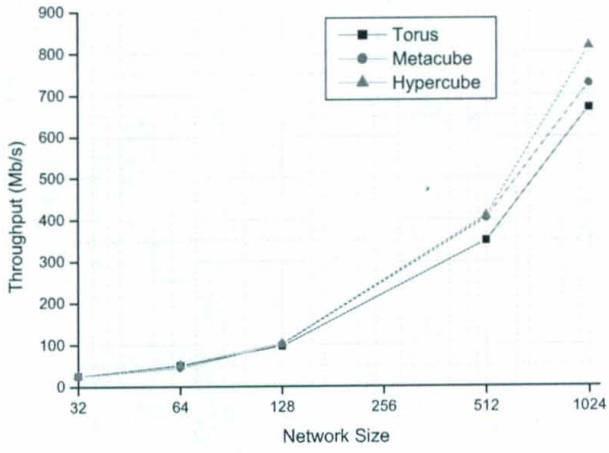

Figure 5.16: Throughput Performance (Bit-complement, MMPP, λ=0.1, r=0.8)

is 0.8% under 0.4 burst rate and 1.8% under 0.8 burst rate. When the injection rate becomes higher, higher loss rate will be encountered, especially for a higher burst rate.

5.5.3 Performance Analysis Based on Uniform Random and Poisson Distribution

Under uniform random traffic, communication pairs are randomly selected during simulations. Each node will have equal probability to be selected as the destination to receive packets from any sender (except from the node itself). Unlike some other traffic patterns, randomness does not favor any particular topology, thus fairness is achieved. Under uniform random traffic, for the hypercube and the Metacube, the average travel distance is greatly reduced from that of the bit-complement distribution, which means communicating pairs are not necessarily far away from each other.

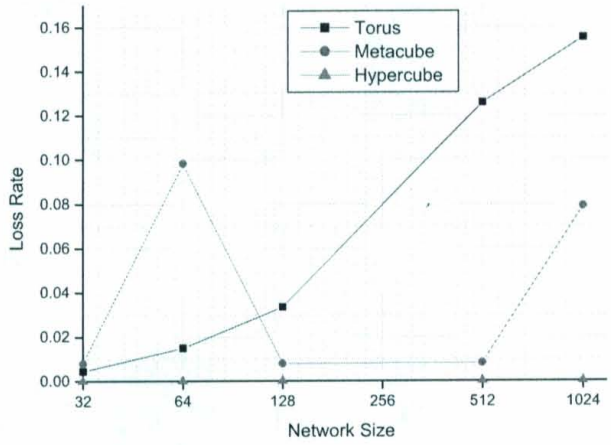

Figure 5.17: Loss Rate Performance (Bit-complement, MMPP, λ=0.1, r=0.4)

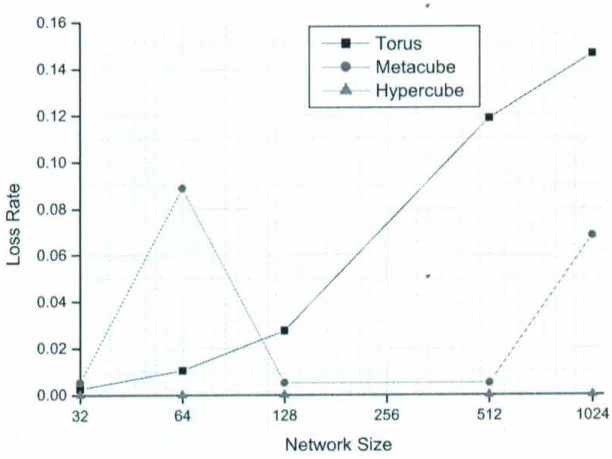

Figure 5.18: Loss Rate Performance (Bit-complement, MMPP, λ=0.1, r=0.8)

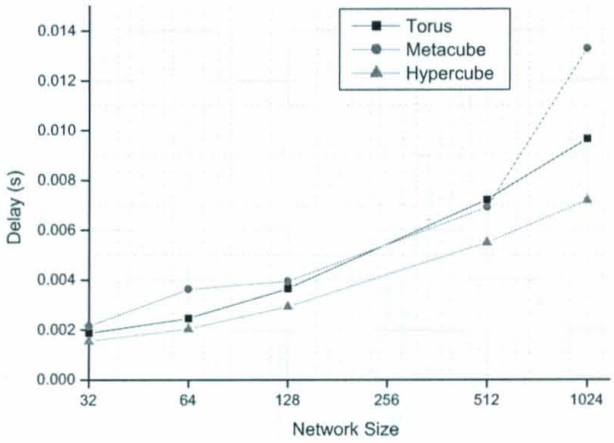

Figure 5.19: Delay Performance (Uniform, Poisson, λ=0.1)

5.5.3.1 Average Delay

The average point-to-point delay of torus, Metacube and hypercube of different sizes is plotted in Figure 5.19 - 5.21 for the injection rate of 10%, 20% and 30% , respectively. When the average hop count becomes the dominating factor, the torus loses the advantage of smaller delay. For the hypercube, the advantage of smaller travel distance completely offsets the disadvantage of longer link delay. The hypercube has the smallest delay among the three for all sizes. As network size increases, the differences among three delay plots increase. Due to the low node degree, the 64-node Metacube still has relatively large delay. Heavier traffic intensifies this intrinsic drawback in the Metacube of this size. Under 10% injection rate, the loss rate of the 64-node Metacube is slightly lower than that of the 128-node Metacube. When the injection rate increases to 30%, the delay of the 64-node Metacube exceeds that of the 128-node Metacube.

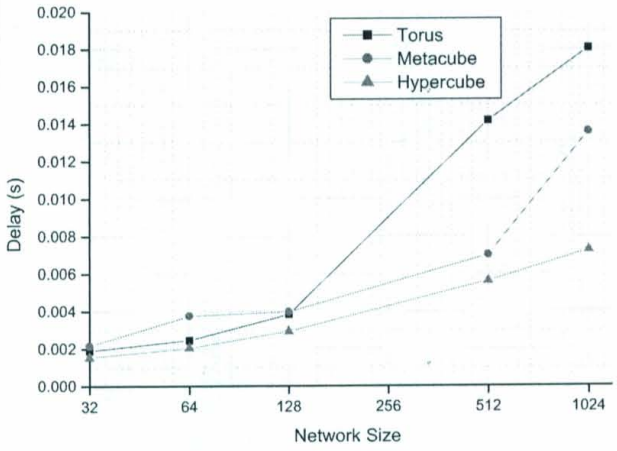

Figure 5.20: Delay Performance (Uniform, Poisson, λ=0.2)

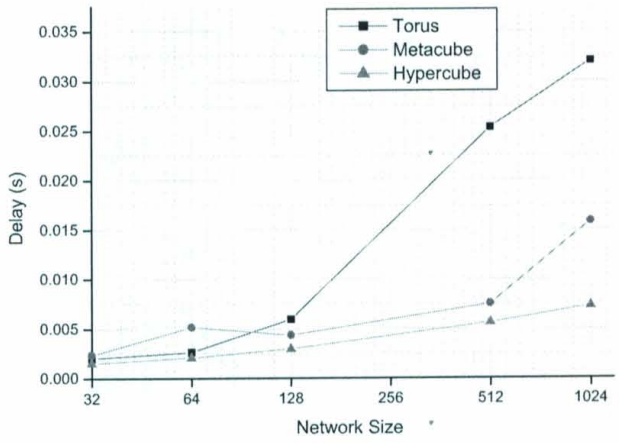

Figure 5.21: Delay Performance (Uniform, Poisson, λ=0.3)

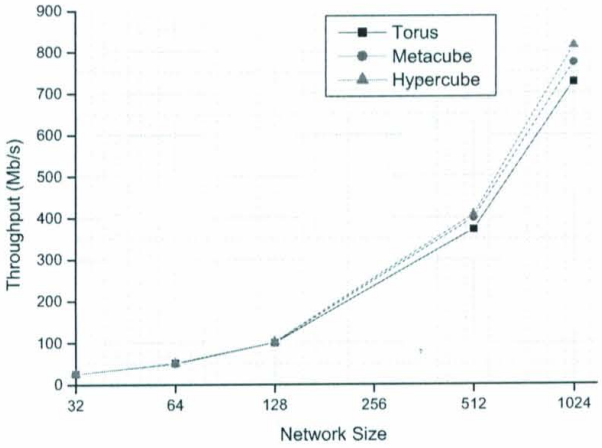

Figure 5.22: Throughput Performance (Uniform, Poisson, λ=0.1)

5.5.3.2 Throughput

Figures 5.22, 5.23 and 5.24 give the throughput of the three networks of different sizes for the injection rates of 10%, 20% and 30%, respectively. In general, throughput increases with the injection rate. The simulation results show that under uniform random distribution, hypercube still has the highest throughput among the three topologies. When the network size is small, the differences among them are not significant. With the increase of injection rate and network size, the throughput performance begins to diverge. The advantage of hypercube becomes more obvious for 1024-node network. For sizes of 128-node and 512-node, the Metacube has very close performance to the hypercube.

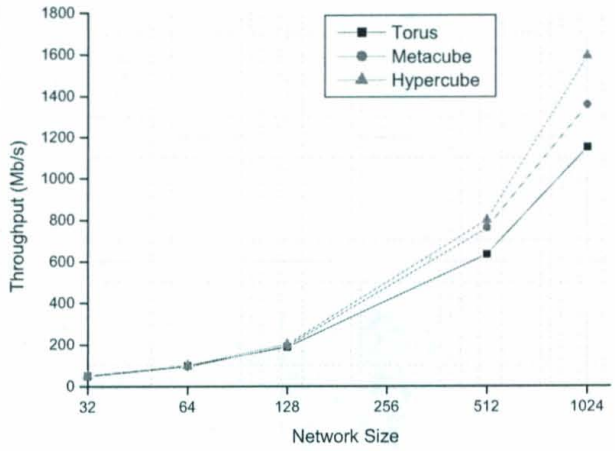

Figure 5.23: Throughput Performance (Uniform, Poisson, λ=0.2)

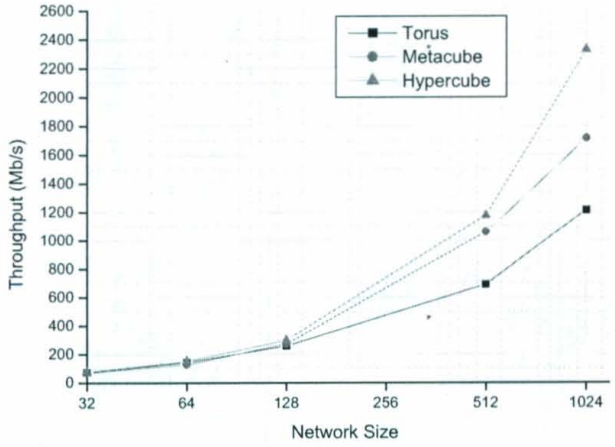

Figure 5.24: Throughput Performance (Uniform, Poisson, λ=0.3)

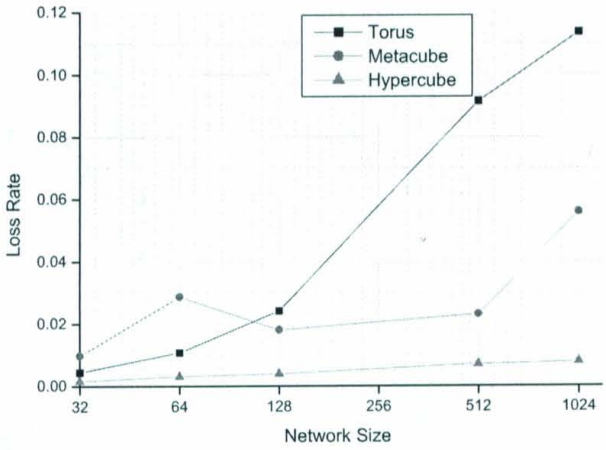

Figure 5.25: Loss Rate Performance (Uniform, Poisson, λ=0.1)

5.5.3.3 Loss Rate

Loss rates are presented in Figures 5.26, 5.26 and 5.27 for injection rate of 10%, 20%, and 30%, respectively. In general, due to contention, the loss rate increases along with the injection rate. Random traffic may cause multiple data flows requesting the same resources, which leads to contention within the network. Because of such contention, unlike under bit-complement traffic, the loss rate of the hypercube network is no longer 0, but it is still much lower than that of the torus and the Metacube. Due to the low node degree, the 64-node Metacube still has relatively high loss rate, but not as dramatic as under bit-complement distribution. The loss rate of the Metacube network lies in between that of the torus and the hypercube.

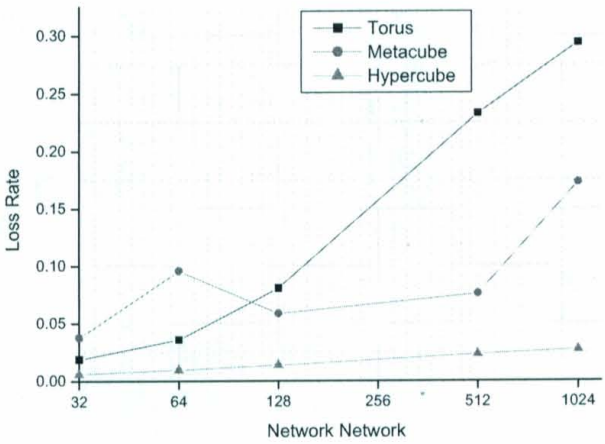

Figure 5.26: Loss Rate Performance (Uniform, Poisson, λ=0.2)

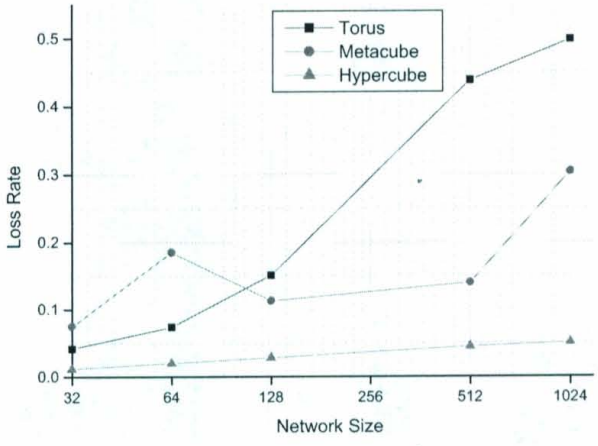

Figure 5.27: Loss Rate Performance (Uniform, Poisson, λ=0.3)

5.5.4 Network Performance Analysis Based on Hotspot Distribution

When the spatial distribution is hotspot and the temporal distribution is Poisson, if the channel load is mild, a small hotspot proportion does not lead to significant degradation of network performance. The reason is that every node generates data with probability P_r independently. If P_r is small, it is unlikely that many nodes request the hotspot at the same time. Thus, the hotspot node does not contributes much to the performance degradation. This is verified by the simulation results.

The probability of a node sending a message to the hotspot node in a time slot is $P_r \times \alpha$. The probability of M $(0 < M \leq N-1)$ nodes sending messages to the hotspot node in the same time slot can be calculated by the binomial distribution.

$$P = \binom{M}{N} \times (P_r \times \alpha)^M \times (1 - P_r \times \alpha)^{N-M}. \tag{5.11}$$

For the BPD ON/OFF traffic model, the ON period lasts for the mean of the ON time, which length is a few multiples of a time interval. If ON periods of different nodes overlap and they are sending messages to the hotspot, the degradation of network performance is significant.

5.5.4.1 Average Delay

The average delay for hotspot traffic with a 10% injection rate is shown in Figure 5.28, which is consistent with that of the uniform random traffic shown in Figure 5.19. For bufferless networks, average delay is largely determined by the average travel distance which is reflected by the spatial distribution. A hotspot distribution with small hotspot proportion shares similar spatial characteristics with the uniform

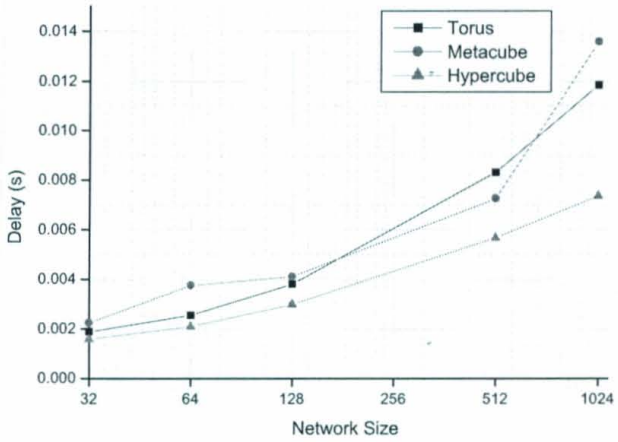

Figure 5.28: Delay Performance (Hotspot, BPD, λ=0.1)

Table 5.20: Confidence Analysis of 128-node Metacube, 10% Injection Rate (Uniform Random, Poisson)

Hotspot Proportion	Average Delay	Throughput	Loss Rate
10%	0.004048	102731.3832	0.066436
20%	0.004338	98681.2203	0.10816
30%	0.004629	94333.2076	0.15901
40%	0.004967	86577.9815	0.22525

random distribution. The only difference lies in the hotspot nodes.

5.5.4.2 Throughput

The throughput for hotspot traffic with a 10% injection rate is shown in Figure 5.29. Compared with other traffic distributions discussed in this thesis, the three target networks consistently have the lowest throughput under this traffic.

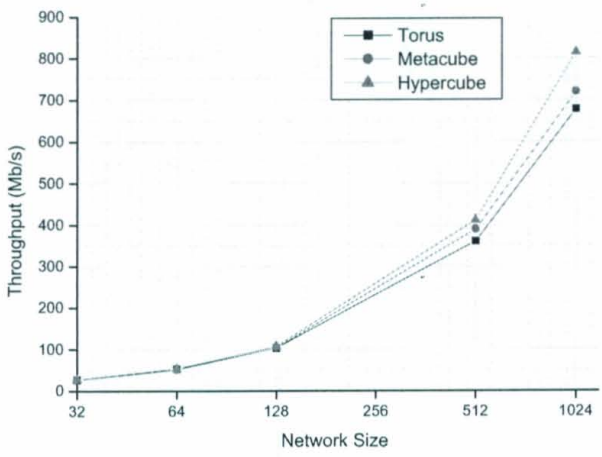

Figure 5.29: Throughput Performance (Hotspot, BPD, λ=0.1)

5.5.4.3 Loss Rate

The loss rate for hotspot and BPD traffic with a 10% injection rate is shown in Figure 5.30. Even with an injection rate of 10%, all the three networks experience significant packet loss. The hypercube network, which usually has a very small loss rate under other traffic scenarios, has a 9% loss rate for a 1024-node network. This traffic scenario can greatly degrade network performance in terms of throughput and loss rate. The loss rates for the 128-node Metacube under different injection rates ranging from 10% to 30% and different hotspot proportion ranging from 10% to 40% are shown in Figure 5.31. It can be observed that the loss rate increases significantly along with the injection rate as well as the hotspot proportion. The loss rates are much higher than those under uniform random distribution with the same injection rate. Due to the overload on hotspots, higher hotspot proportion leads to a higher loss rate.

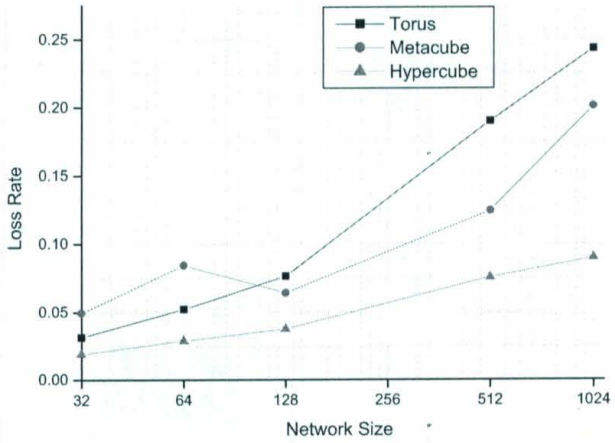

Figure 5.30: Loss Performance (Hotspot, BPD, λ=0.1)

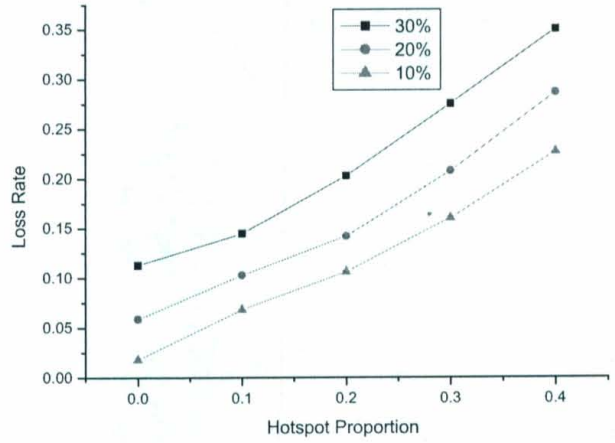

Figure 5.31: Loss Performance (Hotspot, Poisson , λ=0.1, 0.2, 0.3)

5.6 Performance Improvement

In the case of resource contention, buffers can be used to avoid packet loss. In this section, we study the performance improvement by adding an additional buffer to each node. For simplicity, at first one extra buffer space is considered. We chose bit-complement and uniform random as the spatial distributions and the Poisson process as the temporal distribution. To see how the performance improvement varies with the traffic load, three injection rates (*i.e.,* 10%, 20% and 30%) are applied to the target networks. Different sizes, 32-node, 64-node, 128-node, 512-node and 1024-node are used for performance study.

The performance of a 64-node torus network under 30% injection rate is evaluated along with different buffer sizes. Its delay, throughput and loss rate are shown in Figure 5.32 - Figure 5.34, respectively. It is clear that there is a significant improvement in both throughput and loss rate after adding the first buffer. When the buffer size is 3, the loss rate drops to a very low value. Further increasing buffer space to 6, the loss rate becomes 0.

The three important performance metrics are evaluated as shown in Figure 5.35 - Figure 5.52. When the injection rate increases, the performance improvement gained by adding a buffer becomes less obvious. Before adding a buffer to the networks, the three throughput curves in Figure 5.8 begin to split after a node size of 128. After a buffer space is added, the three throughput curves are very consistent for even 1024-node networks as shown in Figure 5.38, which indicates the improvement of throughput for both the torus network and the Metacube network achieved by adding one buffer. Taking the 128-node torus network as an example, under 10% injection rate, the loss rate is reduced from 2.7% to 0.13% after adding a buffer space. However, for networks

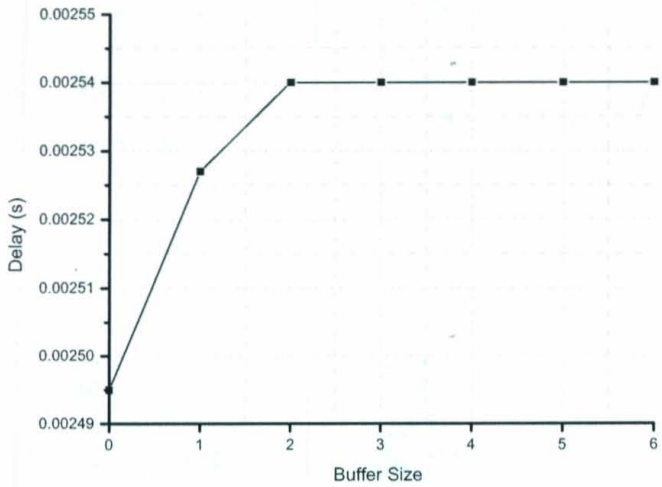

Figure 5.32: Delay Performance (Bit-complement, Poisson, λ=0.3)

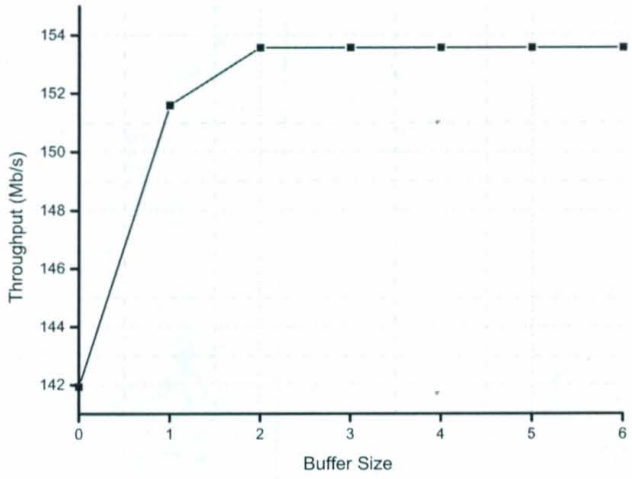

Figure 5.33: Throughput Performance (Bit-complement, Poisson, λ=0.3)

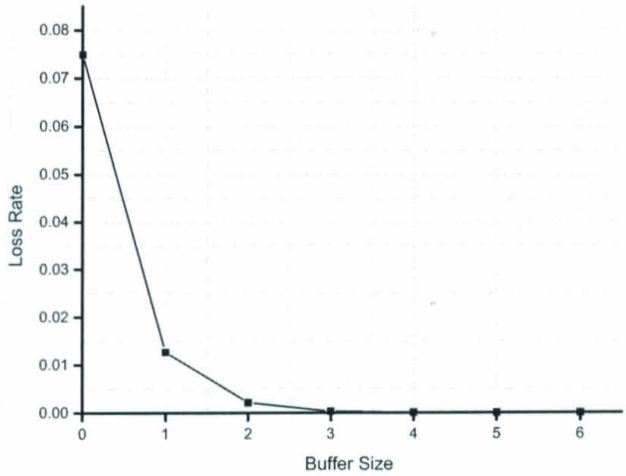

Figure 5.34: Loss Rate Performance (Bit-complement, Poisson, λ=0.3)

of large size, adding a buffer does not always produce significant improvement to the loss rate. When the injection rate is 10%, both the torus and the Metacube have significant loss reduction and the throughput improvement is even significant for the size of 1024 nodes. When the injection rate becomes higher, the improvement for large networks becomes less obvious. For example, under 30% injection rate, there is only 1% improvement for a 1024-node torus network and 4% for a Metacube network after adding a buffer space.

Adding more buffers to the networks does not always incur lower loss rate or higher throughput. If two or more packets arrive at an input port at the same time, only one packet can enter the buffer queue (assuming that particular link is busy and there is vacancy in the queue). The excess packets will be discarded regardless of the buffer size, simply because they cannot enter the same queue at the same time. If this is the case, adding a buffer to this input port cannot lower the loss rate or improve the throughput.

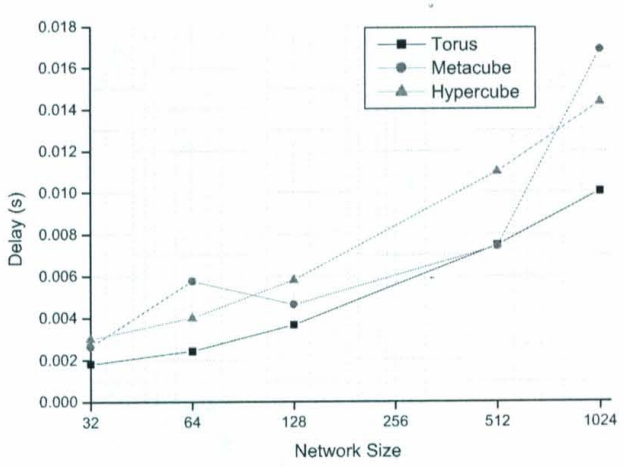

Figure 5.35: Delay Performance (Bit-complement, Poisson, λ=0.1, buffer size=1)

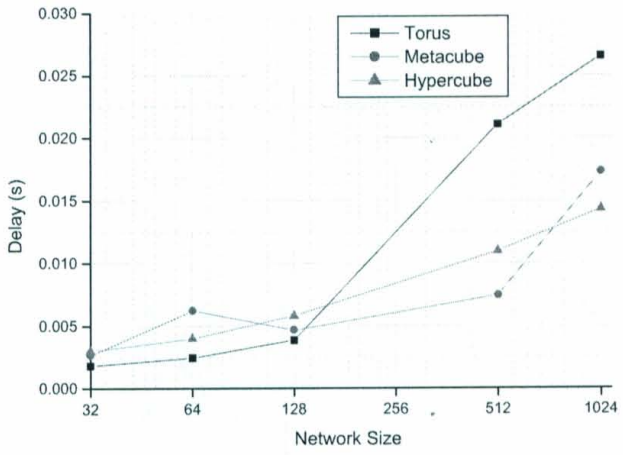

Figure 5.36: Delay Performance (Bit-complement, Poisson, λ=0.2, buffer size=1)

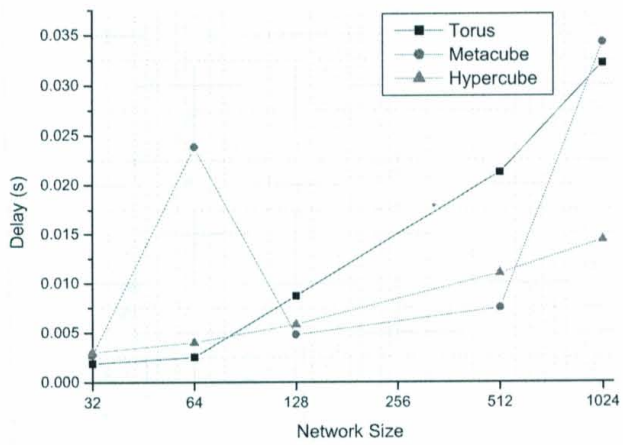

Figure 5.37: Delay Performance (Bit-complement, Poisson, λ=0.3, buffer size=1)

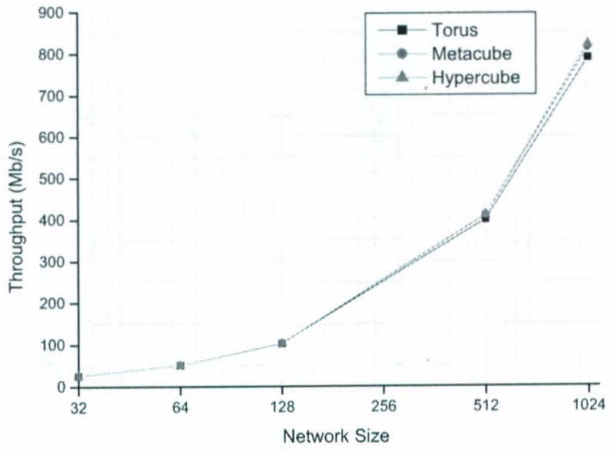

Figure 5.38: Throughput Performance (Bit-complement, Poisson, λ=0.1, buffer size=1)

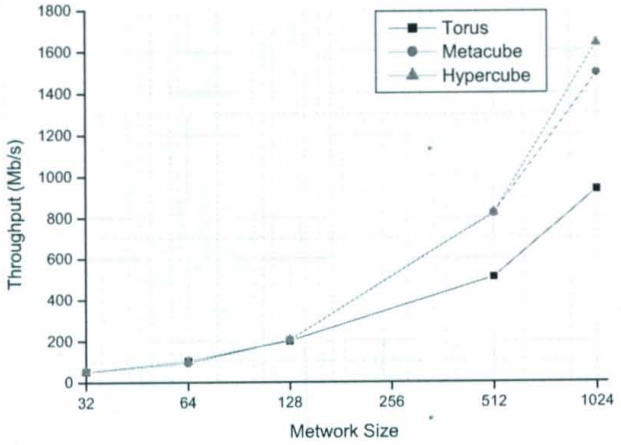

Figure 5.39: Throughput Performance (Bit-complement, Poisson, λ=0.2, buffer size=1)

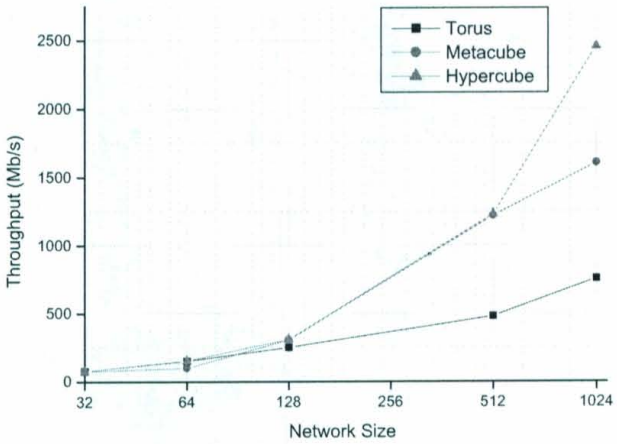

Figure 5.40: Throughput Performance (Bit-complement, Poisson, λ=0.3, buffer size=1)

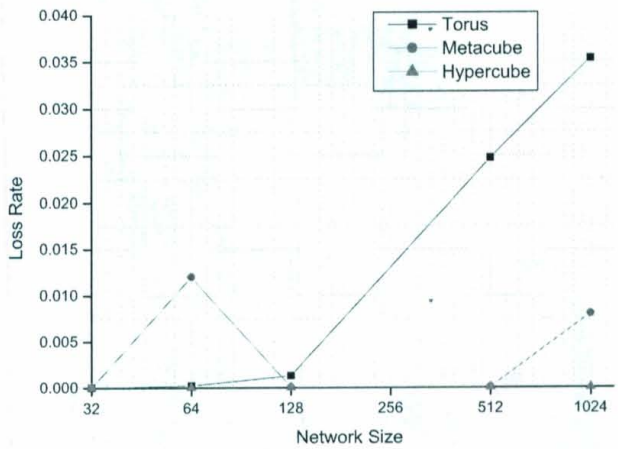

Figure 5.41: Loss Rate Performance (Bit-complement, Poisson, λ=0.1, buffer size=1)

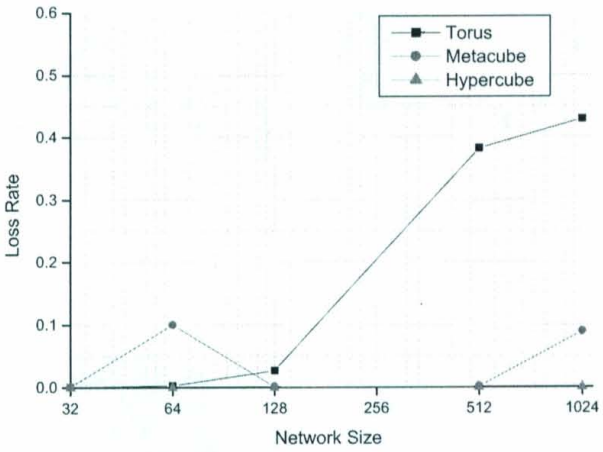

Figure 5.42: Loss Rate Performance (Bit-complement, Poisson, λ=0.2, buffer size=1)

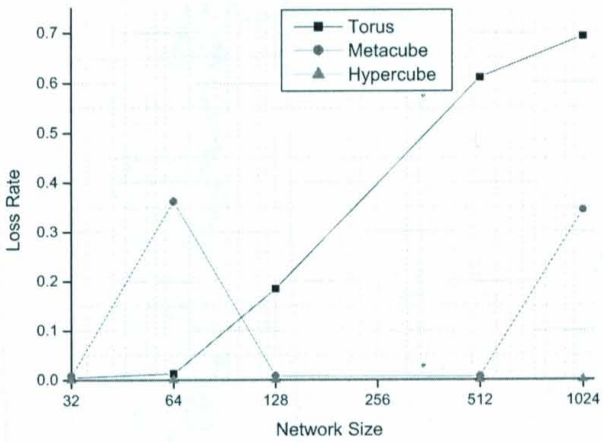

Figure 5.43: Loss Rate Performance (Bit-complement, Poisson, λ=0.3, buffer size=1)

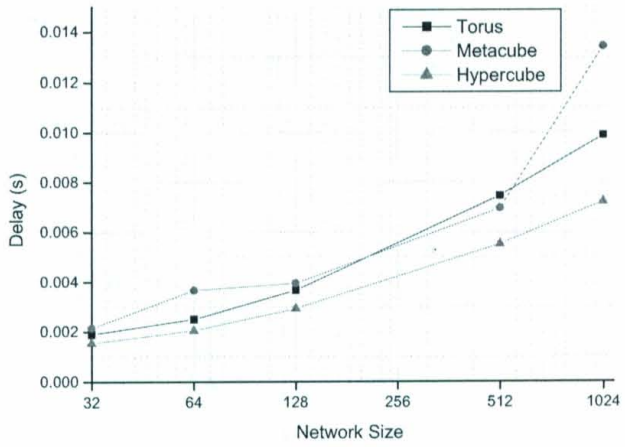

Figure 5.44: Delay Performance (Uniform, Poisson, λ=0.1, buffer size=1)

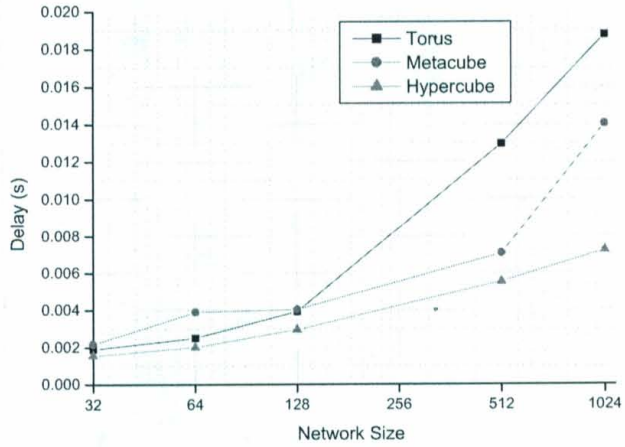

Figure 5.45: Delay Performance (Uniform, Poisson, λ=0.2, buffer size=1)

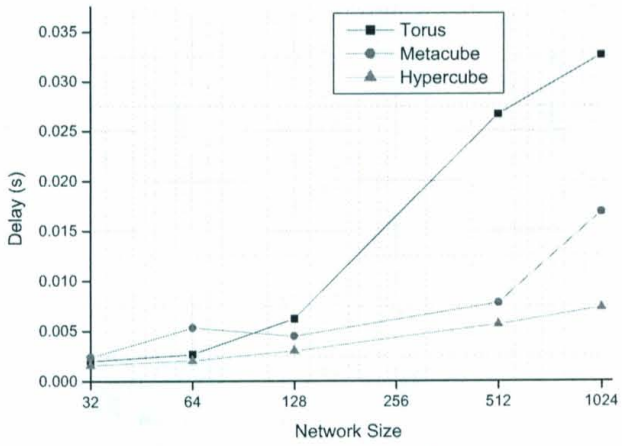

Figure 5.46: Delay Performance (Uniform, Poisson, λ=0.3, buffer size=1)

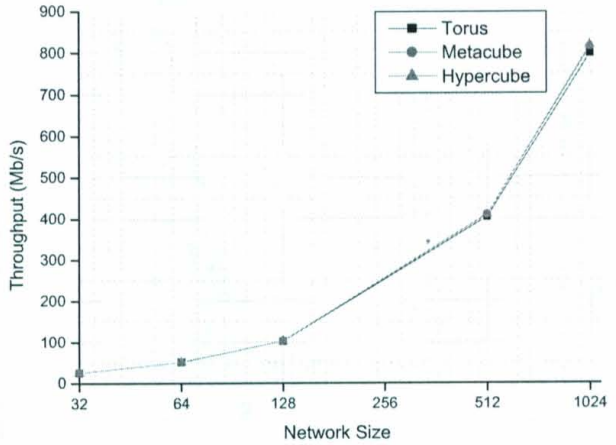

Figure 5.47: Throughput Performance (Uniform, Poisson, λ=0.1, buffer size=1)

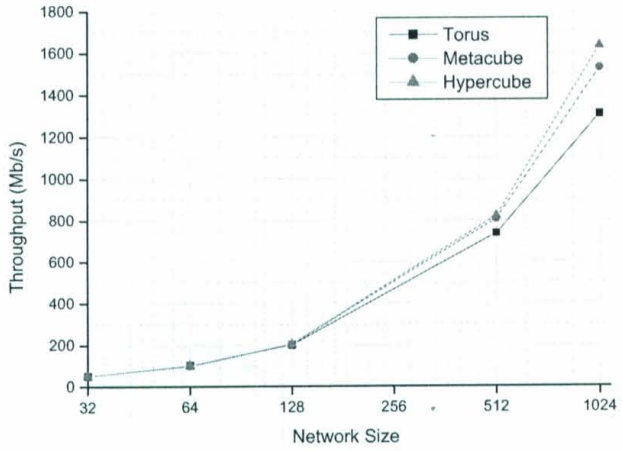

Figure 5.48: Throughput Performance (Uniform, Poisson, λ=0.2, buffer size=1)

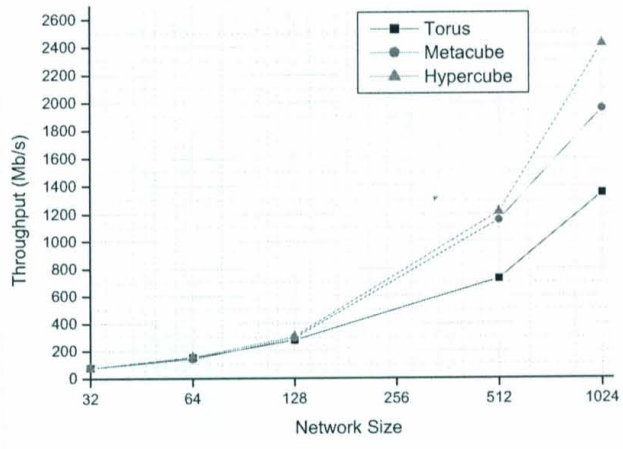

Figure 5.49: Throughput Performance (Uniform, Poisson, λ=0.3, buffer size=1)

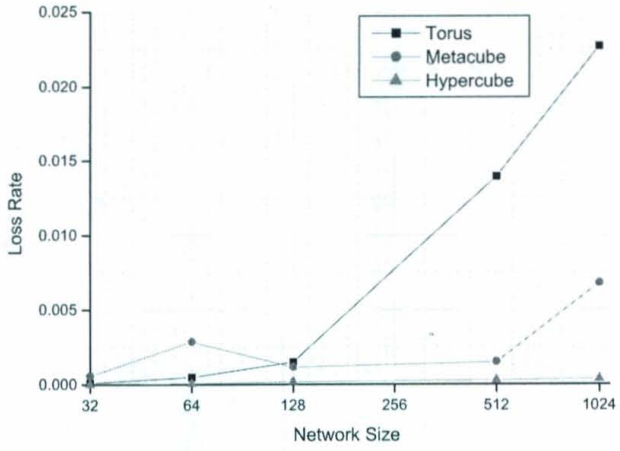

Figure 5.50: Loss Rate Performance (Uniform, Poisson, λ=0.1, buffer size=1)

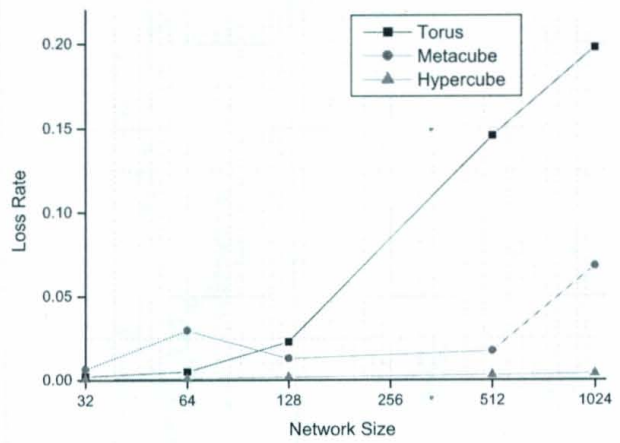

Figure 5.51: Loss Rate Performance (Uniform, Poisson, λ=0.2, buffer size=1)

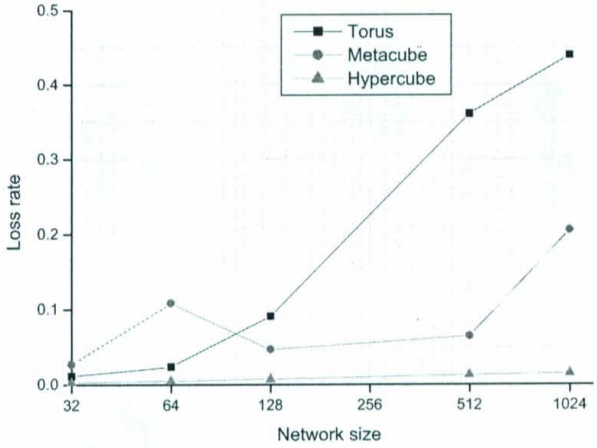

Figure 5.52: Loss Rate Performance (Uniform, Poisson, λ=0.3, buffer size=1)

5.7 Scalability Analysis of Different Topologies

In a scalable network, performance roughly improves in proportion to the total capacity [45]. As the size of NoC increases, it is of great interest to know how its performance scales. The three important performance metrics include point-to-point delay, throughput and loss rate. Networks of 32, 64, 128, 512, and 1024 nodes were studied. Under the same injection rate, if throughput increases approximately linearly with the size of the networks, performance scales. High node degree is a big advantage to maintain scalability. Since it has the highest connectivity among the three topologies, the hypercube outperforms the Metacube and the torus. Through simulations, the performance of the hypercube network scales the best among the three topologies under consideration. The hypercube is also more resilient to load changes and traffic burstiness. Unlike the torus and the Metacube, its loss rate increases very slowly as the network scales. The performance of the torus network degrades the fastest among the three. Although the 32-node and 64-node Metacube networks perform even worse

than the torus, the Metacube networks of 128 and 512 nodes have excellent performance which is very close to the hypercube in terms of the throughput and the loss rate.

5.8 Discussion and Summary

Node degree/link complexity and diameter have great impact on the performance of networks. Networks built on topologies of higher node degree and smaller diameter will outperform networks built on topologies of lower node degree and larger diameter. When choosing topologies, performance is not the only concern for the design. Trade-off needs to be made between performance and cost/implementation complexity. The principle is to pick the topology with the lowest cost from all eligible candidates which can meet the communication requirements, i.e. delay, throughput, and loss rate.

By comparing the performance of the torus, the Metacube and the hypercube, it is found that for networks of 32 nodes and 64 nodes, the Metacube networks have the poorest performance due to its low node degree. The performance differences among the three networks become more and more apparent as the networks scale up. The results show that the Metacube and the hypercube are more resilient to load changes than the torus. The network performance degrades when the injection rate increases, for sizes of 128 nodes or more, the torus degrades the fastest among the three. Under the same injection rate, as network size scales up, the performance degradation is mainly caused by heavier channel load. Increased contention can lead to longer delay, lower throughput and higher loss rate. Under MMPP, the torus and the Metacube networks have different throughput and loss rates under different burst rates. Due to its high connectivity, the hypercube networks is not as much as affected as the torus

networks and the Metacube networks.

Although the hypercube performs the best, the hypercube is expensive in terms of link complexity and node degree. When the sizes of networks are small (32-64 nodes) and channel loads are not heavy, torus is a viable choice. It is cheap and simple to implement. For the 32-node and 64-node cases, the Metacube network performs the worst among the three because it has the smallest node degree and the largest diameter. For 128 nodes and 512 nodes networks, the Metacube starts to outperform torus and exhibits similar performance to the hypercube especially after adding one buffer space. The Metacube of 128-node and 512-node have lower link complexity and fewer long wires, which makes it a viable choice under a moderate load. For a bigger size of network, although the Metacube has similar node degrees to the torus, its smaller diameter contributes to better performance. When network size scales up to 1024 nodes, neither the torus nor the Metacube has very satisfactory performance. The networks have distinct performance for different traffic scenarios. In particular, the loss rate and the throughput are dramatically influenced by the burstiness and the hotspot proportion. With increased burstiness and hotspot proportion, contention is intensified. As a result, more packets will be lost and the throughput becomes lower.

Chapter 6

Conclusions and Future Work

Interconnection network design greatly affects system performance and cost. In this thesis, we study the performance and the scalability of the torus, the Metacube, and the hypercube networks for NoC applications. The performance comparisons among these three networks are investigated under various traffic scenarios.

6.1 Conclusions

The major contributions of the thesis include: A comprehensive study was conducted on network topologies including classic topologies and some recent topologies with their main advantages and disadvantages. We summarize their topological parameters including diameter, node degree, link complexity, *etc.* Three topologies are chosen for performance analysis from the reviewed topologies.

Three traffic generators including Poisson (an independent and memory-less process), MMPP (a short-range dependent process) and BPD (self-similar/long-range correlated process) with tunable parameters, are implemented in NS-2. Due to the desirable characteristics, such as minimal routing distance and freedom from dead-lock,

dimension order routing algorithms are implemented for the three topologies.

The approximate wire cost of each topology is calculated based on a 2D grid layout. The impact of wire length on network performance, especially on delay performance, is analyzed. Performance improvement by adding a buffer space is well studied in this thesis. The performance gain by adding more than one buffer space is also evaluated. When adding a buffer space to a network, there can be a great improvement in its performance, i.e., lower loss rates and higher throughput, especially for smaller networks. When the traffic load within the network is very heavy, adding one buffer space does not seem to contribute much to lower the loss rate. This is because when the traffic load is very heavy, multiple packets are trying to enter the same queue. This conflict cannot be solved by adding buffer space.

A well-rounded performance analysis is conducted under various traffic scenarios for the chosen target topologies. The benefits and drawbacks of them are discussed in terms of their topological characteristics. Based on simulation results, we give some suggestions as to how to select topologies based on traffic loads and size of network. When traffic load is light, the torus can be an option for small networks up to 64 nodes. For sizes of 128 nodes and 512 nodes, the Metacube can be a viable choice, which has a similar performance to the hypercube only with lower node degree. For 1024-node networks, the hypercube has an absolute advantage over the Metacube in terms of performance.

6.2 Future Work

There are a few directions worth further investigation in the future.

This research targets networks from 32 nodes up to 1024 nodes, which is likely rea-

sonable for network sizes for the next at least two or three years in the advancement of NoC. As the size of NoC keeps increasing, it is worth studying the performance of even larger networks, 2048-node or more. This study can be expanded to bigger networks in the future. Implementation of the three networks on FPGA platform will be useful to estimate area cost and more accurately evaluate network performance of the three target network topologies. We are also interested in implementing real applications to the target networks.

Traffic modeling and how to make it applicable to various networks remain challenging to NoC researchers. In this thesis, three popular spatial and three temporal distributions are adopted in this work. If there are other spatial distributions or temporal distributions that can better characterize some NoC traffic, it is definitely worthy to include them as well.

One important issue of a NoC design is to balance its performance in term of throughput or latency and costs such as area usage and power consumption. This thesis focused on performance evaluation, with very little effort on cost estimation except the wire length. When evaluating different topologies, cost is another important issue. Network designers have to make a trade-off between the performance and the cost. It is generally more important to design a capable network with moderate expense than to design a superior network with daunting cost.

Adaptive routing is able to explore all possible paths. The implementation of adaptive routing requires more effort and resources than deterministic routing. Decisions are made based on local information, which may be disadvantageous from the global perspective. For deterministic routing, the same path is taken with the same source and destination addresses. Deterministic minimum routing simplifies the router design, which is implemented with minimal overhead. Design and verification of dynamic network behavior are much more complex than for deterministic networks. Adaptive

routing algorithms can be applied to NoCs as long as their overhead is acceptable. If a more advanced routing algorithm or scheduling scheme can be adopted, network performance can be improved. For example, if there is a highly adaptive, dead-lock and live-lock free routing algorithm applied to these three networks, there will be more paths which could be used to route packets.

References

[1] M. Gordon, "Cramming more components onto integrated circuits," *Electronics magazine*, vol. 38, no. 8, pp. 114–117, April 19 1965.

[2] http://en.wikipedia.org/wiki/32_nanometer.

[3] J. Owens, W. Dally, R. Ho, D. Jayasimha, S. Keckler, and L. Peh, "Research challenges for on-chip interconnection networks," *Micro, IEEE*, vol. 27, no. 5, pp. 96 –108, September 2007.

[4] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "Timing analysis of network on chip architectures for mp-soc platforms," *Microelectronics Journal*, vol. 36, no. 9, pp. 833–845, September 2005.

[5] B. Tobias and M. Shankar, "A survey of research and practices of network-on-chip," *ACM Computing Survey*, vol. 38, no. 1, pp. 1–51, June 2006.

[6] http://www.xbitlabs.com/news/mobile/display/20080603141353_Nvidia_Unleashes_Tegra_System_on_Chip_for_Handheld_Devices.html.

[7] W. Dally and B. Towles, "Route packets, not wires: On-chip inteconnection networks," in *Proceedings of the 38th Annual Design Automation Conference (DAC'01)*, Las Vegas, USA, June 2001, pp. 684–689.

[8] R. Marculescu and P. Bogdan, "The chip is the network: Toward a science of network-on-chip design," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 4, pp. 371–461, March 2009.

[9] H. Lee, N. Chang, U. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transaction on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, pp. 1–20, August 2007.

[10] T. Dumitras and R. Marculescu, "On-chip stochastic communication," in *In Proceedings of the conference on Design, Automation and Test in Europe (DATE '03)*, vol. 1, Messe Munich, Germany, March 2003, pp. 790–795.

[11] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, Pittsburgh, USA, April 2002, pp. 105–112.

[12] R. Marculescu and A. Nandi, "Probabilistic application modeling for system-level perfromance analysis," in *Proceedings of the conference on Design, automation and test in Europe (DATE '01)*, Munich, Germany, March 2001, pp. 572–579.

[13] V. Girish and M. Radu, "On-chip traffic modeling and synthesis for mpeg-2 video applications," *IEEE Transaction on Very Large Scale Integrated Systems*, vol. 12, no. 1, pp. 108–119, January 2004.

[14] M. Radu, Y. Umit, P. Li-Shiuan, J. N. Enright, and H. Atin, "Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives," *IEEE Transaction on Compupter-Aided Design of Integrated Circuit and Systems*, vol. 28, no. 1, pp. 3–21, January 2009.

[15] L. Keong, H. Jing, W. Cai, and S. Nadarajah, "How network topology affects dynamic load balancing," *IEEE Parallel Distributed Technology*, vol. 4, no. 3, pp. 25–35, September 1996.

[16] T. Schonwald, J. Zimmermann, O. Bringmann, and W. Rosenstiel, "Network-on-chip architecture exploration framework," in *in Proceedings of 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools (DSD'09)*, August 2009, pp. 375 –382.

[17] V. Soteriou, H. Wang, and L. Peh, "A statistical traffic model for on-chip interconnection networks," in *in Proceedings of 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Monterey, USA, September 2006, pp. 104 – 116.

[18] A. Weldezion, M. Grange, D. Pamunuwa, Z. Lu, A. Jantsch, R. Weerasekera, and H. Tenhunen, "Scalability of network-on-chip communication architecture for 3-d meshes," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS '09)*, San Diego, USA, May 2009, pp. 114–123.

[19] U. Ogras and R. Marculescu, "It's a small world after all: Noc performance optimization via long-range link insertion," *IEEE Transaction on Very Large Scale Integration Systems*, vol. 14, no. 7, pp. 693–706, July 2006.

[20] U. Ogras, R. Marculescu, H. Lee, and N. Chang, "Communication architecture optimization: Making the shortest path shorter in regular networks-on-chip," in *In Proceedings of the Design Automation and Test in Europe Conference*, Messe Munich, Germany, March 2006, pp. 712–717.

[21] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, USA: Morgan Kaufmann, January 2003.

[22] J. Ullman, *Computational Aspects of VLSI.* New York, NY, USA: Computer Science Press, 1984.

[23] e. NR Adiga, "An overview of the bluegene/l supercomputer," in *in Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, November 2002, pp. 1–22.

[24] K. Efe and A. Fernandez, "Products of networks with logarithmic diameter and fixed degree," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 963–975, September 1995.

[25] A. El-Amawy and S. Latifi, "Properties and performance of folded hypercubes," *IEEE Transaction on Parallel Distributed System*, vol. 2, no. 1, pp. 31–42, August 1991.

[26] K. Efe, "The crossed cube architecture for parallel computation," *IEEE Transaction on Parallel Distributed System*, vol. 3, no. 5, pp. 513–524, September 1992.

[27] Y. Li, S. Peng, and W. Chu, "Efficient collective communications in dual-cube," *The Journal of Supercomputing*, vol. 28, no. 1, pp. 71–90, April 2004.

[28] S. Ziavras, "Rh: A versatile family of reduced hypercube interconnection networks," *IEEE Transaction on Parallel Distributed System*, vol. 5, no. 11, pp. 1210–1220, November 1994.

[29] K. Ghose and K. R. Desai, "Hierarchical cubic networks," *IEEE Transaction on Parallel Distributed System*, vol. 6, no. 4, pp. 427– 435, April 1995.

[30] F. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Communication of ACM*, vol. 24, no. 5, pp. 300–309, May 1981.

[31] Y. Li, S. Peng, and W. Chu, "Metacube - a versatile family of interconnection networks for extremely large-scale supercomputers," *The Journal of Supercomputing*, vol. 53, no. 2, pp. 329 – 351, August 2010.

[32] http://www.eetimes.com/design/signal-processing-dsp/4007691/Using-micro-benchmarks-to-evaluate--compare-Networks-on-chip-MPSoC-designs/.

[33] P. Yew, N. Tzeng, and D. Lawrie, "Distributing hot-spot addressing in large-scale multiprocessors," *IEEE Transactions on Computers*, vol. C-36, no. 4, pp. 388–395, April 1987.

[34] G. Donald and C. Harris, *Fundamentals of Queueing Theory.* Wiley, February 1998.

[35] G. Varatkar and M. Radu, "On-chip communication analysis for multimedia applications," in *Proceedings of 2002 IEEE International Conference on Multimedia and Expo (ICME '02.)*, vol. 2, November 2002, pp. 185–188.

[36] H. Jeong, J. Lee, D. McNickle, and K. Pawlikowski, "Suggestions of efficient self-similar generators," *Simulation Modelling Practice and Theory*, vol. 15, no. 3, pp. 328–353, March 2007.

[37] http://en.wikipedia.org/wiki/Pareto_distribution.

[38] J. Hu and R. Marculescu, "Dyad: Smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference (DAC '04)*, San Diego,USA, February 2004, pp. 260–263.

[39] D. Tutsch, *Performance analysis of network architectures.* Springer, September 2006.

[40] A. Holt, *Network Performance Analysis: Using the J Programming Language.* Springer, October 2007.

[41] W. Ligon and R. Umakishore, "Toward a more realistic performance evaluation of interconnection networks," *IEEE Transaction on Parallel Distributed System*, vol. 8, no. 7, pp. 681–694, July 1997.

[42] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transaction on Computers*, vol. 54, no. 8, pp. 1025–1040, August 2005.

[43] http://en.wikipedia.org/wiki/Packet_loss.

[44] "Ns manual."

[45] http://en.wikipedia.org/wiki/Scalability.

Appendix A

Confidence Analysis

This Appendix includes the confidence analysis for the torus and the hypercube networks and some of the Metacube Networks of different sizes under different traffic scenarios.

Table A.1: 32-node Torus, 10% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.001809507	25.52359735	0.002723507
Standard Deviation	1.71143E-06	0.106305272	0.000140574
CI min	0.001804149	25.19083445	0.002283475
CI max	0.001814864	25.85636025	0.003163538

Table A.2: 128-node Torus, 10% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00363462	99.46274358	0.02736607
Standard Deviation	2.53772E-06	0.287495706	0.000325006
CI min	0.003626746	98.57075457	0.026357701
CI max	0.003642494	100.3547326	0.028374439

Table A.3: 32-node Torus, 30% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00184023	74.58712771	0.02851754
Standard Deviation	9.41254E-07	0.126643704	0.000277833
CI min	0.00183731	74.19420083	0.027655531
CI max	0.00184315	74.98005459	0.029379549

Table A.4: 128-node Torus, 30% Injection Rate (Bit-complement, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.0085576	235.2524179	0.23140289
Standard Deviation	2.09805E-05	0.170539357	0.000683399
CI min	0.008492506	234.7232996	0.229282564
CI max	0.008622694	235.7815362	0.233523216

Table A.5: 128-node Torus, 10% Injection Rate, 0.8 Burst Rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00364258	96.82892801	0.05253152
Standard Deviation	3.5567E-06	0.33743567	0.000589361
CI min	0.003631545	95.78199444	0.050702959
CI max	0.003653615	97.87586159	0.054360081

Table A.6: 128-node Hypercube, 10% Injection Rate, 0.8 Burst Rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00579779	102.3864317	0
Standard Deviation	4.0936E-07	0.346965865	0
CI min	0.00579652	101.3099295	0
CI max	0.00579906	103.4629338	0

Table A.7: 512-node Torus, 10% Injection Rate, 0.8 Burst Rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00736454	348.7051838	0.14904076
Standard Deviation	1.05193E-05	1.19670968	0.000862812
CI min	0.007331903	344.9922522	0.146363784
CI max	0.007397177	352.4181153	0.151717736

Table A.8: 512-node Metacube, 10% Injection Rate, 0.8 Burst Rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00740513	401.9372711	0.01814103
Standard Deviation	1.06983E-06	1.434992331	0.00030523
CI min	0.007401811	397.4850398	0.017194019
CI max	0.007408449	406.3895024	0.019088041

Table A.9: 512-node Hypercube, 10% Injection Rate, 0.8 Burst Rate(Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.01096608	409.3533817	0
Standard Deviation	1.08879E-06	1.312188954	0
CI min	0.010962702	405.2821622	0
CI max	0.010969458	413.4246012	0

Table A.10: 1024-node Torus, 10% Injection Rate, 0.4 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00969956	691.8875932	0.15461908
Standard Deviation	1.05345E-05	1.63444006	0.000874199
CI min	0.009666875	686.8165521	0.151906777
CI max	0.009732245	696.9586344	0.157331383

Table A.11: 1024-node Torus, 10% Injection Rate, 0.8 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.00982393	668.8343192	0.18257007
Standard Deviation	1.57051E-05	2.268172428	0.001118654
CI min	0.009775203	661.7970493	0.179099314
CI max	0.009872657	675.8715891	0.186040826

Table A.12: 1024-node Metacube, 10% Injection Rate, 0.4 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.01667322	754:7625265	0.07914762
Standard Deviation	3.71614E-06	2.007667972	0.000640506
CI min	0.01666169	748.5335022	0.077160375
CI max	0.01668475	760.9915509	0.081134865

Table A.13: 1024-node Hypercube, 10% Injection Rate, 0.4 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.01422525	817.5473272	0
Standard Deviation	2.46747E-06	2.457354639	0
CI min	0.014217594	809.9230974	0
CI max	0.014232906	825.171557	0

Table A.14: 1024-node Hypercube, 10% Injection Rate, 0.8 burst rate (Bit-complement, MMPP)

	Average Delay	Throughput	Loss Rate
Mean	0.01422518	817.4629313	0
Standard Deviation	3.71831E-06	2.885646242	0
CI min	0.014213643	808.5098768	0
CI max	0.014236717	826.4159858	0

Table A.15: 32-node Torus, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00187293	25.5709499	0.00094446
Standard Deviation	1.59629E-05	0.107184789	0.00176608
CI min	0.001823403	25.23839657	0.004535011
CI max	0.001922457	25.90350323	0.006423931

Table A.16: 32-node Hypercube, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00155248	25.59010607	0.00003933
Standard Deviation	1.2106E-05	0.103273298	1.73147E-05
CI min	0.00151492	25.2696886	1.43909E-05
CI max	0.00159004	25.91052354	9.30509E-05

Table A.17: 128-node Torus, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00365307	99.71775291	0.02450292
Standard Deviation	1.89855E-05	0.306520123	0.000483947
CI min	0.003594165	98.76673843	0.023001418
CI max	0.003711975	100.6687674	0.026004422

Table A.18: 128-node Torus, 30% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00591188	259.6866146	0.1530384
Standard Deviation	0.000114396	1.026325417	0.0020711
CI min	0.005556954	256.5023201	0.146612572
CI max	0.006266806	262.870909	0.159464228

Table A.19: 128-node Torus, 30% Injection Rate with one buffer (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00603615	279.7653952	0.088546
Standard Deviation	0.000104218	0.944432475	0.001877295
CI min	0.005712801	276.8351831	0.082721474
CI max	0.006359499	282.6956072	0.094370526

Table A.20: 128-node Hypercube, 10% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00292834	101.935903	0.00387797
Standard Deviation	1.53657E-05	0.285659564	0.000158519
CI min	0.002880666	101.0496109	0.003386148
CI max	0.002976014	102.8221952	0.004369792

Table A.21: 128-node Hypercube, 30% Injection Rate (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00295915	298.4663255	0.02749271
Standard Deviation	0.520393397	1.5502E-05	0.000717615
CI min	0.002911053	296:8517442	0.025266225
CI max	0.003007247	300.0809068	0.029719195

Table A.22: 128-node Hypercube, 30% Injection Rate, Buffer Size=1 (Uniform Random, Poisson)

	Average Delay	Throughput	Loss Rate
Mean	0.00297969	304.9215814	0.00684976
Standard Deviation	1.64849E-05	0.425405196	0.000320167
CI min	0.002928544	303.6017121	0.005856404
CI max	0.003030836	306.2414507	0.007843116

Table A.23: 64-node Torus, 10% Injection Rate (Hotspot, Pareto)

	Average Delay	Throughput	Loss Rate
Mean	0.00254412	52.37116269	0.05261154
Standard Deviation	2.17828E-05	0.339728423	0.001777713
CI min	0.002476536	51.31711558	0.047095979
CI max	0.002611704	53.42520981	0.058127101

Table A.24: 64-node Hypercube, 10% Injection Rate (Hotspot, Pareto)

	Average Delay	Throughput	Loss Rate
Mean	0.00206917	53.73833311	0.02897272
Standard Deviation	1.36071E-05	0.315379085	0.001773063
CI min	0.002026952	52.75983266	0.023471585
CI max	0.002111388	54.71683355	0.034473855