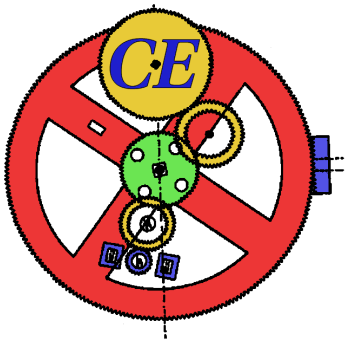


2008

Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

MSc THESIS

Design of a High-Performance Buffered Crossbar Switch Fabric Using Network on Chip

Iria Varela Senín

CE-MS-2008-19

Abstract

High-performance routers constitute the basic building blocks of the Internet. The wide majority of today's high-performance routers are designed using a crossbar fabric switch as interconnect topology. The buffered crossbar (CICQ) switching architecture is known to be the best crossbar-based architecture for routers design. However, CICQs require expensive on-chip buffers whose cost grows quadratically with the router port count. Additionally, they use long wires to connect router inputs to outputs, resulting in non-negligible delays. In this thesis, we propose a novel design for the CICQ switch architecture. We design the whole buffered crossbar fabric as a Network on Chip (NoC). We propose two architectural variants. The first is named the Unidirectional NoC (UDN), where the crossbar core is built using a NoC with input and output ports placed on two opposite sides of the fabric chip. Because of the chip pins layout, we improved the UDN architecture using a Multidirectional NoC (MDN) architecture, by placing the inputs and output around the perimeter (four sides) of the NoC-based crossbar fabric. Both proposed architectures have been analyzed with appropriate routing algorithms for both unicast and multicast traffic conditions. Our results show that the proposed NoC based crossbar switching design outperforms the conventional CICQ architecture. Our designs offers several advantages when compared to traditional CICQ design: 1)Speedup, because short wires allow reliable high-speed signalling, and simple local arbitration per on-chip router. 2) Load balancing, because paths from different input-output port pairs share the same router buffers. 3) Path diversity allows traffic from an input port to follow different paths to its destination output port. 4) Simpler switch design by allowing simple input memory structure such as first-in first-out (FIFO) input queueing, as opposed to traditional design where virtual output queueing (VOQ) is required.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Design of a High-Performance Buffered Crossbar Switch Fabric Using Network on Chip

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Iria Varela Senín
born in A Coruña, Spain

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Design of a High-Performance Buffered Crossbar Switch Fabric Using Network on Chip

by Iria Varela Senín

Abstract

High-performance routers constitute the basic building blocks of the Internet. The wide majority of today's high-performance routers are designed using a crossbar fabric switch as interconnect topology. The buffered crossbar (CICQ) switching architecture is known to be the best crossbar-based architecture for routers design. However, CICQs require expensive on-chip buffers whose cost grows quadratically with the router port count. Additionally, they use long wires to connect router inputs to outputs, resulting in non-negligible delays. In this thesis, we propose a novel design for the CICQ switch architecture. We design the whole buffered crossbar fabric as a Network on Chip (NoC). We propose two architectural variants. The first is named the Unidirectional NoC (UDN), where the crossbar core is built using a NoC with input and output ports placed on two opposite sides of the fabric chip. Because of the chip pins layout, we improved the UDN architecture using a Multidirectional NoC (MDN) architecture, by placing the inputs and output around the perimeter (four sides) of the NoC-based crossbar fabric. Both proposed architectures have been analyzed with appropriate routing algorithms for both unicast and multicast traffic conditions. Our results show that the proposed NoC based crossbar switching design outperforms the conventional CICQ architecture. Our designs offers several advantages when compared to traditional CICQ design: 1)Speedup, because short wires allow reliable high-speed signalling, and simple local arbitration per on-chip router. 2) Load balancing, because paths from different input-output port pairs share the same router buffers. 3) Path diversity allows traffic from an input port to follow different paths to its destination output port. 4) Simpler switch design by allowing simple input memory structure such as first-in first-out (FIFO) input queueing, as opposed to traditional design where virtual output queueing (VOQ) is required.

Laboratory : Computer Engineering
Codenumber : CE-MS-2008-19

Committee Members :

Advisor:	Prof.dr. K.G.W. Goossens, CE, TU Delft
Advisor:	Dr. L. Mhamdi, CE, TU Delft
Member:	Dr. S.D. Cotofana, CE, TU Delft
Member:	Dr.ir. F.A. Kuipers, NAS, TU Delft

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisor, Kees Goossens for guiding me through my MsC project. In addition to this, I want to thank him for always having a different point of view of my work and for taking the time to give me useful advice all along this project.

I also want to express my gratitute to Lotfi Mhamdi for his daily support, dedication, stimulating suggestions and encouragement during 9 months.

Last, but not least, I would like to give my special thanks to Fernando Kuipers and Sorin Cotofana for being part of my graduation committee.

Iria Varela Senín
Delft, The Netherlands
December 3, 2008

INTRODUCTION 1

This chapter provides an overview of the necessary background to understand the purpose of this thesis. It also summarizes the topics covered along with the following chapters. To start with, the motivation and objectives of this thesis are introduced. Finally, it presents the outline and contents of the different sections.

1.1 Overview

The increasing demand of the Internet capacity is leading the network to a fast development in the bandwidth and velocity of the transmission links. Bandwidth has become in the last years a fundamental requirement in the communication and over all in the quality of Internet offered services. This problem has been solved with the development of optical fibres and Wavelength Division Multiplexing (WDM) transmission technique. Using optical carriers it is now possible to achieve the range of multi-terabits per second inside the network. Figure 1.1 shows a typical Internet network.

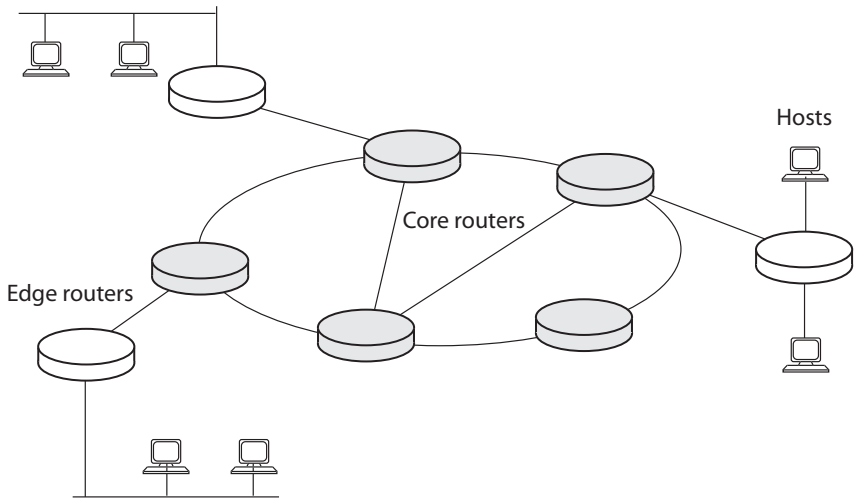


Figure 1.1: Example of an Internet network.

The routers are responsible for sending traffic on the best route through the network. Routers can be divided in two types: edge routers and core routers. Edge routers can route between one or more networks. Hence they have lots of logical interfaces, extensive filtering, policy and traffic shaping and they can be asymmetrical with respect to interfaces. Core routers, on the other hand, designed to operate in the Internet backbone, or core. They have lots of bandwidth, but limited filtering or policy controls and limited traffic conditioning. Additionally, all their interfaces tend to have the same general speed.

An increase in the link speed should be accompanied in performance by the routers and switches that compose the network. If this constraint is not accomplished, these elements become the bottleneck of the system, preventing the Internet services demands from being satisfied.

Routers must perform two fundamental tasks inside a network, routing (path determination) and packet forwarding. Path determination decides which packets will be selected to transfer, while packet forwarding is the actual deliver process. The routing system requires four essential elements to implement in the routing and packet-forwarding processes: routing software, packet processing, a switch fabric and line cards (Figure 1.2). All of these components should be equally powerful, otherwise, performance would be restricted to the weakest element.

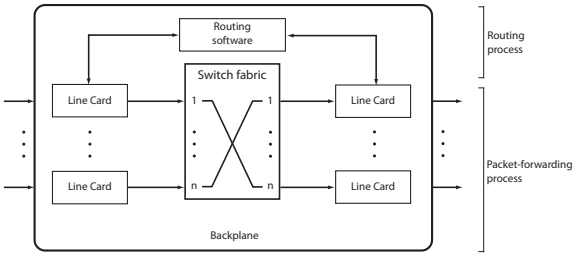


Figure 1.2: Elements of a routing system

In this thesis, we will focus on the switch fabric. It provides the infrastructure for moving packets between router line cards and transports the data from inputs to outputs. The switch fabric is one challenge designers face when creating a system that routes data from one of many inputs to any one of many outputs. The switch fabric is used in many types of applications, from high-speed telecommunications to networking to storage area networks. Typical challenges designers face when implementing switch fabrics in their system include latency, bandwidth, queueing, backplane interfacing, scheduling algorithms, and traffic management interaction. This topic has been covered by designers the last decades, and several solutions have been proposed, like crossbar switches [1][2], shared buffer or shared memory.

Among them, the crossbar-based fabric switch is the dominant architecture for today's high-performance packet switches (IP routers, ATM switches, Ethernet switches) for at least three reasons. First, they are more scalable than their direct competitors, shared-bus and shared memory. This is due to the limitation in bus transfer bandwidth and/or the limitation in the memory access bandwidth. Second, they provide simple point-to-point connections, which means they can operate at very high-speed (up to 10Gb/s). Third, they can support multiple I/O transactions simultaneously. This can increase the aggregate bandwidth of the system, which can be in the hundreds of Gbps. Crossbar-based packet switches come in two flavours depending on their fabric core: unbuffered and internally buffered crossbar fabric. These two categories represent two extremes in a wide range or architectures depending on the number and layout of buffers inside the crossbar fabric.

Recently, Partially Buffered Crossbars have been proposed [3] as a good alternative in packet switching design. The similarity between these variants lies in the quadratic growth of their cost with the number of switch ports. Additionally, these architectures require a sophisticated input queueing structure, known as the virtual output queueing (VOQ) to achieve acceptable performance [4].

Irrespective of whether the crossbar core is unbuffered, partially buffered or fully buffered, a scheduling algorithm is required to configure the crossbar switch matrix, i.e. deciding which

input port sends to which output port by closing their corresponding (input/output) crosspoint. A packet switch contains input line cards with big buffers as well as output buffers for packet reassembly. Due to the big size of these input/output buffers, they usually consist of DRAM memories and hence have high access times.

When the fabric core is unbuffered, the limitation in input/output memories access times implies that the scheduler has to select at most one packet from each input port and send at most one packet to each output. This process is known as the matching (or scheduling) and is often complex to implement in hardware. The unbuffered crossbar fabric is cheaper than its buffered counterpart since it contains no internal buffers. However, it requires a centralized and complex scheduler to configure the crossbar matrix for cell transfer from the inputs to the output ports of the switch [4].

1.2 Motivation and Outline

Using a partially or fully buffered crossbar (CICQ) core can relax the scheduling complexity. It overcomes the scheduling complexity of traditional crossbars by means of parallel and distributed schedulers, one per port of the switch [5] and by allowing multiple packets destined to the same output port to be temporarily stored in the fabric internal buffers. Each internal buffer is usually dedicated to packets belonging to the same input/output pairs. While this results in simplicity in design, it has some drawbacks: first, the internal buffers are over-designed/dimensioned with respect to the dynamic of the switch. Second, it often results in unbalanced internal buffers utilization. Additionally, whether a crossbar switch is buffered or not, the fabric requires long point-to-point wires to interconnect the switch inputs to its outputs. This results in long delays and consumes high power to drive these wires.

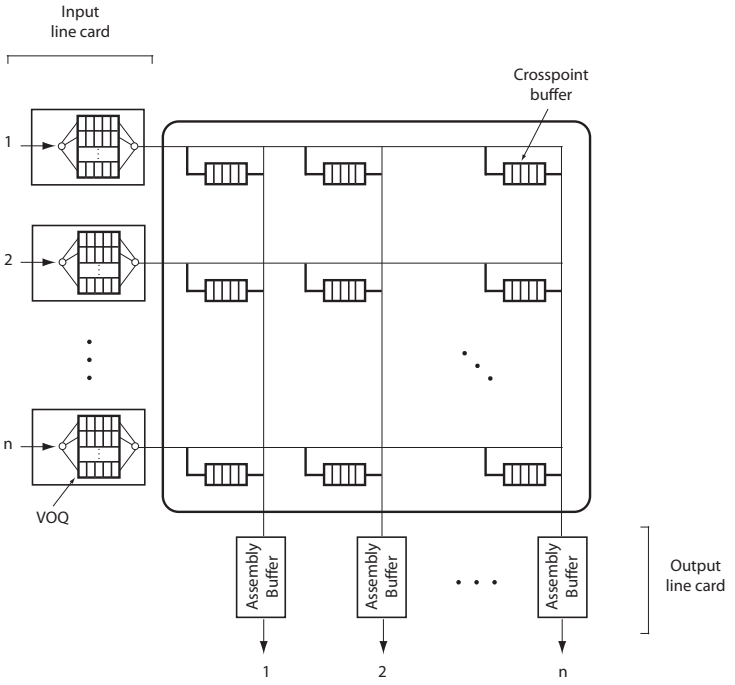


Figure 1.3: CICQ crossbar

Buffered crossbar (CICQ) switches promise to offer scalable Internet router capacity. However, similar to traditional crossbars, they require expensive on-chip buffers, whose cost grows quadratically with the port count. Additionally, point-to-point switching mandates the use of long wires to connect inputs to outputs, resulting in non-negligible delays (see Figure 1.3).

In this thesis, we propose a novel design for the CICQ switch architecture. Instead of using a dedicated internal buffer per input-output pair of ports, we design the whole buffered crossbar fabric as a Network on Chip (NoC) . Our design offers several advantages when compared to traditional crossbar based fabric switches. First, using on-chip routers instead of dedicated internal buffers allows a better load balancing of the traffic passing through the switch. This is achieved by allowing the on-chip routers to switch traffic from any input to any output, resulting in sharing and better use of internal memory. This is in contrast to traditional CICQs that use dedicated internal buffers. Second, the adoption of small routers and shared resources provides path diversity, by allowing traffic from any input to follow more than one path in its way to its destination output port. This results in further load balancing especially in the presence of non-uniform traffic patterns and gives better fault tolerance in presence of interconnect failures. Traditional crossbar have no path diversity, they use expensive redundant planes instead [6].

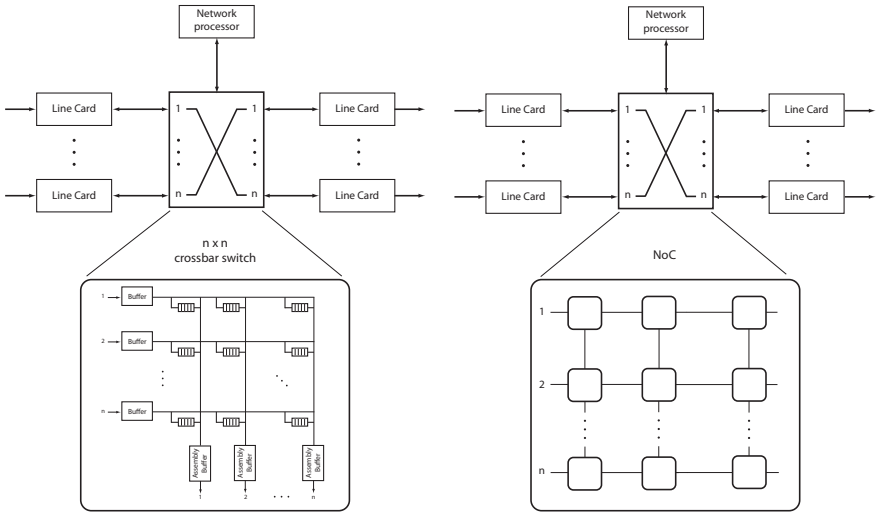
Designing the fabric as a NoC allows scalability in port count and speed per port. This is achieved by using short wires enabling reliable high-speed signaling as opposed to long wires. Using uniform short wires affords significant advantages in cost and performance. Additionally, a NoC based fabric requires simpler switch design by allowing simple input memory structure such as first in first out (FIFO) input queueing in the line cards, as opposed to traditional design that requires sophisticated queueing structures such as the VOQ architecture.

The architecture of current generation of high-speed routers is shown in Figure 1.4(a). The line cards contains the physical layer components necessary to interface the external data link to the switch fabric. The forwarding engine inspects the packet headers, determines to which outgoing line card they should be sent and rewrites the header. The forwarding engine may be a physically separate component or may be integrated with either the line card or the network processor. Currently, the high-end routers deploy forwarding engines directly on line cards. The network processor runs the routing protocols and computes the routing tables that are copied into each of the forwarding engines. Finally, the switch fabric is used to interconnect the components of the gigabit Internet router. It is responsible for interconnecting all ports on all the line cards in the system.

Our claims to use NoC fabrics are the advantages gathered in the following list:

- Better load balancing
- Higher path diversity
- Scalability in port count and speed per port
- Use of shorter wires
- Simpler switch design by using FIFOs in the line cards

The performance of the switch fabric is a major factor in the global behavior of the system. This thesis presents a new design for the switch fabric of the Internet routers. Taking advantage of on-chip characteristics, it proposes to replace the actual crossbar switch fabric by a NoC crossbar. This new architecture is represented in Figure 1.4.



(a) Current Architecture (b) Proposed Architecture

Figure 1.4: High Speed routers.

To introduce this new concept, the remainder of this thesis is structured as follows: Chapter 2 first makes an introduction of on-chip and off-chip networking. Then gives the necessary background for the Internet network and on-chip network design and compares their architectures.

Afterwards, the following two chapters present our fabric switching architectures based on NoC. Chapter 3 introduces the the Unidirectional NoC (UDN) architecture. This first intuitive proposal does not adapt efficiently to the chip layout and it is modified in Chapter 5 to a new architecture. This chapter introduces the Multidirectional NoC (MDN) architecture. These two chapters describe the dynamics of these architectures and explain their properties. A routing algorithm is proposed and analyzed for both designs. The fabric models are further optimize allowing different design trade-offs such as the depth of the NoC as well as input/output ports connections to the fabric. These chapters also present a detailed performance study of the proposed switching architecture along with its variants and compares it to the traditional CICQ switch. Multicast traffic is tested in the new architectures in chapter 8. Three different algorithms are implemented and compared. First of all a simple copy-multicast algorithm is evaluated. Then the unicast routing algorithm is modified to be adapted to the characteristics of the multicast traffic.

Finally we conclude the thesis in chapter9 gathering the conclusions of the performance of the new architectures and pointing out possible future work.

2 Related Work

The proposal of this thesis is to replace the actual crossbar switch fabrics by an NoC in the current Internet routers. To be able to carry out this research, we first present an introduction to the state of the art of the Internet routers and the most common switch fabrics. Then, the emerge of NoC is explained and compared to the actual off-chip networks. How the NoC should be built and placed in the router belong to the last part of this chapter.

2.1 Introduction

During the last decades, the Internet has seen an unprecedented growth. In 1981, there were only 200-odd Internet hosts. By 1992, this number rose over one million. Reaching 2003, the number was of 170 million hosts. This increase of users must be in relation with the increase of the network bandwidth. The widespread use of fiber optic cables is one of the developments that have revolutionized the networking paradigm. Fiber can support bandwidths which are much higher than bandwidths supported by any other media. The maximum bandwidths of fiber is in the range of gigabits. By using WDM techniques, this can be extended up to Terabits. Thus it can be presumed that at least fiber-optic cables will not cause speed bottlenecks in the near future. Instead the routers and switches the are current challenge for researchers when scaling the Internet backbone.

2.2 Internet Routers

Routers architecture has evolved over the last years not to be an obstacle in the development of high-speed networks. Here we summarized the evolution of their design distributed in three different generations.

2.2.1 Architectural Evolution

The first generation of Internet routers was based on software implementations on a single general-purpose central processing unit (CPU) . These routers consist of a general-purpose processor and multiple interface cards interconnected through a shared bus as depicted in Figure 2.1. Each Line Card performs the link layer function, connecting the system to each of the external links. Packets arriving from a link are transferred across the shared bus to the CPU, where a forwarding decision is made. The packet is then transferred across the bus again to its outgoing Line card. The main drawback of this architecture is that the central processor has to process all packets flowing through the router (as well as those destined to it). This represents a serious processing bottleneck. For this reason, the second generation emerged.

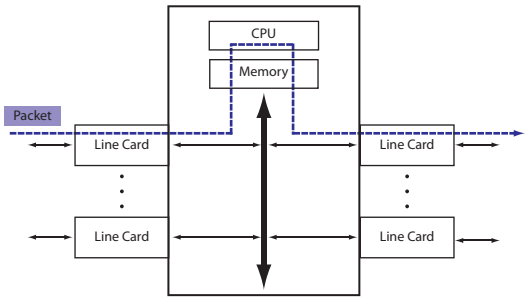


Figure 2.1: First generation of routers

For the second generation of Internet routers, improvement in the shared-bus router architecture was introduced by distributing the packet forwarding operations. Now there are multiple CPUs that process packets in parallel. This parallelism among CPUs can increase system throughput and allow the use of lower-cost CPUs (see Figure 2.2(a)). If the CPUs are placed directly in the Line Cards, local forwarding decisions are made in these dedicate CPUs and the packet is immediately forwarded to its outgoing interface. Packets traverse then the bus only once, and the system throughput is further increased. The central CPU maintains the forwarding tables in the line card CPUs an manages the system. See Figure 2.2(b). Though now the process load is distributed, the shared bus continues to be a bottleneck in the system. The solution lies then in modifying the communication infrastructure.

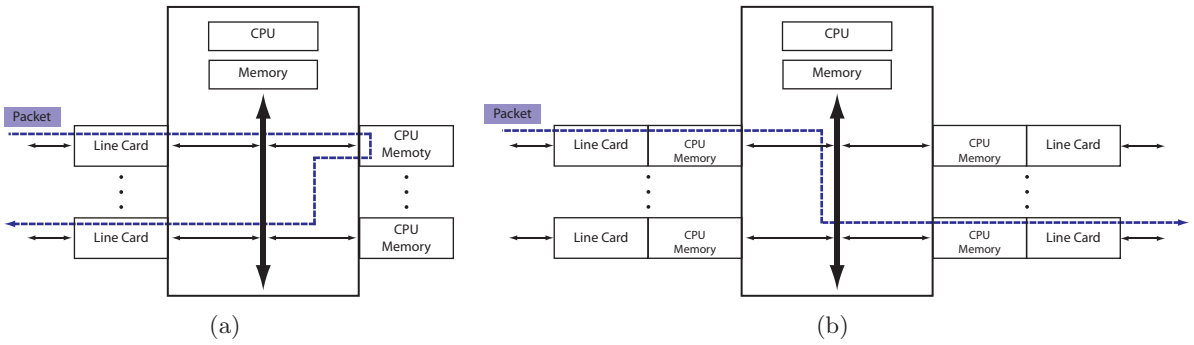


Figure 2.2: Second generation of routers

To alleviate the bottlenecks of the second generation of IP routers, the third generation of routers were designed with the shared bus replaced by a switch fabric. This provides sufficient bandwidth for transmitting packets between interface cards and allows throughput to be increased by several orders of magnitude. Today, the highest performance routers are designed according to this architecture (see Figure 2.3). Over the last years, researchers have focused their effort in developing and improving this crossbar switch fabric.

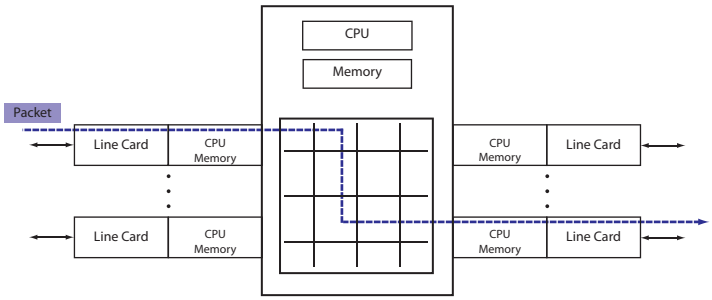


Figure 2.3: Third generation of routers

The following section gives a description of the most significant architectures for switch fabrics. During the last years, different variants of packet switch fabrics have been proposed. This section summarizes the state of the art of these efforts and discusses the major issues from the architectures, its scalability and cost-performance.

2.2.2 Switch fabrics in use today

A packet switch fabric is the key component of a large number of routers. The most common fabric architectures in use today are bus-based [7], shared memory [8] and crossbar [4]. Among all of switch fabric implementations and proposals, the crossbar fabric is proved to be the most attractive choice for performance [4].

A crossbar has the properties of being non-blocking, inherently supports multicast and can provide parallel point-to-point communication making it attractive for real time applications. For these reasons, a lot of research has been carried on the design and optimization of crossbar fabrics. The main challenges in designing a crossbar lies in its scalability beyond a small number of ports or data rates. The cost of a crossbar switch grows as the square of the number of its ports. Additionally, for a switch with a medium to large number of ports, it is difficult to achieve a high data rate due to the dominant delay incurred by the *long* point-to-point wires connecting inputs to outputs.

Several solutions have been proposed and implemented to scale the performance of crossbar switches. To cope with high data rates, one of the most used solutions is bit-slicing [9]. Using bit-slicing, the crossbar core consists of multiple lanes and each of them switches part of the data in parallel. Input signals are fed to a serial-to-parallel converter, sent through a multiple bit-slice core and serialized back at the output. However, for high data rates, using parallel slices the core remains sometimes the bottleneck. Additionally, bit-slicing adds considerably to the cost and die size of the whole crossbar, rendering this approach infeasible [9].

When the crossbar is buffered (e.g. CICQ switch), other constraints are imposed in addition to the above. Aside from using long point-to-point wires, as in a traditional crossbar, a CICQ switch contains a small amount of buffering per crosspoint of the crossbar fabric. While the adoption of internal buffering overcomes the centralized scheduling complexity of unbuffered crossbars, it comes at the cost of an expensive and complex fabric chip. The crossbar fabric of an $N \times N$ CICQ switch has to maintain N^2 dedicated internal buffers, one per input-output pair of ports. These buffers are dedicated and their number grows quadratically with the switch port count, making CICQ less appealing. Recent proposals attempted to overcome these shortcomings. A CICQ switching architecture with flexible access to crosspoint buffers has been recently proposed [10]. This approach tried to achieve better internal buffer use by sharing

access to the internal buffers rather than being dedicated. However, the implementation cost of this approach has proved prohibitive. Other solutions have been recently proposed to minimize the internal buffer requirement by using partial internal buffering instead of dedicated buffers per input-output pair of ports [11][3]. While these approaches present a new trend in designing buffered crossbar switches by solving the high requirement of internal buffering, they suffer the inherent delay of the long crossbar wires.

This collection of drawbacks lead us to propose to replace this current fabric crossbar switch by a NoC. On-chip networks use uniform short wires that reduce the latency and simplifies the synchronization of the switch. A NoC based fabric also requires a simple design as no VOQ are needed. Simple FIFOs can be implemented without detriment in performance reducing the cost and complexity of the system.

Next section presents the NoCs and their motivation. It introduces their basic concepts and explains the advantages of their design and what makes them suitable for the internet routers.

2.3 NoC Basics

The need of NoCs emerged due to the increase of processing power and data intensive applications. The growth in performance of embedded multi-processor architectures has given a boost of communications in a single-chip system, System-on-Chip (SoC). This scaling of microchip technologies will soon require highly scalable interconnection architectures. As the number of processor cores and IP blocks integrated on a single chip is steadily growing, a systematic approach to design the communication infrastructure becomes necessary. NoCs have been proposed to support the trend for SoCs integration.

Although NoCs borrow concepts and techniques from the computer networking domain, this features should be adapted to the on-chip architectures. Computer networks and on-chip networks share many requirements, but there are also many differences that made impractical to blindly reuse classical computer networks designs [12].

Currently, the most common type of on-chip communication structure is the bus. The bus is designed to handle a small number of functional units and becomes clogged when too many masters try to take control. This is the result of sharing a single resource, the communication channel. Buses cannot adapt to changes in the system architecture as they do not decouple the activities as transaction, transport and physical layer behaviors. On the other side, NoCs strive for a modular build that can solve on-chip traffic transport and management challenges. Instead of using buses and dedicated point-to-point links, a grid of routing nodes spread out across the chip and connected by communications links is created, the NoC. A table with a comparison in performance of buses and NoC is presented in table 2.1 by [13].

Criteria	Bus	NoC
Max Frequency	250 MHz	>750 MHz
Peak Throughput	9 GB/s (more if wider bus)	100 GB/s
Cluster min latency	6 Cycles @250 MHz	6 Cycles @250 MHz
Inter-cluster min latency	14-18 Cycles @250 MHz	12 Cycles @250MHz
System Throughput	5 GB/s (more if wider bus)	100 GB/s
Average arbitration latency	42 Cycles @250 MHz	2 Cycles @250 MHz

Table 2.1: Comparison Bus with NoC

Wiring is also cheaper than in the large scale networks. In Internet networks wiring cost

is determined by infrastructure efforts for fiber lines. The wiring of the on-chip network is structured and IP cores are tiled on the chip in a regular manner connected via a structured on-chip network such that the routing of wires is not an issue any more. As a result, interconnect arbitration changes from centralized to distributed.

In bus arbitration, master modules request access to the interconnect, and the arbiter grants the access for the whole interconnect. As there is only one arbiter, arbitration is centralized. The arbiter can see the state of the interconnect as well as all the requests. For this reason it is also a global arbitration.

Arbitration is also performed in NoC since it is a shared interconnect. In this case arbitration is distributed, because it is performed in every router and only uses local information. The different type of arbitration in on-chip and off-chip networks also affect to the different delays of the systems. For a bus, the latency is proportional to the number of masters in the system. For NoC, arbitration is performed router by router so it increases with the number of hops between sender and receiver. However, as its shown in table 2.1, they can run at higher frequencies than buses, minimizing the latency.

Since on-chip communication links are relatively short, pipeling or transmission-line effects are absent. Hence, buffers can be smaller due to the tight synchronization between routers and flow control can be used to prevent buffer overflow [14].

In the Internet network, retransmissions are needed since loss of packets is possible. On-chip wires provide a reliable communication medium and no data is dropped because of errors. Then no reordering modules are necessary if the routing algorithm manages to send the packets in the same order they arrive.

This kind of on-chip networks provide a powerful solution to establish and manage the communication between on-chip processing and storage components. The basic idea is borrowed from traditional large-scale multi-processors and the wide-area networks domain. In the same way, packetized communication takes place in on-chip router-based networks.

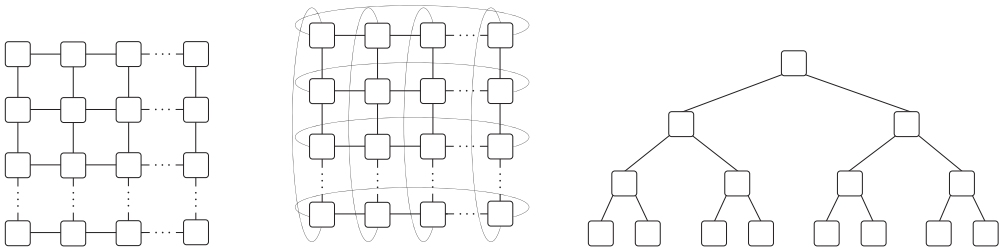
A NoC contains two components: routers and network interfaces (NI). By using NI, the actual structure of the NoC can be hidden from the functional units. They are the connection of the IP view on communication and the router view on communication. They connect all the IP cores to the network, mapping the bus type transitions coming from the IPs into packets that can be propagated inside the Network on chip and, on the opposite side, building the bus transactions that correspond to packets that need to exit the NoC. Extensive research has been done in this area to decouple computation from communication through those interfaces [15].

Routers route the data according to a chosen protocol. Their job is to deliver messages from their source to their designated destination. Their different architectures are discussed in section 2.4. As there are multiple routers between two functional units, there exist different paths among them that can be used in case of failure. This characteristic gives robustness through self-repairing systems and the designer can determine how robust or failsafe the system should be.

The way in which routers are connected is a key factor in NoCs performance. An on-chip network is defined mainly by its topology and the protocol implemented by it. Depending on the connectivity among the links and the layout, several topologies are possible. A simple way of identifying the topologies was described in [16].

Several regular topologies are grouped in the term of *k-ary n-cube*. In this notation, n is the number of dimensions and k is the degree of each dimension. The 4-ary 2-cube mesh is the most simple NoC structure. It consists of a grid of horizontal and vertical lines with the routers placed in the intersections (Figure 2.4(a)). If the mesh connections are also wrapped around, the topology becomes a Torus, refer to Figure 2.4(b). In this network, the routing problems at the edge of the mesh are avoided, but the nodes must be interleaved to make sure all inter-node

connections are of the same length. The k-ary tree, has a central root that is connected to other nodes in an order of hierarchy. The number of connections is the factor k. Regular topologies are shown in Figure 2.4.



(a) Mesh (b) Torus (c) Binary Tree

Figure 2.4: Regular Topologies

Mixing different forms in a hierarchical or asymmetric fashion leads to irregular forms. They are usually based on the concept of clustering. This possible topology is represented in Figure 2.5.

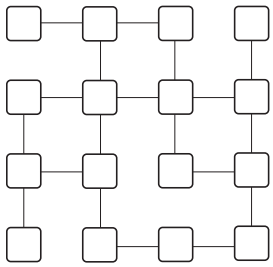


Figure 2.5: Irregular Topology

NoC topology and characteristics are application-specific. NoCs are mainly developed for consumer products so the constraints in area and cost are higher than in computer networks. On-chip routers and NI have to be designed with the minimum number of buffers, a with a fast and simple arbitration. These constraints are partially reduced thanks to the short links used by on-chip networks. Short links allow tight synchronization between routers and buffers can be smaller and their overflow controlled by using flow control techniques. Many researchers think that quality of service is more important in SoCs for consumer electronics than for Internet services due to the real time constraints of many of the applications in which they are used [17]. Contrary to traditional networks, NoC topology is static and cannot be modified once it is built. Conditions on-chip are more stable than off-chip and NoC routers are considered just faulty or correct.

Researches have been working on NoC for the last few years. Several models showed up motivated by the demand for well structured designs in large scale SoCs. Here, just some of those solutions are shown to have a global guideline and vision about the direction this field is moving to.

In the Æthereal model, proposed by Philips [18], the guaranteed services pervade as a requirement for hardware design and also as a foundation for software programming. They argue that guaranteed services are essential to provide predictable interconnects that enable compositional system design and integration. And that combining these services with best-effort (BE) solutions, the typical inefficiently use of the resources produced by guarantees is overcome. Therefore, routers provide both guaranteed throughput (GT) and BE services. All the routers in the network are synchronized with a single, centralized synchronous clock. Input queueing is implemented for buffering and custom-made hardware FIFOs to keep the area costs down [19].

The Nostrum Network-on-Chip, developed by the Royal Institute of Technology (KTH) in Sweden [20], is based on a 2D structure that also provides guaranteed services. The guaranteed bandwidth is accessed via Virtual Circuits (VC). It is a simple architecture and can achieve both GB and BE performance.

Other architectures are Xpipes [21], Intel 80 core research processor [22], Wolkotte [23] or Mango [24] .

The basic needs to design the architecture of a NoC are summarized in the following section. As on-chip and off-chip networks have different characteristics, the trade-offs in their design are different. The architectural options classically used in Internet routers are summarized and compared to those that are used in on-chip networks.

2.4 Router Design

The new crossbar fabrics presented in this thesis are designed as a NoC. Inside a NoC, routers constitute the most important part. As explained in section 1.1, router performance is a key factor for the global response of the whole system. Therefore their design should be carefully studied to understand their characteristics and how they work. This section explains the existing architectural components of Internet networks and which of them are suitable for NoC, like different switching architectures, buffering architectures, or flow control. At this point, there is an analysis of the routing algorithms that can be implemented. This study will help decide which options best fit the constraints of the new design in the following chapters.

2.4.1 Switching Architecture

When a communication is established between two routers, two possibilities arise in networks in the way the messages are sent. Messages can be routed using Circuit switching or Packet switching:

Circuit switching

In Circuit switching, the communication lines are dedicated to passing messages from the source to the destination. Circuit switching uses a fixed amount of bandwidth between the source and the destination for all the communication, and the channels remain inaccessible for other connections though it has a “dead time”. This is ideal when data must be transmitted quickly, must arrive in sequenced order and at a constant arrival rate. Thus, Circuit switched networks are appropriate when transmitting real time data, such as audio and video.

Different kind of techniques for circuit switching exist: Frequency division multiplexing (FDM) and Time division multiplexing (TDM).

In FDM the bandwidth is divided in individual channels. Each user is assigned a different frequency. The signals are sent at the same time through the same communication channel but

they are divided in frequency.

TDM appeared after FDM. In TDM, bandwidth is assigned to each channel during a fraction of the total time (slot).

Packet switching

Packet switching breaks up messages into small packets to retransmit them. Each packet is individually transmitted across the network, and may even follow different routes to the destination. A header information about source, destination or packet numbering is then needed to be able to route the packet in each router. This allows statistical multiplexing for the packets in the network and the resources are available as soon each packet is sent. Packet switching is more efficient and robust for data that is bursty in nature, and can withstand delays in transmission, such as e-mail messages and Web pages.

Some researches [25] [26], have shown that circuit switching is not appropriate for computer communications due to their type of traffic. So packet switching is the most used technique for off-chip networking.

NoCs use packet switched architectures in order to fully utilize the network bandwidth. Some proposals also include the circuit-switched network to support guaranteed communication service between IPs on the chip [27]. Other proposals, like Æthereal [28] use TDM of circuit switching to provide GT. They use time division multiplexing connections over pipelined circuits, that additionally offers bandwidth allocation.

	Packet Switching	Circuit Switching
Off-chip networks	X.25 Frame Relay ATM	ISDN
On-chip networks	Æthereal Nostrum (BE services)	Æthereal Nostrum (GT services)

Table 2.2: Example of the use of switching strategies

Table 2.2 gives some examples of networks with different switching strategies. Both Æthereal and Notrum can work with packet switching or circuit switching architectures. They argue Quality-of-Service is a need for NoCs and combine both architectures to exploit the network capacity that is left over.

A generic packet switched network contains inputs, an interconnection crossbar and outputs. An arbitration algorithm is used to resolve output port contention among the input ports. There are many issues concerning to the design of a packet switched network that have a huge impact in the optimal use of the resources of the network. For example, the size of the buffers in the routers are limited in size. If the queue of packets increases without any bound, the packets must be discarded or queued. In the following section, we describe the basic architectural components of a packet switched network. Its design is a key factor in the network performance.

2.4.2 Buffering Architecture and Arbitration

Most of the issues in an Internet router design pertain to two important aspects: buffering and internal routing. The first aspect relates to the placement of buffers in the router, while the

second relates to the design of the switching fabric. These two are, however, not independent concepts, because fixing one of them imposes constraints on the other. In fact, there are two ways of designing a router. Either one can choose a particular buffering technique and then decide from one of the possible interconnection structures. Alternatively, one can choose a particular interconnection structure and then decide upon the buffering technique that best suits. Section 2.4.2.1 and 2.4.2.2 explain both the buffering architecture and the internal routing or arbitration inside the router.

2.4.2.1 Buffering Architecture

When two or more packets have to go through the same link at the same time, network contention occurs. As only one packet can traverse the link, the rest of the packets must be queued or dropped. This contention can lead to network congestion resulting in a deterioration of the system performance.

Preferably, data should not be discarded when congestion happens. This would imply retransmissions after a possibly lengthy time-out period, further contributing to network congestion and the delay seen by the user. One of the most common solutions involves buffer memory in which a switch can temporarily queue data directed at overloaded outputs. Several buffering architectures are possible, like output queueing (OQ), shared memory, input queueing (IQ) and VOQ.

Output queueing

OQ architectures place the buffering at the output ports to where packets are immediately forwarded once they arrive. Though it is the ideal switching architecture due to its optimal performance, it requires a speed up of N for a N port switch and then no arbitration scheme is used. The N time case occurs when N input ports all simultaneously forward a packet to a single output port. Therefore, for a limited memory speed, the size of such routers is restricted to a small number of ports as link rates increase. For this reason OQ architecture is generally considered unfeasible. Another option is to increase the size of the switch to have a $N \times N^2$ switch (see Figure 2.6).

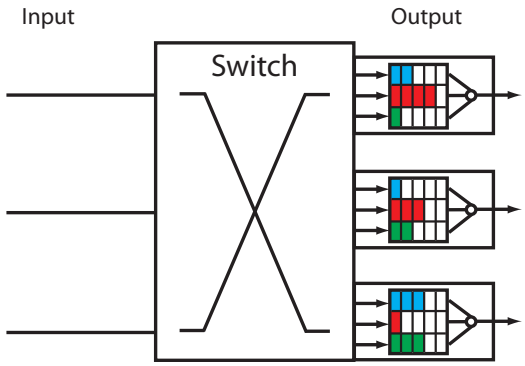


Figure 2.6: Output queueing with a $N \times N^2$ switch

Shared-buffer

In Shared-buffer switches input buffer memories are shared by all the switch output ports and are allotted to one particular input as the occasion demands. Pure shared memory is not able to provide the necessary amount of buffering space to support bursty traffic and suffer from packet losses at asymmetric load patterns due to limited speed-up capabilities.

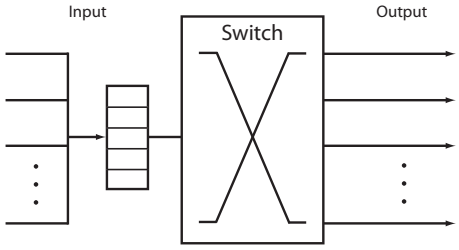


Figure 2.7: Shared queueing

Input queueing

Input Buffering stores each arriving packet in buffers allocated in the input ports, can operate at the speed of the input lines and can be implemented up to arbitrary buffer sizes, making these techniques attractive. However IQ routers need to perform matching between input and outputs to resolve input and output contentions. Two buffering schemes are mainly used: dedicated input buffer [29] and shared-memory input buffer [30] [31]. The dedicated buffering scheme is simple and easy to implement, but the memory utilization is not as high as that for the shared-memory scheme, provided that the total memory size of both schemes is the same. Some inputs can be more heavily loaded while others have empty places in their buffers. The shared-memory scheme can complement this unevenly used memory problem, so that it can achieve the optimal buffer utilization. Its drawback is that it requires extra hardware complexity and speedup.

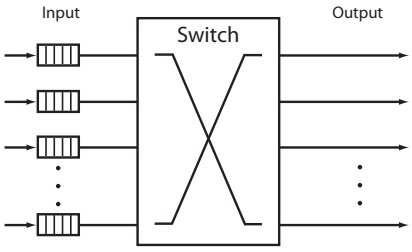


Figure 2.8: Input queueing

The problem of this Input queueing is that its performance is considerably reduced due to the Head of Line (HoL) blocking problem [32].

In FIFO input queueing, when a packet reaches the head of the FIFO, it is considered by the scheduler. The packet contends for its output with packets destined to the same output

but currently at the HoL of other inputs. The scheduler should then decide which packet will be the next. This method entails that packets can be held up by frames ahead of them that are going to a different output. This phenomenon is called HoL blocking. The scheduler only deals with the packet at the head of the FIFO queue, and so the HoL packet blocks packets behind it that need to be delivered to different outputs. HoL blocking can decrease performance to 58.6% of the aggregate bandwidth for fixed of variable length packets [32]. This problem can be solved using VOQ [33].

Virtual Output Queues

At each input, a separate FIFO queue is maintained for each output. HoL is eliminated because each arriving packet is classified and queued in the corresponding VOQ according to its destination port. No packet can be held up by a packet ahead of it that is destined to a different output. When VOQs are used it has been shown possible to increase the throughput of an input-queued switch from 58.6% to 100% for both uniform and nonuniform traffic [34], [35].The input buffer memory does not need to have speed up since the queues are internally implemented within a single memory module. Fig 2.9 shows the architecture for VOQ input buffering.

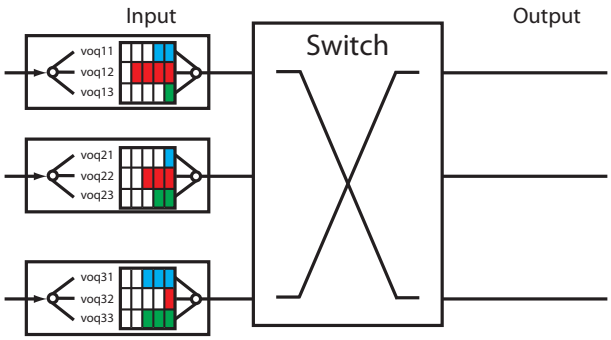


Figure 2.9: Virtual Output queueing

The following table (see table 2.3) gather some examples of the use of this architectures in different switch fabrics in the market, both for Internet routers and NoCs. For the following tables a $\surd$ represents that that technology is used for the type of network where is marked.

	Output queueing	Shared Buffer	Input queueing	VOQ
Off-chip networks		$\surd$	$\surd$	$\surd$
On-chip networks		$\surd$	$\surd$	

Table 2.3: Examples of buffering strategies in the market

An example of NoC that uses shared buffer is the Intel's 80 research core. For Input queueing the most representative is Æthereal.

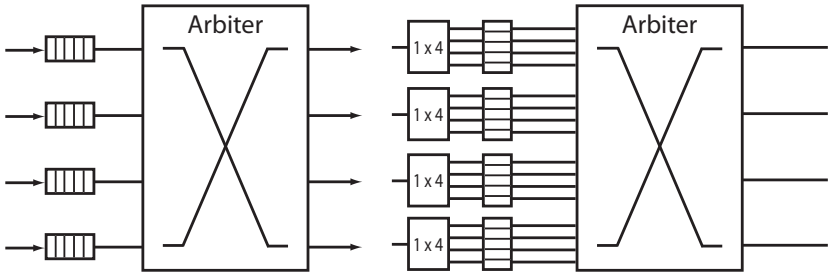
Neither on-chip nor off-chip routers use Output queueing architecture because of its unaffordable cost. Æthereal uses input queueing for both GT and BE services. They argue VOQ

are not necessary in NoCs as they strive for a small router and, therefore, they use small memories and preferably no RAMs . Hence, they design FIFOs with few overhead to use in their architecture. Nostrum implements deflective routing to avoid the use of buffers.

2.4.2.2 Arbitration

Packets of different input ports may have to go to the same output. This election is made by the arbiter, that decides in which order the packets will be served and it is necessary when input buffering, shared buffer or VOQs are implemented. In general, the performance of packet switches are constrained by the efficiency of the arbitration scheme used to select the packets that will traverse the switch at a given time. It must be decided first which packets of each queue will have access to the arbiter. There are several possibilities for this choice both for off-chip or on-chip routers:

- FIFO.
- Multiport RAM.



(a) FIFO (b) Multiport RAM

Figure 2.10: Possibilities to access the Arbiter

In the FIFO model, FIFO queues are implemented and no multiplexors are needed, as only the head packets of each buffer can enter the arbiter. Those packets that are not in the head of the queues can be retained by the heads, leading to more congestion and causing the known HoL blocking. In Multiport RAM each packet of the buffers enter the arbiter and entails bigger routers as there are multiplexors that avoid HoL blocking leading to better performance.

Once the packets are examined by the arbiter, among all the packets that want to go to each exit, only one is chosen. This can be modeled as parallel matchings, where N input and output arbiters perform the parallel selection and communication among them to decide the matching results. This communication adds overhead time to the matching process. This matching process can be divided in three phases: request, grant, and accept [36]. Therefore, the resolution time is the time spent in each of the phases plus the transmission delays for the exchange of the information.

Several algorithms have been proposed by researchers, all of them looking for a high throughput, starvation free, fast and simple to implement method. They can be divided in two major groups, Maximum Size Matching (MSM) and Maximum Weight Matching (MWM) algorithms.

MSM algorithms, find the largest size matching between input and outputs, for example, maximizing the number of connections made in each packet time.

In the MWM algorithms, a weight is assigned to each packet from inputs to outputs. This weight can be the waiting time, the number of packets in the queue, etc. This kind of algorithms maximize $\sum_{i,j} S_{i,j}W_{i,j}$. Where $S_{i,j}$ is a service indicator (a value of 1 indicates that input i is match to output j), and $W_{i,j}$ is the attached weight to the packet.

MSM algorithms

- Random
- Round Robin
- ISLIP
- Priority

MWM algorithms

- LPF
- OCF
- Weighted priority

If the packets are chosen randomly, the election of the port by the arbiter is uniform and no starvation happens in the long term. Random values are complex to obtain hardware wise. Round Robin (RR) is an easier method in hardware. RR and its variants is commonly used in Buffered Crossbar Switches, achieving a 100% throughput [37]. ISLIP has been proved to be simple to implement and to be able to achieve 100% throughput for uniform traffic when the round-robin pointers are carefully controlled [36]. In priority arbitration, the arbiter selects the packets have different priorities. The arbiter selects the one with the highest priority.

Longest Queue First (LQF), takes the number of packets in each queue as weight. Hence, the algorithm picks a packet such that the sum of served queues' lengths is maximized. It can lead to starvation as it does not consider the waiting time of the packets. Oldest Cell First (OCF) chooses the packet that has been more time inside the router. Unlike the LQF algorithm, it does not starve any queue. Other possible solution is to elect the packet that has the longest remaining path. This election can cause starvation in the system.

More complicated solutions can be given, as i.e. giving preference to those buffers that are full or the more congested ones on average. The router must arbitrate traffic when more than one packet arrives concurrently destined for the same output port. They must provide sufficient buffering to handle situations where the packet input rate is greater than the routerss throughput capability. Scheduling algorithms cannot easily exit a packet of each buffer per cycle. In this case, there will be no free space for new packets in the buffers in the next cycle. Instead of dropping packets, some kind of mechanism between routers can be implemented. Flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from over running a slow receiver. This process is explained in the next section.

Table 2.4 summarizes the use of the different strategies in on-chip and off-chip networks.

	Random	RR	ISLIP	LPF	OCF	Priority	Weighted priority
Off-chip networks		√	√	√			
On-chip networks		√				√	√

Table 2.4: Examples of buffering strategies in the market

Æthereal uses RR in the BE architecture to arbitrate the packets.

2.4.3 Link Level Flow Control

In the longer term, no amount of buffering is sufficient: instead, sources should be indicated not to send more data until the congestion is solved. This indication should be done based on the amount of buffer space available or in use in the router. Then, the sources can control how much data they send. This control loop has a delay since the control signal is sent until the router stops receiving packets. When this delay is minimized, elastic buffering is avoided. Also, links between routers should be used at full capacity whenever possible. The flow control mechanism should be robust; loss or delay of control messages, for instance, should not cause increased congestion. This is called link level flow control (LLFC). For this thesis, three different mechanisms were considered.

ON/OFF

In ON/OFF flow control, the upstream node communicates through a pin with the downstream node. It sends a signal each clock cycle; if it has space in its buffer, an ON signal is sent and an OFF signal otherwise. This flow control method forces to use elastic buffers due to its delay. The timing diagram is shown in Fig 2.11(a).

ACK/NACK

In ACK/NACK flow control, the transmitter does not keep any state for the buffers of the downstream node. It is the downstream node who sends back to the transmitter acknowledgments (ack) or negative acknowledgments (nack) depending on the state of the buffer when receiving a packet. If there is enough space in the buffer for the arriving packet, an ack is sent, otherwise a nack is sent. Then the upstream node has to hold the packet until it receives an ack. If it receives a nack, it retransmits the packet. Due to this behavior, it is also known as optimistic flow control. This method avoids the round-trip delay between the time a buffer becomes empty, triggering a credit or an ON signal, and when a packet arrives to occupy the buffer. Its drawback is that the transmitter has to hold for an additional delay time waiting for an acknowledgment, turning this method into inefficient in terms of use of buffer and bandwidth as it sends flits when even no buffer could be available. When flits and not packets are being sent, the downstream node has to wait for all of them to arrive to maintain the order.

Credit based

In credit based flow control, the upstream node decreases a counter each time it sends a packet. That counter starts with the value of the buffer size of the downstream node. When the downstream node releases a packet, it sends a credit back to the upstream node. When the

credit is received, the transmitter node increments the counter again. Credit provides precise control over buffer use , and can stop transmission automatically to avoid buffer overrun. This has been shown to be a fair flow control method, that minimizes the delay and uses links efficiently, guarantying no packet loss due to congestion [38].

ACK/NACK is rarely used due to its inefficient use of buffers and bandwidth. ON/OFF is typically used in systems with large number of buffers and credit based in those with small number of buffers. This types of flow control are used in NoC.

Internet routers use protocols like High-Level Data Link Control (HDLC), where the congested receiver block transmission from the sender using an Receiver Not Ready command. After the congestion is cleared, the receiver issues an Receiver Ready command to reopen transmissions. In the simplest case, the node is congested if it has no free buffers. Thus, packets are indiscriminately accepted until all buffers are full. Another solution consists of introducing a selective definition for congestion. The Ethernet networks use the "pause frame" to stop the sender from transmitting packets during a period of time that is indicated inside the frame.

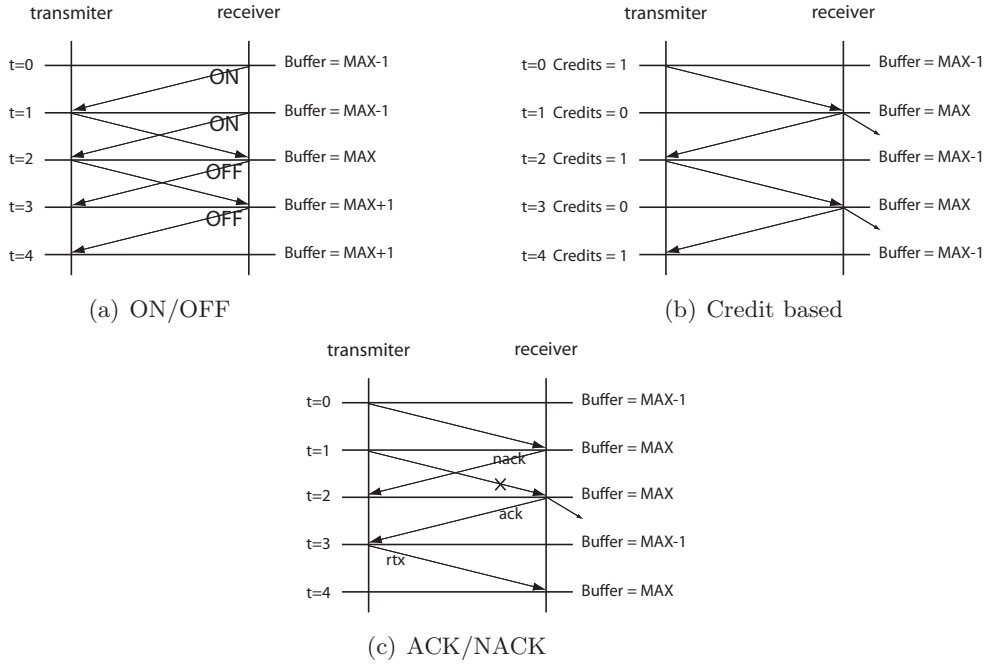


Figure 2.11: Flow control

	ON/OFF	ACK/NACK	Credit based	Pause Frame	HDLC
Off-chip networks				Ethernet	ATM Frame Relay X.25
On-chip networks	Wolkotte	Xpipes	Æthereal		

Table 2.5: Flow control mechanisms in off-chip and on-chip networks

Table 2.5 gathers the use of the different flow control mechanism in the different networks.

2.4.4 Switching modes

There are three basic approaches to routing packets, based on what a switch does with a packet as it begins to arrive: cut-through, store and forward and wormhole.

Store and Forward

This kind of switches wait for the whole packet to arrive before switching it to the output port. While the packet is in the buffer the error correction and filtering table look-ups are performed. If everything is correct, the packet can be forwarded. In Networks on Chip no correction error is needed as the mean time between failure is very large [15]. This method causes a delay in packet forwarding that is dependent upon the length of the packet.

Cut-through

In cut-through, the packet starts to be sent as soon as the header arrives and the resources are acquired. The difference with store and forward is that each hop is started as soon as possible, without waiting for the whole packet to arrive. Latency penalty is then overcome [39].

Wormhole

Wormhole combines packet switching with the data streaming quality of circuit switching to attain a minimal packet latency. Packets are divided in flits that follow the same path. First, the node looks at the header of the packet, determines the next hop and immediately forwards it. The subsequent flits are forwarded once they arrive. The latency within the router is not that of the whole packet.

Protocol	Latency	Buffering
Store and Forward	packet	packet
Cut-through	flit	flit
Wormhole	flit	flit

Table 2.6: Cost for Switching modes

Table 2.6 summarizes the differences in latency and buffering per router of the switching modes.

	Store and Forward	Cut-through	Wormhole
Off-chip networks	√	√	√
On-chip networks		√	√

Table 2.7: Switching modes in on-chip and off-chip networks

Table 2.7 represents what methods are implemented in the different networks. Æthereal employs the wormhole switching mode, meanwhile for example, Nostrum uses Cut-through.

2.4.5 Routing Algorithm

When routing a packet, a path among all the possible paths has to be elected. The routing algorithm has to find the best path for each packet. To achieve this purpose, first it should be defined what is a "best path" and how to measure its cost. The cost of the path can be for example, the number of jumps needed to go from one node to the next one. Though this is not an optimal metric, as it is supposed "1" for all the links, it is a simple technique and can offer good results. Another kind of measure is the delay between neighbor nodes; here the metric is time and its values are not constant but depend on the traffic of the network.

A path is considered a "best path" if it accomplishes the following conditions:

- It has the minimum average delay
- It achieves a high performance in throughput.

The easiest criterion is to choose the shortest path, that is, the route that has to pass through the least number of nodes. A generalization of this criterion is the "minimum cost". Generally speaking, the concept of cost in a link is a measure of the quality of the link based on the defined metric. In practice, several metrics are simultaneously used.

When the network uses virtual circuits, usually the routing algorithm establishes a path that is not changed during the time of life of this VC. In this case, the routing algorithm is elected per session. When it is a datagram network, it does not have to keep the order of sequence of the packets. In this case, if it is not necessary to keep the sequence of packets in order, the routing criterion can be changed per packet. If the order of the packets has to be kept, other solutions, such as routing per flow, are possible.

Classifying the routing algorithms in terms of how they select between the set of possible paths R_{xy} from source node x to destination y, there are three types.

- Deterministic
- Oblivious
- Adaptive

Deterministic routing programs the path of each packet in advance of transmission. This kind of algorithms allow a simple and in-order packet delivery routing. The problem with existing deterministic routing algorithms is that they cannot make the best use of the network.

In oblivious routing, a system of optional paths is chosen in advance for every source-destination pair, and every packet for that pair must travel along one of these optional paths. Thus, the path a packet takes only depends on its source-destination pair (and maybe a random choice to select one of the options).

In adaptive routing, however, the path taken by a packet may also be on other packets or events taking place in the network during its travel. This events can be based on local or on global information. Load distribution turns to be one of the most important parameters for adaptive routing [40].

2.4.5.1 Deadlock

Algorithms for routing messages should have low latency and high network throughput. A main features contributing to those characteristics is freedom from deadlock.

Deadlock occurs when a packet waits for an event that cannot happen. For example, two or more packets are waiting for the other to finish, and thus neither ever does. This circular wait

condition shown in Figure 2.12 causes deadlock because a packet holds resources while waiting and excludes other packets from using those resources. The packets that want to go East are blocked by all the packets that want to go South. In the same condition, the packets willing to go South are blocked by those that want to go West. This last packets cannot move neither because of the packets that try to move to North. To end the circular blocking, the packets that want to go North are blocked by those that want to go East.

The routing algorithm should take into account these situations to avoid them. Deadlock prevents a packet from moving inside the network. It rapidly spreads over the network unless the routing algorithm includes preventive or recovery measures.

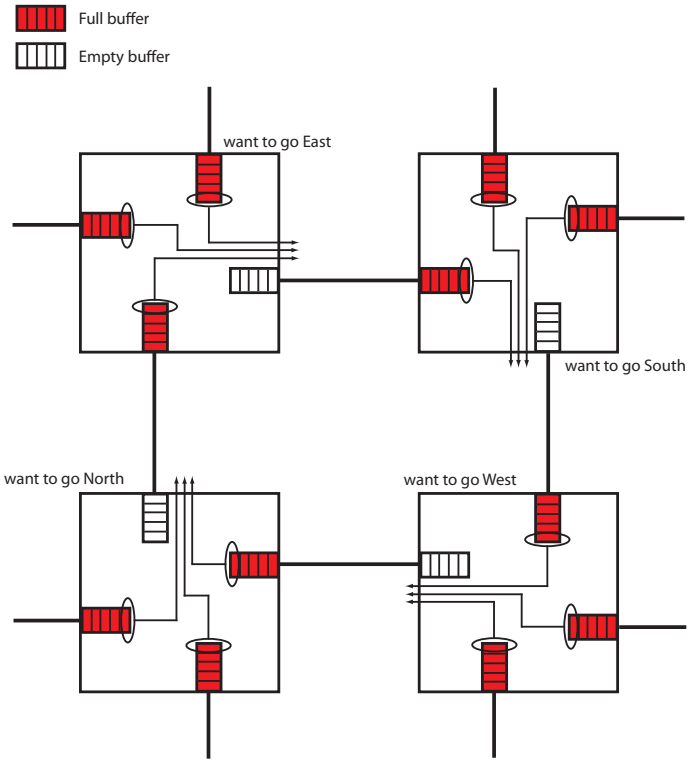


Figure 2.12: Example of a deadlock situation

Two possibilities for solving the deadlock problem exist. Deadlock recovery and deadlock avoidance. Most deadlock avoidance schemes [41] [42] prevent deadlocks by enforcing routing restrictions, e.g., dimension-order restrictions and turn restrictions, resulting in low routing adaptiveness and low network performance. However, these routing restrictions can increase router speed by simplifying router designs.

In contrast, deadlock recovery routing schemes [43] [44] maximize routing adaptiveness by relaxing routing restrictions enforced by deadlock avoidance schemes to prevent deadlock. This increased routing adaptiveness, however, can compromise router speed by complicating the design of router components, i.e., crossbar, channel selection logic, routing and arbitration logic [45]. Preserving the increased routing adaptiveness by implementing the router in this way not only increases router delay but also limits the ability to optimize the router design to reduce router delay.

2.4.5.2 Deadlock Avoidance

Flow Control

Using flow control across the switch will eliminate the possibility of creating deadlocks. Flow control can forbid to have full buffers in scenarios where deadlock could occur.

Virtual Channels

This technique allows adaptiveness in the network at the expense of buffer space and control logic so that the virtual channels can share the physical channels. It also entails more bits for the link flow control [46], [47].

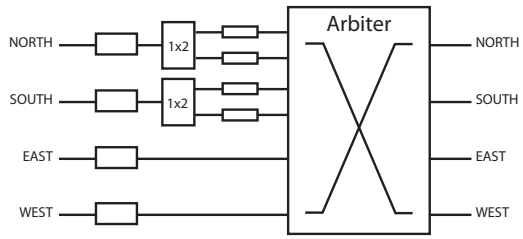


Figure 2.13: Virtual Channel control logic

Turn model

This model is not based on adding physical or virtual channels to network topologies. Instead, it analyzes the directions in which packets can turn in a network and the cycles that the turns can form. It can be applied to adaptive and nonadaptive algorithms.

Partially Adaptive routing in 2D meshes.

- West-First Routing Algorithm
- North-Last Routing Algorithm
- Negative-First Routing Algorithm

XY routing is a turn model algorithm. In this algorithm, a packet is first routed along the x dimension and then along the y dimension. The XY algorithm prevents the deadlock by prohibiting four of the turns. It doesn't allow adaptiveness with detriment of performance.

Dropping A possible solution to avoid deadlock is discarding packets that are already in the buffers.

The following table differs what methods are used in on-chip and off-chip networks.

	Flow Control	Virtual Circuits	Turn Model	Dropping
Off-chip networks	√	√	√	√
On-chip networks	√	√	√	

Table 2.8: Deadlock avoidance in on-chip and off-chip networks

An example of flow control to avoid deadlock in NoC is Æthereal for BE and GT. For BE Æthereal also uses the turn model.

2.5 Conclusions

The advantages of on-chip communications, and of NoCs in particular, allow us to claim a new vision of fabric crossbar switches. Our work differs from previous art by addressing both issues of optimizing the crossbar delay as well as the internal buffer use and requirement. This is addressed by adopting a *multi-hop buffered* NoC-based paradigm for the design of buffered crossbar switches. Designing the fabric as a NoC permits to use short wires instead of the long wires used in traditional CICQ crossbars. This allows scalability in port count and speed per port and enables reliable high-speed signaling. Using uniform short wires affords significant advantages in cost and performance. Additionally, a NoC based fabric requires simpler switch design by allowing simple input memory structure such as FIFO input queueing, as opposed to traditional design that requires sophisticated queueing structures such as the VOQ architecture.

Next, we summarize the characteristics of the NoCs that we think can enhance the high-speed routers performance:

- short wires ⇒ tighter synchronization ⇒ smaller buffers.
- local arbitration vs global arbitration ⇒ less delay.
- higher frequencies ⇒ speedup.

3 Unidirectional NoC

The previous chapters have explained the existing proposals for crossbar switches and how they are not scalable and cannot follow the explosive growth of the Internet. They illustrate how the scaling of microchip technologies opens a new door for on-chip communications. And they summarize how the advantages of NoCs in scalability, its robustness thanks to the multi-route possibility and its support to redundancy can solve the current problems of crossbar switches.

This thesis proposes to use a buffered crossbar switch fabric based on NoC for moving packets between router line cards. This novel design is depicted in Figure 3.1. The NoC should be designed to accomplish the needs of a high-performance Internet router. The main requirements of the new switch fabric will be based on its performance, cost and area.

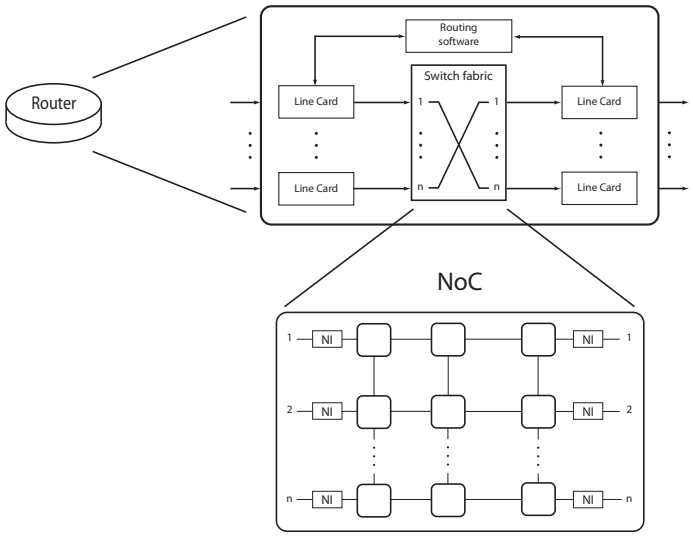


Figure 3.1: The proposal architecture.

Routers and NIs are the main components of the NoC switch fabric. First, the topology of the network should be chosen. Then, the NIs and routers are designed. The study of the different possibilities to build the NoC routers in section 2 will be the base of their layout. Afterwards, a routing algorithm is presented for the new architecture.

In this chapter, we first introduce the simulation environment used in this thesis to evaluate the proposal architectures performance. A simulator of Stanford University was used [48] and modified to fulfill the desired objectives: test the NoC architectures. This simulator is used to test the architectures proposed in this thesis. The behavior of the system is based on its throughput, average cell delay and distribution of the load in the mesh. This simulation environment, the models of traffic employed and the parameters to measure in the following simulations are explained in appendix A.

3.1 UDN Architecture

The proposed architecture is shown in Figure 3.2. We will call it the Unidirectional NoC (UDN) crossbar. It is a two-dimensional mesh of packet-switched routers, with network interfaces (NI) on two opposing sides of the mesh. The mesh is scalable in the number of stages M between the N inputs and outputs. Where N is the number of rows of the mesh and M the number of columns. This scalability allows to play with the number of routers that compose the mesh to obtain the best cost-performance architecture. Packets contain an ATM cell (including the header), and flow in one direction through the mesh.

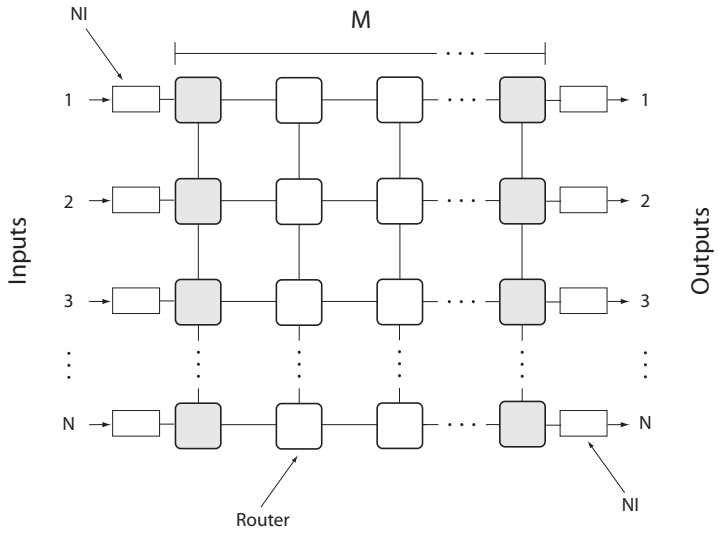


Figure 3.2: The Unidirectional NoC (UDN) crossbar architecture.

3.1.1 Architectural Design

Based on chapter 2 the architectural components of the UDN router are elected. Constraints in size, simplicity and performance are the factors that may lead us to choose the different options for building the new architecture.

3.1.1.1 NI design

The switch is connected to the outside through the NI. As packets flow only in one direction, there are two types of NIs in the UDN architecture. Those of the East side, simply unwrap the packet to send it to the Line cards of the router. The packets are sent in order through the network, so this NIs do not have to reorder them.

The NIs of the West side are more complex. These NIs are based on [15]. Their architecture is shown in Figure 3.3. When a message arrives, the remote queue id is taken from the table. Then, they include the header for the packet, with the destination port. We implement flow control using credit based LLFC between the routers and the NIs using credits. The NI has a counter (credit) that is initialized with the remote buffer size. This counter tracks the empty buffer space of the router. When the router sends a packet, it generates a credit back to the NI to indicate that more empty space is available. Then the counter of the NI is incremented.

Whenever a queue contains data, the request generator issues a signal specifying that queue can be scheduled.

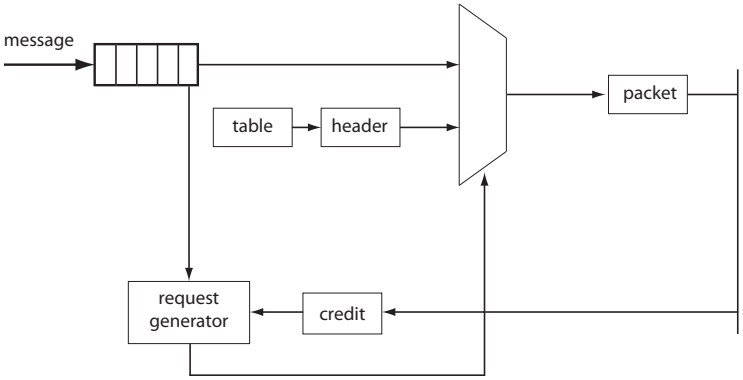


Figure 3.3: UDN Network Interface

3.1.1.2 Router design

Firstly, it should decided what kind of communication is using in the network . In this case, the packets received in the inputs are sent through the network by packet switching, as it is the most appropriate for computer communications and we are implementing only BE (section 2.4.1).

For the buffering architecture, input queueing is implemented. Shared buffer has a more efficient use of the buffer space, but it needs extra hardware. As it needs speedup, shared buffer architecture is not considered (refer to section 2.4.2.1). Though HoL blocking is caused, it is proved in section 3.2.3 that it is negligible due to the routing algorithm elected. Then, the use of input queues simplifies and reduces the size of the switch.

RR is implemented for the arbitration as it is a fair (starvation free) scheduling policy. This is the also the scheduling policy because of its performance in Buffered Crossbars (see section 2.4.2.2) and its simplicity in hardware. Therefore, comparisons between both architectures will be done with this type or arbitration.

Packets are received entirely by a router before being forwarded to the next router, also known as store and forward. Though as explained in section 2.4.4 wormhole is more efficient, in terms of simplicity it is not tested.

Buffering credit flow control is performed to avoid elastic buffers. This implies that buffer size (also called buffer depth) should be at least 2 cells to avoid the delay generated by the flow control. The delay of the credit sent back by the receiver makes it impossible to have 100% throughput when the transmitter sends 1 packet per cycle. This effect is shown in Figure 3.4. In architectures like Æthereal, this handshake in the flow control is minimized because it only has the delay of one flit instead of one packet.

In most NoC architectures, buffers account for the main part of the router area. As such, it is a major concern to minimize the amount of buffering necessary under given performance requirements. In [49] it is shown that increasing the buffer size is not a solution towards avoiding congestion, though it is useful to absorb bursty traffic. So a compromise between area and performance should be achieved.

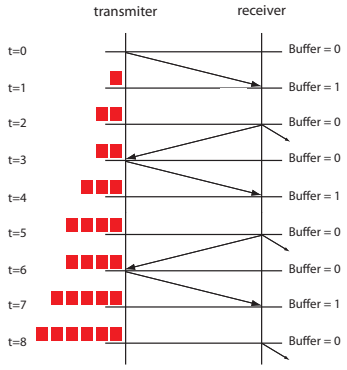
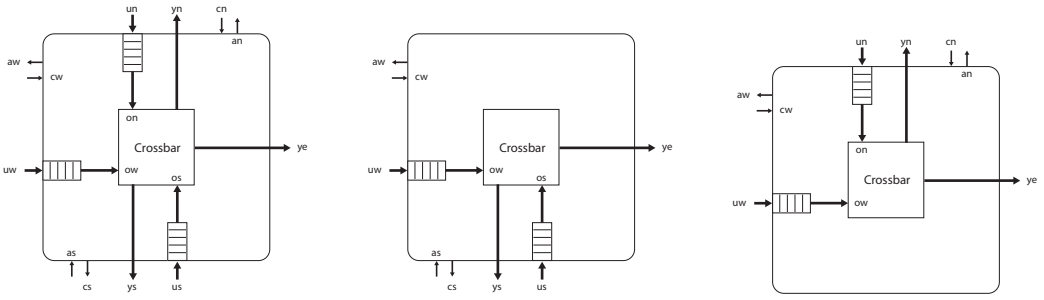


Figure 3.4: Case of Buffer Size 1

Packets advance at a maximum rate of one packet per cycle. For simplicity, we show synchronous implementation but mesochronous or asynchronous are possible. Because packets do not flow from right to left, the router is asymmetric, and no deadlock can occur. The routing algorithm elected is explained in the next section.

Figure 3.5 shows the router architecture. Where: u =[un us ue uw] means input; y=[yn ys ye yw] means output of the router; o=[on os oe ow] means output of the input ports; a=[an as ae aw] means pin flow control of the input ports; c=[cn cs ce cw] means pin flow control of the neighbors For reasons of cost, area and performance, buffer size is 4 in each input port. In section 3.3 we calculate the size of the switch fabric with this amount of buffering. Routers belonging to the first and last row differ from the rest of the routers in their degree. Then, fewer buffers are required in those cases. For credit based flow control, special pins for communications are needed on each router.



(a) Center Router (b) Router for 1st row (c) Router for last row

Figure 3.5: Different kind of routers for UDN architecture

Architectural component	UDN
Switching Modes	Packet Switching
Buffering Architecture	Input queueing
Access to the Arbiter	FIFO
Arbiter Algorithm	RR
LLFC	Credit-Based
Switching mode	Store and Forward

Table 3.1: UDN architectural components

Table 3.1 collects all the architectural decisions for UDN routers. Once the architecture of the fabric is chosen, a routing strategy should be decided. Here we proposed a new algorithm, that achieves a good load balance making the most of the multi-hop characteristics of the UDN mesh. The next section explains this routing election and makes an analytical and simulation study of its performance.

3.2 Routing in UDN and Performance Analysis

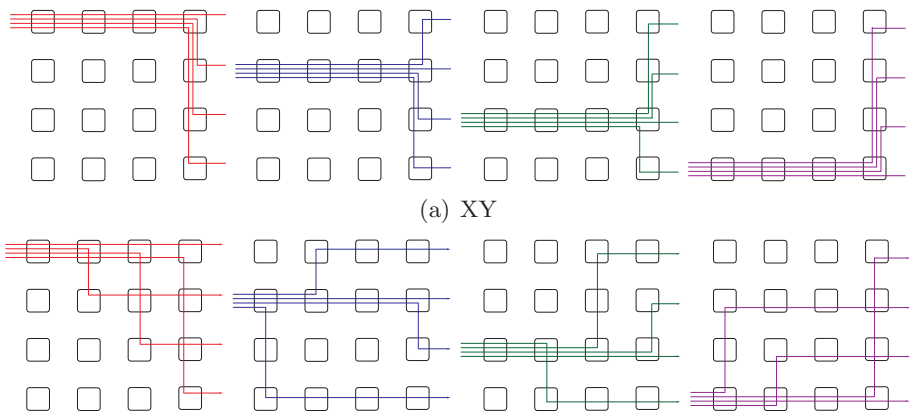
This section first proposes two routing algorithms for UDN. They are the traditional XY routing algorithm and a new algorithm called Modulo algorithm. It presents and makes an analytical study of the Modulo algorithm. This study first calculates the number of packets per cycle the system can egress based on the assumption that there is no HoL blocking. Thereafter, the calculations are done with HoL blocking presence. Both values of are compared to obtain the throughput of the switch. The following section compares the Modulo algorithm analysis with the simulation values to estimate the validity of the mathematical model. Finally, there is a comparison of the Modulo algorithm with other existing routing algorithms.

3.2.1 Routing Analysis

The UDN packet consists of the ATM cell that is now the payload and a new header. ATM stands for Asynchronous Transfer Mode (ATM). Is a transmission mode that encodes data traffic into small fixed-sized cells. The goal of Asynchronous Transfer Mode (ATM) is to integrate the transmission of various type of data (e.g. video, IP-traffic) into one high-speed network. An ATM cell is comprised of 53 bytes. Five of the bytes make up the header field and the remaining 48 bytes form the user information field.

The ATM cell is now wrapped with a new header. This packet routing header contains the path from ingress to egress, and is shifted at every router.

Packets follow deterministic minimal paths through the NoC, using one of three different routing algorithms: XY, balanced XY, and balanced flows . In standard XY (see Figure 3.6(a)) routing, packets travel East to the right column (X) and then to the correct row (Y). This results in a very unbalanced NoC usage because all vertical traffic occurs in the column of egress routers.



(a) XY

(b) Balanced XY

Figure 3.6: Example of XY and Balanced XY routing in a 4x4 UDN

Balanced XY, illustrated in Figure 3.6(b), remedies this by introducing an extra turn in one of the earlier columns. A packet for output (x, y) turns South/North when $x \ mod \ M = (N - i + j + t) \ mod \ M$, and East when $y = j$, where i, j indicate the current router position in the mesh, t the location of the extra turn, N the number of inputs/outputs, and M the number of stages of the mesh. Hence, we refer to this as the Modulo routing algorithm as described in Algorithm 1. It can be set for both balanced XY, balanced and balanced flows . If t=0 then it is set for balanced XY.

The following algorithm is performed by *Router[i,j]* that receives a packet whose desired exit is *output*.

Algorithm 1 Modulo UDN

```
Switch(Packet Buffer Input)
    case(North):
      if(i == output) then East
      else South
    case(South):
      if(i == output) then East
      else North
    case(West):
      if ((output%M) == (N-i+j+t)%M) then
          if(output > j) Down
  else North
      else East
```

In balanced XY, each input-output pair has a fixed t. As a result, all packets of an input-output pair follow the same path. In balanced flow routing, t increases modulo M for every new flow. A flow is defined here as the group of packets that arrive in consecutive time slots and that have the same input-output pair. Hence, for a given input-output pair, all packets (cells) of one flow follow the same path, but packets of different flows take different paths. Hence successive flows from one input-output pair may be interleaved at the output, although they were not interleaved at the input. This is not an issue as all the packets of each flow follow the same path and FIFO scheduling is used.

UDN packet

If the path is fixed since the beginning, the number of bits of the header is determined by the size of the mesh. For this architecture, the router degree is 2×2 or 3×3. Hence the longest path requires $(N + M - 1) * \lceil {}^2log\ 3 \rceil$ bits, e.g. $(32 + 8 - 1) * 2 = 78$ bits. Whether this is less than the width of a link depends on serialization.

In UDN architecture, with the Modulo Algorithm 1, only $\lceil {}^2log\ M \rceil$ bits are needed if balanced XY is performed. Then, for a 32x8 Switch, this field has a size of 3 bits. If balanced flows is elected, the number of bits needed are $2\times \lceil {}^2log\ M \rceil$

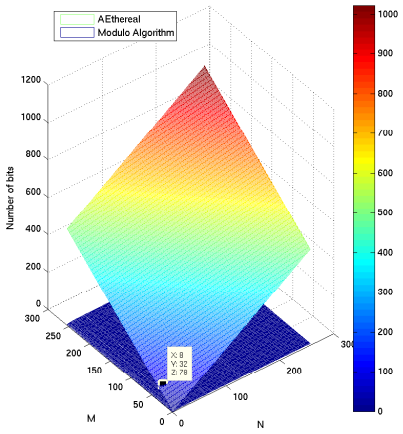


Figure 3.7: Number of bits for the header.

Figure 3.7 shows the different number of bits needed depending on the way the path is elected. Æthereal, i.e., fixes the whole path since the beginning meanwhile the Modulo Algorithm calculates in each router the next step.

Figure 3.8 represents the UDN the packet for balanced XY:

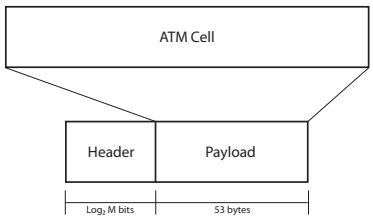
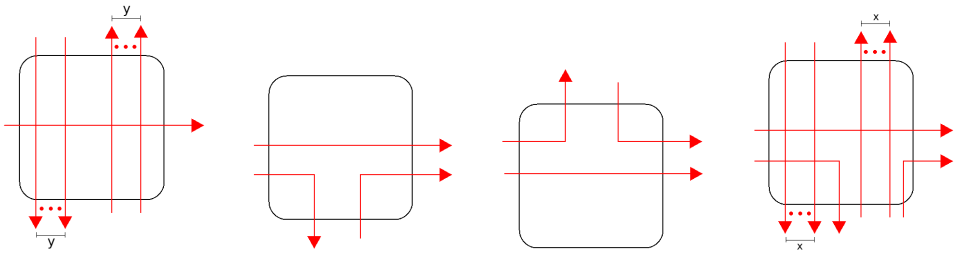


Figure 3.8: UDN packet for balanced XY.

3.2.2 UDN Throughput Analysis with Modulo Algorithm without HoL

In this section, we study the throughput of an $N \times N$ UDN-based crossbar switch using the Modulo routing algorithm described above. We assume that the input traffic is random, the

arbitration used is random, the switch speedup is one and the number of ports of the switch is even. We use random traffic, that performs as Bernoulli Uniform traffic, and for terms of simplicity, random arbitration is used though UDN really uses RR . The buffer depth is considered infinite to avoid dependencies among the routers. An $N \times N$ UDN-based crossbar switch contains four types of routers based on the kind of flows that pass through them. The possible kind of flows are: West→East, West→South, West→North, South→East, South→North, North→South. These types of routers are illustrated in Figure 3.9.



(a) Type a (b) Type b (c) Type c (d) Type d

Figure 3.9: Router types

- Type a: To this type belong those routers that do not have packets that go West-South nor South-East or West-North nor North-East.

$$Router\ type\ a = R[i][2imodM] | i \in 0..M-1$$

The number of flows that can go North-South (that is the same that go South-North) depends on the position of the router in the matrix.

$$y = 0...\frac{N-2}{2}$$

Where y is the number of flows for North-South and South-North. For each y there are 2 routers. Then there are N routers of type a.

- Type b: A router is a type b router if the packets can travel West-East or West-South and South-East.

$$Router\ type\ b = R[i+1][i] | i \in 0..N-2 \cup R[i][0] | i \in 0..N-1$$

There are $2N-3$ of these routers.

- Type c: These routers have packets that can go West-East, West-North and North-East. There are $2N-3$ of these routers. It is the symmetrical case of type b.

$$Router\ type\ c = R[i][i] | i \in 1..N-1 \cup R[N-1][i] | i \in 0..M-2$$

- Type d: To this type belong the routers that have the same kind of packets that type b or c plus packets that can travel from North-South or South-North.

$$Router\ type\ d = R[i][j] \notin (Router\ type\ a \cup Router\ type\ b \cup Router\ type\ c)$$

The number of flows that can go in those directions depend on the position in the matrix.

$$x = 1..\frac{N-2}{2}$$

Where x is the number of possible flows for North-South and for South-North. For each number of flows there are $2(2N-4x-3)$ routers. The number of routers in total for this type is:

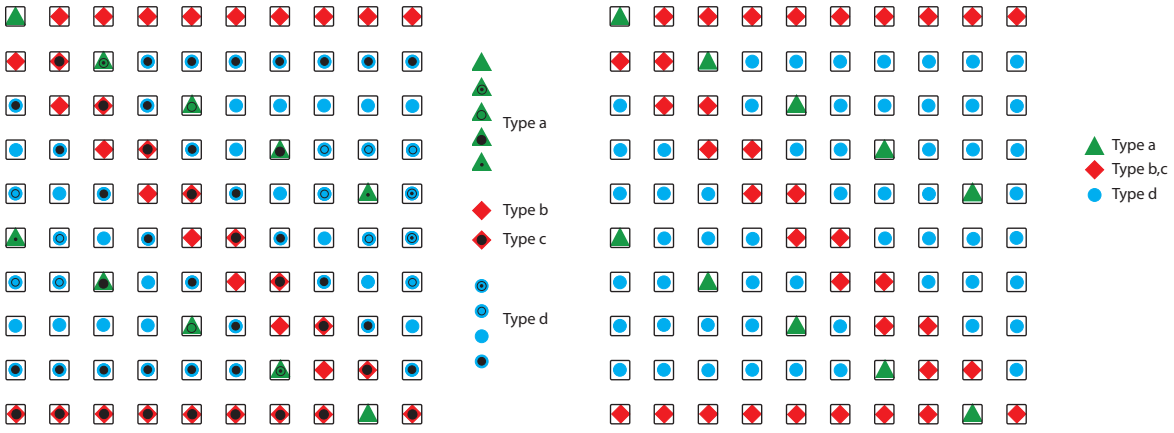
$$\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)$$

Finally, summing the number of routers of each type:

$$N+(2N-3)+(2N-3)+\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)=N^2$$

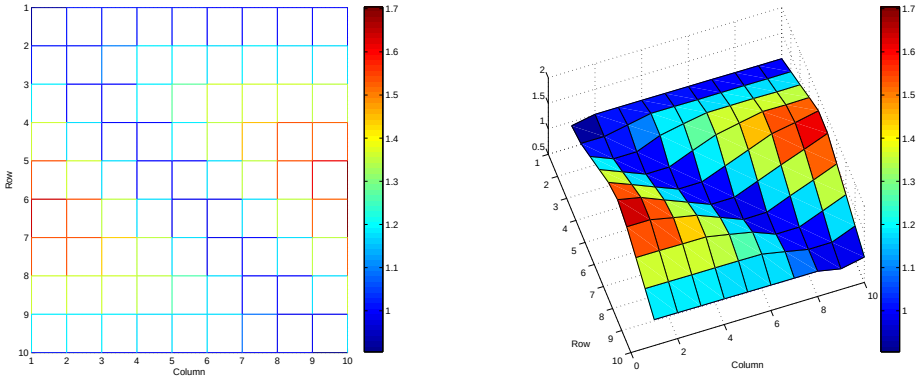
Now the types of routers are identified, their throughput is studied. Firstly, we calculate it for the ideal case, where the router is capable to exit all the packets that enter in each cycle Then, we calculate the average throughput for each type in case of HoL blocking presence is calculated. Then . Finally both cases are compared, to obtain the throughput of the switch.

The placements of each type in a 10×10 mesh is shown in Figure 3.10. Type a and type d are divided in subtypes. This subtypes depend on the value of y for type a and of x in type d. In this case, for example, for a 10×10 mesh, there are $\sum_{y=0}^{\frac{N-2}{2}} = 10$ routers of type a, but as there are 2 routers per subtype, there are 5 subtypes of type a. This case is analog to type d.



(a) Router types (b) Router types simplified

Figure 3.10: Router types for a 10x10 mesh



(a) Router types front vision

(b) Router types 3D vision

Figure 3.11: Simulation of Router types for a 10x10 mesh

Figure 3.11 is the result of a simulation of a 10x10 mesh for Bernoulli Uniform traffic. The environment of this simulation and the model of traffic employed is explained in appendix A. This graph shows the number of packets per cycle for each router in the mesh. We can see, that each type of router, deals with the same average of packets per cycle. Figure 3.10(a) that showed the analytical placement of the routers, draws the same shape as the simulation. We can conclude our analytical study of each type or router is close enough mathematically and simulation wise.

In the next calculations, NOP means the number of packets per cycle of a router of the corresponding type. $ANOP$ means the average number of packets per cycle for all the routers of that type.

To calculate the throughput of each type of routing, the different probabilities should be taken into account.

P(packet going from input i to output j) = P(i $\rightarrow$ j) = P(i)P(i/j) = $1\frac{1}{N}$

As t = 0 in this study for the modulo Algorithm, the different probabilities are as follows:
P(West$\rightarrow$South) = $\frac{1}{N}$
P(North$\rightarrow$South) = P(South$\rightarrow$North) = $\frac{x}{N}$ or $\frac{y}{N}$
P(South$\rightarrow$East) = $\frac{1}{N}$
P(West$\rightarrow$East) = 1

First, it is calculated the ideal number of packets the switch should exit to have 100% throughput.

If the ideal throughput happens when each router is able to deal with each packet that enters: Average ideal throughput per type:

- Type a:

$$NOP = \frac{2y}{N} + 1$$

$$ANOP = \frac{2 * \sum_{y=0}^{\frac{N-2}{2}} \left(\frac{2y}{N} + 1\right)}{N} = \frac{3N-2}{2N}$$

- Type b,c:

$$NOP = 1 + \frac{2}{N}$$

$$ANOP = 1 + \frac{2}{N}$$

- Type d:

$$NOP = \frac{2x}{N} + \frac{2}{N} + 1$$

$$\begin{aligned} ANOP &= \frac{\sum_{x=1}^{\frac{N-2}{2}} \left(2(2N-4x-3)\left(\frac{2x}{N}+\frac{2}{N}+1\right)\right)}{\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)} \\ &= \frac{\frac{8N^3-27N^2-14N+72}{6N}}{\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)} \\ &= \frac{8N^2-11N-36}{6N(N-3)} \end{aligned}$$

The average desired throughput in the switch, is then:

$$ATSwitch = \frac{\frac{3N-2}{2} + \left(1+\frac{2}{N}\right)(4N-6) + \left(\frac{8N^3-27N^2-14N+72}{6N}\right)}{N^2} = \frac{4N^2+3N-4}{3N^2}$$

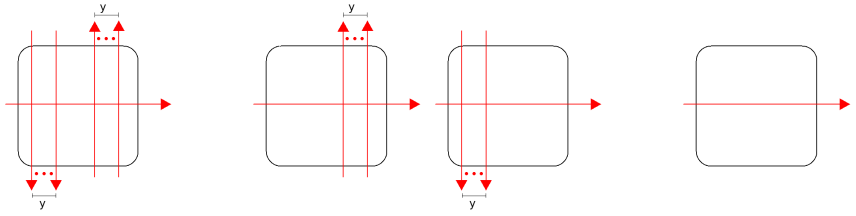
3.2.3 UDN Throughput Analysis with Modulo Algorithm with HoL

Next, the Average Throughput per type taking into account HoL blocking is calculated. For the calculation of the analytical response of the mesh, to simplify the calculations, it is assumed that we flush all HoL blocked packets at the end of every time slot. Then, only the actual time slot has to be taken into account.

- Type a:

$$NOP = 3 * P(3\ diff\ dest) + 2 * P(2\ diff\ dest) + 1 * P(1\ dest)$$

Figure 3.12 represents the possible destinations for type a.



(a) 3 destinations (b) 2 destinations (c) 1 destination

Figure 3.12: Possible destinations for type a.

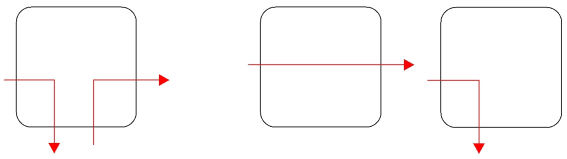
$$NOP = 3\left(\frac{y}{N}\right)^2 + 1\left(1-\frac{y}{N}\right)^2 + 4\left(1-\frac{y}{N}\right)\frac{y}{N} = 1+\frac{2y}{N}$$

$$ANOP = \frac{2*\sum_{y=0}^{\frac{N-2}{2}}\left(1+\frac{2y}{N}\right)}{N} = \frac{\frac{3*N-2}{2}}{N} = \frac{3*N-2}{2N}$$

- Type b,c:

$$NOP = 2*P(2\ diff\ dest) + 1*P(1diff\ dest)$$

Figure 3.13 represents the possible destinations for type b.



(a) 2 destinations (b) 1 destination

Figure 3.13: Possible destinations for type b.

$$NOP = 2\left(\frac{1}{N^2}\right) + 1\left(\frac{1}{N}\left(1-\frac{1}{N}\right)\right) + 1\left(1-\frac{1}{N}\right) = \frac{1}{N^2}+1$$

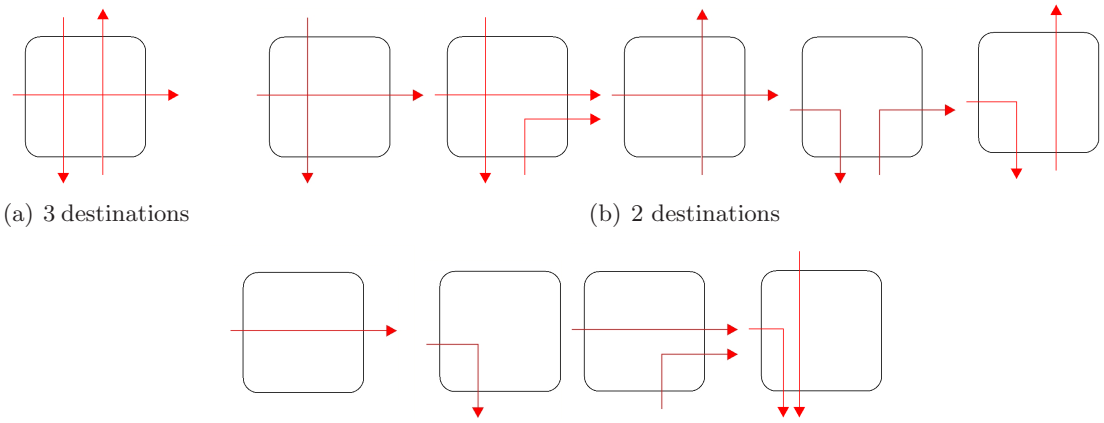
The $4N-6$ routers are the same, so the average throughput (AT) for type a is

$$AT = \frac{1}{N^2}+1$$

- type d:

$$NOP = 3*P(3\ diff\ dest) + 2*P(2\ diff\ dest) + 1*P(1\ diff\ dest)$$

This possible destinations are gathered in Figure 3.14.



(a) 3 destinations

(b) 2 destinations

(c) 1 destination

Figure 3.14: Possible destinations for type d.

$$
\begin{aligned}
NOP = 3 &\left[\left(1-\frac{1}{N}\right)\frac{x^2}{N^2}\right] + 2\left[\frac{x}{N}\left(1-\frac{x}{N}-\frac{1}{N}\right)\left(1-\frac{1}{N}\right) + \left(\frac{x}{N}\right)\left(1-\frac{1}{N}\right)\left(\frac{1}{N}\right)\right] \\
&+ 2\left[\left(1-\frac{x}{N}\right)\left(1-\frac{1}{N}\right)\left(\frac{x}{N}\right) + \left(\frac{1}{N}\right)\left(\frac{1}{N}\right) + \left(\frac{1}{N}\right)\left(\frac{x}{N}\right)\right] \\
&+ 1\left[\left(1-\frac{1}{N}\right)\left(1-\frac{x}{N}\right)\left(1-\frac{x}{N}-\frac{1}{N}\right)\right] \\
&+ 1\left[\left(1-\frac{1}{N}\right)\left(1-\frac{x}{N}\right)\frac{1}{N} + \frac{1}{N}\frac{x}{N}\left(1-\frac{x}{N}-\frac{1}{N}\right) + \frac{1}{N}\left(1-\frac{x}{N}\right)\left(1-\frac{x}{N}-\frac{1}{N}\right)\right]
\end{aligned}
$$

$$
NOP = \frac{(2*N-1)x + N^2 + 1}{N^2}
$$

$$
ANOP = \frac{\sum_{x=1}^{\frac{N-2}{2}} \left(2(2N-4x-3)\left(\frac{(2*N-1)x+N^2+1}{N^2}\right)\right)}{\frac{1+(N-2)}{2}(N-2) + \frac{1+(N-3)}{2}(N-3) - (N-2)} = \frac{16N^3-48N^2+17N-36}{12N^2(N-3)}
$$

The average throughput of the switch, is then:

$$
\begin{aligned}
ANOP\ Switch &= \frac{\frac{3N-2}{2} + (4N-6)\left(\frac{1}{N^2}+1\right)}{N^2} \\
&+ \frac{\left(\frac{16N^3-48N^2+17N-36}{12N^2(N-3)}\right)\left(\frac{1+(N-2)}{2}(N-2) + \frac{1+(N-3)}{2}(N-3) - (N-2)\right)}{N^2} \\
&= \frac{16N^3-14N^2+29N-22}{12N^3}
\end{aligned}
$$

Once we have the ideal throughput in number of packets and the throughput taking into account the HoL blocking, we can calculate the ratio of all the packets that are indeed egressed by the system. Figure 3.15 represents the final throughput of the system.

$$Throughput = \frac{\frac{4N^2+3N-4}{3N^2}}{\frac{16N^3-14N^2+29N-22}{12N^3}}$$

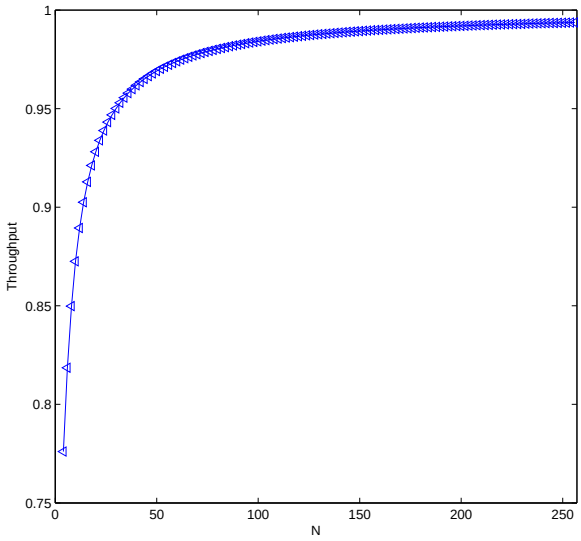


Figure 3.15: Throughput of UDN switch with Modulo algorithm

The throughput of the switch is always above 0.775 and it rapidly improves as the mesh increases. The reason is that as N has a greater value, the probability P(i → j) decreases. The blocking caused by packets coming South→East and West→South is reduced in each router. Only type *a* has to deal with more packets per cycle because the packets North→South and South→North increase with the number of inputs of the Switch. Though type a has more packets per cycle, this type of routers, because of the possible direction of their packets, never have HoL blocking.

UDN throughput is better that in the traditional CICQ buffered crossbar switch (see section 2.4.2.1). Then it can allow FIFO input queues both at line cards and routers, instead of VOQ as the HoL occurred is shown to be insignificant.

Conclusion 3.1. *Modulo algorithm allows to use FIFO queues in both line cards and routers of the UDN architecture instead of VOQ. The HoL blocking caused is shown to be negligible.*

In the following section we analyze the behavior of the system by simulation and compare it with the mathematical study.

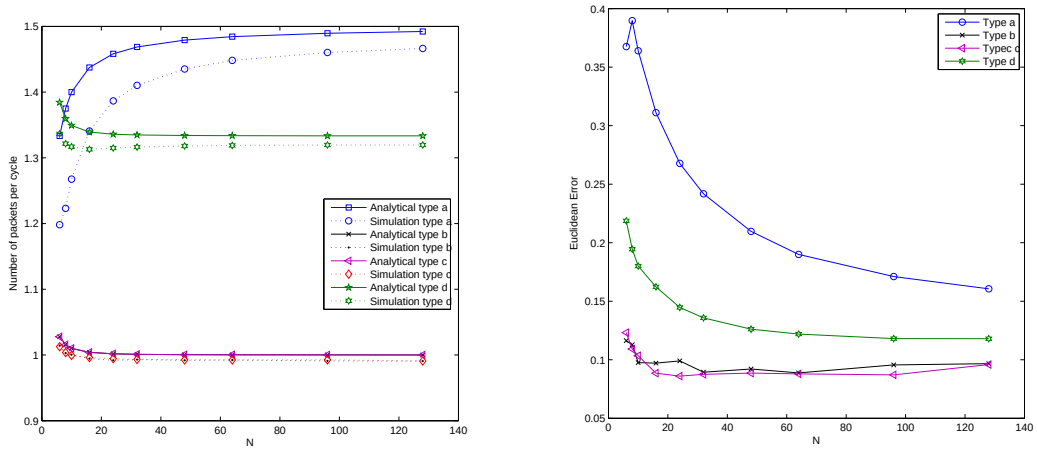
3.2.4 Analytical study VS simulation results

The following graphs represent the simulation results compared to the analytical values obtained in the previous sections.

Figure 3.16(a) collects the number of packets for different switch sizes for each type or router. Then, the formula used for the analytical values is:

$$ANOPSwitch = \frac{16N^3 - 14N^2 + 29N - 22}{12N^3}$$

For the system simulations, Bernoulli Uniform traffic was tested with an input load of 1 packet per cycle. The parameter measured is also the number of packets per cycle per type of router.



(a) Packets/cycle

(b) Euclidean error

Figure 3.16: Number of packets per cycle per type with Modulo algorithm

Type *a* is the less accurate as it has the least number of routers inside the mesh. That is why its analysis improves as N increases. The Euclidean error is shown in figure 3.16(b) for each type of router. It also shows how the analysis becomes more accurate when the number of input/output ports is higher.

Conclusion 3.2. *The analytical model is a good approximation. Its accuracy increases with the switch size.*

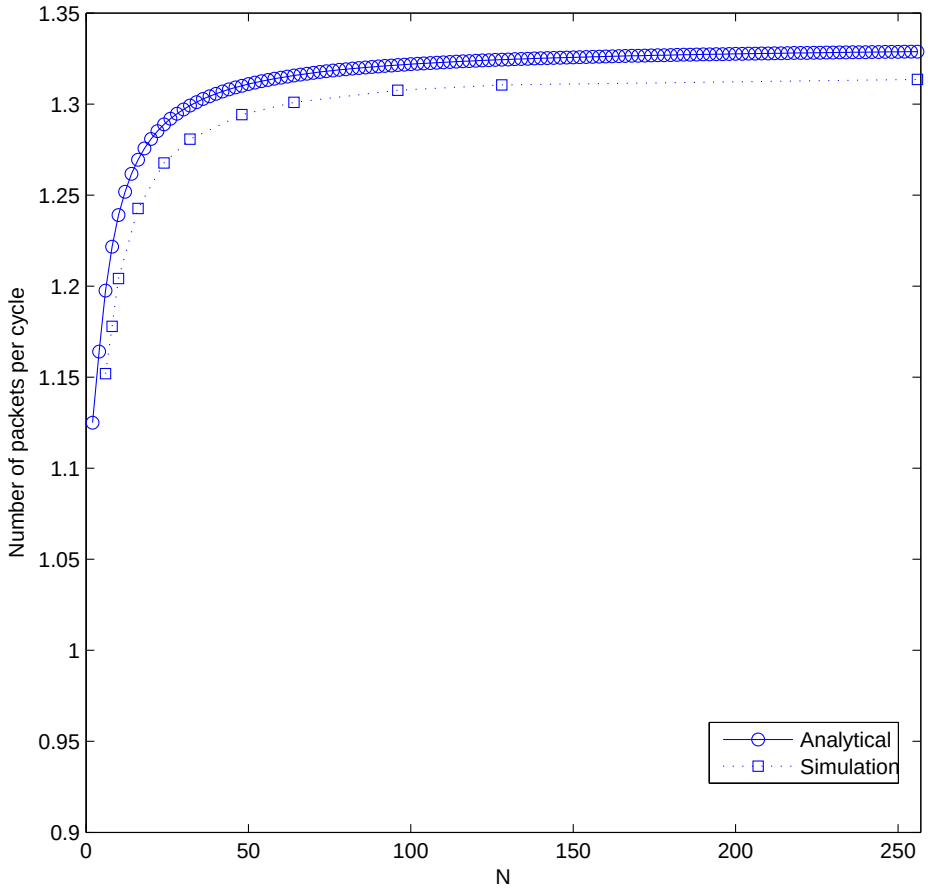


Figure 3.17: Total number of packets per cycle for the UDN switch with Modulo algorithm

The global number of packets per cycle for the simulation and the analysis study is in figure 3.17. The model is calculated for an infinite buffer size. Simulations are done with a buffer size of 500. This value is big enough to be considered infinite in the simulations.

3.2.5 Modulo Algorithm VS XY algorithm

In this section, Modulo algorithm is compared with XY routing algorithm under simulation. First it compares the number of packets/cycle each link of the mesh can exit in a 32x32 UDN switch under Bernoulli Uniform traffic. Modulo algorithm is tested for both balanced flows and balanced XY .

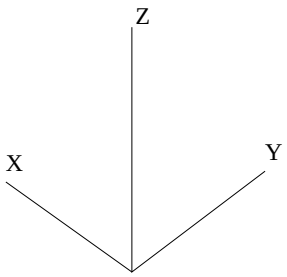
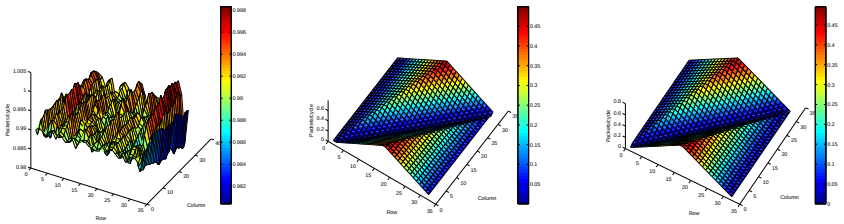


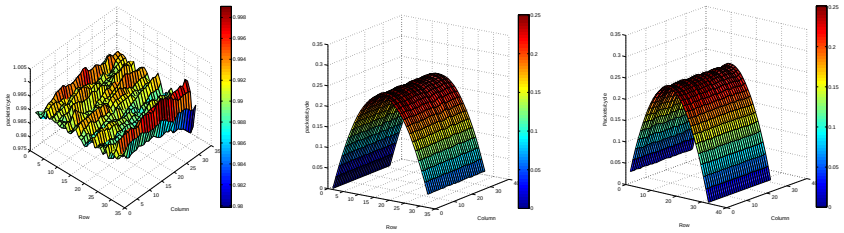
Figure 3.18: Placement of the coordinates for the figures below.

Following graphs have a 3D representation (see Figure 3.18). The switch mesh is represented in the X and Y planes by its rows (N) and columns (M). Each crosspoint is a router or a link depending on the simulation study. In the Z axis, the number of packets/cycle egressed by the router or the link of belonging to that coordinate (X,Y) is displayed. The axis position is shown in Figure 3.18.



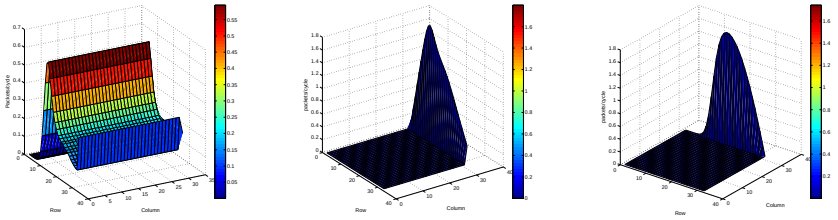
(a) East (b) North (c) South

Figure 3.19: Number of packets/cycle per link for the switch for balanced XY.



(a) East (b) North (c) South

Figure 3.20: Number of packets/cycle per link for the switch for balanced flows.



(a) East (b) North (c) South

Figure 3.21: Number of packets/cycle per link for the Switch for XY.

Graphs 3.19 and 3.21 show the load of the East, South and North links in packets per cycle in the UDN mesh. West links are not represented as there are no packets flowing in that direction. The characteristics of this simulation are explained in appendix A.

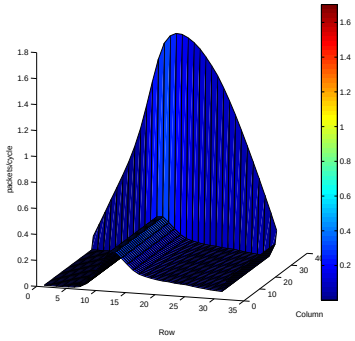
For Modulo algorithm, figures 3.20 and 3.19 show that East links have a good load balance. With this algorithm, the network is able of egressing nearly 1 packet/cycle in these links with a maximum variance of less than 0.018 packets/cycle. North and South links are distributed differently for balanced XY and balanced flows . The fact that balanced flows varies the path for each flow, distributes better the load in the mesh. balanced flows has a maximum variance of 0.25 packets/cycle meanwhile balanced XY has a maximum variance of 0.45 packets/cycle.

In XY algorithm, packets are directed first in the horizontal direction and then in the vertical direction. Then, only the links that belong to the routers of the last row of the mesh, will have packets travelling along their North and South links (Figure 3.21). In contrast all the East links of the routers of the mesh will have to deal with packets. Because of the congestion caused in the last row by all the cells that want to go North or South, East links are asymmetrically distributed. The reason is that the links of the middle have greater probability of being able to send their packets North and South (i.e. those routers of the corners will only try to send packets to the North or only to the South). Then, the rows of the middle will also have less congestion and their load will be higher.

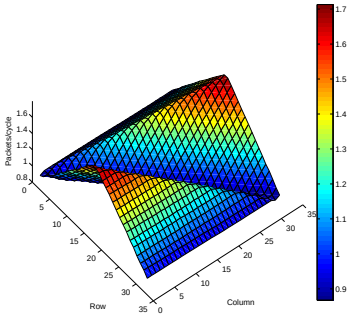
Modulo algorithm achieves a better performance thanks to a more efficient distribution of the load. For each column j in which a row i has to turn to row k, row k turns to row i. In this way, links are better balanced all over the network. If a performance comparison is done between the XY and Modulo Algorithm, it is again proved that our algorithm, with its both variants (balanced XY and balanced flows) outperforms the typical XY algorithm. Figure 3.22 shows the load in number of packets of each router of the mesh for SP1 for the routing algorithms. This values are the same as the addition of the three previous graphs. That is, i.e, Figure 3.22(a) is the addition of Figure 3.21(a) and Figure 3.21(b) and Figure 3.21(c). Figure 3.23 shows a comparison in average cell delay and throughput of both algorithms for SP1. Both cases, reflect how the Modulo Algorithm outperforms XY thanks to its better load balance.

Conclusion 3.3. *Modulo algorithm distributes the load per link more efficiently than the XY algorithm.*

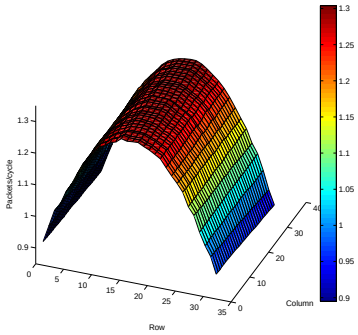
The following study compares traditional XY with balanced XY and balanced flows in terms of average cell delay and throughput.



(a) XY

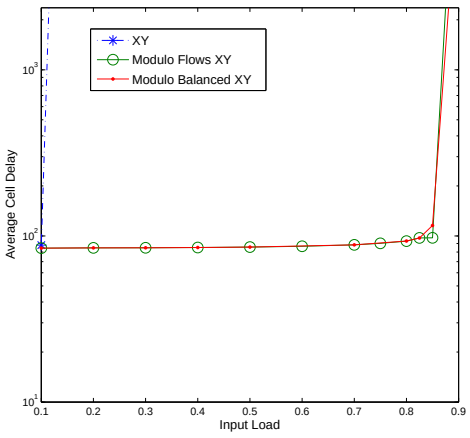


(b) Balanced XY

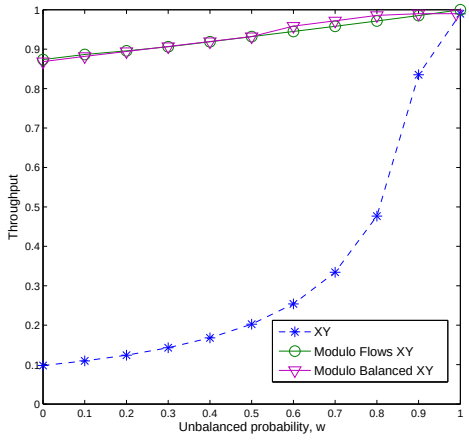


(c) Balanced flows

Figure 3.22: Comparison of XY and balanced XY and balanced flows in packets/cycle for Bernoulli Uniform Traffic



(a) Bernoulli Traffic



(b) Unbalanced Traffic

Figure 3.23: Comparison of XY and Modulo Algorithm

Conclusion 3.4. *Modulo algorithm has better performance than XY algorithm.*

We conclude, then, that Modulo algorithm is a good option to be performed in UDN architecture. Because it achieves a better load balance, balanced flows is elected.

In the next section we synthesize the UDN architecture to have a clue of the area and frequency of the switch.

3.3 Hardware Implementations

To assess the cost and performance of our proposed architectures we synthesized a NoC in an ASIC 65 nm CMOS technology. We use the Æthereal NoC [18], with input-queued worm-hole routers, flit size three, two VCs, and a non-blocking crossbar. Static-priority arbitration is used between the two VCs, and round robin per VC. This router and the UDN router will not differ significantly in terms of area and speed. The area of a router is dominated by the number of registers, which is the same for both routers. The arbitration of the Æthereal router is more complex than that of the UDN router and hence will be slower. The Æthereal NIs are more general than required here because they implement shared-memory transaction semantics, resulting in response buffers that are unused here. They also implement end to end flow control and they have 4 queues instead of 1. Two for semantics and two for input and output packets.

We generated a 3×3 UDN topology, RTL VHDL, and SystemC models of the NoC from a high-level specification [50]. This instance contains all different router degrees, and allows us to compute the area of any size UDN crossbar. Synthesis for a 65 nm CMOS technology, without any optimizations, achieved 413 MHz with a total network area of 4.8 mm^2 for a buffer depth of 4 packets. Routers of degree 3 and 4 occupy 0.29 and 0.38 mm^2 respectively, and NIs 0.32 mm^2. The area of a crossbar with N ports and M stages is $\mathit{3x3\ Router} M * 2 + \mathit{4x4\ Router} M * (N-2) + NI * N * 2$, then $0.29 * M * 2 + 0.38 * M * (N-2) + 0.32 * N * 2$ mm^2, e.g. 411 mm^2 for $N = M = 32$. The registers that are used for FIFO buffers dominate the area. By using dedicated hardware ripple-through FIFOs, described in [15], the area drops significantly. In a 90 nm CMOS process a 48-word 37-bit FIFO occupies a third of a register-based FIFO. Using the same scaling factor for all FIFOs in 65 nm, an N port crossbar would occupy approximately $0.10 * M * 2 + 0.20 * M * (N-2) + 0.11 * N * 2$ mm^2, e.g. 210 mm^2 for $N = M = 32$.

The data path of the router is 32 bits. Each ATM cell is 53 bytes, hence the cycle time is $\frac{53 bytes \times 8 \frac{bits}{byte}}{32 \frac{bits}{cycle}}$ cycles. The maximum sustainable throughput of an $N \times N$ crossbar (diagonal traffic with no contention), is therefore $N/(14 * 2.4 \text{ ns}) = 30 * N * 10^9$ cells/second, or $95 * 10^9$ cells/second for $N = 32$. The minimum cell latency is $3*M+14$ cycles, or $(3*M+14)*2.4 = 0.26$ milliseconds for $M = 32$. Doubling the data width of the NoC would double the throughput, at limited area cost because the number of registers remains the same.

Table 3.2 shows the area values and the frequency for a switch using dedicated hardware register-based FIFOs.

Buffer size	1	2	4
Frequency (MHz)		526	413
3x3 Router (mm^2)	0.043	0.143	0.289
4x4 Router (mm^2)	0.059	0.195	0.388
NI (4 buffers) (mm^2)	0.047	0.151	0.321

Table 3.2: UDN hardware implementation with register-based FIFOs

Table 3.3 shows the different sizes for different buffer depths for a switch implemented with ripple-through FIFOs. Areas are now significantly smaller than in the previous case.

Buffer size	1	2	4
Frequency (MHz)		526	413
3x3 Router (mm^2)	0.020	0.057	0.106
4x4 Router (mm^2)	0.036	0.109	0.205
NI (4 buffers) (mm^2)	0.015	0.041	0.108

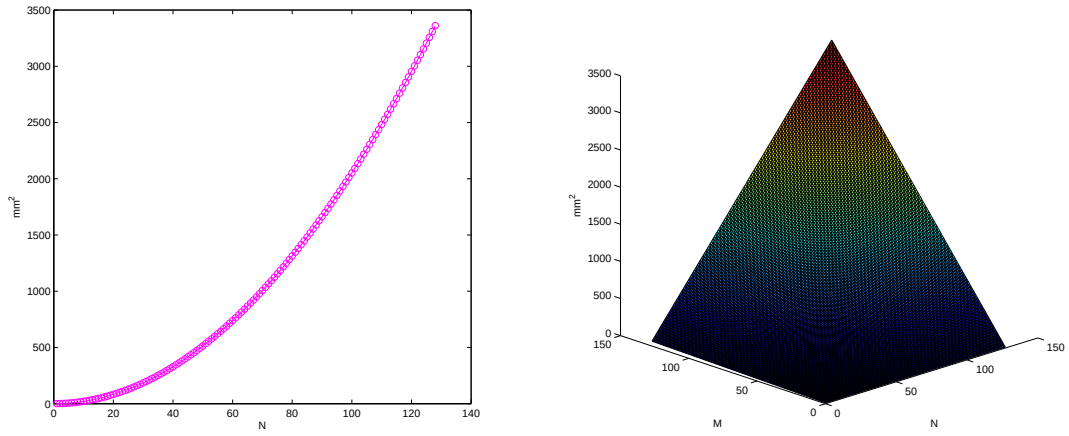
Table 3.3: UDN hardware implementation with dedicated hardware FIFOs

The total area for different switch sizes are depicted in table 3.4 both for register-based FIFOs and dedicated hardware ripple-through FIFOs.

Switch size	3	16	32	64
Register FIFOs	mm^2	mm^2	mm^2	mm^2
Buffer size 1	0.714	16.062	62.291	245.250
Buffer size 2	2.350	53.030	205.749	810.256
Buffer size 4	4.822	106.347	411.165	1616.214
Dedicated HW FIFOs	mm^2	mm^2	mm^2	mm^2
Buffer size 1	0.319	9.195	36.829	174.415
Buffer size 2	0.916	27.528	110.792	444.525
Buffer size 4	1.894	52.673	210.134	839.425

Table 3.4: UDN area for different switch sizes.

Figure 3.24 shows the variance of the switch area as N and M is increased for a buffer size 4. In Figure 3.24(a) N=M and in Figure 3.24(b) both parameters can be modified independently.



(a) (b)

Figure 3.24: Switch sizes for Buffer size 4.

3.4 Conclusions

The goal of this chapter was to present a new architecture to use as switch fabric in the high-speed routers. Its architectural components were designed based on the information of chapter 2 about NoC and Internet routers. A routing algorithm was proposed to be implemented and analytical and simulation studies were performed. Once the architectural components and the routing algorithm of the NoC are elected next chapter studies the UDN architecture under different conditions.

4 UDN System Analysis

The main part of the Internet traffic is unicast traffic. In unicast traffic each packet is sent to a single destination. This section presents the experimental results of the proposed UDN-based. The experiments are carried for different switch sizes and different internal stages of the UDN crossbar. Each point in the resulting figures is obtained after a simulation duration of one million time-slots and we gather the data when tenth of the simulation time has elapsed. The performance evaluation is carried under various traffic conditions, including: i) Bernoulli uniform and bursty uniform with different burst sizes; ii) The double diagonal traffic as defined in [51] and; iii) The Unbalanced traffic model as defined in [52]. The motivation of these traffic models and the parameters used during the simulations are in appendix A.

Firstly, UDN is compared to traditional CICQ buffered crossbar switch. Then, the architectural parameters are modified to see the response of the switch. This parameters are: switch size, speedup (SP), depth of the system, buffer depth, routing algorithm and scheduling algorithm. Some graphs represent the cost/performance study of the simulations. Where $Cost = SP \times$ *area* and $Performance = 1/delay$. Area is calculated according to data of table 3.3. Some simulations have buffer depth 20 and 6. Though we do not have the area for those routers, we estimate a router with buffer depth 20 is at least 4 times a router with buffer depth 4. Buffer depth 6 is estimated as 1.5 a router of buffer depth 4 (table shows that buffer depth 4 is $\simeq$twice buffer depth 2).

In each figure, a vector represents the value of each parameter: switch size (SS), speedup (SP), depth (D), buffer depth (BD), routing algorithm (RA), and scheduling algorithm (SA). When a x is represented, it means that parameter has several values in the graph. When there are two or more subfigures, a | separates the values for each figure if they are different. The first parameter represents what architecture is simulated. We show an example of this vector:

$$<\text{UDN, SS} = x, \text{SP} = x, \text{D} = x, \text{BD} = x, \text{RA} = x, \text{SA} = x >$$

4.1 Comparison with the traditional CICQ crossbar

First, UDN is compared to traditional CICQ crossbar in terms of throughput and average cell delay.

Figure 4.1 studies the throughput stability of the UDN and CICQ switch. We can see that even with SP1, the UDN outperforms the fully buffered CICQ switch. As the unbalanced probability increases, UDN improves its behavior.

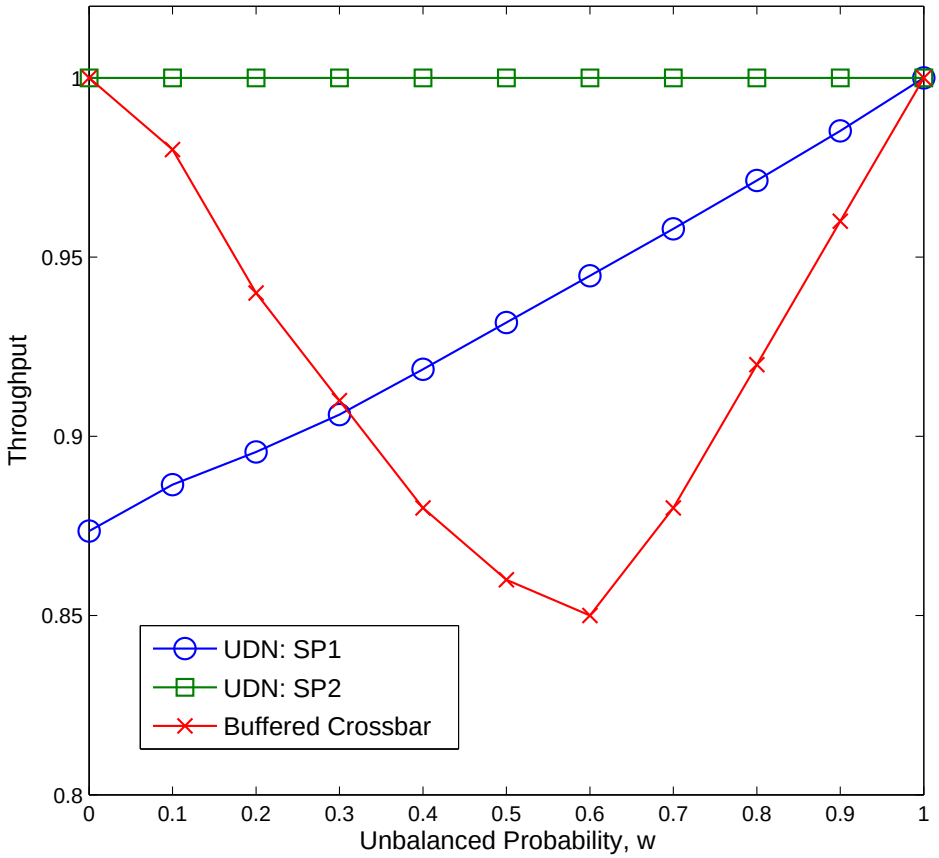


Figure 4.1: Throughput Stability of a 32x32 Switch under Unbalanced traffic.
<UDN, SS = 32, SP = x, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.1. *UDN performs better than CICQ under Unbalanced traffic.*

Figure 4.3 depicts the average cell delay of the proposed UDN architecture and compares it to that of a fully buffered crossbar. The UDN architecture employs a SP1 and SP2. We can see that our proposed UDN architecture outperforms the CICQ switch under heavy unbalanced traffic loads (see Figure 4.1). When traffic load is light ($< 90\%$), the CICQ performs better. This is due to the multi-hop delay of the UDN. Each packet has to cover at least M routers to achieve its destination in UDN meanwhile in CICQ, only one hop is necessary to exit the crossbar. Similarly, under uniform (Figure 4.2(a)) and double diagonal (Figure 4.2(b)) traffic, the CICQ performs better.

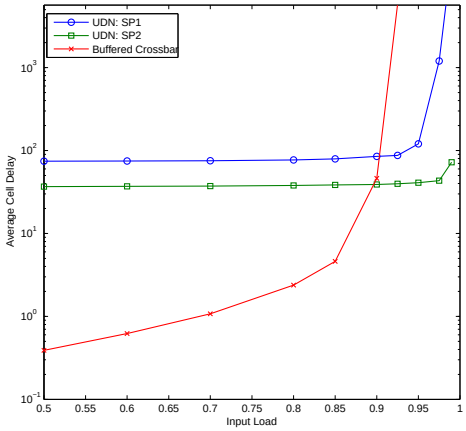
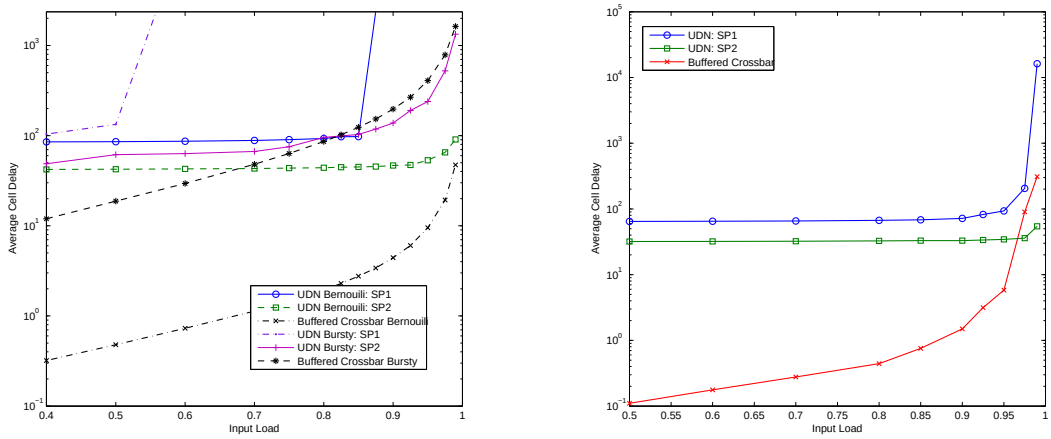


Figure 4.3: Cell delay comparison between the UDN and a CICQ switch of size 32 x 32 each under Unbalanced traffic ($w = 0.5$).

$< x$, SS = 32, SP = x, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >



(a) Uniform Traffic

(b) Double diagonal traffic

Figure 4.2: Cell delay comparison between the UDN and a CICQ switch of size 32 x 32 each under Uniform and Double diagonal traffic.

$< x$, SS = 32, SP = x, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.2. *UDN employing SP2 has less average cell delay than CICQ under heavy loads under Unbalanced traffic.*

4.2 Parameter study

This section, studies the UDN architecture under different traffic patterns and architecture parameters. Varying the architectural parameters we try to achieve the best relation in

cost/performance. Three patters of traffic are simulated: Unbalanced, Bernoulli Uniform and Bursty Uniform traffic. The SP, switch size, depth, buffer size and scheduling algorithm are adjusted to increase the performance and reduce the cost of the switch. Each variation can be done individually, or by couples. Varying i.e. the depth of the system with a determined buffer depth may have no impact in the system. But if this buffer depth is reduced, the depth of the system can be critical.

Table 4.1 summarizes the parameter variations for each type of traffic. Each crosspoint in the table represents that an analysis of the system varying those parameters together is performed in the next sections for that model of traffic.

Three models of traffic are tested: Unbalanced traffic is evaluated in section 4.2.1, Bernoulli Uniform traffic, in section 4.2.2 and Bursty traffic in section 4.2.3.

For the calculation of the area for the cost/performance graphs of the switch we use the values of table 3.3.

Parameter	Switch Size	Speedup	Depth	Buffer depth	Sched. Algorithm
Switch Size	Unbalanced Bernoulli Bursty	Unbalanced	Unbalanced	Unbalanced	
Speedup	Unbalanced Bernoulli	Unbalanced Bernoulli Bursty	Bernoulli	Bernoulli Bursty	
Depth	Unbalanced	Bernoulli	Unbalanced Bernoulli	Unbalanced Bernoulli	
Buffer depth	Unbalanced	Bernoulli Bursty	Unbalanced Bernoulli	Unbalanced Bernoulli Bursty	
Sched. Algorithm					Unbalanced Bernoulli Bursty

Table 4.1: Study of parameters for each type of traffic.

4.2.1 Unbalanced Traffic

The simulations for several values of the unbalanced parameter w and SP1 are illustrated in Figure 4.4. It shows that UDN architecture, performs better under higher values of w, this is, when the traffic tends to be more unbalanced. The position of the inputs/outputs in the mesh, simplify the balance of the load when the traffic goes directly from input i to output i that means going straight forward in the mesh, causing less congestion.

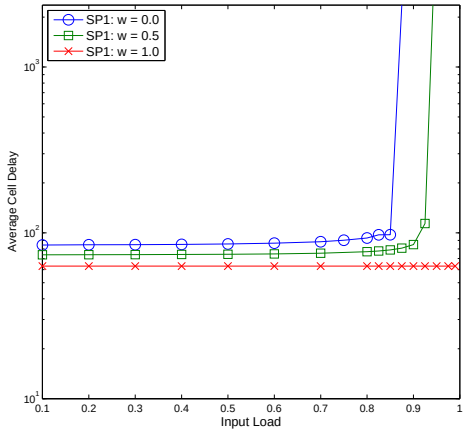


Figure 4.4: 32x32 UDN Switch varying w.

<UDN, SS = 32, SP = 1, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.3. *The more unbalanced is the traffic, the less congestion is caused in UDN*

Unbalanced traffic already has a good performance with SP2. This is shown in Figure 4.5. This figure shows how SP affects the average cell delay of the switches. SP has the effect of modifying the switch size by a factor of the SP. A 16x16 UDN switch with SP1 has the same average cell delay than the 32x32 UDN switch with SP2 and that a 64x64 switch with SP4. This fact holds meanwhile the mesh is not congested.

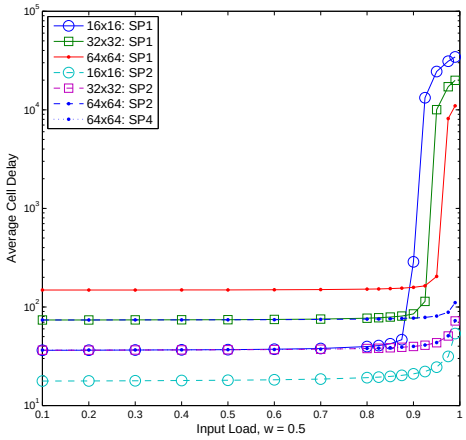
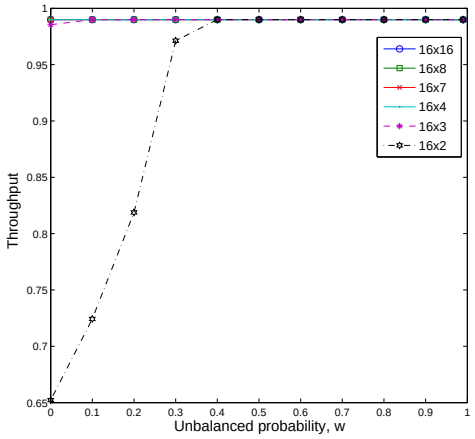


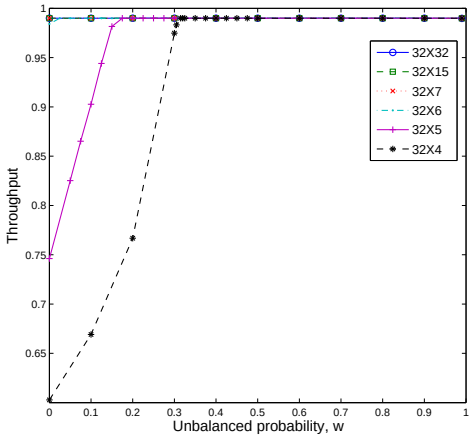
Figure 4.5: UDN Unbalanced Traffic $w = 0.5$ for different switch sizes and SPs.
<UDN, SS = x, SP = x, D = N, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.4. *SP2 is sufficient to provide 100% throughput for Unbalanced Traffic in UDN.*

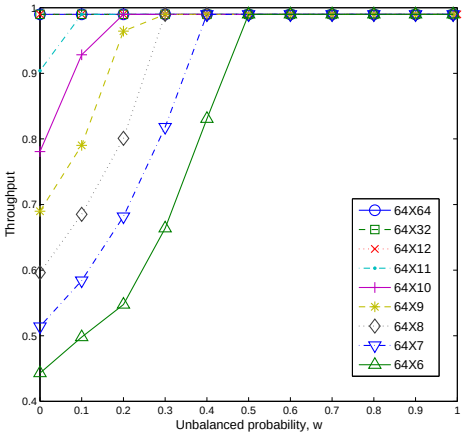
As 100% performance is achieved employing SP2, we try modifying the numbers of columns (depth) of the mesh to save cost. We can see that when the depth is more than 3 columns in the 16x16, 6 columns in the 32x32 and 12 in the 64x64, the UDN achieves full throughput. This is a saving in the interconnect cost by a factor of 5. Once that decrease factor is more than 5, performance becomes significantly worse. The number of hops is then not enough to distribute the load in the mesh. For small switch sizes, this deterioration is more noticeable because the factor N/N^2, that is the number of switches removed, is higher as N decreases. Modulo algorithm distributes now N different destinations in M routers. The turn position of some input/output pairs is now the same, incrementing the load of the routers (see Figure 4.14 for Bernoulli Uniform traffic).



(a) 16x16



(b) 32x32

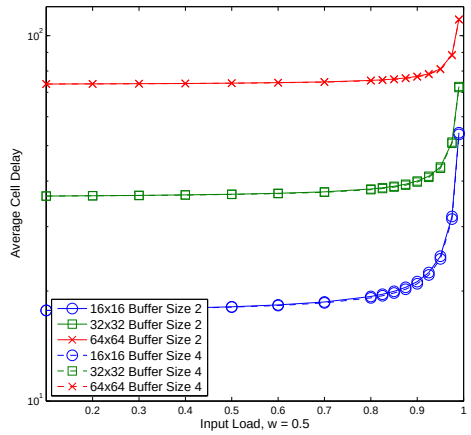


(c) 64x64

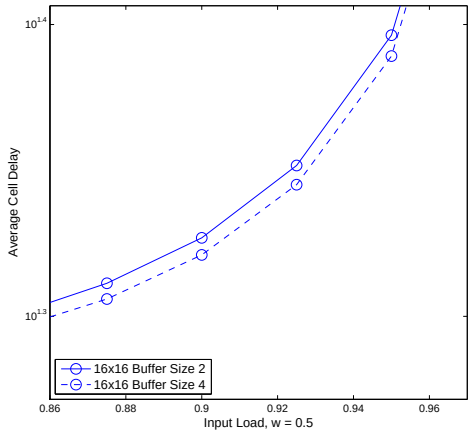
Figure 4.6: UDN varying depths for several switch sizes under Unbalanced traffic. <UDN, SS = 16|32|64, SP = 2, D = N, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.5. *Employing SP2 in UDN under Unbalanced traffic, the depth can be reduced by a factor of 5 keeping throughput performance. Above that factor, degradation is more significant for small switches.*

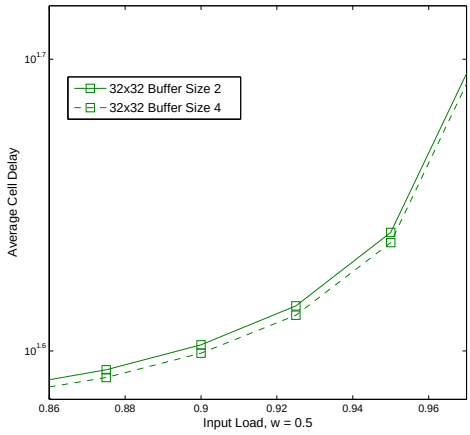
Figure 4.7 shows that reducing buffer size to 2 is not a drawback if SP2 and maximum depth (N=M) are employed. Delay is not significantly different with buffer size 2 or with buffer size 4. So a buffer size of 2 is more appropriate as the area of the switch is reduced. Buffer size of 4, slightly improves the delay, but it becomes negligible when the number of inputs in the system increases. This study is reflected in Figures 4.7(b), 4.7(c), 4.7(d)).



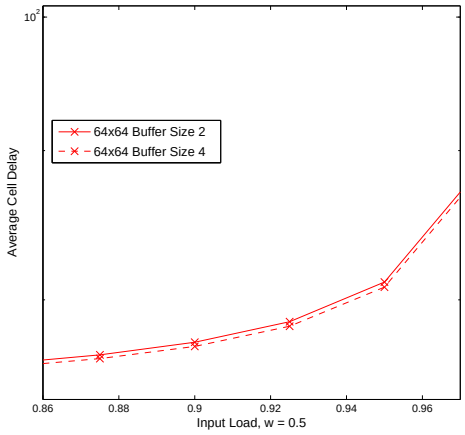
(a) Different Switch Sizes



(b) 16x16



(c) 32x32



(d) 64x64

Figure 4.7: Unbalanced Traffic, w = 0.5.
<UDN, SS = x, SP = 2, D = N BD = x, RA = Modulo, SA = FIFO:RR >

Conclusion 4.6. *Decreasing buffer size does not deteriorate performance under Unbalanced Traffic for UDN with full depth and SP>1.*

If the depth of the mesh is reduced, then buffer size plays an important role in performance. A 32x32 UDN switch is tested varying the depth and the buffer. With buffer size 2, the switch does not achieve 100% throughput if the depth is just reduced a factor of 2 (depth of 15 columns) though it is working with SP2 (see Figure 4.8). Performance deteriorates mainly for low values of w, that is, when the traffic tends to be more balanced.

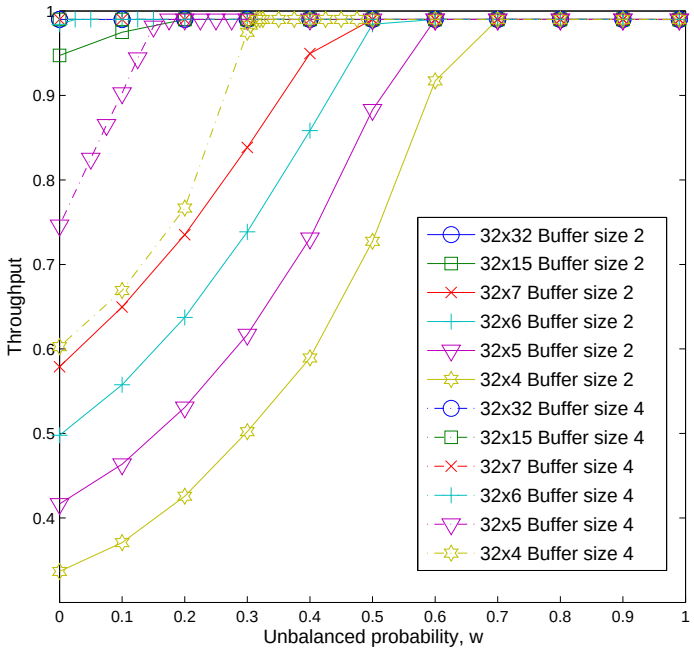


Figure 4.8: UDN Switch performance for different buffer sizes and depths and Unbalanced traffic. <UDN, SS = 32, SP = 2, D = x, BD = x, RA = Modulo, SA = FIFO:RR >

Conclusion 4.7. *Buffer size strongly impact performance when depth is reduced for Unbalanced traffic.*

Performance does not improve significantly for the UDN switch if the arbiter algorithm is modified due to the small number of ports in the routers. Multiport RAM is explained in section 2.4.2.2.

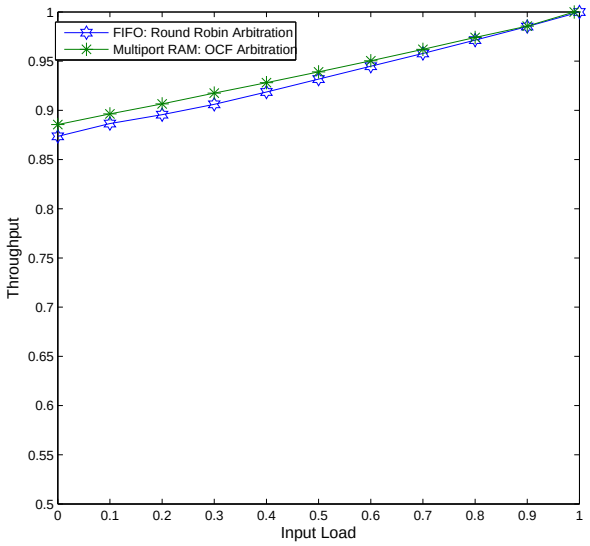


Figure 4.9: 32x32 UDN switch under Unbalanced traffic for different arbiter algorithms. <UDN, SS = 32, SP = 2, D = 32, BD = 4, RA = Modulo, SA = x >

Conclusion 4.8. *The arbiter algorithm does not have a significant impact in UDN under Unbalanced traffic.*

Table 4.2 summarizes different options to improve the response of the UDN fabric for Unbalanced traffic.

	Performance	Average Cell Delay	Area	Cost
$\uparrow w$	$\uparrow$	$\downarrow$	=	=
$\uparrow$Speedup	$\uparrow$(SP2 enough)	$\downarrow$	=	$\uparrow$
$\uparrow$Switch Size	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$
$\downarrow$Depth	$\downarrow$ if SP1	$\uparrow$	$\downarrow$	$\downarrow$
	$\simeq$ if SP2 and $\frac{N}{M} < 5$	$\downarrow$	$\downarrow$	$\downarrow$
	$\downarrow$ if SP2 and $\frac{N}{M} \geq 5$	$\uparrow$	$\downarrow$	$\downarrow$
$\downarrow$Buffer Size	$\simeq$ if SP>1 and $N = M$	$\simeq$	$\downarrow$	$\downarrow$
	else $\downarrow$	$\uparrow$	$\downarrow$	$\downarrow$

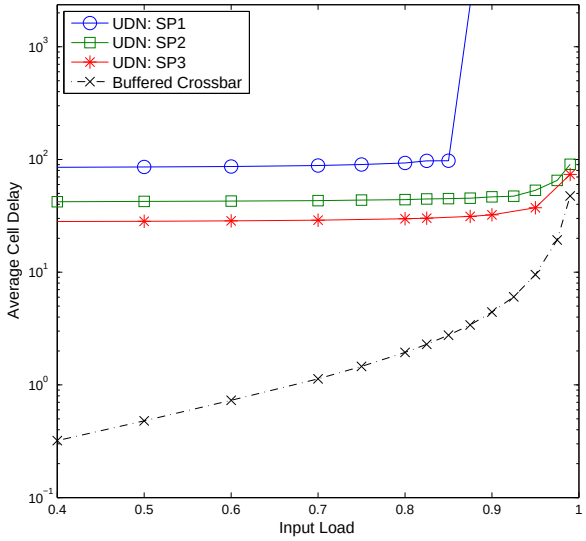
Table 4.2: UDN parameter conclusions under Unbalanced

Comparison in term of sizes is relative. A 32x32 switch has a higher number of packets per cycle than a 16x16 switch due to the higher number of ports. To make the a relative comparison we relate the throughput of both switches.

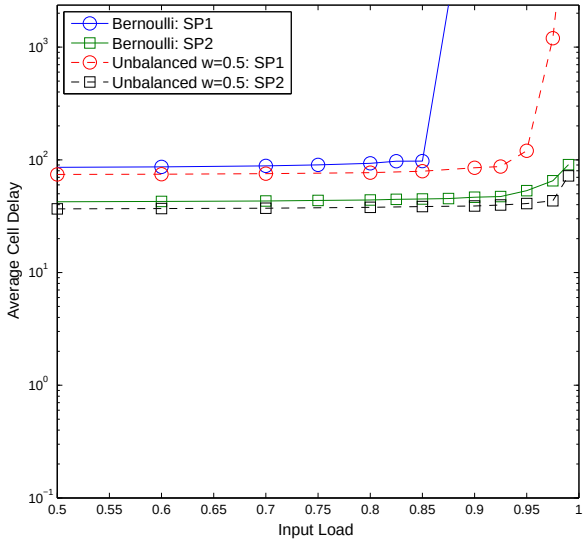
4.2.2 Bernoulli Uniform Traffic

The following study is focused in the performance of UDN architecture under Bernoulli Uniform traffic. Figure 4.4 (with $w = 0$) already depicted that this kind of traffic has a worse behavior

than the Unbalanced traffic. Figure 4.10(a) shows that increasing the speedup can lead the switch to improve its performance to manage 100% throughput. The behavior is again shown to be worse than that for Unbalanced traffic in Figure 4.10(b).



(a) UDN vs CICQ under Bernoulli Uniform Traffic

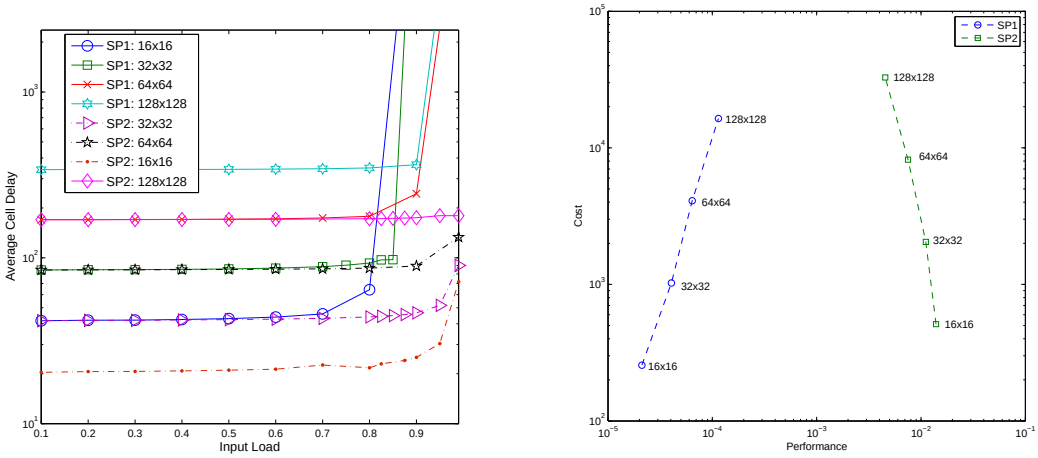


(b) Bernoulli Uniform vs Unbalanced Traffic

Figure 4.10: UDN 32x32 switch under Bernoulli Uniform Traffic for different speedups. <UDN, SS = 32, SP = x D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.9. *Bernoulli Traffic has good performance running at SP2 in UDN. Average cell delay is higher than for Unbalanced Traffic.*

According to the analytical study (see section 3.2.3), under this traffic, increasing the switch size should enhance the behavior of the network. Figure 4.11(a) shows that average cell delay becomes lower when the load is high for meshes with higher number of ports. Small switches have lower delay with SP2. Figure 4.11(b) represents the cost/performance of this conclusion.



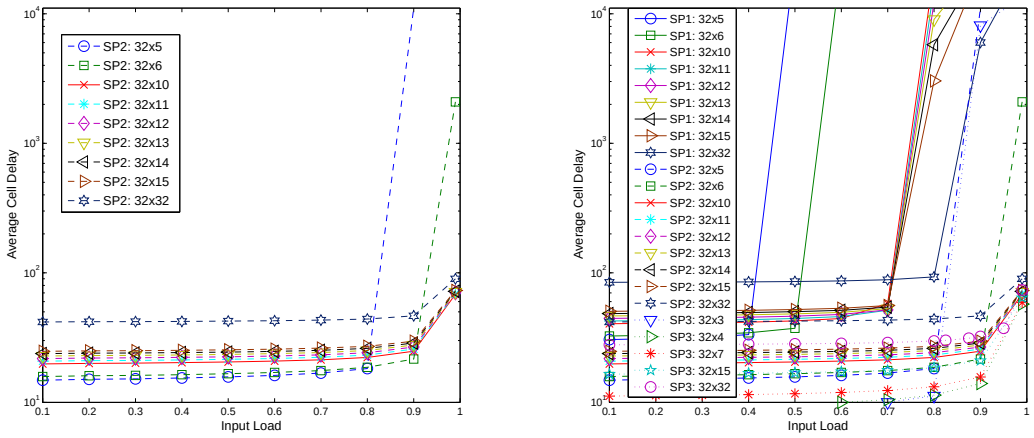
(a) Performance

(b) Cost-performance

Figure 4.11: UDN under Bernoulli Uniform Traffic for different switch sizes. <UDN, SS = x, SP = x, D = N, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.10. *UDN is scalable in performance and the switch size under Bernoulli Uniform traffic. Bigger switches reach congestion with higher loads but have higher average cell delay for light loads.*

Once we are employing SP2 and throughput is 100%, depth can be reduced to save routers. Figure 4.12(a) depicts a 32x32 switch for different depths. The value for which the average cell delay does not tend to infinite coincide with that of Figure 4.6(b) for *w=0*. This is, when a factor of 5 in number of routers reduced. SP2 has a better response than SP1 for all the ranges of depth. Increasing SP by a factor x entails reducing the average cell delay by the same factor if the network is not congested. When the system becomes highly congested, this equation does not apply. For SP2, a depth of 6 is enough for dealing with all the traffic. For SP1, they should be necessary then, just 12 columns, but in this case, not even a depth of 32 manages with all the input load. If the switch is running with SP3, then more routers can be saved, now, only 4 columns are necessary. For low input loads, it is always preferable to decrease the number of columns of the fabric. When the load is higher, two solutions are possible: either the depth should be increased, or the speedup of the system is raised.



(a) SP2 (b) SP1,SP2,SP3

Figure 4.12: UDN under Bernoulli Uniform Traffic for different depths.
<UDN, SS = 32, SP = 2|x, D = x, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.11. *Increasing the speedup allows saving routers in the mesh by reducing its depth without deteriorating performance in UDN under Bernoulli Uniform traffic.*

The different cost of each layer and speedup is represented in Figure 4.13.

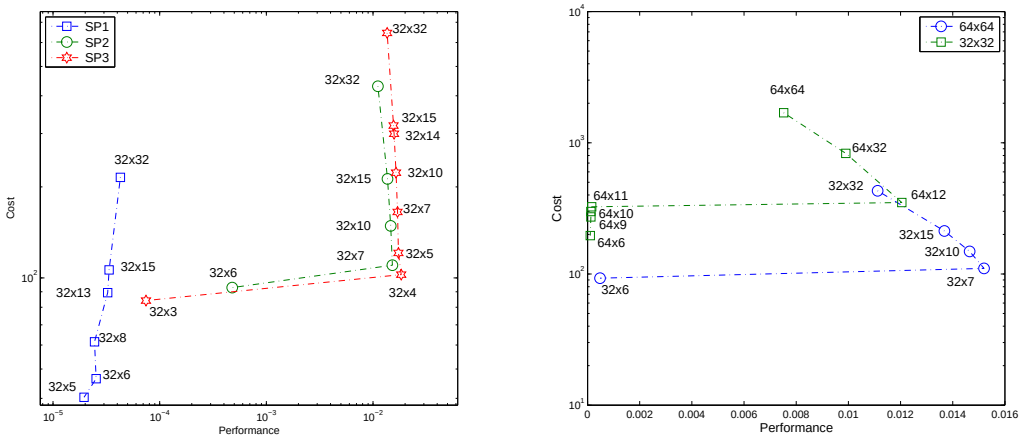
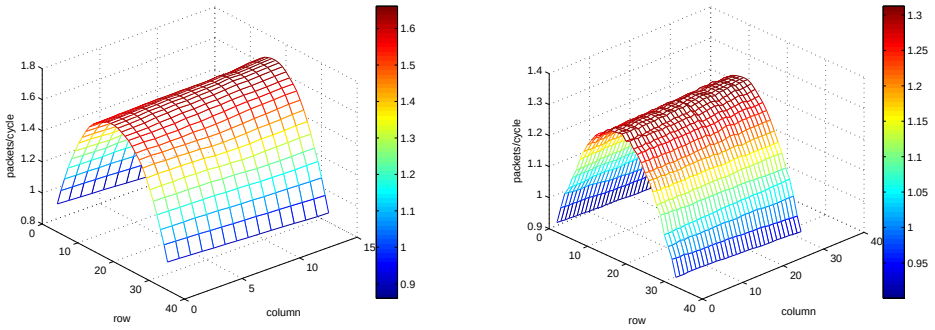


Figure 4.13: UDN 32x32 switch: Cost under Bernoulli Traffic for different depths

The following Figure 4.14, shows how the load is equally distributed along the mesh, even when the number of columns is reduced. In this way, Figure 4.14(a) with a 32x15 switch

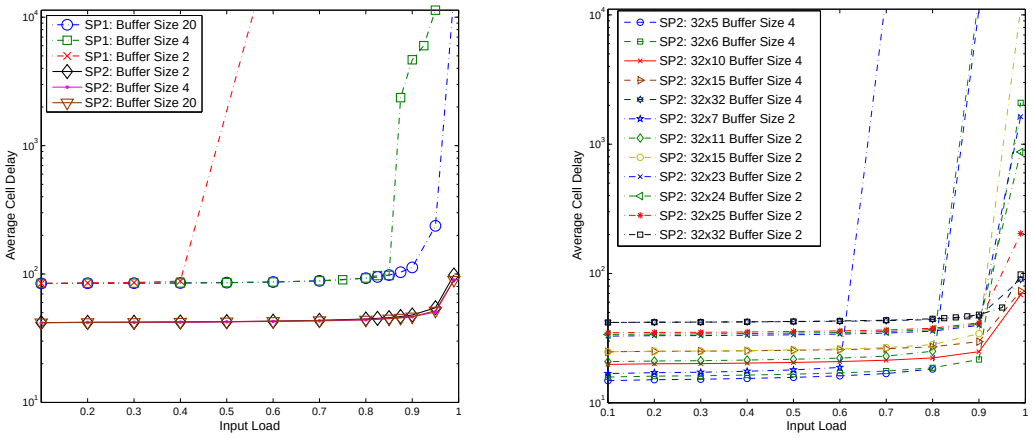
and 4.14(b), with depth 32, have the same shape for the distribution of the packets. This distribution is thanks to the Modulo algorithm, that takes into account the number of columns in the mesh to equally distribute the packets in the routers.



(a) Depth15 (b) Depth32

Figure 4.14: Router Load for Bernoulli Uniform Traffic

The following experiment varies the buffer size of each router. As it happened under Unbalanced traffic, decreasing the buffer size deteriorates performance when the SP1 or when SP2 and the depth of the mesh is reduced. These data are collected in Figure 4.15. When buffer size 4 is used, a 32x7 switch performs 100% throughput, meanwhile if buffer size 2 is used, the switch should be 32x25 to have full throughput.

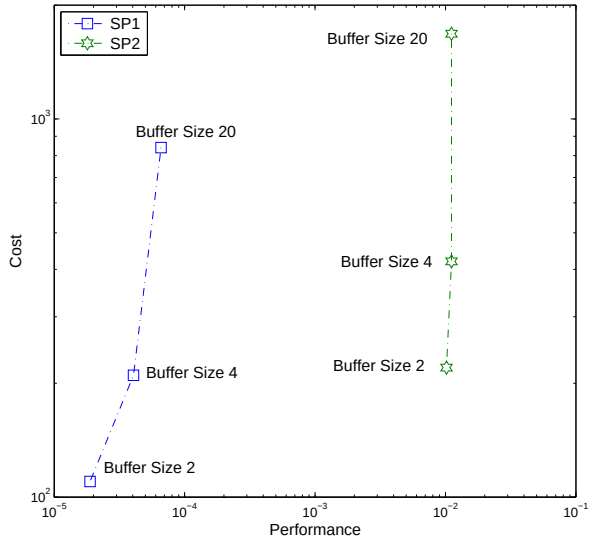


(a) Buffer Size for different speedups (b) Buffer Size for different depths and SP2

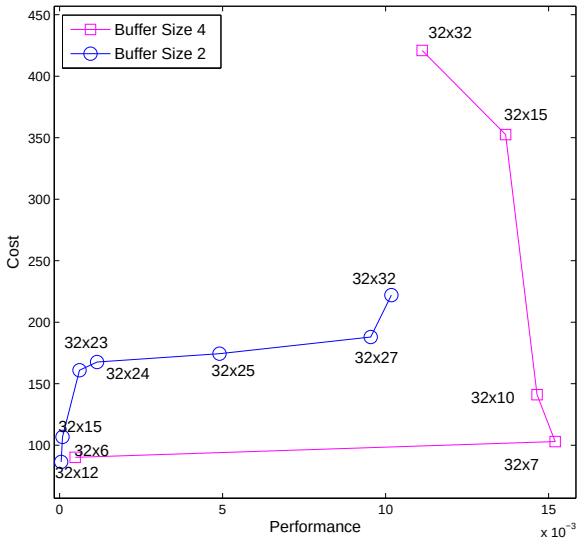
Figure 4.15: UDN 32x32 Switch under Bernoulli Uniform Traffic modifying the buffer size. <UDN, SS = 32, SP = $x|2$, D = $32|x$, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.12. *Buffer size can be reduced if depth is not reduced and SP is* ≥ 2 *without resulting in worse behavior of the system in UDN under Bernoulli Uniform traffic.*

It is better to increase speedup than buffer size. This can be seen in Figure 4.16(a). It has a better performance buffer size 2 with SP2 than buffer size 20 with SP1. Figure 4.16(b) shows the cost/performance for different buffer sizes and SP2.



(a) Speedup and Buffer size



(b) Switch depth and Buffer size

Figure 4.16: UDN 32x32 switch: Cost under Bernoulli Traffic for different depths and buffer sizes and SPs

Conclusion 4.13. *It is a better cost/performance option to increase SP than buffer depth in UDN under Bernoulli Uniform traffic.*

Modifying the scheduling algorithm might improve the performance of the system. This is tested in Figure 4.17. The ideal case is the Multiport RAM arbitration with OCF (refer to section 2.4.2.2). The current FIFO arbitration with RR manages only 0.2% less load than the ideal one and it performs better than FIFO arbitration with OCF. The number of ports of the system has no significant impact on the variance in performance of the different arbitration algorithms.

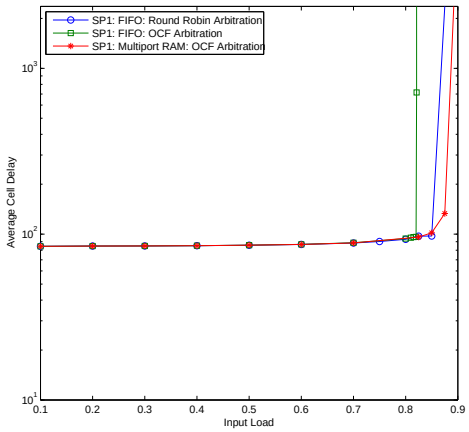


Figure 4.17: 32x32 UDN switch for different arbitration algorithms under Bernoulli Uniform traffic.

$$<\text{UDN, SS} = 32, \text{SP} = 1, \text{D} = 32, \text{BD} = 4, \text{RA} = \text{Modulo, SA} = x >$$

Conclusion 4.14. *Modifying the scheduling algorithm in UDN does not modify significantly the switch performance under Bernoulli Uniform traffic.*

The conclusions for Bernoulli Uniform traffic are the same than those for Unbalanced traffic. They are summarized in table 4.3

	Performance	Average Cell Delay	Area	Cost
↑Speedup	↑(SP2 enough)	↓	=	↑
↑Switch Size	↑	↓	↑	↑
↓Depth	↓ if SP1	↑	↓	↓
	$\simeq$ if SP2 and $\frac{N}{M} < 5$	↓	↓	↓
	↓ if SP2 and $\frac{N}{M} \geq 5$	↑	↓	↓
↓Buffer Size	$\simeq$ if SP>1 and $N = M$	$\simeq$	↓	↓
	else ↓	↑	↓	↓
FIFO OCF Arb.	↓	↑	↑	↑
Multiport RAM OCF Arb.	↑	↓	↑	↑

Table 4.3: UDN parameter conclusions under Bernoulli

4.2.3 Bursty Uniform Traffic

Figure 4.2(a) already showed that UDN has no good performance under Bursty Uniform traffic. Now we will study how burst size affects the behavior of the network. As the burst size increases, the response of the system gets worse. This is shown if Figure 4.18. Even with SP2, the throughput achieved does not reach the 100%. The mesh is loaded with 1 packet per cycle and the burst size varies from 1 to 500.

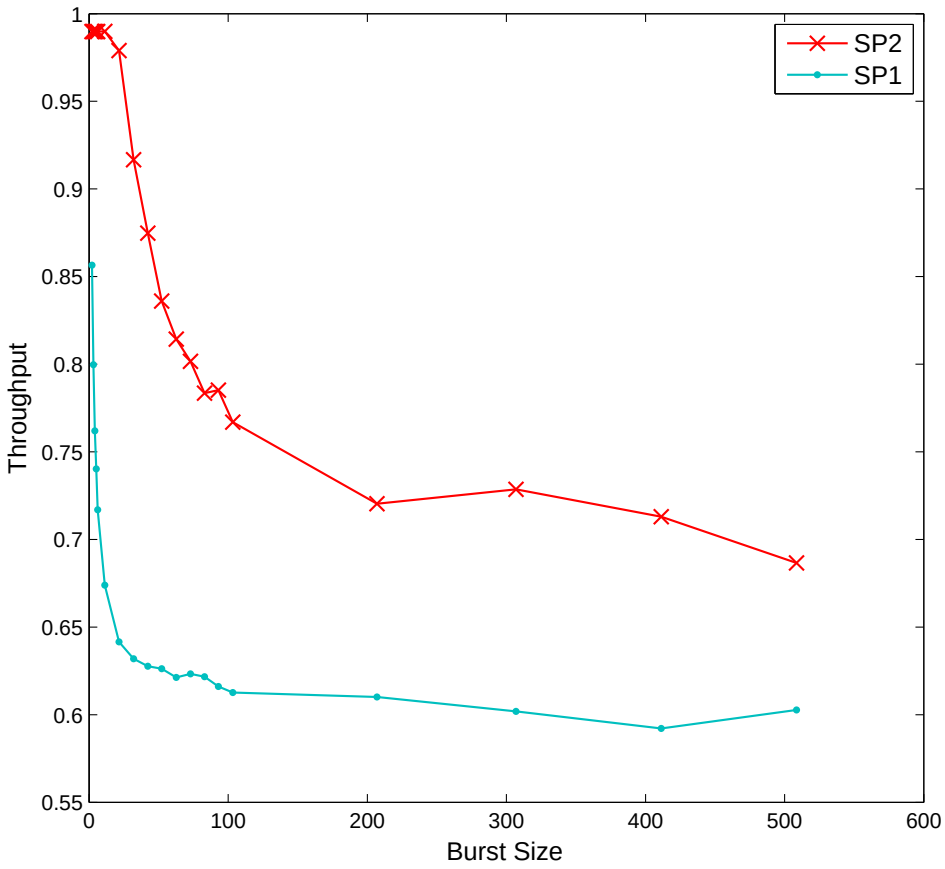


Figure 4.18: Bursty Uniform Traffic 32x32 Switch.
<UDN, SS = 32, SP = x, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.15. *UDN does not perform well under Bursty traffic with high average burst flows.*

The following experiments are done with an average burst size of 16 packets. Running at SP1, a 32x32 switch cannot manage more than $\simeq$50% of the traffic. SP2 improves the throughput to achieve the behavior of the traditional CICQ (see Figure 4.19).

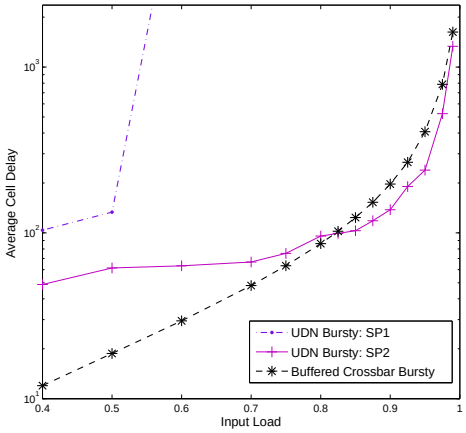
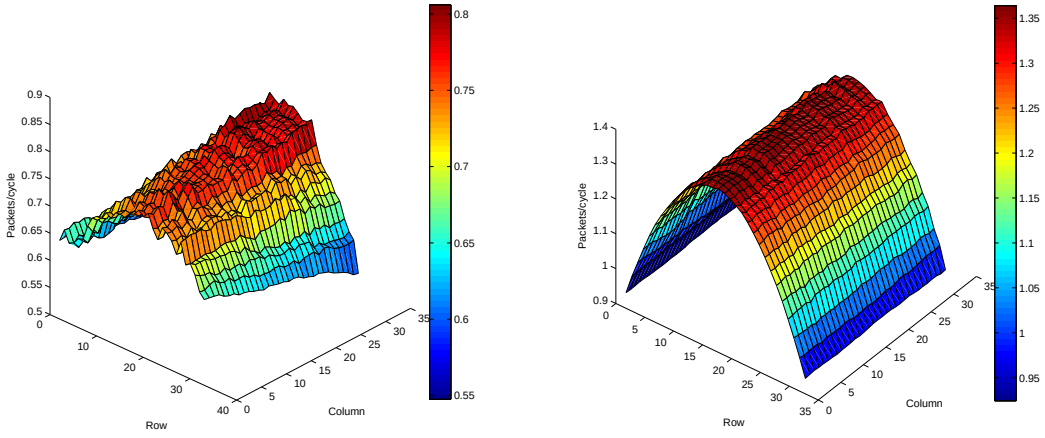


Figure 4.19: 32x32 UDN under Bursty Uniform Traffic for different speedups.
<UDN, SS = 32, SP = x, D = 32, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.16. *Burst sizes of average 16 packets with SP2 in UDN perform as CICQ. Average cell delay is high.*

The distribution of the load is similar for both SP1 and SP2. They differ on the number of packets per cycle that the fabric is able to deal with. For SP2, the load is more balanced because the system is not congested. The simulation is done with a load of 0.9 packets/cycle and a burst size of average 16 packets. This is represented in Figure 4.20.



(a) SP1 (b) SP2

Figure 4.20: Router load distribution for a 32x32 UDN Switch under Bursty traffic

Increasing the size of the switch, once again, improves the performance of the fabric. The values of average cell delay for different switch sizes is represented in Figure 4.21 for SP1.

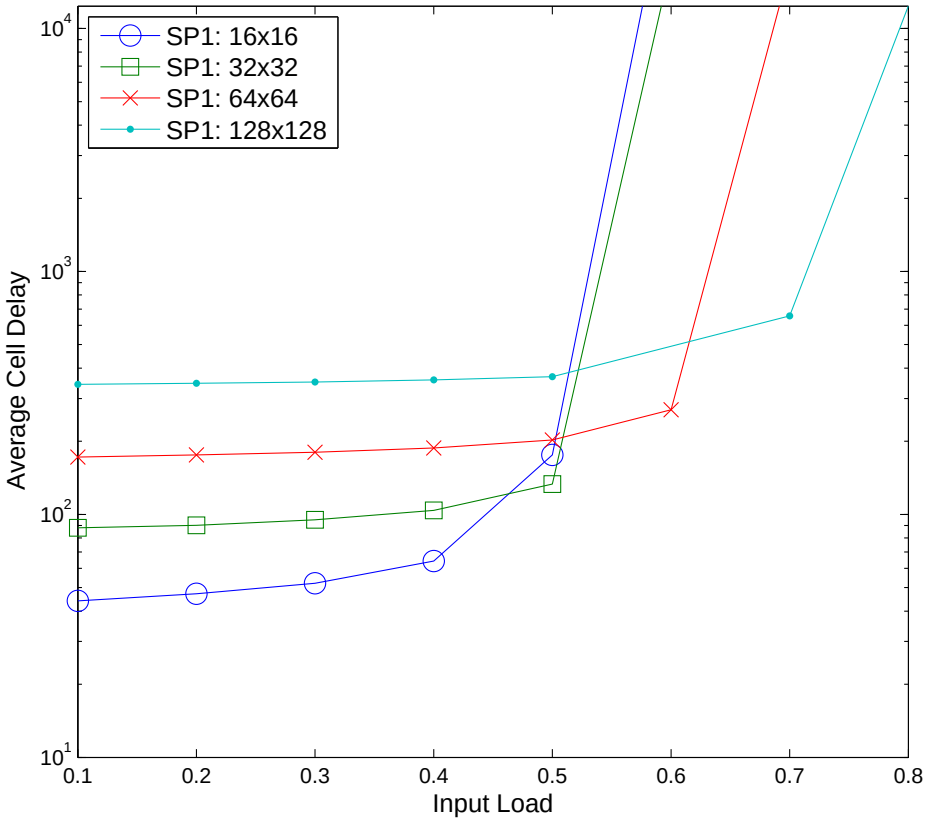


Figure 4.21: UDN under Bursty Uniform Traffic for different switch sizes.
<UDN, SS = x, SP = 1, D = N, BD = 4, RA = Modulo, SA = FIFO:RR >

Conclusion 4.17. *Increasing the size of the switch improves the performance of the system with bursty traffic in UDN.*

In this case, reducing the depth of the mesh when working at SP2, will just decrease the behavior of the system. The UDN switch already becomes congested with a NxN switch for high loads. Another solution to improve the performance is increasing the buffer size. As it is said in section 3.1.1 though buffer size does not help avoid congestion, it does help absorb burstiness. Figure 4.22 shows how a buffer size of 20 significantly outperforms the result of the simulation of buffer size 4 for SP1. The cost of this improvement is, however, very high. Running the fabric at SP2, gets a better response for buffer size 4 than for buffer size 20 with SP1.

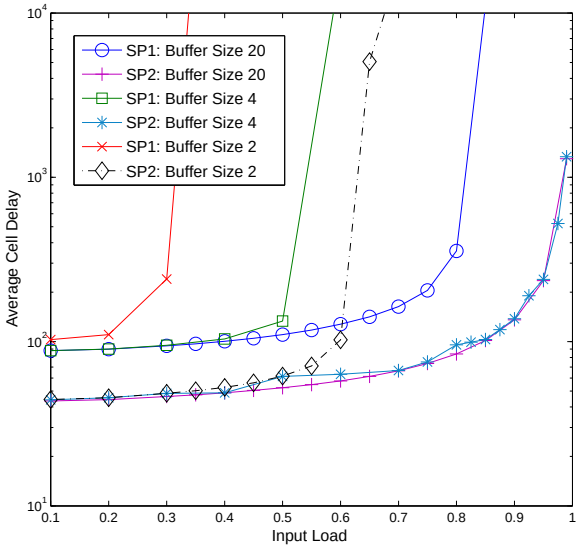
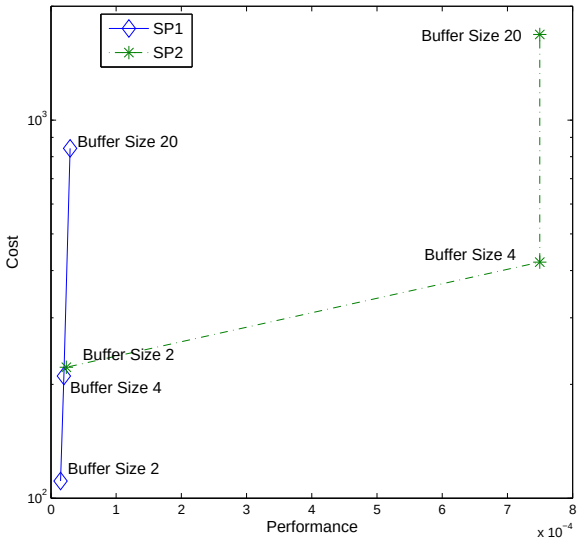


Figure 4.22: Bursty Uniform traffic in a 32x32 UDN for different buffer sizes. <UDN, SS = 32, SP = x, D = 32, BD = x, RA = Modulo, SA = FIFO:RR >

Conclusion 4.18. *Increasing speedup is a better choice than increasing the buffer size.*

Figure 4.2.3 shows this conclusion in terms of cost/performance.



Varying the arbiter algorithm affects the systems as is shown in Figure 4.23. The ideal case of Multiport RAM with OCF has only 0.05% more throughput than the current arbitration with FIFO and OCF .

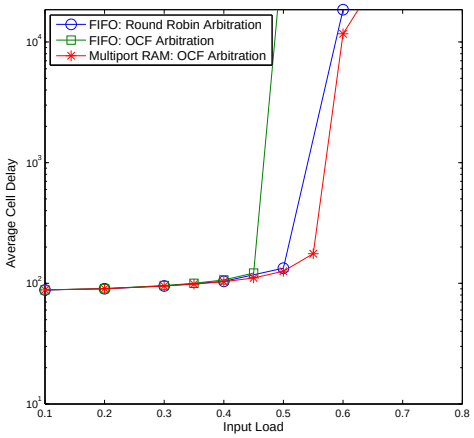


Figure 4.23: 32x32 UDN switch under Bursty uniform traffic for different arbitration algorithms. <UDN, SS = 32, SP = 2, D = 32, BD = 4, RA = Modulo, SA = x >

Conclusion 4.19. *Modifying the scheduling algorithm does not modify significantly the switch performance under Bursty Uniform traffic.*

	Performance	Average Cell Delay	Area	Cost
↑Speedup	↑(SP2 enough)	↓	=	↑
↑Switch Size	↑	↓	↑	↑
↓Depth	↓	↑	↓	↓
↓Buffer Size	≃ if SP>1 and Depthmax else ↓	≃ ↑	↓ ↓	↓ ↓
FIFO OCF Arb. Multiport RAM OCF Arb.	↓ ↑	↑ ↓	↑ ↑	↑ ↑

Table 4.4: UDN parameter conclusions under Bursty Uniform Traffic

4.3 Conclusions

This chapter gathers the system simulations of UDN architecture. First, its performance was first compared to CICQ. Finally, its architectural parameters were modified to test the different responses of the system.

In the introduction of this thesis, we argued our reasons to propose this new NoC architecture as a crossbar fabric. Those claims where the following:

- Better load balancing
- Higher path diversity
- Scalability in port count and speed per port
- Use of short wires
- Simpler switch design by using FIFOs

The aim of the performance evaluation of the new architecture is to conclude if those claims are satisfied and if our architecture outperforms the CICQ crossbar switch. Summarizing this chapter we can conclude the following characteristics of the UDN architecture:

- Better load balanced is achieved with the Modulo Algorithm 1 and the multi-hop structure of the mesh. This is depicted in Figures like 3.19 and 3.20 and in Conclusion 3.3.
- Path diversity is enforced by varying the path of each flow. Modulo Algorithm 1 can modify the path per flow reaching a higher load balance (see Figure 3.20).
- It was proved analytically and by simulations that increasing the switch size does not deteriorate the performance of the system (see Figures 4.11(a), 4.21) and that delay is increased linearly with the size of the switch (Conclusions 4.10, 4.17). Short wires connect the routers and they are not lengthened with the switch size.
- Analysis study of section 3.2.3 and Figure 3.15 shows how HoL blocking is not an issue in UDN mesh and the routing algorithm employed so FIFOs can be used avoiding to use VOQ both in the line cards and in the routers (see Conclusion 3.1). Section 3.3 refers to the size of these FIFOs and the saving in area.

Simulations showed how UDN outperforms CICQ if is employing SP2 for Unbalanced traffics. For Uniform traffics, like Bernoulli UDN performs better when for heavy loads (>90%) and with SP2. Bernoulli Uniform traffic has higher delays in the UDN architecture due to the multi-hop routing. We can conclude after this experiments, that UDN is a feasible architecture that outperforms CICQ in performance and delay. Its drawback is the physical distribution of its inputs and outputs, than make it unsuitable for the chip layout. NIs of UDN are disposed in the West and East side of the architecture and the pins of a chip are placed all around its perimeter. Three solutions are then possible, using bigger chips, using longer wires to connect the NIs or to modify the UDN architecture.

Table 4.5 summarizes the concepts to achieve the best cost/performance with the UDN architecture.

	Performance	Average Cell Delay	Area	Cost
↑Speedup	↑(SP2 enough)	↓	=	↑
↑Switch Size	↑	↓	↑	↑
↓Depth (not for Bursty)	↓ if SP1 $\simeq$ if SP2 and $\frac{N}{M} < 5$ ↓ if SP2 and $\frac{N}{M} \geq 5$	↑ ↓ ↑	↓ ↓ ↓	↓ ↓ ↓
↓Buffer Size	$\simeq$ if SP>1 and N=M else ↓	$\simeq$ ↑	↓ ↓	↓ ↓
FIFO OCF Arb. Multiport RAM OCF Arb.	↓ ↑	↑ ↓	↑ ↑	↑ ↑

Table 4.5: UDN parameter conclusions

Gathering conclusions 4.12,4.13, 4.14, 4.18 it is shown that the importance of the parameters is as follows:

1. Speedup.
2. Depth.
3. Buffer size.
4. Scheduling algorithm.

Short wires are used to interconnect the routers and the NIs. The main drawback of UDN is the way the NIs should be connected to the pins of the chip. This is shown in Figure 4.24. The layout of the chip has its pins all around its perimeter meanwhile UDN architecture has them only on the sides. Either a bigger chip is used, or longer wires should be used to connect the switch to the chip. Though UDN is a logically nice architecture, it is not physically appropriate. For this reason, we propose a second architecture in chapter 5 in which these issues are solved.

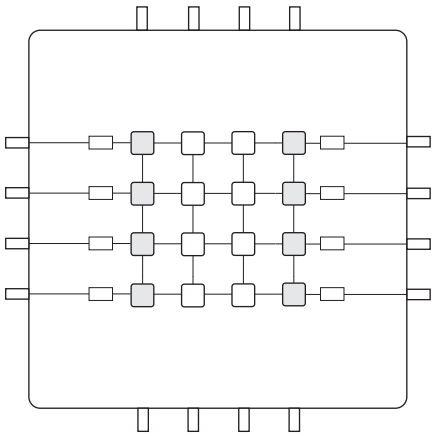


Figure 4.24: Placement of 4x4 UDN in a chip.

Multidirectional NoC

5

Chapter 3 presented a new NoC architecture for a crossbar switch fabric. Alongside all the advantages of this architecture, there exist several drawbacks related to the pin distribution that should be taken into account. As it was tested, one of the advantages of the UDN architecture is the use of short wires. This allows scalability in the port count and speed per port and enables reliable high-speed signaling.

UDN fabric does not have the inputs/outputs all around the perimeter. Then, when connecting them to the pins of the Chip, long wires should be used. To be able to wrap the inputs/outputs around the mesh, the connections should be twisted. With several foldings and unfoldings, we get to the next result.

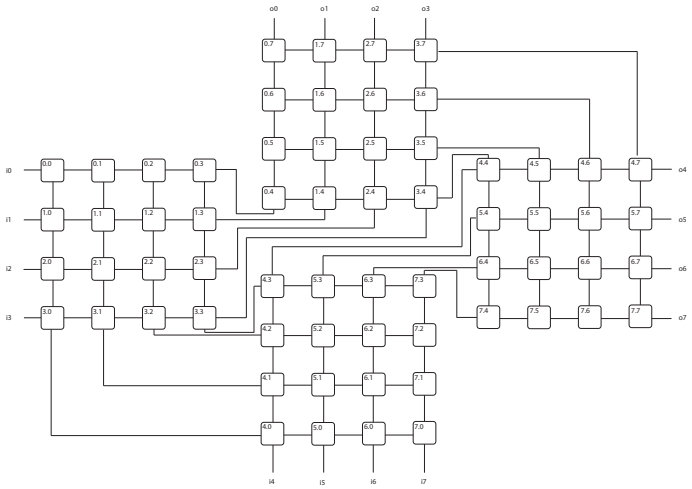


Figure 5.1: 8x8 UDN switch twisted

Now, all the links do not have the same length, augmenting complexity and reducing the scalability of the chip. To overcome this problem, his chapter introduces a new NoC architecture in line with the pin distribution inside the chip. This is depicted in Figure 5.2 for a 4x4 twisted UDN switch.

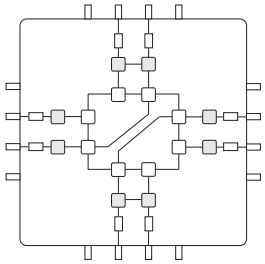
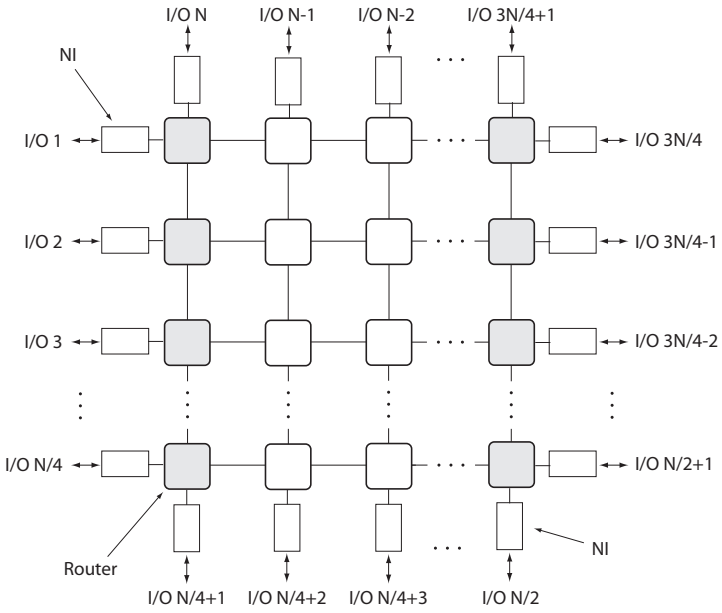


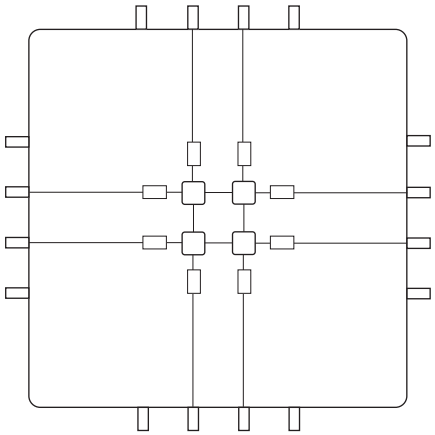
Figure 5.2: Placement of 4x4 UDN twisted in a chip.

5.1 MDN Architecture

The Multi-directional NoC (MDN) crossbar architecture, shown in Figure 5.3(a), takes advantage of the fact that I/O pads (and NIs) are placed on the perimeter of the chip layout, and not on two sides as in the UDN architecture. The mesh is now $(N/4) \times (N/4)$, with packets traveling in all directions.



(a) MDN



(b) 4x4 MDN placement

Figure 5.3: The Multi-directional NoC (MDN) crossbar architecture.

The placement of MDN in a chip is shown in Figure 5.3(b). Now, the distribution of the NIs is according to the layour of the pins in a chip.

To make the architecture scalable, we allow multiple (P) planes (also called layers) that are vertically connected only at the edges (i.e. i or j is 0 or $N/4 - 1$). Routers then have degree

4, except the borders in planes 1 and $P-2$, where it is 6. In planes 0 and $P-1$ the routers of the borders have degree 3. NIs are placed centrally, in plane $\lfloor N/2 \rfloor$ (this will be referred as P_m). Figure 5.4 shows an example of the MDN architecture with 4 planes and N=4.

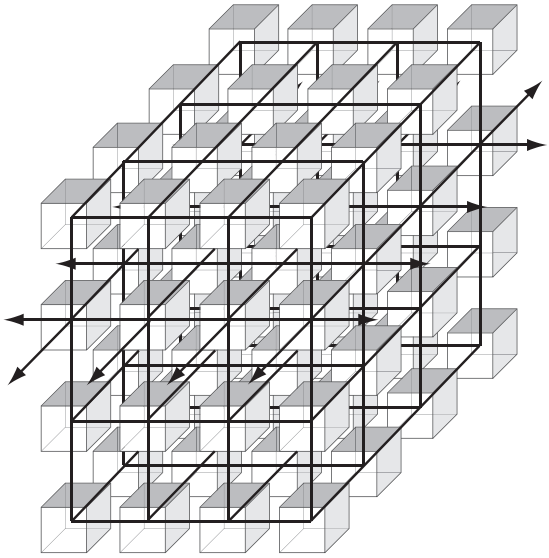


Figure 5.4: Example of the MDN crossbar architecture with multiple planes.

In this chapter, when we refer to a NxN switch it is a MDN switch with N inputs and N outputs and 1 plane. When more planes are used this architecture it is referred as NxNxP.

5.1.1 Architectural Design

5.1.2 NI design

All the NIs of the MDN architecture have the same design. Unlike UDN, they all act now as input and output interfaces (see Figure 5.5)

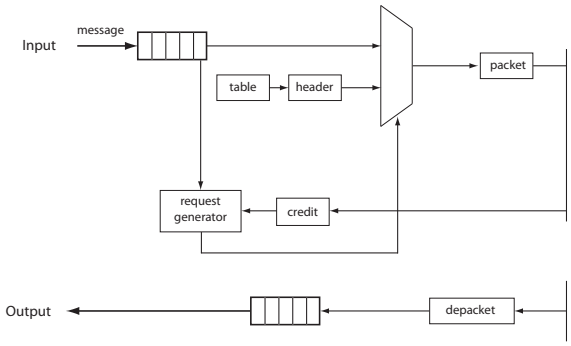


Figure 5.5: MDN NI

5.1.3 Router design

The main design of MDN routers is based on UDN architecture. As in UDN, packet switching is implemented with buffering credit based flow control and store and forward for packets delivery. Input queuing is the buffering architecture and Round Robin is the arbitration algorithm. It differs now from UDN because packets flow in all directions and deadlock can occur. We avoid this by using two virtual channels (VC) at North and South inputs of the router. Routers at the East and West edges of the mesh additionally have the VCs at their East and West inputs, respectively. The use of VCs entails more pins for the flow control. Packets use VC 1 if their destination is East of their starting position, and VC 2 otherwise. West and East links only requires VC 1 and VC 2 respectively. In this way, the network is virtually divided in two UDN networks and deadlock cannot occur. Figure 5.6 is an example of the VC distribution in a 4x4 mesh.

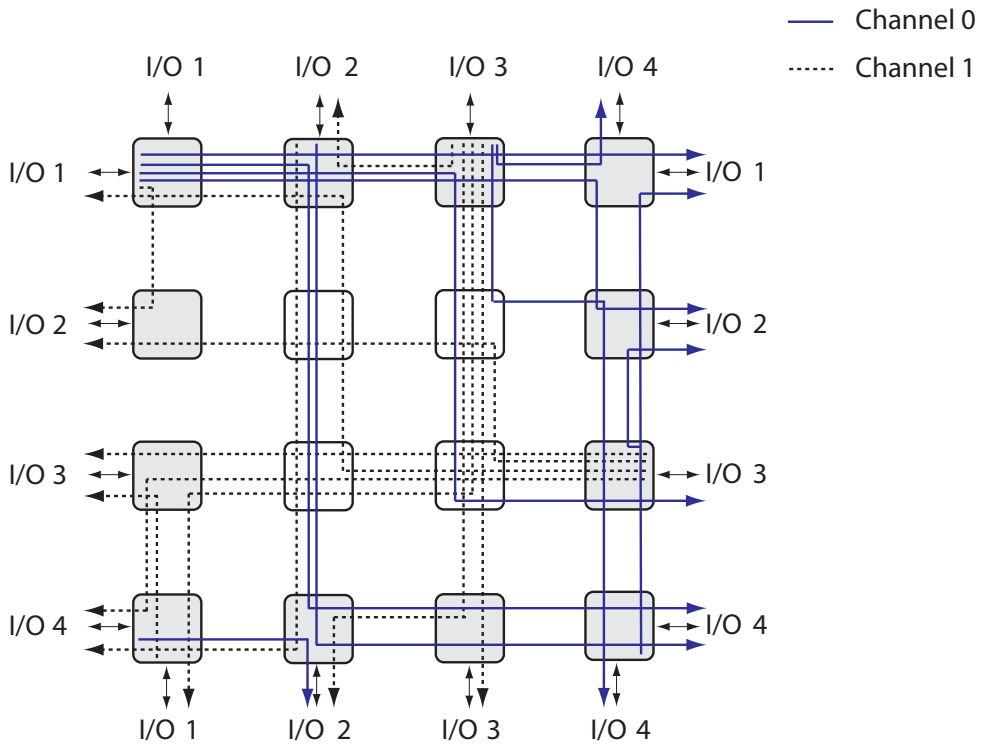
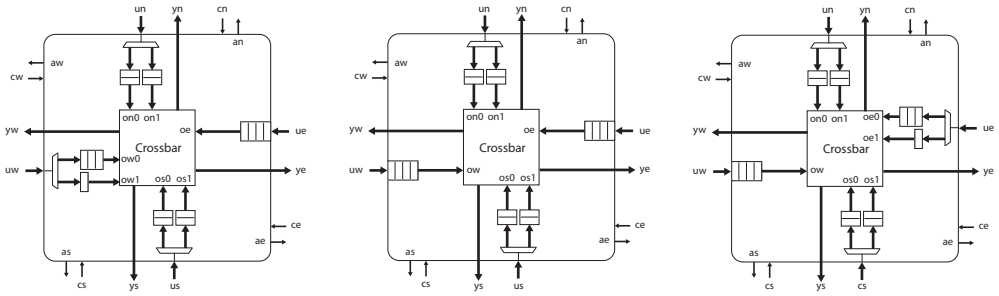


Figure 5.6: 4x4 MDN with VC paths

Note that the total amount of buffering is equal to that of the UDN router. West and East routers have asymmetrical buffer sizes in their virtual channels. In this way a better balance of the load is achieved as the probability of a packet to belong to channel 1 is different to the probability of belonging to channel 2 in those routers for Uniform traffic. Then West routers have $\frac{2}{3}$ of the buffer depth for VC0 and $\frac{1}{3}$ for VC1. East routers $\frac{1}{3}$ for VC0 and $\frac{2}{3}$ for VC1. In the following figures we compared that proposal with the symmetrical proposal, that is when all the VC have the same depth.

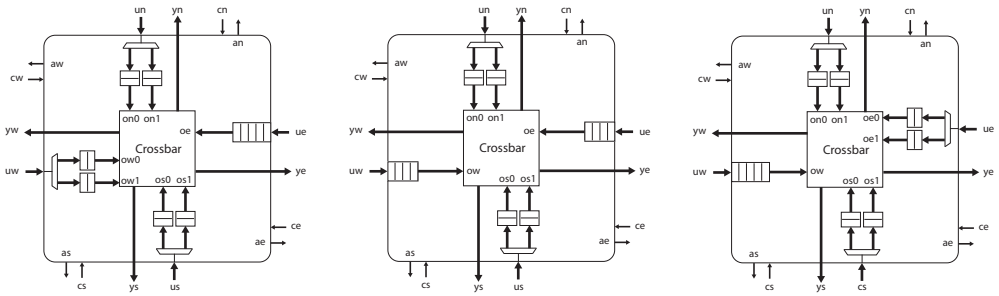


(a) West (b) Center, North, South (c) East

Figure 5.7: The MDN router architectures for P=1. Asymmetrical proposal.

Where: u =[un us ue uw] means input; y=[yn ys ye yw] means output of the router; o=[on os oe ow] means output of the input ports; a=[an as ae aw] means pin flow control of the input ports; c=[cn cs ce cw] means pin flow control of the neighbors.

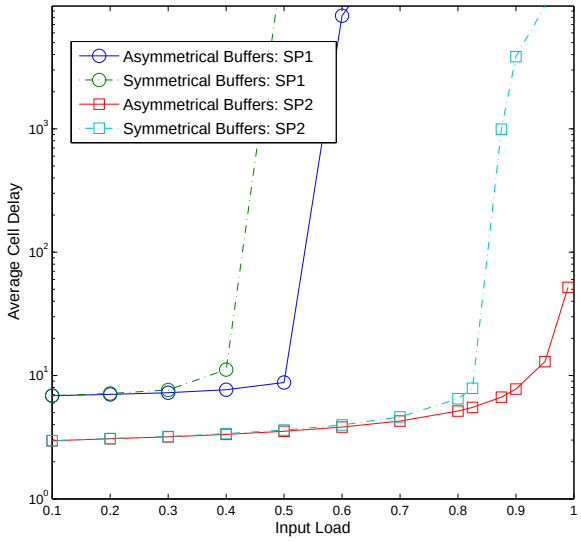
If the switch only has 1 plane, the architecture of the routers is shown in Figure 5.7. This Figure represents the asymmetrical proposal for MDN routers. Simulations in Figure 5.9 show that this option, is better than the symmetrical one (see Figure 5.9) when the fabric is employing SP2. For SP1, and for unbalanced traffic, performance is worse with the asymmetrical buffers because of the pipeline stall caused by the handshake of the credit based flow control when buffer size is 1 (refer to Figure 3.4).



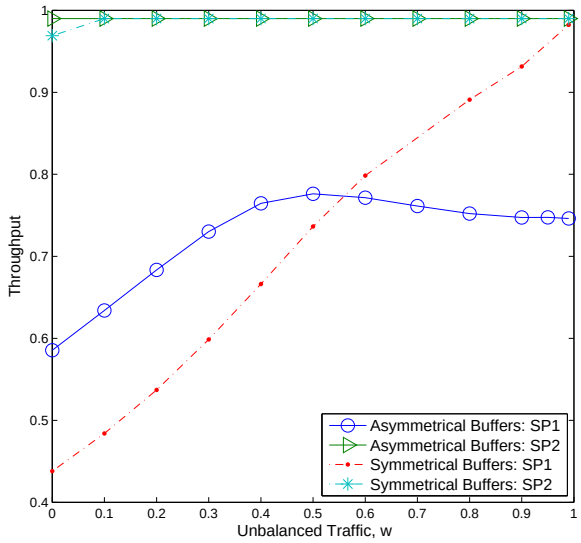
(a) West (b) Center, North, South (c) East

Figure 5.8: The MDN router architectures for P=1. Symmetrical proposal.

We are going to implement the asymmetrical proposal since one of our claims is that with NoC we can have higher clock frequencies and SP2 is achieved.



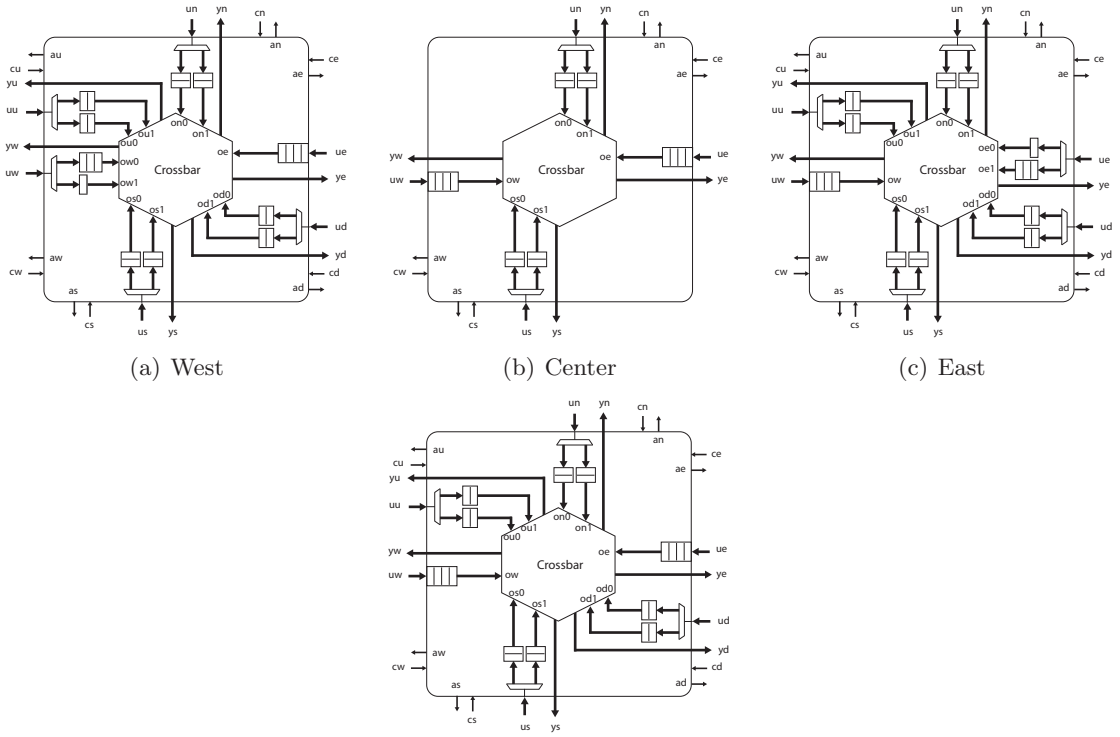
(a) Bernoulli Uniform Traffic



(b) Unbalanced Traffic

Figure 5.9: Study of the asymmetry of buffers in MDN

When there is more than one plane constituting the mesh, the routers at the perimeter have a higher degree. Those routers belonging to the outer layers have degree 3×3 and 4×4. The routers of the other layers have degree 4×4 and 5×5.



(a) West (b) Center (c) East

(d) North, South

Figure 5.10: The MDN Cube router architectures

Table 5.1 summarizes the characteristics of the MDN architecture.

Architectural component	MDN
Switching Architecture	Packet Switching
Buffering Architecture	Input queuing
Access to the Arbiter	FIFO
Arbiter Algorithm	RR
Flow Control	Credit-Based
Switching mode	Store and Forward
Deadlock Avoidance	VCs

Table 5.1: MDN architectural components

5.2 Routing in MDN

In MDN we implement an algorithm based on the Modulo algorithm 1 of UDN. Now the I/O are connected in a different way so some modifications should be applied to that algorithm.

First, the routing algorithm inside each horizontal plane of MDN is explained. Then we study how the load is distributed along the different planes.

Inside each plane, this architecture uses the Modulo Algorithm 1 for the packets which input port is physically in front of the output port (i.e. North to South, West to East) and the XY

algorithm for the packets whose origin is perpendicular to the exit. Each router checks the input port of the packet, then based also on its desired output port, the path is elected. Below there is an example of how the router performs this election. The complete code can be found in appendix B.

Router in coordinate $< i, j >$ receives a packet whose destination is in coordinate $< x, y >$ in the mesh.

Algorithm 2 MDN Modulo and XY for each Stage

```
Switch(Packet Input Port)
    case(North):
        Switch(Packet Output Port)
            case(North):
                Switch(Packet Buffer Input):
                    case(North):
                        if(j == y) then North
                        if(j < y) then East
                        if(j > y) then West
                    case(East):
                        if(j == y) then North
                        if(j > y) then West
                    case(West):
                        if(j == y) then North
                        if(j < y) then East
            case(South):
                Switch(Packet Buffer Input)
                    case(West):
                        if(j == y) then South
                        else East
                    case(East):
                        if(j == y) then South
                        else West
                    case(North):
                        if(j == y) then South
                        else if (((y)%M) == (N-i+j+t)%M) then East/West
                        else then South
            case(East):
                Switch(Packet Buffer Input)
                    case(West):
                        East
                    case(North):
                        if (x == i) then East
                        else South
            case(West):
                Switch(Packet Buffer Input)
                    case(East):
                        West
                    case(North):
                        if (x == i) then East
                        else South
```

The algorithm shows how depending on the position of the input port with respect to the output port, the path is computed differently. How the router decides to what plane send the packet is explained in the next algorithm.

When a packet enters the NoC it is first decided through which plane is going to be routed. Packets enter through the plane that is in the middle of the switch. Depending on its desired output they will be distributed to the upper or the lower layers using a modulo algorithm .

Algorithm 3 Modulo MDN

Switch(packet condition):
- case(new packet ‖ packet going to its destined plane):
  - case(output even):
    - case(# planes even):
      - if $((N-j+P+t)\%P_m)+P_m = (output\%P_m)+P_m$ then algorithm 2
      - else Up
    - case(# planes odd):
      - if $((N-j+P+t)(\%P_m+1))+P_m = (output(\%P_m+1)+P_m$ then algorithm 2
      - else Up
  - case(output odd):
    - if $(N-j+P+t)\%(P_m+1) = (output)\%(P_m+1)$ then algorithm 2
    - else Down
- case(packet inside a plane):
  - if (!edge of the plane) algorithm 2
  - else
    - if($P < P_m$) then Up
    - if($P > P_m$) then Down
    - if($P = P_m$) then algorithm 2
- case(packet going back to the center plane):
  - if($P < P_m$) then Up
  - if($P > P_m$) then Down
  - if($P = P_m$) then algorithm 2

MDN packet

Like in UDN architecture, Modulo Algorithm 3 only need $\lceil {}^2log\ M \rceil$ bits if *t=0* in the routing algorithm. Otherwise, $2\times \lceil {}^2log\ M \rceil$ bits are needed. Those bits contain the information of the destination port and the value of the t parameter. Figure 5.11 shows the MDN packet for *t=0*.

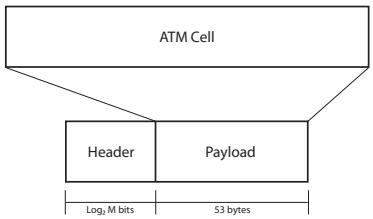


Figure 5.11: MDN packet for *t=0*.

5.3 Hardware implementations

The area of MDN is not synthetised, so we used the values of UDN from table 3.3. Like in UDN cost/performance graphs, we estimate buffer depth 20 as 4 times buffer depth 4 and buffer

depth 6 as 1.5 times buffer depth 4. Now the formula to calculate the switch size is different: *3x3 Router*$(N-4)$ + *4x4 Router* $\left(\frac{N}{4}-2)\right)^2 + NI * N$.

Table 5.2 shows some values for different switch sizes.

Switch size	16	32	64
Register FIFOs	mm^2	mm^2	mm^2
Buffer size 1	1.5040	4.8320	17.1520
Buffer size 2	4.9120	15.8560	56.4640
Buffer size 4	10.1560	32.3320	113.9320
Dedicated HW FIFOs	mm^2	mm^2	mm^2
Buffer size 1	0.624	2.336	9.216
Buffer size 2	1.776	6.832	27.408
Buffer size 4	3.820	13.804	53.452

Table 5.2: MDN area for different switch sizes.

5.4 Conclusions

This chapter proposed a new NoC architecture for the crossbar switch fabric. It solves the disadvantages of the proposal architecture of chapter 3. Those disadvantages were the position of the NI in relation to the pins of the chip. Log wires had to be used to interconnect them or bigger chips should be employed. The architectural components of MDN are designed based in the previous architecture plus some new features to take into account like deadlock. Virtual channels were added to solve this issue. The routing algorithm proposed is based in Modulo algorithm.

6 MDN System Analysis

This chapter presents the performance analysis of MDN for unicast traffic. It compares this new architecture with the traditional CICQ and with the previous UDN. The experiments are carried varying different parameters of the mesh and under different traffic conditions. Like in the UDN simulation analysis, Bernoulli Uniform traffic, Bursty Uniform traffic, Double diagonal traffic and Unbalanced traffic are tested. The simulation environment is also the same as the one explained in appendix A. For the simulations the architecture of asymmetrical buffers are used.

Some cost-performance study is done for this architecture. Where $Cost = SP * area$ and $Performance = 1/delay$.

In each figure, a vector represents the value of each parameter: switch size (SS), speedup (SP), number of planes (P), buffer depth (BD), routing algorithm (RA), and scheduling algorithm (SA). When a x is represented, it means that parameter has several values in the graph. When there are two or more subfigures, a | separates the values for each figure if they are different. The first parameter represents what architecture is simulated. We show an example of this vector:

<MDN, SS = 32, SP = 2, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

6.1 Comparison with the traditional CICQ crossbar

First we compare the base MDN architecture with CICQ. We simulate a 32x32x1 switch with buffer size 4.

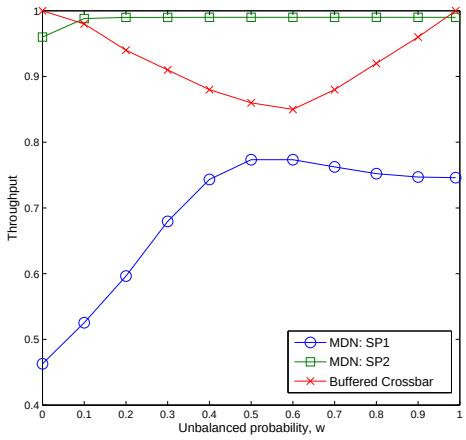


Figure 6.1: Performance of a 32x32 MDN and CICQ switch under Unbalanced traffic.
$< x$, SS = 32, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Figure 6.1 compares the current CICQ with MDN in terms of throughput. Only for SP1, and due to the asymmetrical use of the buffers of the VC, CICQ outperforms MDN. This is due to the delay caused by the handshake of the flow control. The buffer depth is 4 in total, so VCs of West and East sides of the switch have on 1 packet per buffer. For SP2, our new architecture performs better for all the ranges of w. This new fabric, like UDN, outperforms CICQ for unbalanced patterns of traffic.

For uniform traffic (Figure 6.2), CICQ outperforms MDN both for Bursty Uniform and Bernoulli Uniform traffic . Though the MDN switch runs at SP2, CICQ has lower average cell delay. Figure 6.3 depicts the average cell delay for the MDN architecture and the traditional CICQ. MDN performs better with Unbalanced traffics. For heavy unbalanced loads (see Figure 6.3(b)) and double diagonal traffic (Figure 6.3(b)) MDN outperforms CICQ when it is employing SP2. For light loads, due to the multi-hop delay, our proposed architecture is defeated by the CICQ.

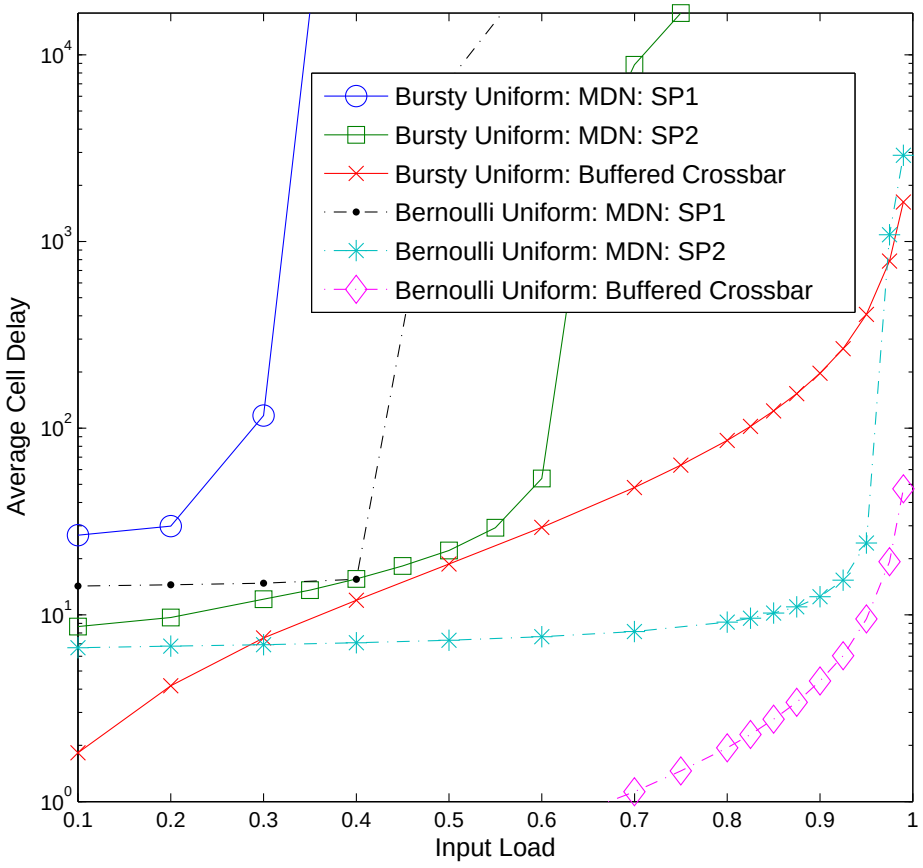
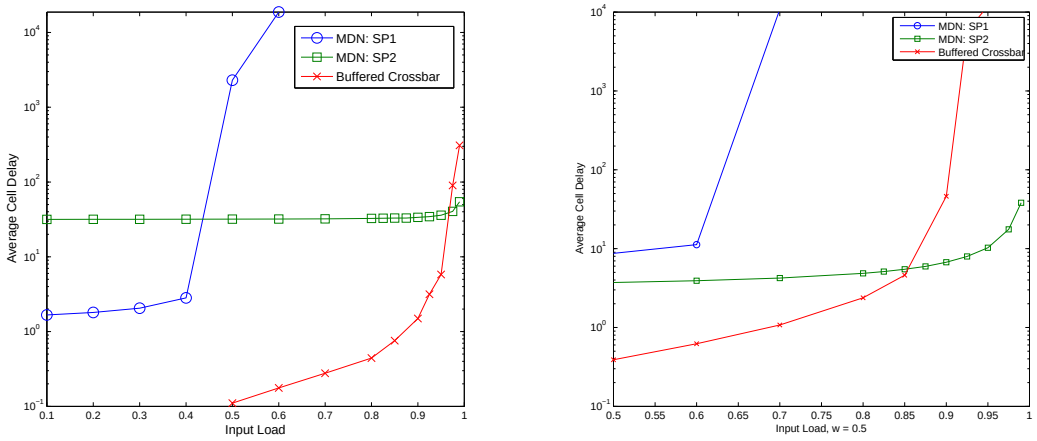


Figure 6.2: Cell delay comparison between the MDN and CICQ switch of size 32x32 under Uniform traffic.

$< x$, SS = 32, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >



(a) Double Diagonal Traffic

(b) Unbalanced Traffic $w = 0.5$

Figure 6.3: Cell delay comparison between the MDN and a CICQ switch of size 32 x 32 for non-Uniform traffic.

$< x$, SS = 32, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

The results for the MDN can be improved if some of the parameters of the architecture are modified. This research is done in the next section.

6.2 Parameter study

In this section, we will study the response of the MDN architecture for different traffic patterns. In the same way we did for the UDN architecture, we will adjust the switch size, speedup, number of planes and buffer depth of the system to enhance its response to heavy loads. Table 6.1 summarizes the different experiments for each type of traffic. Each crosspoint of the table means that both parameters are varied and studied together.

Parameter	Switch Size	Speedup	# Planes	Buffer depth
Switch Size	Unbalanced Bernoulli	Unbalanced	Unbalanced Bernoulli Bursty	Bernoulli Bursty
Speedup	Unbalanced	Unbalanced Bernoulli Bursty		
# Planes	Unbalanced Bernoulli Bursty	Unbalanced	Unbalanced Bursty	 Bursty
Buffer depth	Bernoulli Bursty		 Bursty	Bernoulli Bursty

Table 6.1: Study of parameters for each type of traffic in MDN.

6.2.1 Unbalanced Traffic

First, we test how the system reacts to unbalanced traffic and how this behavior can be altered by modifying the architectural parameters.

Figure 6.4 shows how like in UDN, Unbalanced traffic helps to improve the performance of the switch. For higher values of w cells have higher probability of going from input i to output i. The I/O distribution of this architecture makes of this an advantage thanks to the full duplex ports.

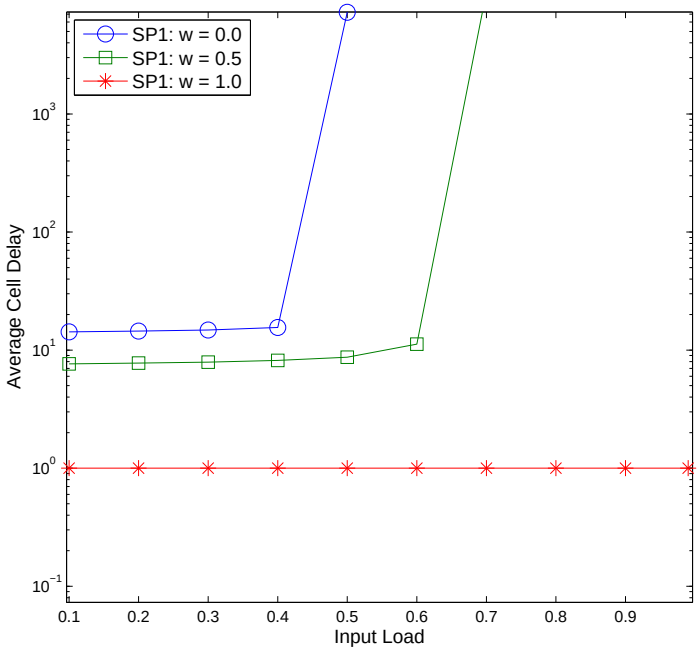


Figure 6.4: 32x32 MDN switch with different Unbalanced traffic.
<MDN, SS = 32, SP = 1, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.1. *MDN architecture performs better for Unbalanced Traffic.*

The size of the switch becomes a drawback in MDN with 1 plane because of the number of ports versus number of routers. This ratio is depicted in Figure 6.5 and compared to that of UDN.

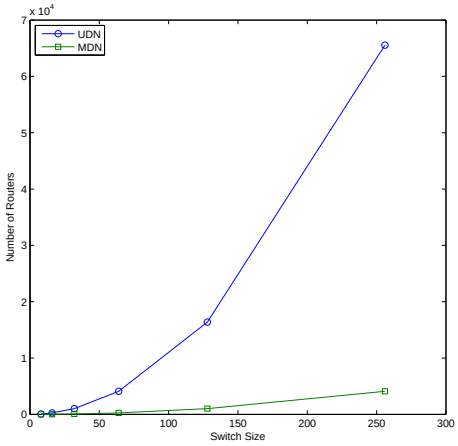


Figure 6.5: Number of routers for UDN and MDN architecture

If in UDN the number of routers grows quadratically with the number of ports but in MDN grows with a factor $\frac{N^2}{16}$, that is, there are 16 times fewer routers for the same number of inputs/outputs. The performance of the fabric for different sizes is shown in Figure 6.6. Bigger switches have worse performance as the number of routers to manage the traffic is not enough. Only employing SP3, full throughput is achieved for a 64x64 switch. The 128x128 and 256x256 MDN meshes cannot get to the 100% throughput for w<0.3 and w<0.7 respectively though they are employing SP3.

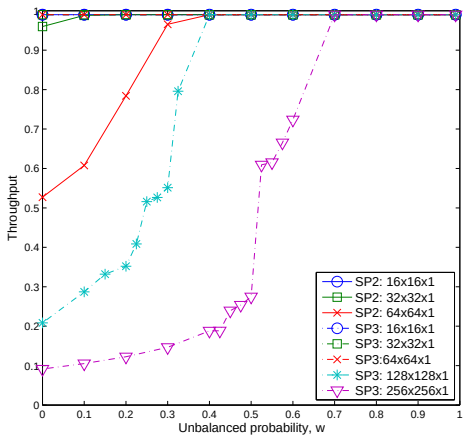


Figure 6.6: Throughput performance for MDN with different switch sizes and speedup values. <MDN, SS = x, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.2. *MDN architecture with only 1 plane is not scalable in performance for switch sizes under Unbalanced traffic.*

The average cell delay of MDN with 100% throughput increases with the switch size because of the multi-hop architecture. Figure 6.7 shows how the 16x16 switch like in the UDN, has half of the cell delay of the 32x32 switch and one forth of the 64x64 switch.

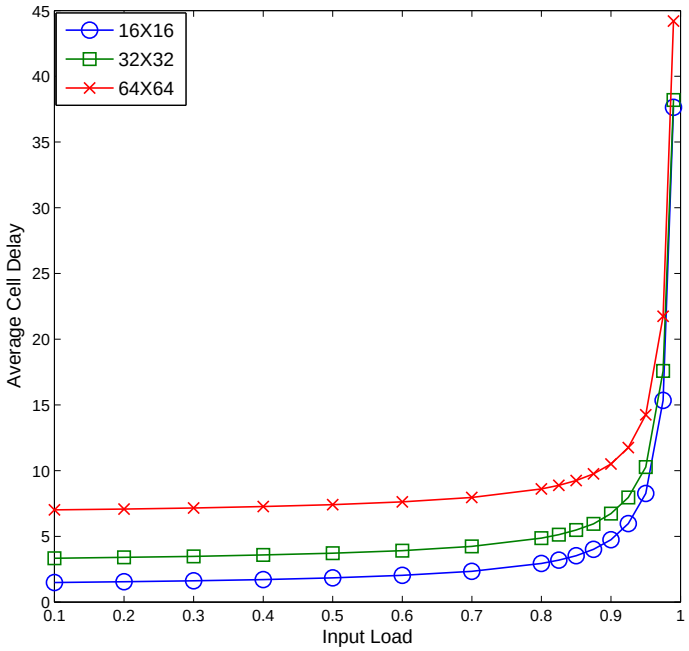


Figure 6.7: Average Cell Delay for MDN with different switch sizes for Unbalanced Traffic. <MDN, SS = x, SP = 2, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.3. *Small switch sizes have smaller delay due to the multi-hop routing when 100% throughput is achieved in MDN under Unbalanced traffic.*

The low ratio of routers to inputs can be solved increasing the number of planes of the mesh. Figure 6.9 shows how the cube architecture ($\frac{N}{4} \times \frac{N}{4} \times \frac{N}{4}$) can manage all the input load with SP2 even for a 64x64 switch.

In this case, the number of routers is incremented by $\frac{N}{4}$. This is represented in Figure 6.8. Compared to the number of routers of the N×N UDN architecture has now $\frac{N}{4^3}$ more routers. To have the same number of routers in both UDN and MDN, 16 layers are needed for any size of MDN switch.

The equation of number of routers in the cube makes that bigger sizes of the switch perform better, similar to UDN. As the number of layers is reduced, it first affects to big switches.

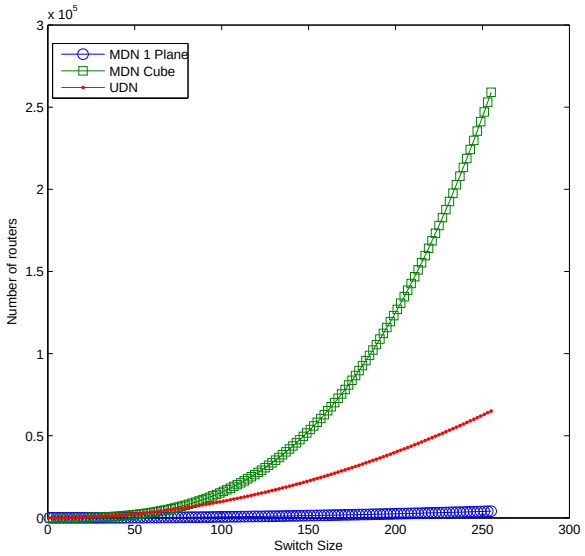


Figure 6.8: Number of routers in MDN cube.

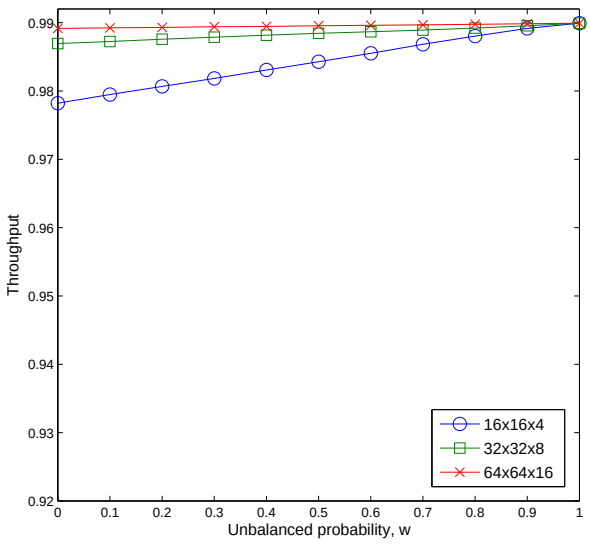


Figure 6.9: A MDN Cube with different switch sizes.
<MDN, SS = x, SP = 2, P = N/4, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.4. *The Cube Architecture performs 100% throughput with SP2 even for big switches.*

The distribution of the packets in the routers in the different planes for the 64x64x16 switch is represented in Figure 6.10. The inputs and outputs are connected to the routers of the perimeter of the P_m plane. For this reason, this plane has the higher peaks of load. In this case, this is *Plane 8*. The routers of the corners are connected to two inputs/outputs instead of only to one and, therefore, they have twice the load of the other routers of the perimeter.

The modulo algorithm 3 is employed to distribute the packets in the mesh. It achieves a very good balance of the load distributing in the different planes as is shown in the graphs.

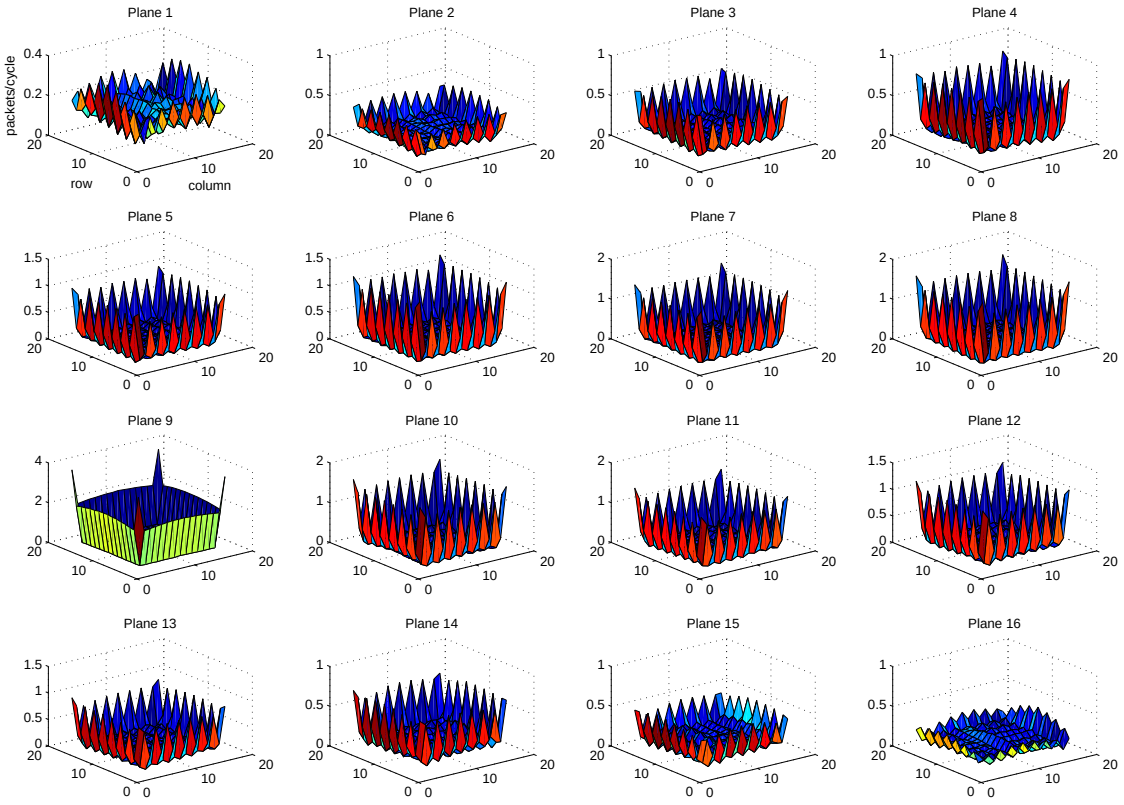
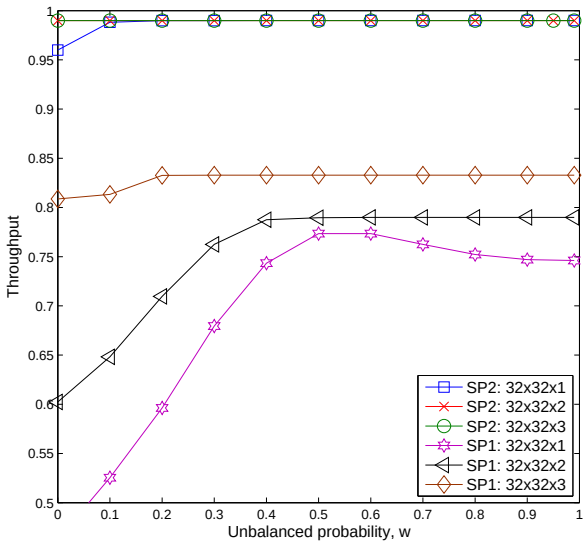


Figure 6.10: Load of each layer for a 64x64x16 MDN Cube

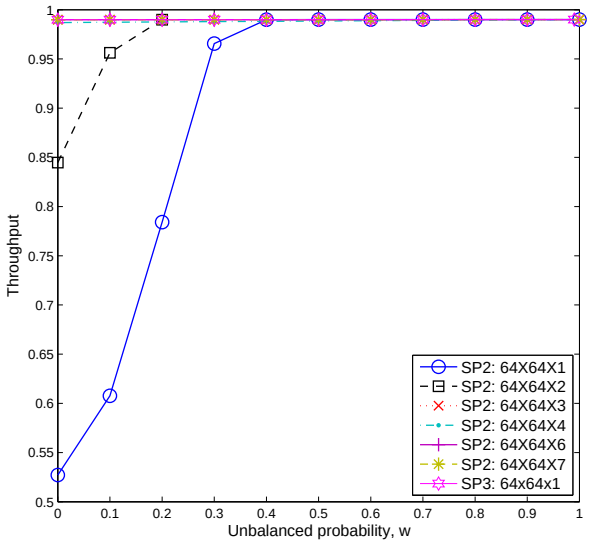
Making a cube is too expensive. The number of planes can be reduced to get to the best cost/performance option. Figure 6.11(a) shows the throughput of a 32x32xP switch for 1, 2, and 3 planes with SP1 and SP2. For SP1, throughput is never 100% due to the handshake of the link level flow control. With SP2, it already has full throughput with 2 layers for any value of w.

In Figure 6.11(b) we represent the performance of a 64x64xP MDN in which the number of planes P goes from 1 to 7 for SP2. It is also tested the performance of a 64x64x1 MDN with SP3. Only 3 layers instead of the 16 of the cube are needed with SP2 to have 100% throughput for unbalanced traffic. This is saving a factor of 5 in number of routers, that is 53248 routers

in a 64x64 switch. If SP3 is employed, 1 layer is enough to deliver full throughput in a 64x64 switch.



(a) 32x32



(b) 64x64

Figure 6.11: MDN with different planes under Unbalanced traffic with SP2.
<MDN, SS = 32|64, SP = x, P = x, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.5. *3 planes are enough to perform 100% throughput in MDN for Unbalanced traffic and SP2 in a switch smaller or equal than 64x64.*

Figure 6.12 represents the cost/performance ratio for a 64x64 switch with different layers and SPs. In this case, *performance* = *throughput* It shows that it is a better option to increase SP than to increase the number of planes.

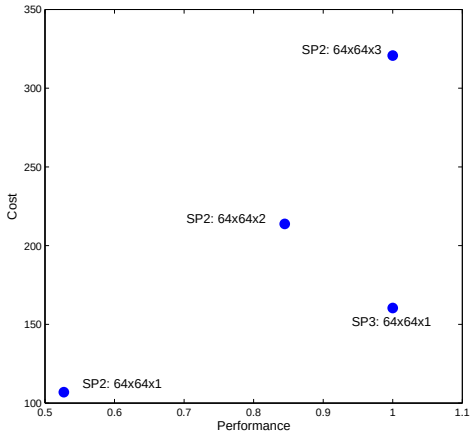


Figure 6.12: Cost/performance of a 64x64 MDN switch for different layers and SPs

Conclusion 6.6. *It is better a better cost/performance option to increase SP than the number of planes in MDN under Unbalanced traffic.*

The load distribution for the 64x64x4 MDN switch is represented in Figure 6.13. The shape is the same as the distribution of packets for the 64x64x16 cube. This means the Modulo algorithm 3 distributes the load well over the available planes.

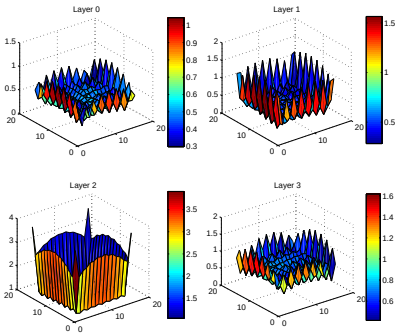


Figure 6.13: Load distribution for 64x64x4 MDN

Conclusion 6.7. *MDN Modulo algorithm keeps load balanced independently of the number of planes.*

	Performance	Average Cell Delay	Area	Cost
↑ w	↑	↓	=	=
↑Speedup	↑(SP2 enough if switch size <64x64)	↓	=	↑
↑Switch Size	↓	↑	↑	↑
↑Planes	↑(3 enough)	↓	↑	↑

Table 6.2: MDN parameter conclusions under Unbalanced Traffic

6.2.2 Bernoulli Uniform Traffic

The following graphs test MDN architecture for Bernoulli Uniform traffic. As was already shown in Figure 6.4 with $w = 0$, this traffic has a worse response in the system than the Unbalanced traffic. Next graph (see Figure 6.14) shows the average cell delay of a 32x32 MDN for different speedups and compares it with the average cell delay under unbalanced traffic for $w = 0.5$.

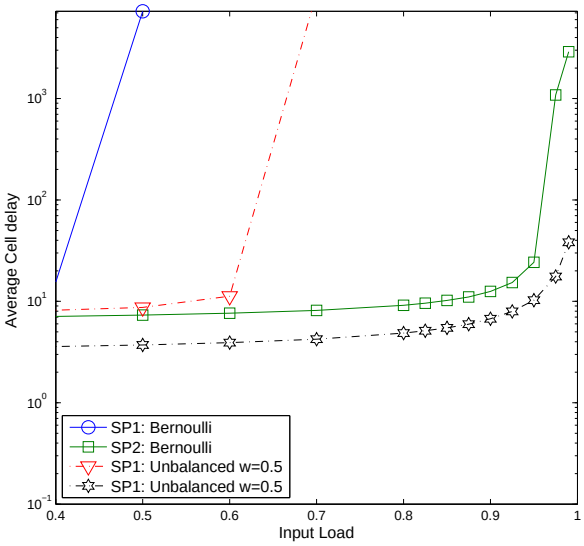
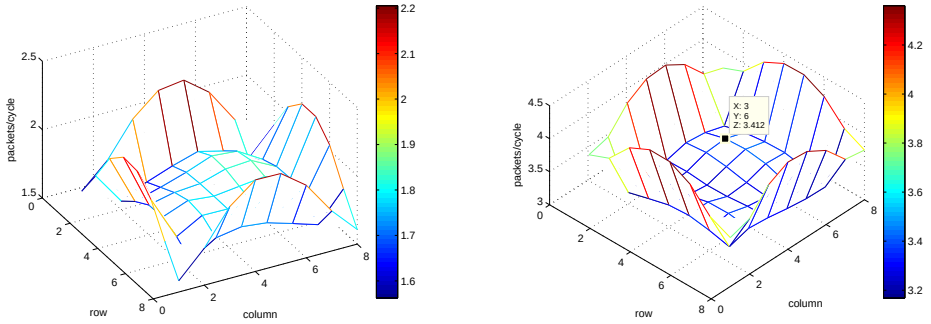


Figure 6.14: MDN under Bernoulli Uniform Traffic for different speedups.
<MDN, SS = 32, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

The distribution of the load in the mesh is represented in Figure 6.15 for SP1 and SP2. Employing SP1, the switch manages half of the load per cycle. Both shapes are similar and have lower load in the routers of the corners. Though those routers have 2 I/O instead of only 1, they have less probability of being in the path of other flows. On the other side, the routers of the middle of each side have higher load because they are in the way of many flows.



(a) SP1 (b) SP2

Figure 6.15: Load Distribution for a 32x32 MDN with Bernoulli Uniform Traffic

The scalability for the switch size in performance is not kept for the MDN with only 1 plane. As it is shown in Figure 6.16, average cell delay increases for bigger switches. This is the same case as for the Unbalanced traffic as the number of routers in the mesh is not enough to manage all the input load.

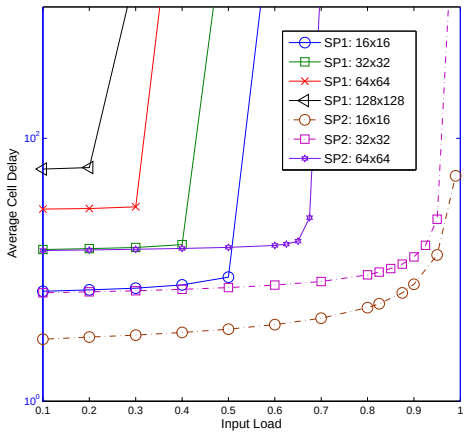


Figure 6.16: MDN under Bernoulli Uniform Traffic for different switch sizes.
<MDN, SS = x, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.8. *MDN architecture with only 1 plane is not scalable in performance for switch sizes for Bernoulli Uniform traffic.*

Some cost-performance study is done in Figure 6.17 for the scalability in the switch size.

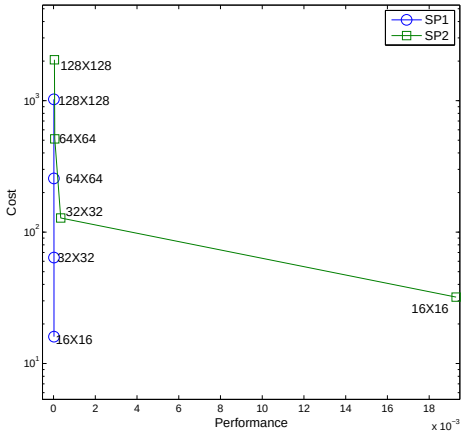


Figure 6.17: MDN: Cost under Bernoulli Traffic

To solve the scalability problem, the number of planes is incremented for fabrics employing SP2. Now, full throughput is achieved in the 64x64 switch with 3 planes (see Figure 6.18). To increase the number of planes when the mesh already has good performance leads in an increase of the delay. This fact can be seen in the average cell delay of the 16x16x3 switch that has higher delay than the 16x16x2 switch.

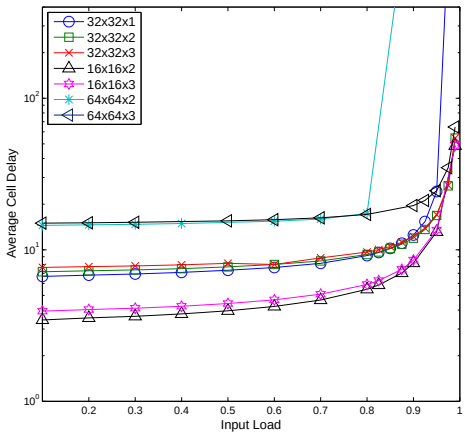


Figure 6.18: MDN under Bernoulli Uniform Traffic for different stages.
<MDN, SS = x, SP = 2, P = x, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.9. *3 layers are enough to manage Bernoulli Uniform traffic employing SP2 in MDN.*

We continue our study of the MDN architecture under Bernoulli Uniform traffic modifying the buffer size. This parameter has an important effect on the system performance. Simply increasing it to 6 cells, the behavior of a 32x32 switch under SP2 reduces the average cell delay by a factor of 52 when the system is heavy loaded (100%). Bigger depth buffer, i.e. 20 in the figure, does not help reducing the delay of the fabric as it is already working in its best possible performance. This is shown in Figure 6.19.

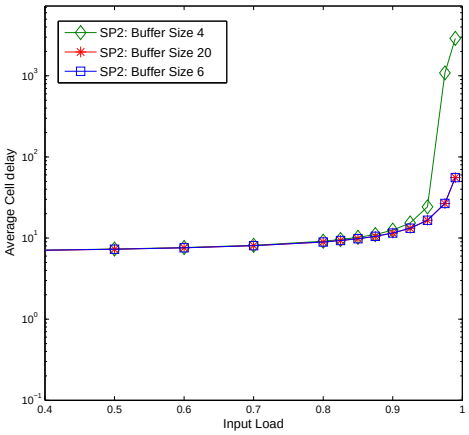


Figure 6.19: 32x32 MDN under Bernoulli Uniform Traffic for buffer sizes.
<MDN, SS = 32, SP = 2, P = 1, BD = x, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.10. *MDN architecture is done well under Bernoulli Uniform traffic with larger buffers.*

	Performance	Average Cell Delay	Area	Cost
⇑Speedup	↑(SP2 enough if switch size <64x64)	↓	=	↑
⇑Switch Size	↓	↑	↑	↑
⇑Planes	↑(3 enough)	↓	↑	↑
⇑Buffer Size	↑(6 enough)	↓	↑	↑

Table 6.3: MDN parameter conclusions under Bernoulli Uniform Traffic

6.2.3 Bursty Uniform Traffic

Bursty Uniform traffic is the traffic pattern with the worse performance. SP2 is not enough to outperform the traditional CICQ. In Figure 6.20 Bursty Uniform Traffic of average size 16 is tested. Another enhancement appart from speedup should be done to the switch to have a better performance than the buffered crossbar.

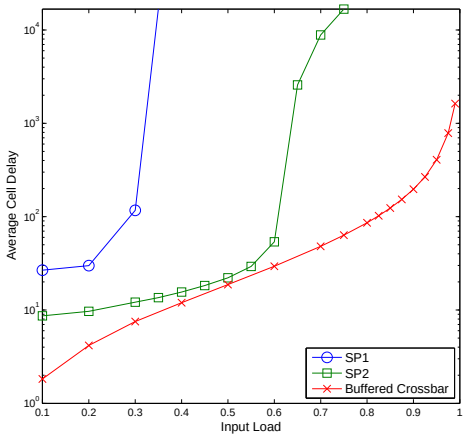


Figure 6.20: Bursty Uniform traffic in the 32x32 MDN switch with different speedups.
<MDN, SS = 32, SP = x, P = 1, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.11. *MDN with 1 plane does not perform well under Bursty Uniform traffic for SP2.*

Adding planes to the mesh improves the performance of the switch. This is shown in Figure 6.21. The 32x32x3 has less than 100 time slots of average cell delay for input loads <90% meanwhile the 32x32x1 switch was in that range of delays for loads <60%.

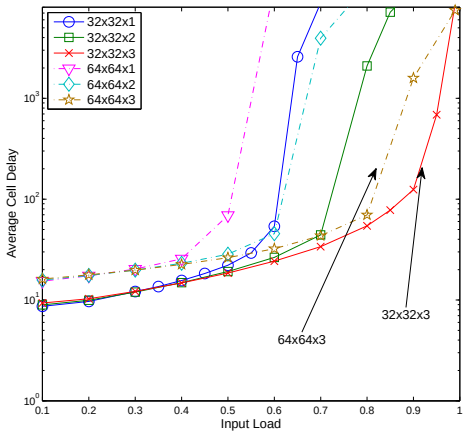
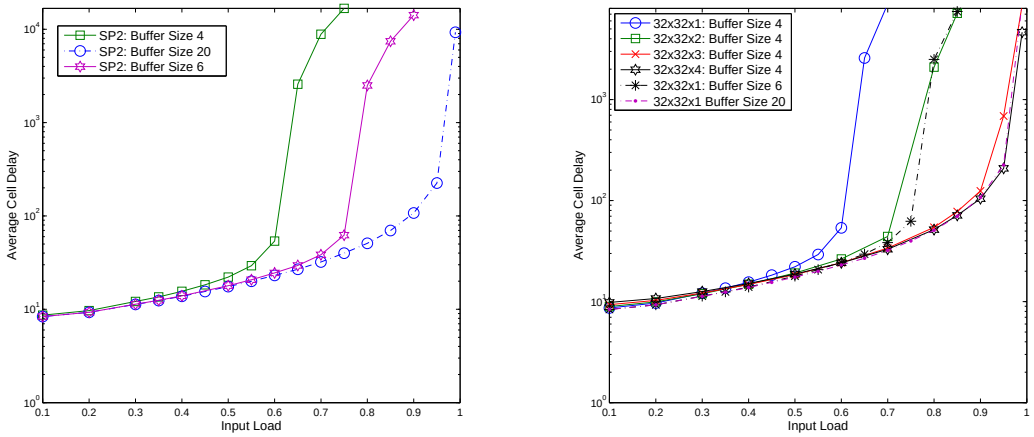


Figure 6.21: MDN under Bursty Uniform Traffic for different stages.
<MDN, SS = x, SP = 2, P = x, BD = 4, RA = MDN Modulo, SA = FIFO:RR >

Figure 6.22(a) shows the performance of a 32x32 MDN switch employing SP2 for different buffer sizes. A buffer size of 20 is needed to reduce significantly the average cell delay of the

crossbar. However, in Figure 6.22(b) it can be seen how adding 4 planes to the switch we get to the same behavior as in the case of the buffer size 20 with a significant save in area.



(a) Buffer Size

(b) Buffer Size and Planes

Figure 6.22: 32x32 MDN under Bursty Uniform Traffic for different buffer sizes. <MDN, SS = 32, SP = 2, P = 1|x, BD = x, RA = MDN Modulo, SA = FIFO:RR >

Conclusion 6.12. *It is better to increase the number of planes than buffer size for cost/performance ratio in MDN under Busty Uniform traffic.*

Figure 6.23 represents this conclusion in a cost/performance graph.

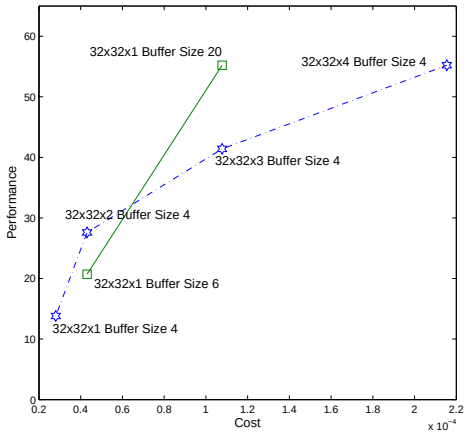


Figure 6.23: MDN cost/performance for Bursty Uniform Traffic and different planes and buffer sizes for SP2

Table 6.4 summarizes the conclusions for Bursty traffic under MDN.

	Performance	Average Cell Delay	Area	Cost
⇑Speedup	↑(SP2 enough if switch size <64x64)	↓	=	↑
⇑Switch Size	↓	↑	↑	↑
⇑Planes	↑(3 enough)	↓	↑	↑
⇑Buffer Size	↑(20 enough)	↓	↑	↑

Table 6.4: MDN parameter conclusions under Bursty Uniform Traffic

6.3 Conclusions

Of all the claims summarized in the introduction, the only one that was not accomplished with the UDN architecture was the use of short wires of the same length. With this new architecture, this aim is finally reached.

- Better load balancing
- Higher path diversity
- Scalability in port count and speed per port
- Use of short wires
- Simpler switch design by using FIFOs

The previous are accomplished:

- Load balanced is kept as it is shown in Conclusion 6.7.
- Higher path diversity is maintained as MDN Modulo algorithm allows to modify the path per flow.
- Scalability, is now higher because fewer number of routers are needed to have full throughput (Conclusion 6.9).
- No VOQs are used neither in the line cards nor in the routers.

Moreover, the position of the input/output pins allow to have less average cell delay. This becomes a problem for heavy loads since less routers are employed. The number of routers can be incremented adding new planes to the system and solving the problem of scalability of the switch for bigger sizes (see Conclusion 6.5). CICQ is outperformed for all kind of traffics if SP2 is employed and the number of planes is adapted.

The following table gathers the parameters variations in MDN.

	Performance	Average Cell Delay	Area	Cost
⇑Speedup	↑(SP2 enough if switch size <64x64)	↓	=	↑
⇑Switch Size	↓	↑	↑	↑
⇑Planes	↑(3 enough)	↓	↑	↑
⇑Buffer Size	↑(20 enough)	↓	↑	↑

Table 6.5: MDN parameter conclusions.

Many parameters were varied along the simulations. Which of them is has a bigger effect in performance can be withdrawn from the conclusions. Conclusions 6.6 and 6.12 shows that in terms of cost/performance: 1) it is better to increase the speedup than the number of planes, 2) it is better to increase the number of planes than the buffer depth. Then, if we order the parameters in terms o which causes a major enhancement in the switch:

1. Speedup.
2. Number of planes.
3. Buffer depth.

7 MDN VS UDN

This chapter compares both MDN and UDN. They are studied togheter under Unbalanced, Bernoulli Uniform and Bursty Uniform traffic. We will consider the best cost/performance of UDN and MDN architecture. Both MDN and UDN employ SP2 and the minimum number of depth/planes to achieve 100% throughput. The buffer size is kept 4 for both architectures. Routing algorithm is Modulo algortihm for UDN and MDN Modulo algorithm for MDN, both for balanced flows. The scheduling algorithm elected is FIFO with RR.

7.1 Unbalanced traffic

We first compare the average cell delay of the UDN to its MDN counterpart under Unbalanced traffic conditions (unbalanced probability, $\omega = 0.5$), as depicted in Figure 7.1. The MDN architecture achieves lower delay than UDN because packets traverse fewer hops to reach their destinations. This in contrast to the UDN where every packet has to cross at least N on-chip routers before reaching its destination, therefore increasing its likelihood to get congested. Both new architectures outperform CICQ crossbar for loads >90%.

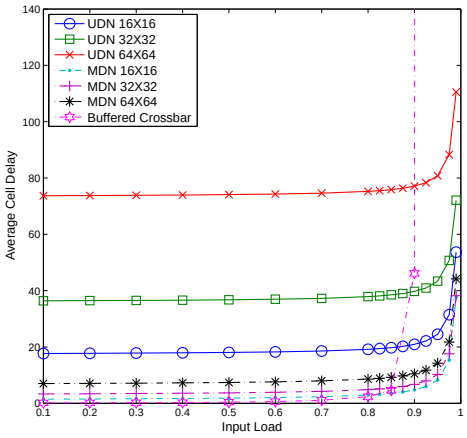
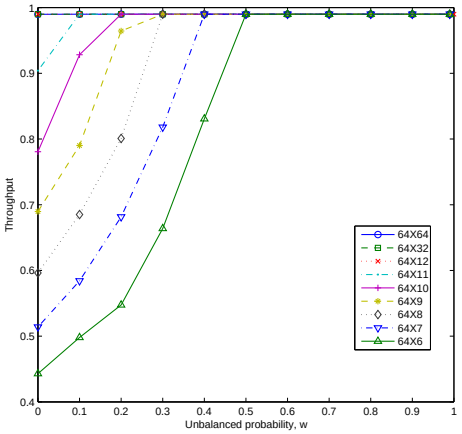


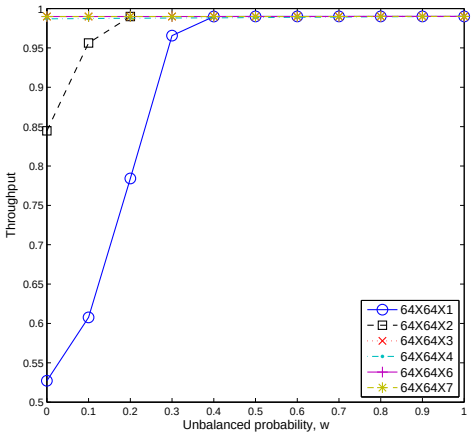
Figure 7.1: Cell delay comparison between the UDN and MDN architectures.
$<$ x, SS $= x$, SP $= 2$, D $= N$, P $= 1$, BD $= 4$, RA $=$ Modulo&MDN Modulo, SA $=$ FIFO:RR $>$

Conclusion 7.1. *MDN has smaller average cell delay than UDN for Unbalanced Traffic.*

We wish to study the effect of the internal buffer requirement. To this end we studied the throughput performance of both UDN and MDN. For the UDN architecture, we varied the depth of the mesh (number of columns) for a 64×64 UDN switch. As illustrated in Figure 7.2(a), when the depth is small (< 12) the throughput is low due to the highly congested NoC. However, as the depth goes beyond 11, we can achieve a throughput comparable to that of a CICQ that uses a depth of 64. This is equivalent to a saving worth of more than 3500 crosspoints while sustaining the same switching throughput. With the MDN architecture the performance is even better, as depicted in Figure 7.2(b). MDN uses stages (planes) instead of depth (column), and we can see from the Figure that increasing the number of planes has a significant impact on the switch throughput. With just 3 stages, we can achieve a full throughput.



(a) A 64×64 UDN with different depths



(b) A 64×64 MDN with different planes

Figure 7.2: Throughput performance comparison between the UDN and MDN architectures with different sizes.
<UDN|MDN, SS = 64, SP = 2, D = x, P = x , BD = 4, RA = Modulo|MDN Modulo, SA = FIFO:RR >

Conclusion 7.2. *Both 64x64x3 MDN and 64x12 UDN are the minimum size needed to perform full throughput under Unbalanced traffic. Then, MDN employs 3500 fewer routers to achieve 100% throughput. MDN is a best cost/performance option.*

Figure 7.3 shows the cost/performance comparison of a 64x64 switch both for UDN and MDN under Unbalanced traffic. In this case *performance* = *throughput*. Performance in this case is measured in terms of throughput, maximum performance is then 1. We see that MDN is a better option than UDN.

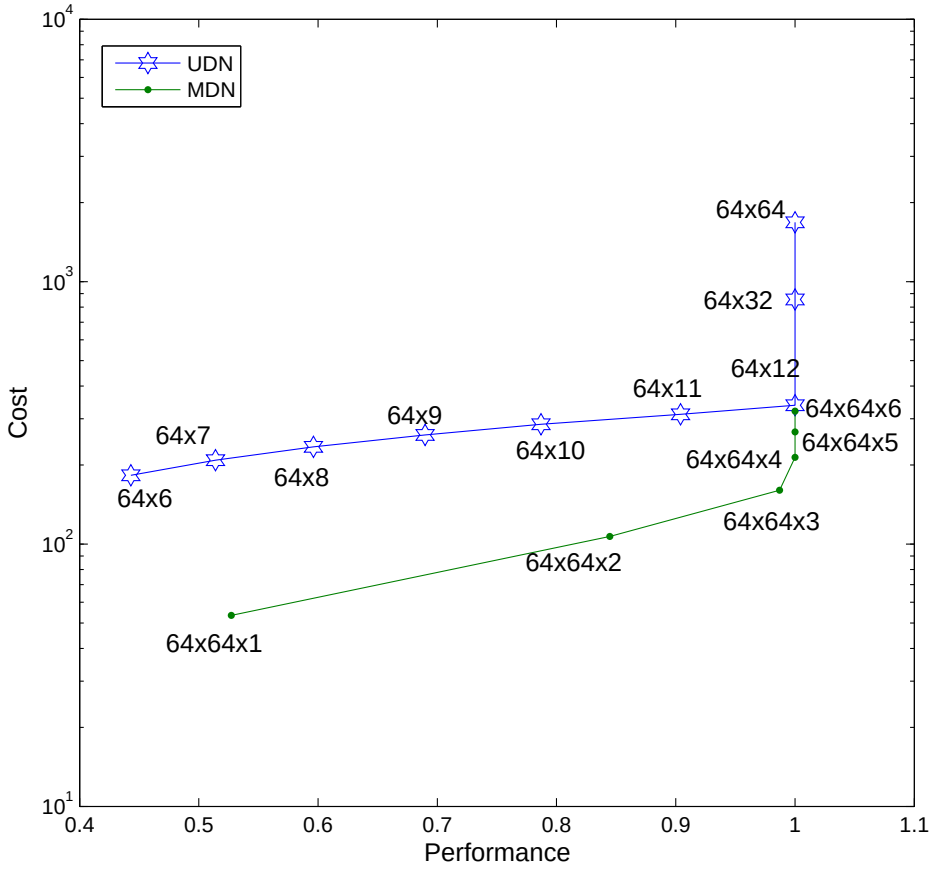


Figure 7.3: UDN and MDN cost/performance comparison for a 64x64 switch with different depths/planes.

Conclusion 7.3. *MDN is a better cost/performance option than UDN for Unbalanced traffic.*

7.2 Bernoulli Uniform traffic

Figure 7.4 makes a comparison of both architectures and CICQ for a 32x32 switch for Bernoulli Uniform traffic. Several parameters of UDN and MDN are changed. Now UDN is using 7 columns and employing SP2 meanwhile MDN has 3 planes and also employs SP2. The 32x7 switch is the optimized mesh for Bernoulli Uniform traffic in UDN. This means 224 crosspoints in UDN and 192 crosspoints in MDN saving 32 routers in the switch. MDN not only uses less routers, but it has a lower average cell delay as it is shown in the figure, though it does not outperform CICQ this is due to the multi-hop routing. As both MDN and UDN can run at higher clock frequencies compared to CICQ they are the best option for switching Bernoulli Uniform traffic in comparison with the traditional CICQ crossbar.

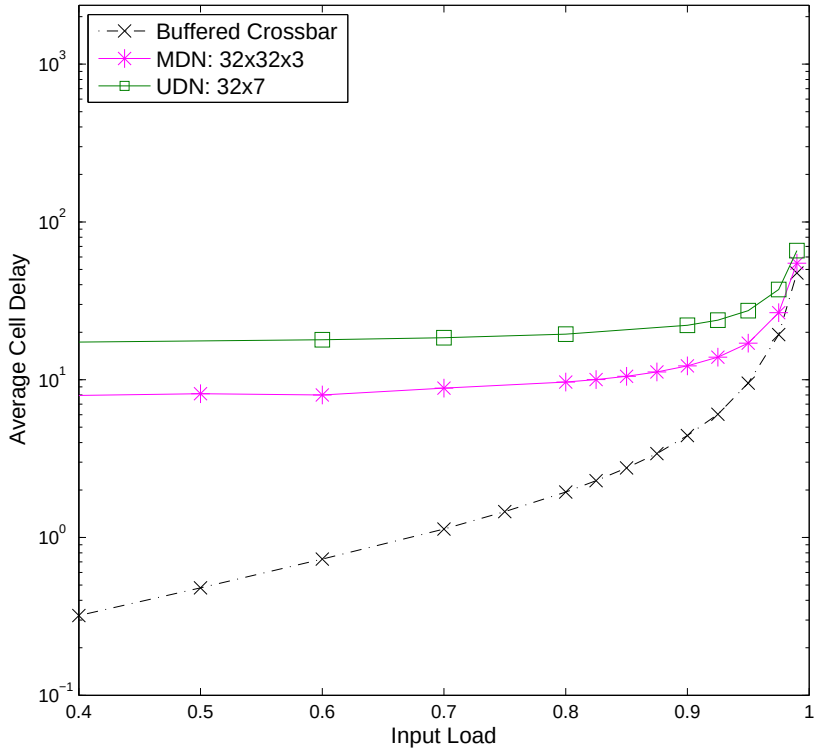


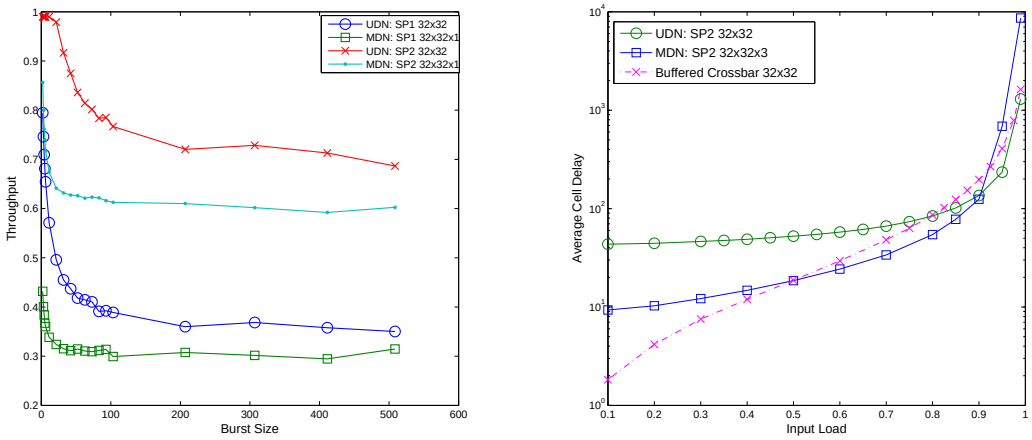
Figure 7.4: 32x32 switch with Bernoulli Uniform Traffic for different architectures. $< x$, SS = 32, SP = 2, D = 7, P = 3 , BD = 4, RA = Modulo&MDN Modulo, SA = FIFO:RR $>$

Conclusion 7.4. *MDN outperforms UDN and CICQ for Bernoulli Uniform traffic. They outperform CICQ as they run at higher frequencies.*

7.3 Bursty Uniform traffic

The advantage of MDN over UDN does not always hold, especially under heavy loads where MDN cannot cope due to its small number of hops compared to UDN. This is the case of Figure 7.5(a), where different burst sizes are used for a 32×32 switch. Here, UDN outperforms MDN both with speedup values of 1 and 2. As a conclusion, when both architectures can deliver 100% throughput, MDN has a lower delay. However, in the presence of high congestion UDN performs slightly better. For lower sizes of bursts like 16, UDN performs better than MDN for light-medium loads (<95%) and becomes worse for high loads. For this simulation, the UDN switch is made out of 32x32 routers meanwhile MDN has only 8x8x3 crosspoints. This is 1024 routers in UDN against 192 routers in the MDN mesh. Then UDN uses 5 times more routers than MDN and only outperforms MDN for loads higher than 95% (see Figure 7.5(b)). Hence, in a cost/performance comparison, MDN outperforms UDN for bursty uniform traffic. MDN

outperforms CICQ in the range of loads 0.5→0.92. Then average cell delay becomes higher for MDN though it should be again taken into account that MDN works with higher frequencies.



(a) Different burst sizes

(b) Burst size of 16

Figure 7.5: 32x32 switch with Bursty Uniform Traffic for different architectures. $<$ x, SS $=$ 32, SP $=$ $x|2$, D $=$ 32 , P $=$ $1|3$, BD $=$ 4, RA $=$ Modulo&MDN Modulo, SA $=$ FIFO:RR $>$

Though UDN has better performance than MDN for high loads ($>$ 90%), it is 5 times more expensive. This is depicted in Figure 7.6.

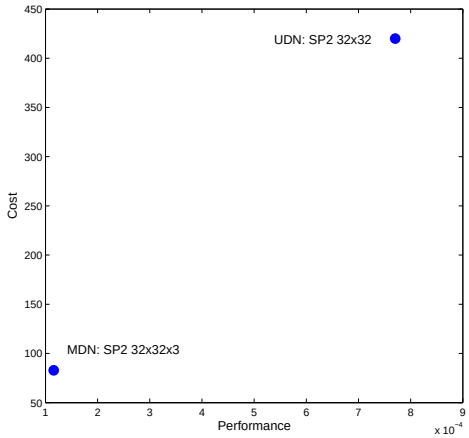


Figure 7.6: UDN and MDN cost/performance for Bursty Uniform traffic.

Conclusion 7.5. *MDN is a best cost/performance option than UDN and CICQ for Bursty Uniform traffic with burst of average size 16.*

7.4 Conclusions

This chapter compared UDN and MDN architectures in terms of performance and cost/performance. Conclusions 7.3, 7.4 and 7.5 show that MDN is a better cost/performance option than UDN for all kind of traffic. Figures also show that if, as expected, MDN runs at higher frequency than CICQ it also outperforms CICQ.

In the last chapters, we discussed the capabilities of MDN and UDN under unicast traffic. It is not the only type of traffic of the Internet. The use of multicast traffic is increasing with the new applications. The performance of the proposal architectures is studied for multicast traffic in the next chapter.

8 MULTICAST

Though the main part of the Internet traffic is unicast, the development of new applications has lead to an increasing demand of switches with multicast capabilities. In multicast traffic, an input packet can have more than one destination. The first multicast switches were based on copy networks and a point-to-point routing networks. Buffer overflow is particularly likely to occur in this kind of communications [53]. If several sources transmit packets to the same destination at the same time over a network, that destination might be unable to process fast enough to avoid buffer overflow. When overflow happens, fairness is another serious problem. It is related to providing equal access to all the input ports.

Many studies have tried to develop new suitable architectures [54] for this kind of traffic. Copy networks have a simple way to service the packets of the input queues, that is to replicate as many times as the number of outputs it goes to. Copying cells involve two disadvantages: increasing the require bandwidth and making the packets contend for access multiple times. For that reason, most of the studies are now focused on the performance of the crossbar switches for multicast traffic. Its natural multicast capability [55] avoid copying the packet to all the destinations. It has the ability to transfer simultaneously, a packet to multiple outputs using simultaneous switching paths.

The number of destinations of a packet is known as the fanout of the packet. When scheduling multicast traffic, two different service disciplines can be used. The first is called no fanout-splitting, in which all the copies of a packet must be sent in the same packet time. If any of the output packets loses contention for an output port, none of the output packets are transmitted and the packet must try again in the next packet time. The second discipline is called fanout-splitting, in which output packets may be delivered to output ports over any number of packet times. Only those output packets that are unsuccessful in one packet time continue to contend for output ports in the next packet time. Research has shown that fanout-splitting enables a higher switch throughput for little increase in implementation complexity [56].

In a NxM router with multicast capabilities, a packet arriving to any of the N input ports can have any set of destinations between 2 and M. To avoid HoL blocking would imply to maintain up to $2^N - 1$ separate FIFO queues per input. This architecture, known as the multicast VOQ (MC-VOQ) switching architecture [57], is considered unfeasible due to the high number of queues required. Most of the the multicast scheduling research work, is based in this FIFO queue architecture [56]. This architecture has a similar HoL blocking problem as in the unicast case. Figure 8.1 shows an example of an 3x4 multicast CICQ with FIFO queues.

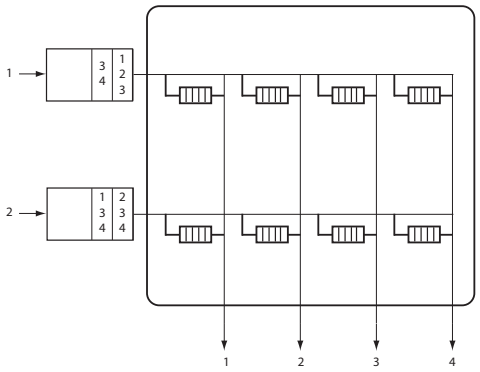


Figure 8.1: 3x4 multicast CICQ switch with FIFO queues.

In this example we consider fanout splitting. For the first input, packets to output port 1 have no contention to exit. In the second input, the packet to output 4 is also free to exit. Packets with destination 2 and 3 have to compete to exit. There are three options: the first input exit both packets, the second input exit both packets or each of them exits only one packet. The set of packets that lose contention for output ports and remain at the HoL of the input queues at the end of each time slot is called *residue*. The residue can either be concentrated on the input ports or it can be distributed over the input ports depending on the policy used. For this example, in concentrated policy, the remaining packets would be only in one of the input ports. This policy leaves the unsent packets in the minimal number of ports. On the other hand, if a distributed policy is used the residue will be distributed in the maximal number of ports, then both ports would have remaining packets.

It is proved, however, that crossbar switches could never reach a 100% when it is increased the number of ports [58]. For this reason, it was proposed to use a small number of queues per input, though maximal throughput is not achieved [59]. This solution is to find a compromise to use k FIFO queues per input ($1 \leq \ k \ll \ 2^M - 1$). In this architecture, as the number of queues is smaller than the possible fanout set, a placement strategy is necessary to enqueue each incoming cell to its corresponding queue. Figure 8.2 shows an example of a multicast k FIFO architecture for an NxM buffered crossbar switch.

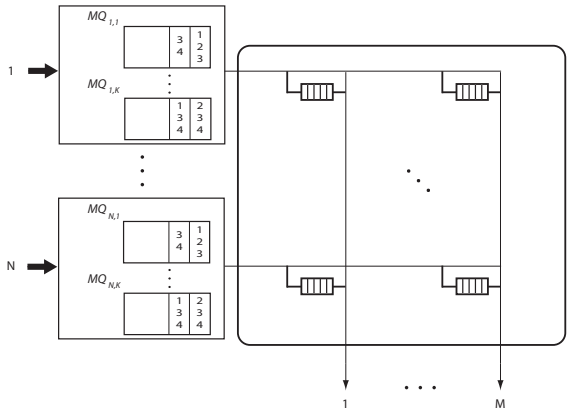


Figure 8.2: NxM multicast k FIFO queues CICQ switch with.

In this chapter, multicast traffic is tested for UDN and MDN architectures. Three different algorithms are implemented and compared. In the first algorithm the NIs are in charge of copying the multicast packets into unicast packets for the network. In the other two algorithms, they are the NoC routers who make the copies of the multicast packets.

8.1 Implementation

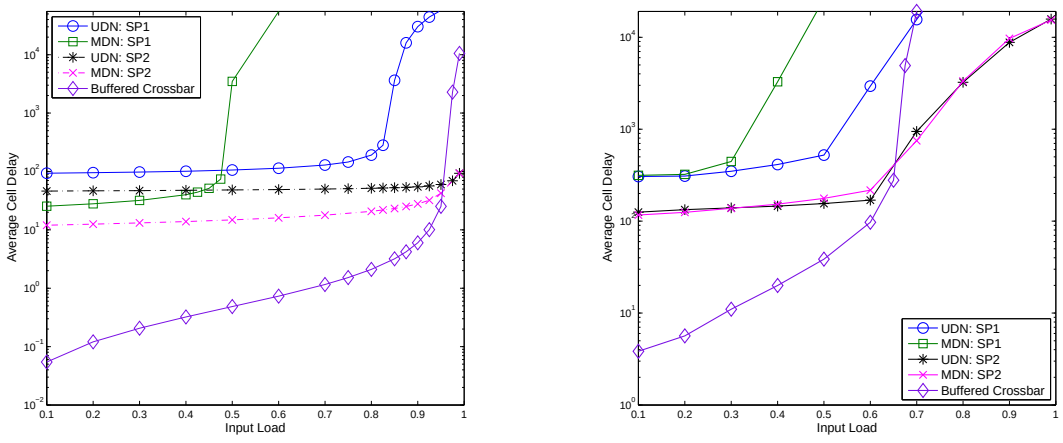
The multicast destinations are coded in the header of a packet with a bit mask. This field has as many bits as the possible number of destinations in the switch. A bit set to "1" means that the packet wants to go to the output port indexed by that bit.

The following analysis will be done using fanout-splitting in the multicast scheduling algorithms because of its simplicity, and performance benefits. FIFO queues are implemented both for the UDN and MDN architecture and the CICQ crossbar.

8.2 Copy multicast

In the first algorithm tested, the NI are responsible of copying the packets and converting them into unicast packets for the network. The copy network generates the copies requested by incoming packets, and the switch routes the replicated copies to their final destinations. The behavior of the routers is then not modified. UDN and MDN use the same algorithms as in the unicast case. The NI converts each packet to a unicast packet and the routers do not have to access the bit mask.

Like in the unicast case, the CICQ performs better under light load (<95%) under Bernoulli Uniform traffic as it does not have multihop delay (Figure 8.3(a)). With bursty traffic, the longer size of the flows causes even more HoL blocking and UDN and MDN perform better than the CICQ above 60% load. This is shown in Figure 8.3(b). Under Diagonal traffic, the new architectures only perform better than CICQ for loads above 95% and when they are employing SP2 (see Figure 8.4).



(a) Bernoulli Uniform Traffic

(b) Bursty Uniform Traffic -b 16

Figure 8.3: Cell delay comparison between the UDN, MDN and CICQ for Copy multicast under Uniform traffic. $< x$, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = Copy Mcast, SA = FIFO:RR >

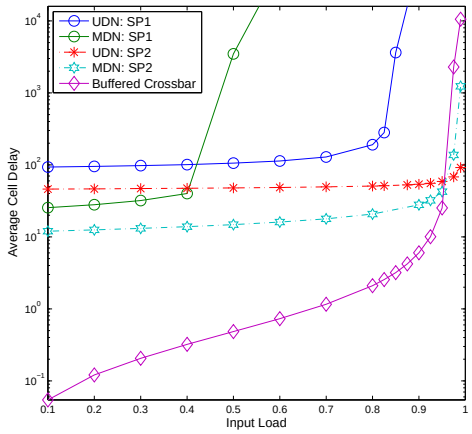
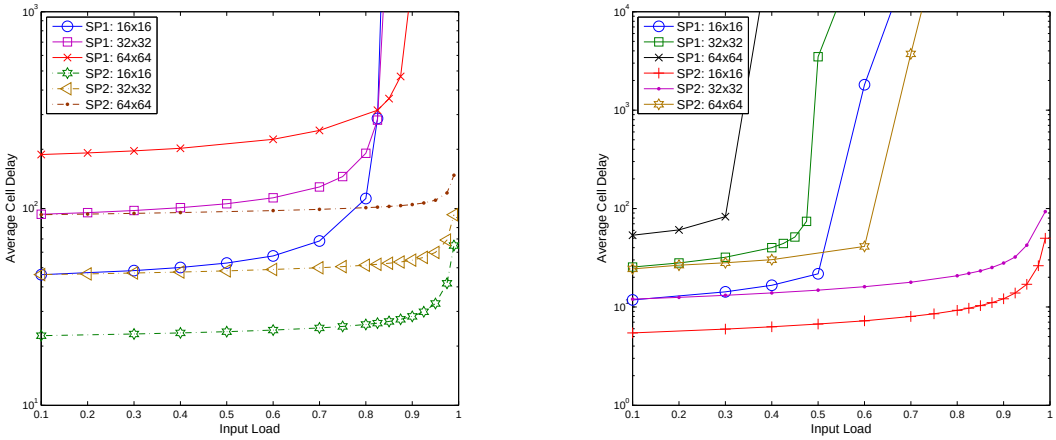


Figure 8.4: Cell delay comparison between the UDN, MDN and CICQ for Copy multicast under Diagonal traffic. $< x$, SS $= 32$, SP $= x$, D $= 32$, P $= 1$,BD $= 4$, RA $=$ Copy Mcast, SA $=$ FIFO:RR $>$

Conclusion: Both MDN and UDN outperform CICQ for Uniform and Non-Uniform multicast traffic for loads>95% when they employ SP2.



(a) UDN (b) MDN

Figure 8.5: Cell delay comparison between UDN and MDN for different switch sizes. <UDN,MDN, SS $= x$, SP $= x$, D $= 32$, P $= 1$,BD $= 4$, RA $=$ Copy Mcast, SA $=$ FIFO:RR $>$

In the previous experiment we test the scalability of the new architectures when the switch size is increased. Figure 8.5 shows the different response of UDN and MDN under Bernoulli

Uniform traffic . For SP1 and SP2 UDN improves its behavior as the switch size increases (Figure 8.5(a)) thanks to a lower HoL blocking (section 3.2.3). On the other hand, the rate of input-output ports of the MDN architecture related to the number of routers, leads this architecture to a worse performance (Figure 8.5(b)).

Conclusion: UDN is more scalable when MDN only has 1 plane.

When UDN is employing SP2 under Bernoulli Uniform traffic, its depth can be reduce to have a better cost/performance relation. Figure 8.6 shows that like under unicast traffic, UDN can be reduced a factor of 5 in depth until it becomes congested.

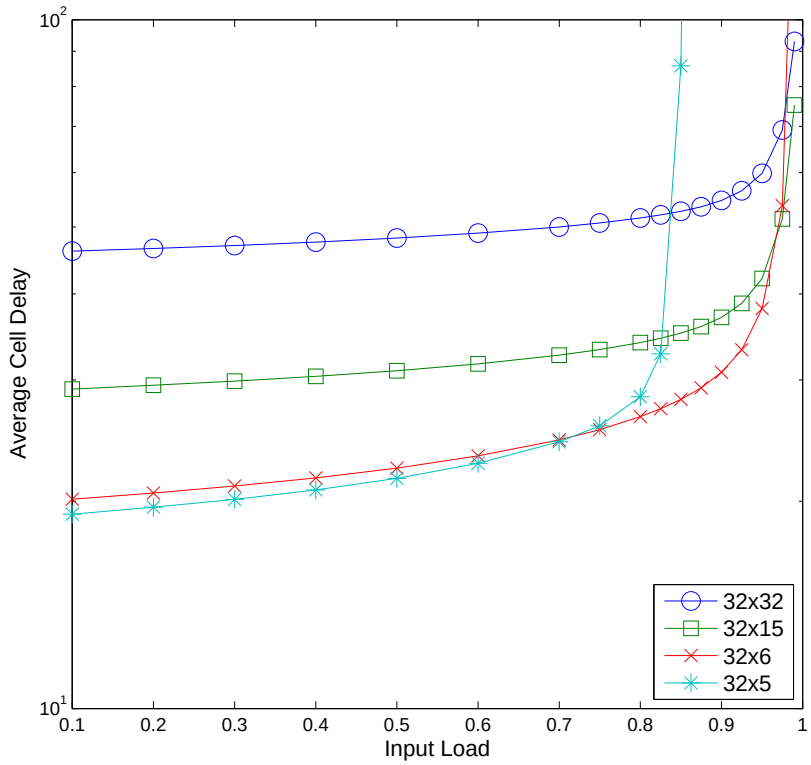
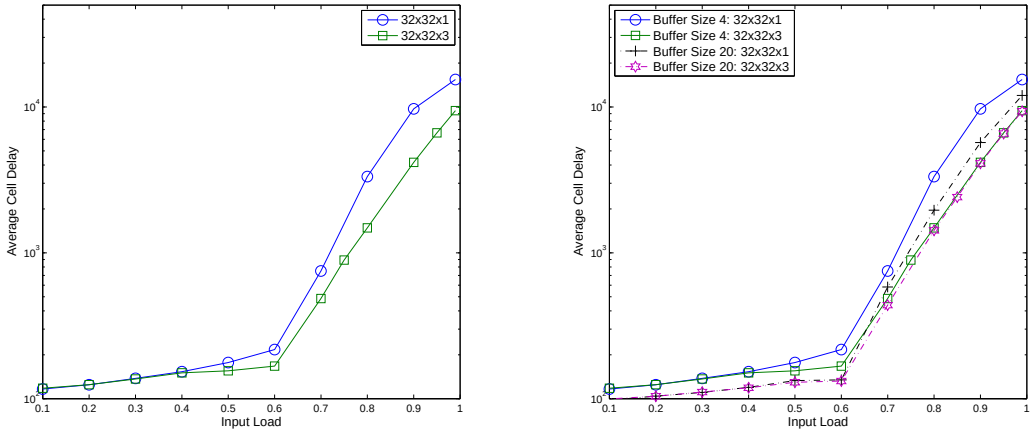


Figure 8.6: Bernoulli Uniform Multicast traffic in 32x32 UDN for different depths. <UDN, SS = 32, SP = 2, D = x ,BD = 4, RA = Copy Mcast, SA = FIFO:RR >

Conclusion: UDN depth can be reduced by a factor of 5 without performance degradation for Bernoulli Uniform multicast traffic when it employs SP2.

On the other side, MDN architecture should increase its number of planes for Bursty traffic. Increasing the planes does not improve the performance of the system significantly This is shown in Figure 8.7(a). Other solution is increasing the buffer depth of each router. Figure 8.7(b) shows the performance of MDN with a buffer depth of 20. Though it improves the behavior of the switch, it is not a good cost/performance option.



(a) Planes

(b) Planes VS Buffer Size

Figure 8.7: Bursty Uniform Multicast traffic in 32x32 MDN for different planes and buffer size. <MDN, SS = 32, SP = 2, P = x ,BD = 4|x, RA = Copy Mcast, SA = FIFO:RR >

Conclusion: Increasing the buffer depth and/or the number of planes odes not increase the performance of MDN under Bursty Uniform multicast traffic.

8.3 Modulo Multicast algorithm

The Modulo Multicast algorithm, is based on the modulo algorithm of unicast traffic. In this case, the NI sends the multicast packet to the router without replication. It is the router itself, who decides when a copy of the packet should be made. The algorithms used for UDN and MDN architecture are the same as in sections 3 and 5, the Modulo Algorithm 1 and Modulo MDN Algorithm 3. When a router applies the Modulo algorithm, as a packet can have now several directions, copies of the packet should be done. The router access the bit mask and makes as many copies as the number of directions the packet wants to go to. Then, it updates the correspondent bit mask of the new packets. An example of this situation is represented in Figure 8.8.

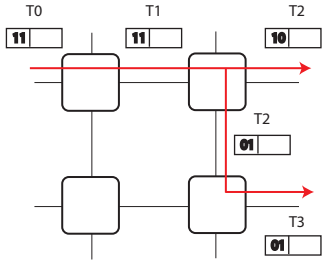
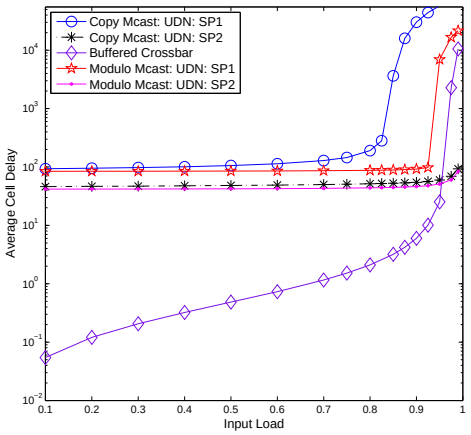
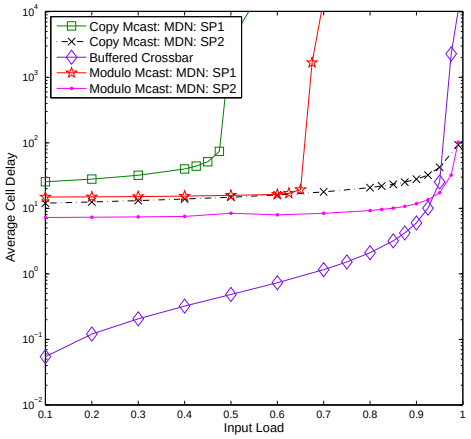


Figure 8.8: Example of the bitmask update in multicast.

It is a 2x2 switch in which a multicast packet enter in time 0 (T0) input port 0. The packet has two destinations, output 0 and output 1. These destinations are market in the bitmask with "11". In T1 it reaches Router[1][1]. This router makes a copy of the packet to send one packet to the East and another packet to the South. Each copy of the packet has an updated bitmask, and now the bit is set to 0 to the destinations to which there where the packet is split.



(a) UDN



(b) MDN

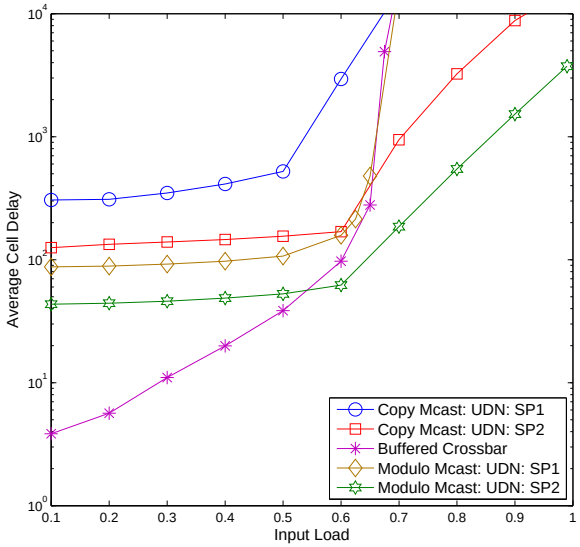
Figure 8.9: 32x32 switch under Bernoulli multicast traffic for Copy multicast and Modulo multicast.

$< x$, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = x, SA = FIFO:RR >

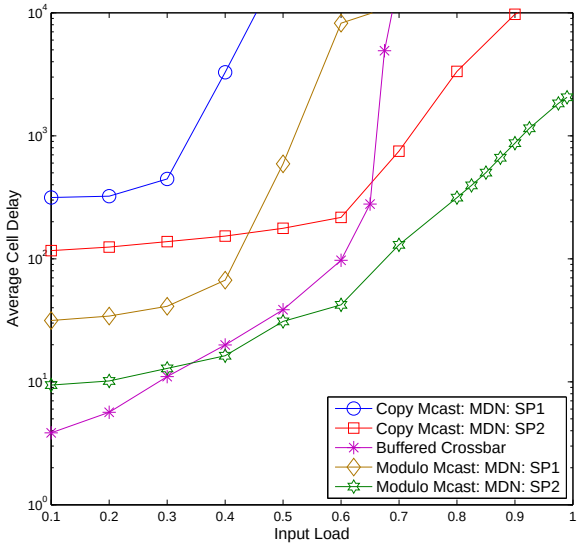
Figure 8.9 shows that employing the Modulo Multicast algorithm, UDN improves its behavior significantly. Now, both for Bernoulli Uniform and Bursty Uniform running at SP1 it becomes

closer to the CICQ performance. MDN has ≃20% more capability to manage Bernoulli traffic and ≃10% for Bursty traffic for SP1. With SP2, both MDN and UDN have less average cell delay than the CICQ for loads >95% in Bernoulli traffic. Under Bursty traffic , UDN performs better than CICQ when load>55% and MDN when load>35%.

MDN outperforms the UDN using 64 routers instead of the 1024 routers of UDN. Therefore is a better choice in terms of cost/performance.



(a) UDN



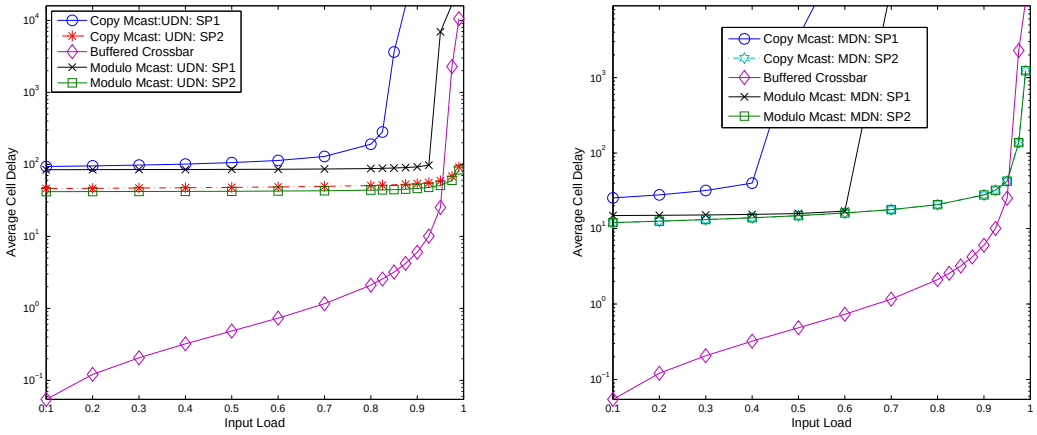
(b) MDN

Figure 8.10: 32x32 switch under Bursty multicast traffic for Copy multicast and Modulo multicast.

$< x$, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = x, SA = FIFO:RR >

Conclusion: Modulo multicast improves the performance of both architectures for Uniform multicast traffic.

For Non-Uniform traffic diagonal traffic is tested. UDN and MDN do not improve in performance. when they are employing SP2 compared to when they use the Copy multicast algorithm. This is shown if Figure 8.11. UDN has less average cell delay than MDN for both SP1 and SP2 when MDN only has 1 plane.



(a) UDN (b) MDN

Figure 8.11: 32x32 switch under Diagonal multicast traffic for Copy multicast and Modulo multicast.

$<$ x, SS $=$ 32, SP $=$ x, D $=$ 32 , P $=$ 1 ,BD $=$ 4, RA $=$ x, SA $=$ FIFO:RR $>$

Conclusion: For Non-uniform multicast traffic, Modulo multicast does not perform better than Copy multicast when SP2 is employed.

The buffer size of MDN is increased to try to enhance its performance under Bursty traffic. Figure 8.12 shows that this modification does not reduce the average cell delay of the system.

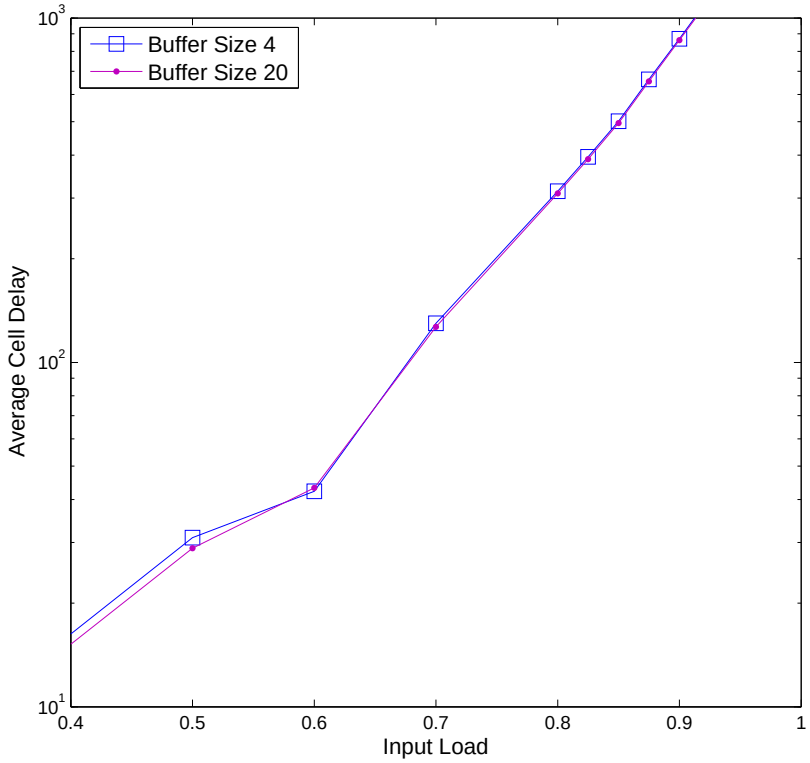
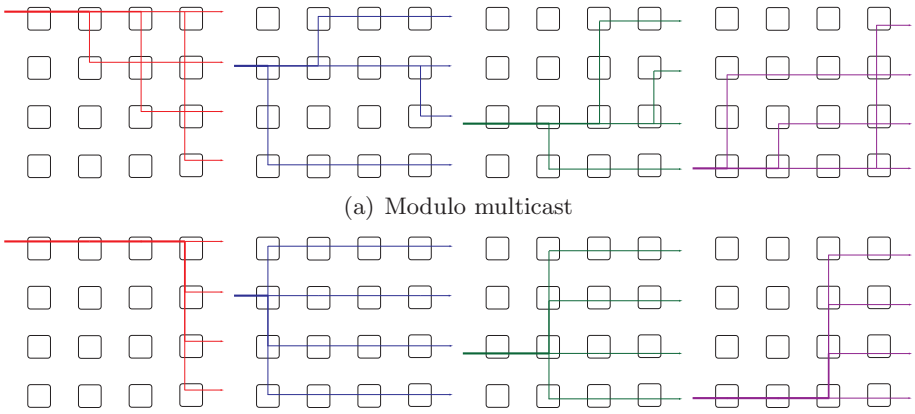


Figure 8.12: 32x32 MDN switch under Bursty traffic for different buffer sizes.
<MDN, SS = 32, SP = 2, P = 1 ,BD = x, RA = Modulo Mcast, SA = FIFO:RR >

Conclusion: Increasing the buffer size does not improve performance of MDN under Bursty multicast traffic.

8.4 Simplified Modulo multicast algorithm

The Modulo algorithm can be simplified if we only allow for each input one column of its row to turn. All the packets from input i to any output, will turn in the same column j. In this way, the number of packets that can have the maximum destination is augmented. This path selection is reflected in Figure 8.13. Figure 8.13(a) shows the paths for the Modulo multicast algorithm, were each flow input/output selects a different row to turn. On the other hand, Figure 8.13(b) shows how in the Simplified Modulo multicast algorithm, all the flows turn in the same row. For MDN, the same algorithm is applied when the input port is in front of the output port in the mesh. To check the performance of this new algorithm we will compare it with the previous simulations.

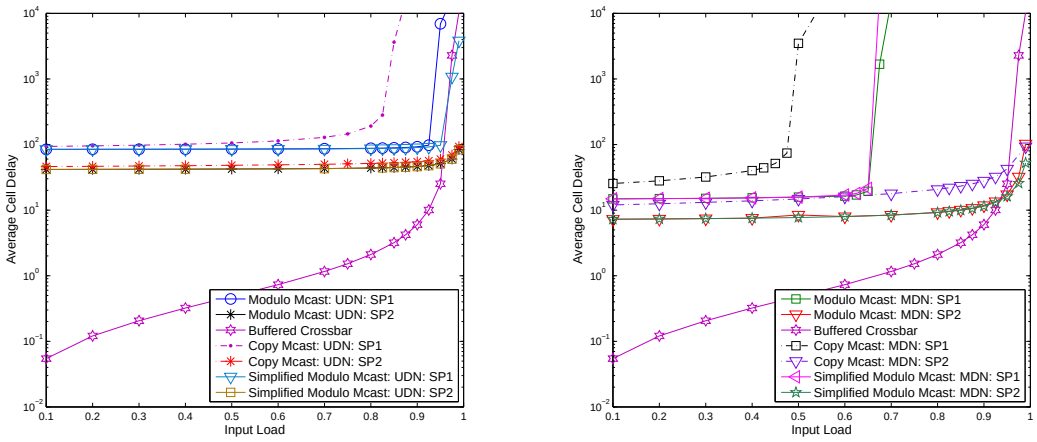


(a) Modulo multicast

(b) Simplified Modulo multicast

Figure 8.13: Modulo multicast algorithm and Simplified Modulo multicast algorithm routing paths in a 4x4 UDN

This new algorithm is studied under Bernoulli traffic in Figure 8.14. It improves the behavior of UDN when it employs SP1. For the other cases, its enhancement is not considerable.

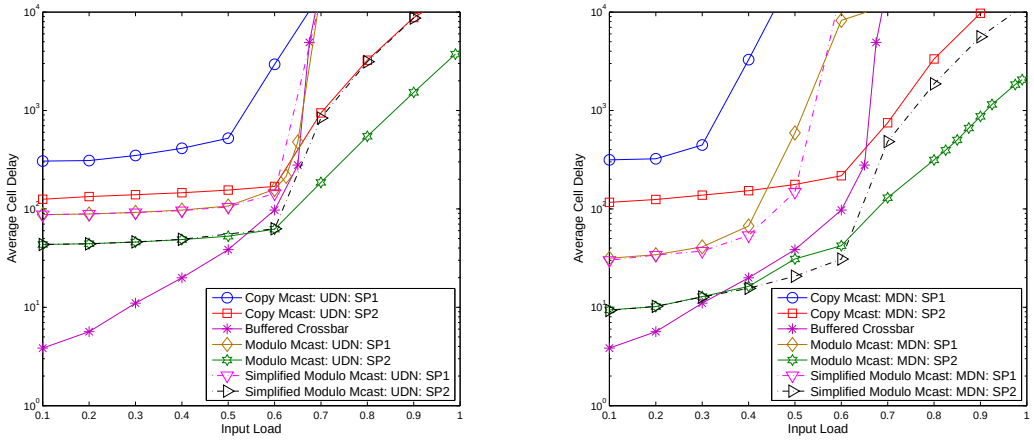


(a) UDN

(b) MDN

Figure 8.14: 32x32 Switch under Bernoulli traffic for different algorithms.
$< x$, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = x, SA = FIFO:RR >

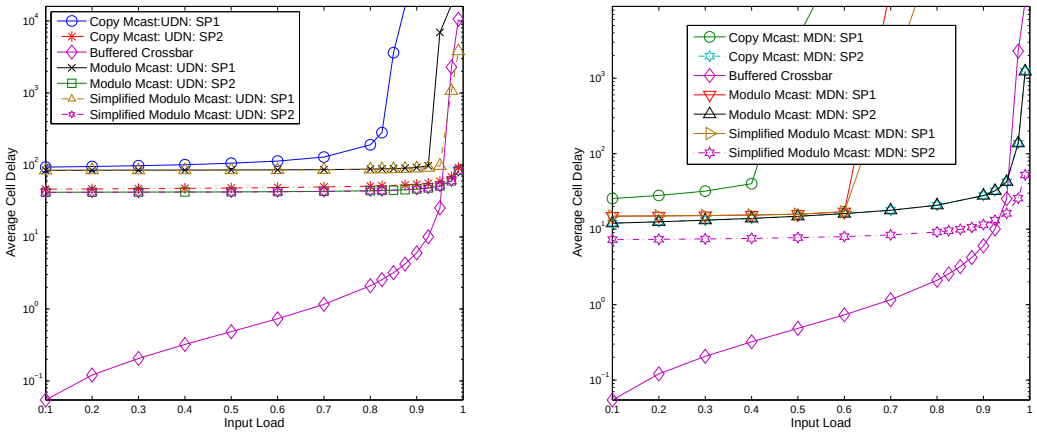
Under Bursty traffic , the Simplified modulo multicast algorithm does not perform better than the previous Modulo Algorithm. Figure 8.15 depicts how both for UDN and MDN, the new algorithm does not reduce the average cell delay of the system neither for SP1 or SP2 compared to the Modulo algorithm.



(a) UDN (b) MDN

Figure 8.15: 32x32 Switch under Bursty traffic for different algorithms.
$<$ x, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = x, SA = FIFO:RR $>$

Figure 8.16 compares the three algorithms for Diagonal traffic. The Simplified Modulo multicast algorithm does not improve considerably the performance of the switch. It is only remarkable the reduce in average cell delay in UDN employing SP1, and in MDN employing SP2.

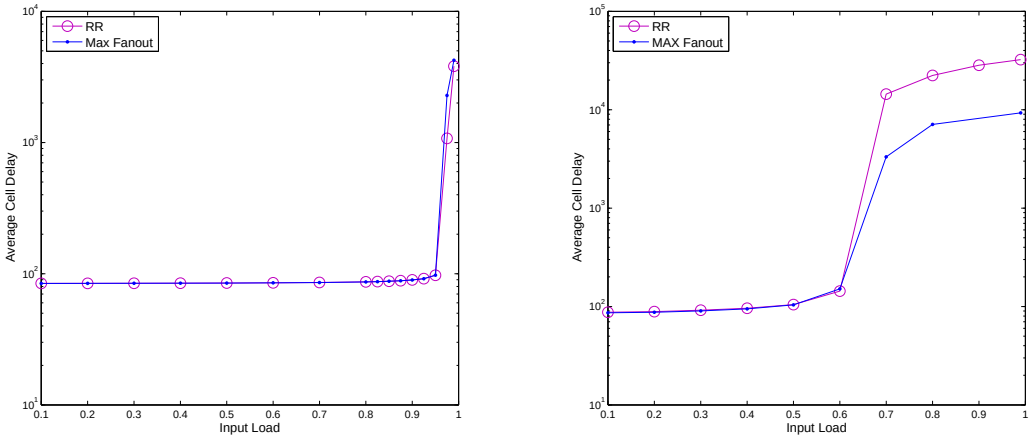


(a) UDN (b) MDN

Figure 8.16: 32x32 Switch under Diagonal traffic for different algorithms.
$<$ x, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = x, SA = FIFO:RR $>$

Conclusion: Simplified Modulo multicast and Modulo multicast have a similar performance in both UDN and the MDN fabrics.

In the next figures, we vary the arbitration algorithm of the switch employing SP1. Previously RR was implemented. Now the arbiter selects the packet that has the higher number of bits to the desired output port of the switch. If two packets are trying to exit through the North port, the arbiter will select the packet that has more bits to "1" for that port. Figure 8.17(b) shows the results under Bernoulli Uniform traffic and Figure 8.17(b) under Bursty Uniform traffic. They show that under Bernoulli traffic, performance does not vary. Under Bursty traffic , to select the maximum fanout improves slightly the average cell delay. The few number of ports of the switch does not make a big difference between these two algorithms.



(a) Bernoulli Uniform Traffic

(b) Bursty Uniform Traffic

Figure 8.17: 32x32 Switch with different Arbiter Algorithms.
$< x$, SS = 32, SP = x, D = 32 , P = 1 ,BD = 4, RA = Simplified Modulo Mcast, SA = $x >$

8.5 Conclusions

In this chapter we tested the proposed architectures under multicast traffic and compared their performance to the traditional CICQ crossbar. Several algorithms were tested for all the architectures. Simulations suggest that when MDN and UDN are employing SP2, they outperform CICQ for Uniform and Non-uniform traffic. Modulo algorithm and Simplified Modulo multicast algorithm have similar performance. This performance is better than the one showed by the Copy Multicast.

9 Conclusions and Future Work

This chapter presents the conclusions for the performed research. First, a summary of this thesis is presented. Then, the main contributions of our work are discussed. Finally, suggestions for future work are proposed.

9.1 Summary

The subject of this thesis was to propose the use of NoC for the crossbar switch fabrics in the high-speed routers. The development of new transmission techniques in the Internet increased data-rate to the range of multi-terabits per second. Routers inside the network must be capable of reaching these speeds to avoid bottlenecks.

The architecture of routers has evolved along the years, but their basic function remains unchanged. These basic tasks are scheduling and data forwarding. Data forwarding is the process of delivering the packets from inputs to outputs. Switch fabrics are usually responsible of this assignment. Compared to shared buffer and shared memory switches, the buffered crossbar switch fabrics is the most popular design. Its scalability in relation to its predecessors and its higher speed made of them the favorite election for high-speed routers. This switch fabrics, however, have several drawbacks that should be taken into account. Firstly, they require expensive on-chip buffers, whose cost grows quadratically with the port count. Moreover, point-to-point switching commands the use of long wires to interconnect inputs and outputs that result in non-negligible delays.

In this thesis we proposed two different NoC architectures to overcome the disadvantages of current CICQ crossbars. We claim that NoCs have the necessary characteristics to enhance the performance of the current switch fabrics. They use short wires leading to a tighter synchronization. Global arbitration is replaced by a local arbitration entailing less delays. As no errors happen in the transmission, reordering modules are not necessary if packets are sent in order through the mesh.

Using a network allows different paths per flow in the system, achieving a higher load balancing and path diversity. HoL blocking is reduced and no VOQ are needed. Instead, simple FIFOs can be implemented.

This thesis begins with an introduction that gives a general view of our research. We present the actual switch fabrics and show their limitations, that motivate our thesis. Chapter 2 provides the background of the work and explains the differences between NoCs and Internet networks. Chapter 3 presents the first of our two proposed architectures. In this chapter, we analyze a possible solution for a NoC switch fabric. This new architecture, referred as UDN, is analyzed analytically and by simulations. Simulations conclude that it has better performance than the traditional CICQ crossbar and it accomplishes the reasons we argued in the beginning of this thesis to use NoCs. Those were: 1) Better load balancing; 2) Higher path diversity; 3) Scalability in port count and speed per port; 4) Use of short wires; 5) Easier synchronization; 6) Simpler switch design by using FIFOs

Finally, we conclude that this architecture is not suitable for the layout of the fabric chip. Some modifications should be done to this architecture to better fit the pin distribution and

avoid the use of long wires to connect the switch.

In Chapter 5 another architecture is proposed. Now the distribution of the pins is around the perimeter of the switch. This fabric, called MDN, is studied by simulations. It results in outperforming both UDN and CICQ when it employs SP2. Our study shows that MDN is a better cost/performance option than UDN. It requires less crosspoints and its size does not increase quadratically with the number of ports. Using NoC reaches higher frequencies and average cell delay is reduced for both UDN and MDN.

Finally, Chapter 8 addresses the problem of multicast traffic flows. Three algorithms are proposed for UDN and MDN. They are all based on fanout-splitting policies. In the first algorithm, the packets are converted to unicast packets in the NIs. The other algorithms make the copies of the cells inside the network. This results in a better use of resources as less packets are spread in the mesh.

9.2 Main contributions

We presented two different NoC architectures to perform as a switch fabric in high-speed routers. In the introduction of this thesis, we aimed to prove that the following characteristics were part of a buffered crossbar based on NoC.

- Better load balancing
- Higher path diversity
- Scalability in port count and speed per port
- Use of short wires
- Simpler switch design by using FIFOs

UDN: we designed a unidirectional mesh scalable in its number of columns. Its NoC characteristics make of it a promising architecture in terms of speed and performance. Its architectural parameters were adjusted to find the best cost/performance solution.

After the experiments done all along this thesis in UDN, we concluded:

- Better load balancing was achieved with the Modulo Algorithm 1 and the multi-hop structure of the mesh. This was depicted in Figures like 3.19 and 3.20 and in conclusion 3.3.
- Path diversity was enforced by varying the path of each flow. The Modulo Algorithm 1 could modify the path per flow reaching a higher load balance (see Figure 3.20).
- It was proved analytically and by simulations that increasing the switch size does not deteriorate the performance of the system (see Figures 4.11(a), 4.21) and that delay increases linearly with the size of the switch. These facts were shown in conclusion 4.10, and conclusion 4.17. The inter routers wires are kept of short length, independently of the switch size.

- The Analytical study of section 3.2.3 and Figure 3.15 showed how HoL blocking was not an issue in UDN mesh and the routing algorithm employed FIFOs queues could be used avoiding to use VOQ both in the line cards and in the routers. This was shown in conclusion 3.1.

The trade off of the UDN parameters is shown in table 9.1.

	Performance	Average Cell Delay	Area	Cost
↑Speedup	↑(SP2 enough)	↓	=	↑
↑Switch Size	↑	↓	↑	↑
↓Depth (not for Bursty)	↓ if SP1	↑	↓	↓
	$\simeq$ if SP2 and $\frac{N}{M} < 5$	↓	↓	↓
	↓ if SP2 and $\frac{N}{M} \geq 5$	↑	↓	↓
↓Buffer Size	$\simeq$ if SP>1 and N=M	$\simeq$	↓	↓
	else ↓	↑	↓	↓
FIFO OCF Arb.	↓	↑	↑	↑
Multiport RAM OCF Arb.	↑	↓	↑	↑

Table 9.1: UDN performance as function of different parameters.

After the simulations, we concluded that these parameters do not have the same importance in affecting the performance of the system. Their weight in the system was obtained by the conclusions. Gathering conclusions 4.12, 4.13, 4.14 and 4.18:

We concluded that the weight of the parameters is: 1)Speedup; 2)Depth; 3)Buffer Size; 4) Scheduling algorithm.

Finally, we found that though UDN was the logical first architecture to implement, its physical placement in the chip layout was not appropriate. Then, we introduced our second proposal, MDN.

MDN: we enhanced our first architecture in terms of port scalability and distribution of the pins in the chip. A $N/4 \times N/4$ architecture variable in the number of planes was presented.

The MDN has the following properties:

- Load balanced was kept as it was shown in conclusion 6.7.
- Higher path diversity was maintained as MDN Modulo algorithm allows to modify the path per flow.
- Scalability, was now higher because fewer number of routers were needed to have full throughput. See conclusion 6.9.
- No VOQs are used neither in the line cards nor in the routers.

We also concluded that this was a better architecture. Its input/output pins allowed to have less average cell delay and it fit with the chip layout. The number of planes could be incremented to solve the problem of the low ratio routers/input ports.

Table 9.2 gathers the best cost/performance values for the MDN architectural parameters.

	Performance	Average Cell Delay	Area	Cost
↑Speedup	↑(SP2 enough if switch size <64x64)	↓	=	↑
↑Switch Size	↓	↑	↑	↑
↑Planes	↑(3 enough)	↓	↑	↑
↑Buffer Size	↑(20 enough)	↓	↑	↑

Table 9.2: MDN performance as function of different parameters

Many parameters were varied along the simulations. From conclusions 6.6,and 6.12, we can see which of these parameters had a higher effect.

Then, if we order the parameters in terms of which causes a major enhancement in the switch: 1) Speedup; 2) Number of planes; 3) Buffer depth.

UDN VS MDN: we compared our two proposed architectures in terms of cost/performance. MDN outperforms UDN as it needs fewer routers to achieve the same behavior as UDN. The position of the inputs/outputs facilitates the communication between the NIs and the placement in the chip.

MULTICAST: we tested both UDN and MDN under multicast traffic. We concluded that their performance was at least as good as that of CICQ crossbar. Both proposed architectures responded well under multicast traffic due to the few HoL blocking of the system. We implemented three different algorithms for routing the multicast traffic. We concluded that Copy multicast had the worse performance and that Modulo multicast algorithm and Simplified Modulo multicast algorithm had a similar response.

To summarize the research done in this thesis we can conclude that the proposed two novel architectures to use as switch fabrics in high-speed routers are feasible. Simulations suggest that their area and performance are better than the ones presented by the CICQ crossbar architecture. MDN outperforms UDN in scalability and cost for unicast and multicast traffic. It was shown to have smaller average cell delay when the system is not congested. For high loads, MDN needs to add planes to behave as well as UDN.

9.3 Future Work

Some features require further study to have a deeper knowledge of these architectures.

End-to-end flow control: First of all, End-to-end flow control should be implemented to have a control of the load in the network when NIs are not considered infinite.

Improved performance: Bursty traffic is the kind of traffic with the worse performance in our system. New methods to improve it can be studied. For example, adding VOQ to the NIs or to allow the disorder of packets. Also a deeper research in multicast traffics and combining it with unicast flows.

Guarantees: the way to provide Guaranteed Throughput or Quality of Services is a main factor in nowadays networks and this kind of traffic was not taken into consideration during this thesis. For this purpose, recovery to failure can be included as future work in these architectures.

Bibliography

[1] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, "The Tiny Tera: A Packet Switch Core," *IEEE Micro*, pp. 26–33, January/February 1997.

[2] F. Abel, C. Minkenberg, P. Luijten, M. Gusat, and I. Iliadis, "A Four-Terabit Packet Switch Supporting Long Round-Trip Times," *IEEE Micro*, vol. 23, no. 1, pp. 10–24, January/February 2003.

[3] L. Mhamdi, "A Partially Buffered Crossbar Packet Switching Architecture and Its Scheduling," in *Proceeding of IEEE International Symposium on Computers and Communications (ISCC)*, July 2008.

[4] N. McKeown, "Scheduling Algorithms for Input-Queued Cell Switches," Ph.D. dissertation, University of California at Berkeley, May 1995.

[5] L. Mhamdi and M. Hamdi, "MCBF: A High-Performance Scheduling Algorithm for Buffered Crossbar Switches," *IEEE Communications Letters*, vol. 07, no. 09, pp. 451–453, September 2003.

[6] R. R. Cessa, "Design and Analysis of Reliable High-Performance Packet Switches," Ph.D. dissertation, Plytechnic University, April 2001.

[7] T. Aramaki, H. Suzuki, S. Hayano, and T. Takeuchi, "Parallel 'ATOM' Switch Architecture For High Speed ATM Networks," *IEEE International Conference on Communications (ICC)*, pp. 250–254, 1992.

[8] M. Devault, J. Y. Cochennec, and M. Servel, "The Prelude ATD Experiment: Assessments and Future Prospects," *IEEE Journal on Selected Areas in Communications*, vol. 06, no. 09, pp. 1528–1537, December 1998.

[9] K. K. Chang, S. Chuang, N. McKeown, and M. Horowitz, "A 50 Gb/S 32x32 CMOS Crossbar Chip Using Asymmetric Seriallinks," *Symposium on VLSI Circuits*, pp. 19 – 22, 1999.

[10] R. Rojas-Cessa and Z. Dong, "Combined Input-Crosspoint Buffered Packet Switch with Flexible Access to Crosspoint Buffers," *IEEE International Caribbean Conference on Devices, Circuits and Systems, Playa del Carmen*, April 2006.

[11] N. Chrysos and M. Katevenis, "Scheduling in Switches with Small Internal Buffers," *IEEE Globecom*, pp. 614–619, November 2005.

[12] B. Vermeulen, J. Dielissen, K. Goossens, and C. Ciordas, "Bringing communication networks on chip: Test and verification implications," *IEEE Communications Magazine*, vol. 41, pp. 74–81, 2003.

[13] Arteris, "A comparison of Network-on-Chip and Busses," *White Papers*, 2005.

[14] A. Radulescu and K. Goossens, "Samos, ii(),communication services for networks on chip," pp. 275–299.

[15] A. Rdulescu, J. Dielissen, S. G. Pestana, O. P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Trans. on CAD of ICs and systems*, vol. 24, p. 2005, 2005.

[16] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 775–785, 1990.

[17] A. Jantsch and H. Tenhunen, *Networks on Chip*. Kluwer, 2003.

[18] K. Goossens, J. Dielissen, and A. Radulescu, "Aethereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test*, vol. 22, no. 5, pp. 414–421, 2005.

[19] P. Wielage, E. Marinissen, M. Altheimer, and C. Wouters, "Design and dft of a high-speed area-efficient embedded asynchronous fifo," *Design, Automation and Test in Europe Conference and Exhibition*, vol. 0, p. 160, 2007.

[20] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed Bandwidth using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip," *Design, Automation and Test in Europe Conference Proceedings, IEEE,*, February 2004.

[21] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. D. Micheli, "Design, synthesis, and test of networks on chips," *IEEE Des. Test*, vol. 22, no. 5, pp. 404–413, 2005.

[22] T. G. Mattson, R. V. der Wijngaart, and M. Frumkin, "Programming the intel 80-core network-on-a-chip terascale processor," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–11.

[23] N. Kavaldjiev, G. J. M. Smit, P. G. Jansen, and P. T. Wolkotte, "A virtual channel network-on-chip for gt and be traffic," in *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*. Washington, DC, USA: IEEE Computer Society, 2006, p. 211.

[24] T. Bjerregaard, "The MANGO clockless network-on-chip: Concepts and implementation," Ph.D. dissertation, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2005, supervised by Assoc. Prof. Jens Sparsø, IMM. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?4025

[25] L. G. Roberts, "Data by the Packet," vol. 11, no. 2, pp. 43 –51, February 1974.

[26] B. Leiner, V. Cerf, D. Clark, R. Khan, L. Kleinrock, D. Lynch, and J. Postel, "The Past and Future History of the Internet," vol. 40, no. 2, pp. 102–108, February 1997.

[27] H.-C. Chi, C.-M. Wu, and S.-T. Wu, "A switch supporting circuit and packet switching for on-chip networks," in *DDECS*, 2006, pp. 226–227.

[28] E. Rijpkema, K. G. W. Goossens, A. Rdulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterl, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," 2003, pp. 350–355.

[29] Chang and L. C-J, "Overflow controller in copy network of broadband packet switch," *Electron. Lett*, vol. 27, no. 11, pp. 937–939, 1991.

[30] J. Bianchini, R.P. and H. KIM, "Design of a nonblocking shared/memory copy network for ATM," *Proceedings of IEEE Infocom '92*, pp. 876–885, 1992.

[31] w. Zhong, Y. Onozato, and J. Kaniyil, "A copy network with shared buffers for larte/scale multicast ATM switching," *IEEE/ACM Trans. Networking*, vol. 1, no. 2, pp. 157–165, 1993.

[32] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, vol. 35, no. 9, pp. 1337–1356, December 1987.

[33] Y. Tamir and G. Frazier, "High performance multi-queue buffers for VLSI communication switches," *Proc. 15th Annyu. Symp. comput. Arch.*, pp. 343–354, June 1988.

[34] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in Input-Queued Switch," *IEEE Transastions On Communications*, vol. 47, no. 08, 1999.

[35] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm for achieving 100San Francisco, vol. 2, pp. 117–129, 1998.

[36] N. McKeown, "iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions On Networking*, vol. 07, no. 02, pp. 188–201, April 1999.

[37] R. Rojas-Cessa, "High-Performance Round-Robin Arbitration Schemes for Input-Crosspoint Buffered Switches," pp. 19–21, April 2004.

[38] H. T. Kung and R. Morris, "Credit-Based flow control for ATM Networks," *IEEE Network Magazine*, March 1995.

[39] P. Kermani and L. Kleinrock, " Virtual cut-through: A new computer communication switching technique ," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

[40] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the Proximity congestion Awareness in a Network on Chip," *Proceedings of th eDesign, Automation and Test in Europe Conference and Exhibition*, pp. 1530–1591, 2003.

[41] W. Dally and C. Seitz, " Deadlock-free message routing in multiprocessor interconnection Networks," *IEEE Trans. Comput.*, pp. 547–553, May 1987.

[42] J. Duato, " A necessary and sufficient condition for dead lock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel Distrib. Systems*, pp. 1055–1067, October 1995.

[43] J. Kim, Z. Liu, and A. Chien, " Compressionless routing: A framework for adaptive and fault-tolerant routing," *Proceedings of the 21st International Symposium on Computer Architecture," IEEE Computer Society, Los Alamitos, CA*, pp. 289–300, April 1994.

[44] K. V. Anjan, T. M. Pinkston, and Disha, " A deadlock recovery scheme for fully adaptive routing," *Proceedings of the 21st International Symposium on Computer Architecture," IEEE Computer Society, Los Alamitos, CA*, pp. 201–210, June 1995.

[45] A. A. Chien, " A cost and speed model for k-ary n-cube wormhole routers," *Proceedings of the Symposium on Hot Interconnects*, August 1993.

[46] D. Xiang, Y. Zhang, Y. Pan, and J. Wu, " Deadlock-Free Adaptive Routing in Meshes Based on Cost-Effective Deadlock Avoidance Schemes," *International Conference on Prallel Processing*, 2007.

[47] X. Liu, S. Zhang, and T. J. Li, "A cost-effective load balanced adaptive routing scheme for mesh-connected networks," *Proc. of 8th Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pp. 532–538, 2000.

[48] "SIM," High-Performance Networking Group, Stanford University http://klamath.stanford.edu/tools/SIM/.

[49] S. Kumar, R. Venkatesh, J. Philip, , and S. Shukla, "Implementing parallel packet buffering."

[50] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate soc design and verification," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1182–1187.

[51] P. Giaccone, D. Shah, and B. Prabhakar, "An implementable Paraller Scheduler for Input-Queued Switches," *IEEE Micro*, vol. 19, no. 01, pp. 1090–1096, January/February 1999.

[52] R. Rojas-Cessa, Z. J. E. Oki, and H. J. Chao, "CIXB-1: Combined Input One-Cell-Crosspoint Buffered Switch," *IEEE Workshop on High Performance Switching and Routing (HPSR)*, pp. 324–329, 2001.

[53] X. Liu and H. T. Moufath, "Overflow control in multicast networks," *Proc. of Canadian Conf. on Eleectrical and computer Engineering*, pp. 542–545, 1993.

[54] M.-H. Guo and R.-S. Chang, "Multicast atm switches: survey and performance evaluation," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 2, pp. 98–131, 1998.

[55] L. Mhamdi and M. Hamdi, "Scheduling Multicast Traffic in Internally Buffered Crossbar Switches," *IEEE International Conference on Communications (ICC)*, pp. 1103–1107, June 2004.

[56] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast Scheduling for Input-Queued Switches," *IEEE Journal in Selected Areas in Communications (JSAC)*, vol. 15, pp. 855–866, June 1997.

[57] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 465–477, 2003.

[58] P. Giaccone and E. Leonardi, "Asymptotic Performance Limits of Switches with Buffered Crossbars Supporting Multicast Traffic," *IEEE Infocom*, pp. 1–10, April 2006.

[59] Andrea Bianco, Paolo Giaccone, Chiara Piglione, Sonia Sessa, "Practical Algorithms for Multicast Support in Input Queued Switches ," *IEEE, High Performance Switching and Routing*, June 2006.

Simulation Environment

A

This appendix describes the simulation environment in which the proposed architectures and the CICQ crossbar are tested. It introduces the software simulation tool and the traffic models to run the experiments. Finally it describes the parameters we use to value the performance of the switch.

A.1 Simulator

For the simulations of the architectures, a simulator of Stanford University was used [48]. It is a slotted time simulator, a fixed-length packet switch simulator written in ANSI C. It can be used to evaluate the performance of a variety of switching architectures using different queuing, scheduling or transmission schemes, as specified by the user. The new architectures, routing algorithms and types of traffic where included in this simulator to perform the UDN and MDN architectures.

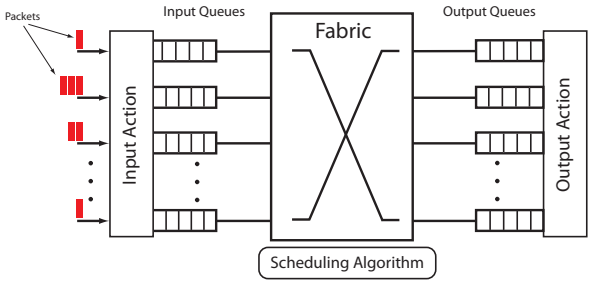


Figure A.1: Architecture of SIM.

Figure A.1 represents the schematic of the switch architecture assumed by the simulator. Packets arrive to the Input Action module that decides whether to accept them and in which input queue place them. Note that packets arrive (depart) at most one per input (output) per time slot . Input queues are one FIFO per input port. The fabric interconnects input and output ports. Here, we add our MDN and UDN architectures. The scheduling algorithm decides what packet is transmitted each time slot. For UDN and MDN architectures it is not used, because each router makes the scheduling locally. However, it is used when simulating the CICQ crossbar. The fabric transmits the packets to the output queues, that are implemented as simple FIFOs. Finally, the Output Action removes the packets from the system.

A.2 Traffic models

Next, we describe the traffic models used to test the crossbar fabrics ant their motivation. Throughout the simulations, we used both uniform and non-uniform traffic models as described below.

Uniform traffic is represented by Bernoulli and Bursty traffic. Diagonal and Unbalanced traffic are the non-uniform traffic tested.

Bernoulli Uniform traffic

Bernoulli Uniform traffic is commonly used to test the performance of switches. In each time-slot a cell is generated with probability p. As it is uniform traffic, this load is equally distributed among the N destinations.

Bursty Uniform traffic

Bursty traffic is closely related to real Internet traffic. Cells are highly correlated in the internet [4] and they arrive in bursts.

The transmission of the voice signals constitutes a big part of the bandwidth in the Internet. During this transmissions there are intervaled periods of silence with periods of speech. This periods make the bursty traffic. Studying this type of traffic is then essential in Internet networks.

Bursty traffic is generated from one source with alternating series of "ON" and "OFF" periods. During the "ON" state, cells are generated each time-slot. During the "OFF" periods, no traffic is generated. The average number of cells generated in the "ON" state follow a geometric distribution known as the burst size and denoted by b.

Non uniform traffic varies its distribution of the probability of sending one cell from one input port to a determined output port. Client-server applications follow this kind of traffic. There is a small number of clients that are connected to a determined server. In this thesis we are using two kinds of non-uniform traffic: unbalanced traffic and diagonal traffic.

Unbalanced traffic

The Unbalanced traffic is defined by using an unbalanced probability, w. For a N $\times$ N switch, the traffic load at each input port is defined by ρ. Then, for each input port s and output port d, the traffic load, $\rho_{s,d}$, is given by:

$$\rho_{s,d} \begin{cases} \rho(w + \frac{1-w}{N}) \\ \rho \frac{1-w}{N} \quad otherwise \end{cases}$$

When w is fixed to 0 it is uniform traffic and when it is fixed to 1, traffic is completely unbalanced.

Double Diagonal traffic

The Diagonal traffic is defined as in the following example:

$$\frac{1}{3} \begin{pmatrix} 2\rho & \rho & 0 & 0 \\ 0 & 2\rho & \rho & 0 \\ 0 & 0 & 2\rho & \rho \\ \rho & 0 & 0 & 2\rho \end{pmatrix}$$

In this kind of traffic, input i only sends packets to output i and i+1. It sends $\frac{2}{3}$ of the load to i and $\frac{1}{3}$ to i+1.

A.3 Performance parameters

The main part of this thesis is constituted by the simulation experiments of UDN, MDN and CICQ architectures. In this section, we explain the different parameters to meassure and the variables used to quantify its behavior.

Speedup
The speedup of a switch is defined as the speed ratio at which the switch fabric can run with respect to the input/output ports. A switch with speedup S can remove up to S packets from one input and send up to S packets to one output per time slot.

In our simulations, we refer to the speedup as SP, where SP1 refers to S=1 and SP2 to S=2.

Through the simulations two parameters of the network are obtained: average cell delay and throughput.

Average cell delay
The average cell delay is the time spent by every cell entering the switch, averaged over all cells. The cell delay includes the time spent by the cell inside the switch queues until it reaches its output port. This time includes the waiting in the NIs and the time it needs to travel through the switch.

Throughput
The throughput is the ratio between the input load and the output load. It is a meassure of the percentage of traffic that the network can manage. The maximum throughput is the value for which the network becomes congested. Congestion happens when the input load is higher than the throughput of the switch. Then, queues grow indefinitely. If the value of that saturation is 1, that means the switch achieves 100% throughput. When comparing two systems with 100% throughput, the one with lower average cell delay is better. A scheduling algorithm is considered stable if it provides 100% throughput.

In the following sections we describe the main performance parameters of both UDN and MDN.

A.3.1 UDN

We present the architectural parameters that constitute the UDN architecture and we motivated why we study them in our simulations.

- N: number of rows in the UDN mesh. It coincides with the number of input and output ports.
- M, depth: number of columns in the UDN mesh. Depth is a trade off factor. Reducing it helps to reduce the cost of the switch.

- Buffer depth, size: capacity in number of packets of each input buffer of each UDN router.

During the simulations these parameters are varied to find the best cost/performance option for UDN. N is increased to test if switch size affects the performance of the system. When $N = M$, the ratio inputs/routers is kept though the switch size increases. Reducing the depth of the system, less routers are needed, reducing the cost of the switch. Buffer depth also plays an importante role in the design of the crossbar. Having higher depths entail more area and better performance.

A.3.2 MDN

MDN parameters are listed below:

- N: number of inputs/outputs of the switch. The mesh is then a $N/4 \times N/4$ square.
- Planes, layers: number of $N/4 \times N/4$ interconnected in MDN. Adding layer increases the ratio inputs/routers in the mesh and helps to improve performance.
- Buffer depth, size: capacity in number of packets of each input buffer of each UDN router.

Throughout the simulations we modify N to check the switch stability under bigger switch sizes. MDN has fewer routers compared to UDN for the same number of inputs, hence, the number of planes can be increased. A NxN MDN mesh with 16 planes has the same number of routers than a NxN UDN switch. Here, as in UDN, buffer depth is another trade off. Finding the most appropriate size for the buffer is very important in terms of area and performance.

Algorithms

B

In this appendix we show the complete code of the algorithms for UDN and MDN.

B.1 UDN algorithm

B.1.1 MODULO UDN

The following algorithm is performed by $Router[i, j]$ that receives a packet whose desired exit is *output*.

```
Switch(Packet Buffer Input)
    case(North):
        if(i == output) then East
        else South
    case(South):
        if(i == output) then East
        else North
    case(West):
        if ((output%M) == (N-i+j+t)%M) then
            if(output ¿ j) Down
            else North
        else East
```

B.2 MDN algorithms

B.2.1 MODULO MDN

The following algorithm is performed by $Router[i, j]$ that receives a packet whose desired exit is *output*.

```
Switch(packet condition):
    case(new packet || packet going to its destined plane):
        case(output even):
            case(# planes even):
                if ((N − j + P + t)%P_m) + P_m = (output%P_m) + P_m then algorithm 2
                else Up
            case(# planes odd):
                if ((N − j + P + t)(%P_m + 1)) + P_m = (output(%P_m + 1) + P_m then algorithm 2
                else Up
        case(output odd):
            if (N − j + P + t)%(P_m + 1) = (output)%(P_m + 1) then algorithm 2
```

```
        else Down
    case(packet inside a plane):
        if (!edge of the plane) algorithm 2
        else
            if(P < P_m) then Up
            if(P > P_m) then Down
            if(P = P_m) then algorithm 2
    case(packet going back to the center plane):
        if(P < P_m) then Up
        if(P > P_m) then Down
        if(P = P_m) then algorithm 2
```

B.2.2 MDN MODULO AND XY

Router in coordinate $< i, j >$ receives a packet whose destination is in coordinate $< x, y >$ in the mesh.

```
Switch(Packet Input Port)
    case(North):
        Switch(Packet Output Port)
            case(North):
                Switch(Packet Buffer Input):
                    case(North):
                        if(j == y) then North
                        if(j < y) then East
                        if(j > y) then West
                    case(East):
                        if(j == y) then North
                        if(j > y) then West
                    case(West):
                        if(j == y) then North
                        if(j < y) then East
            case(South):
                Switch(Packet Buffer Input)
                    case(West):
                        if(j == y) then South
                        else East
                    case(East):
                        if(j == y) then South
                        else West
                    case(North):
                        if(j == y) then South
                        else if (((x)%M) == (N-i+j+t)%M) then
                            if(j > y) then West
                            if(j < y) then East
                        else South
```

```
    case(East):
        Switch(Packet Buffer Input)
            case(West):
                East
            case(North):
                if (i == x) then East
                else South
    case(West):
        Switch(Packet Buffer Input)
            case(East):
                West
            case(North):
                if (i == x) then West
                else South


case(South):
Switch(Packet Output Port)
    case(North):
        Switch(Packet Buffer Input):
            case(South):
                if (j == y) then North
                else if (((x)%M) == (N-i+j+t)%M) then
                    if (j < y) then East
                    if (j > y) then West
                else North
            case(East):
                if (j == y) then North
                else West
            case(West):
                if (j == y) then North
                else East
    case(South):
        Switch(Packet Buffer Input):
            case(South):
                if (j == y) then South
                if (j < y) then East
                if (j > y) then West
            case(East):
                if (j == y) then South
                if (j > y) then West
            case(West):
                if (j == y) then South
                if (j < y) then East
    case(East):
        Switch(Packet Buffer Input):
            case(South):
                if (i == x) then East
```

```
                else North
            case(West):
                East
    case(West):
        Switch(Packet Buffer Input)
            case(South):
                if (i == x) then West
                else North
            case(East):
                West


 case(East):
Switch(Packet Output Port)
    case(North):
        Switch(Packet Buffer Input):
            case(East):
                if (j == y) then North
                else West
            case(South):
                North
    case(South):
        Switch(Packet Buffer Input):
            case(East):
                if (j == y) then South
                else West
            case(North):
                South
    case(East):
        Switch(Packet Buffer Input):
            case(East):
                if (i == x) then East
                if (i < x) then South
                if (i > x) then North
            case(North):
                if (i == x) then East
                else South
            case(South):
                if(i == x) then East
                else North
    case(West):
        Switch(Packet Buffer Input)
            case(East):
                if (i == x) then West
                else if (((y)%M) == (N-i+j+t)%M) then
                    if (i < x) then South
                    if (i > x) then North
                else West
```

```
            case(North):
                if (i == x) then West
                else South
            case(South):
                if (i == x) then West
                else North


 case(West):
Switch(Packet Output Port)
    case(North):
        Switch(Packet Buffer Input):
            case(West):
                if (j == y) then North
                else East
            case(South):
                North
    case(South):
        Switch(Packet Buffer Input):
            case(West):
                if (j == y) then South
                else East
            case(North):
                South
    case(East):
        Switch(Packet Buffer Input):
            case(West):
                if (i == x) then East
                else if (((y)%M) == (N-i+j+t)%M) then
                    if (i < x) then South
                    if (i > x) then North
                else East
            case(South):
                if (i == x) then East
                if (i > x) then North
            case(North):
                if (i == x) then East
                if (i < x) then South
    case(West):
        Switch(Packet Buffer Input)
            case(West):
                if (i == x) then West
                if (i < x) then South
                if (i > x) then North
            case(South):
                if (i == x) then West
                else North
            case(North):
```

```
if (i == x) then West
else South
```

Nomenclature

ACK	Acknowledgement
ANSI C	American National Standards Institute for C programming language
ASIC	Application-Specific Integrated Circuit
AT	Average Throughput
ATM	Asynchronous Transfer Mode
BE	Best Effort
CICQ	combined input-crosspoint queueing
CMOS	Complementary metaloxidesemiconductor
CPU	Central Processing Unit
DRAM	Dynamic random access memory
FDM	Frequency Division Multiplexing
FIFO	First in first out
GT	Guaranteed Throughput
(HDLC)	High-Level Data Link Control
HoL	Head of Line
IP	Internet Protocol
IQ	Input Queuing
ISDN	Integrated Services Digital Network
ISLIP	Iterative Round Robin matching with Slip
(LLFC)	Link Level Flow Control
LPF	Longest Path First
LQF	Longest Queue First
MC-VOQ	Multicast VOQ
MDN	Multidirectional NoC
MSM	Maximum Size Matching
MWM	Maximum Weight Matching
NACK	Negative acknowledgement
NI	Network Interface
NoC	Network on Chip
OCF	Oldest Cell First
OQ	Output Queuing
RAM	Random-Access Memory
(KTH)	Royal Institute of Technology
RR	Round Robin

RTL	Register transfer language
SoC	System on Chip
SP	Speedup
TDM	Time Division Multiplexing
UDN	Unidirectional NoC
VC	Virtual Channel
VC	Virtual Circuits
VHDL	VHSIC hardware description language
VHSIC	Very-High-Speed Integrated Circuits
VOQ	Virtual Output Queueing
WDM	Wavelength Division Multiplexing

Index