UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTMENT OF COMPUTER ENGINEERING

Design of Efficient Packet Marking-based Congestion Management Techniques for Cluster Interconnects

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

(COMPUTER SCIENCES)

Author:

JOAN-LLUÍS FERRER I PÉREZ

Advisors:

DR.ELVIRA BAYDAL CARDONA,

DR.ANTONIO ROBLES MARTÍNEZ,

DR.PEDRO JUAN LÓPEZ RODRÍGUEZ.

VALÈNCIA, 2012

Dedication

Aquest treball està dedicat a tots els que esteu sempre al meu costat.

De manera especial a Mar. @>–>—

“Tenim tantes coses a celebrar...”

Als meus Pares i Germans per tot. :*)

“...ja s’ha acabat això del Doctorat”

Als meus nebots Adrià, Olaf, Aitana, Clàudia i Arianna. ;-)

“*Per la tendresa que vosaltres sabeu donar,*

perquè teniu la gràcia per arroves,

perquè sou la salsa de la família.”

“El Tio Padrí”

Acknowledgments

A Elvira, Antonio y Pedro.

A Elvira por la dedicación durante todo este tiempo. Te agradezco tus sugerencias, comentarios y anotaciones. Ha sido un placer trabajar bajo tu dirección, así es que voy a hacer buena propaganda y creo que no te van a faltar doctorandos. ;-)

A Antonio por haber encaminado la tesis desde el principio. Además, por darle ese toque personal a los papers que hemos confeccionado y que tu has revisado. Antonio eres un repositorio inacabable de ideas, sugerencias y vocabulario.

A Pedro, de manera muy especial, el haber despertado en mi el interés por la arquitectura de computadores desde que asistí a sus clases por primera vez allá por el año..., bueno mejor no recordarlo. Además, por haber aceptado dirigirme el PFC y que tanto disfruté. No se si recordarás que tuve que insistirte pues tu ibas muy liado con tu tesis y no tenías tiempo. Ahora lo entiendo!!!. Además, por haberme facilitado siempre todo desde que decidí continuar los estudios de doctorado y realizar la tesis con vosotros. Ha sido un privilegio trabajar con todos vosotros.

Finalmente, agradecer al GAP su apoyo y ayuda cuando lo he necesitado. Especialmente a Ricardo Marín que siempre me ha solucionado mis dudas técnicas. También a administración del departamento por todas las facilidades en la confección de la documentación que he necesitado pues aunque no he pertenecido a la UPV, gracias a ellos no he sido un “sin papeles”.

Contents

Preface

The current document has been elaborated with the aim to obtain the PhD degree. This work has been done under the advising and guidance of professors Elvira Baydal, Antonio Robles, and Pedro López.

This document can be divided in three parts. The first part introduces the state-of-art in the interconnection networks field, focusing on congestion management mechanisms.

The second part presents all the research work and its evaluation. This research activity has been developed within the Parallel Architectures Group (GAP) in the Department of Computer Engineering (DISCA) at the Universitat Politècnica de València.

In the last part, the contributions and conclusions are summarized. Related publications of the contributions are also presented together with future directions in the research.

Abstract

The growth of parallel computers based on high-performance networks has increased the interest and effort of the research community in developing new techniques to achieve the maximum performance from these networks. In particular, the development of new techniques for efficient routing to reduce packet latency and increase network throughput. However, high utilization rates of the network could result in what is known as *network congestion*, which could cause a degradation of the network performance because all, or a part of the network has exceeded the maximum utilization, which is imposed by the saturation point of the network.

Congestion management in multistage interconnection networks is a serious problem not completely solved. In order to avoid the degradation of network performance when congestion appears, several congestion management mechanisms have been proposed. Most of these mechanisms are based on explicit congestion notification. For this purpose, switches detect congestion and depending on the applied strategy, packets are marked to warn the source hosts. In response, source hosts apply some corrective actions to adjust their packet injection rate. Although these proposals seem quite effective, they either exhibit some drawbacks or are partial solutions. Some of them introduce some penalties over the flows not responsible for congestion, whereas others can cope only with congestion situations that last for a short period of time.

The aim of this dissertation is to analyze the different strategies to detect and correct congestion in multistage interconnection networks and propose new congestion management mechanisms targeted to this kind of lossless networks. The new approaches will be based on a more refined packet marking strategy combined with a

fair set of corrective actions in order to make the mechanisms capable of effectively managing congestion regardless of the congestion degree and traffic conditions.

Resum

El creixement dels computadors paral·lels basats en xarxes d'altes prestacions ha augmentat l'interés i esforç de la comunitat investigadora a desenvolupar noves tècniques que permeten obtindre el millor rendiment d'estes xarxes. En particular, el desenvolupament de noves tècniques que permeten un encaminament eficient i que reduïxquen la latència dels paquets, augmentant així la productivitat de la xarxa. No obstant això, una alta taxa d'utilització d'esta podria comportar el que es coneix com a *congestió de xarxa*, el qual pot causar una degradació del rendiment, perquè tota o part de la xarxa ha excedit la utilització màxima, el valor de la qual ve imposat pel punt de saturació.

El control de la congestió en xarxes multietapa és un problema important que no està completament resolt. A fi d'evitar la degradació del rendiment de la xarxa quan apareix congestió, s'han proposat diferents mecanismes per al control de la congestió. Molts d'estos mecanismes estan basats en notificació explícita de la congestió. Per a este propòsit, els switchos detecten congestió i depenent de l'estratègia aplicada, els paquets són marcats amb la finalitat d'advertir els nodes orígens. Com a resposta, els nodes orígens apliquen algunes accions correctives per a ajustar la seua taxa d'injecció de paquets. Encara que estes propostes pareixen prou efectives, tenen alguns inconvenients o són solucions parcials. Algunes d'estes solucions introduïxen una certa penalització sobre els fluxos no responsables de la congestió, mentre altres només poden manejar situacions de congestió que es donen durant un curt període de temps.

El propòsit d'esta tesi és analitzar les diferents estratègies de detecció i correcció

de la congestió en xarxes multietapa, i proposar nous mecanismes de control de la congestió encaminats a este tipus de xarxes sense descart de paquets. Les noves propostes estaran basades en una estratègia més refinada de marcatge de paquets en combinació amb un conjunt d'accions correctives justes que faran al mecanisme capaç de controlar la congestió de manera efectiva amb independència del grau de congestió i de les condicions de tràfic.

Resumen

El crecimiento de los computadores paralelos basados en redes de altas prestaciones ha aumentado el interés y esfuerzo de la comunidad investigadora en desarrollar nuevas técnicas que permitan obtener el mejor rendimiento de estas redes. En particular, el desarrollo de nuevas técnicas que permitan un encaminamiento eficiente y que reduzcan la latencia de los paquetes, aumentando así la productividad de la red. Sin embargo, una alta tasa de utilización de la red podría conllevar el que se conoce como *congestión de red*, el cual puede causar una degradación del rendimiento, porque toda o parte de la red ha excedido la utilización máxima, cuyo valor viene impuesto por el punto de saturación.

El control de la congestión en redes multietapa es un problema importante que no está completamente resuelto. Con el fin de evitar la degradación del rendimiento de la red cuando aparece congestión, se han propuesto diferentes mecanismos para el control de la congestión. Muchos de estos mecanismos están basados en notificación explícita de la congestión. Para este propósito, los switches detectan congestión y dependiendo de la estrategia aplicada, los paquetes son marcados con la finalidad de advertir a los nodos origenes. Como respuesta, los nodos origenes aplican algunas acciones correctivas para ajustar su tasa de inyección de paquetes. Aunque estas propuestas parecen bastante efectivas, tienen algunos inconvenientes o son soluciones parciales. Algunas de estas soluciones introducen cierta penalización sobre los flujos no responsables de la congestión, mientras otras solo pueden manejar situaciones de congestión que se dan durante un corto período de tiempo.

El propósito de esta tesis es analizar las diferentes estratégias de detección y

corrección de la congestión en redes multietapa, y proponer nuevos mecanismos de control de la congestión encaminados a este tipo de redes sin descarte de paquetes. Las nuevas propuestas estarán basadas en una estrategia más refinada de marcaje de paquetes en combinación con un conjunto de acciones correctivas justas que harán al mecanismo capaz de controlar la congestión de manera efectiva con independencia del grado de congestión y de las condiciones de tráfico.

List of Abbreviations

ACK	Acknowledge Packet
AIMD	Additive-Increase, Multiplicative-Decrease
ATM	Asynchronous Transfer Mode
BECN	Backward Explicit Congestion Notification
BMIN	Bidirectional Multistage Interconnection Network
BTH	Base Transport Header
CC-NUMA	Cache Coherent Non-Uniform Memory Access
CMM	Congestion Management Mechanism
DSM	Distributed Shared Memory
DW	Dynamic Window
ECN	Explicit Congestion Notification
FECN	Forward Explicit Congestion Notification
FIFO	First-In, First-Out
FIMD	Fast-Increase, Multiplicative-Decrease
FLIT	Flow Control Unit
FVOQ	Full-Virtual Output Queuing
HCA	Host Channel Adapter
HOL	Head-of-Line
HPC	High-Performance Computing
HS	Hot-Spot
HSD	Hot-Spot Degree
I/O	Input/Output

IOCM	Input-Output Congestion Management
IOPM	Input-Output Packet Marking
IP	Internet Protocol
IPM	Input Packet Marking
ISC	InterState Connection
LIPD	Linear Inter-Packet Delay
MB	Marking Bit
MBin	Input Marking Bit
MBout	Output Marking Bit
MIN	Multistage Interconnection Network
MPI	Message-Passing Interface
MPPs	Massive Parallel Processors
MVCM	Marking-Validation Congestion Management
MVPM	Marking-Validation Packet Marking
NoCs	Networks-on-Chip
NUMA	Non-Uniform Memory Access
OPM	Output Packet Marking
PC	Personal Computer
PHIT	Physical Unit
PVOQ	Partial-Virtual Output Queuing
QoS	Quality-of-Service
RPM	Renato’s Packet Marking
RTT	Round-Trip Time
SAN	System Area Network
SB	Shared-Buffer
SMP	Symmetric MultiProcessing
SW	Static Window
TCP	Transmission Control Protocol
UMA	Uniform Memory Access
UMIN	Unidirectional Multistage Interconnection Network

VB Validation Bit

VLSI Very Large Scale Integration

WS Waiting Slot

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This chapter introduces the main issue that has motivated the approach and development of this thesis.

1.1 Facing "The Big Issue"

The traffic control centre which daily monitors traffic in the big city plans the effective working of traffic lights in an equitable way. This tries to prevent traffic jams which lead to increase travel costs. Due to the fact that traffic behavior is not uniform throughout the day, and sometimes totally unpredictable, traffic chaos often occurs at rush hours causing slower and a more dense movement of vehicles. Although predictive models can be developed to forecast the traffic, it often stops without any apparent reason and causes hold-ups. Incidents such as a freak accident force us to stop our car and cause other vehicles to stop because we are unable to let them pass.

Faced with a problem of this type, the traffic control center reacts quickly to try to prevent more vehicles from accessing the affected area. To get that, it offers alternative routes that avoid congestion. And, the traffic is regulated in real time through the traffic lights that reduce the flow of cars to the affected area. The success of the taken actions lies in the early detection of the incident, the fair application of a set of corrective actions and the speedy removal of those actions to recover the

previous traffic rate as soon as possible. The “Big Issue” lies in which mechanism is used to detect and quickly report the incident causing congestion to the control centre, and what actions are taken to cope with it.

This problem of traffic congestion in the Big City is a simple analogy to the congestion problem in interconnection networks that is attacked in this thesis. The overall objective of this thesis is to analyze the congestion problem in Multistage Interconnection Networks (MINs), commonly used in PC clusters, evaluating the proposed strategies for congestion management and developing new proposals that are fair and efficient on applying corrective actions.

1.2 Motivation

In recent years, many commercial applications are demanding a large volume of data communication, real-time response and a great interactivity degree with the user. In particular, computing and communication needs have expanded from purely scientific and research applications to a more common range of areas such as database management systems, in which thousands of user requests have to be served simultaneously. As Figure 1.1 shows, more than 96 percent of the High-Performance Computing (HPC) systems referenced at the Top500 list [93] (November 2011) are intended for application areas such as research and user services.

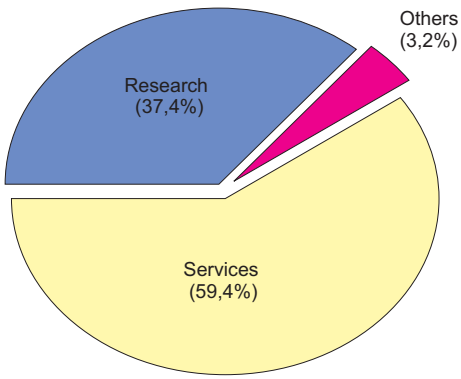


Figure 1.1: Application area of high-performance computing.

The need to attend such increase of users, as well as computational power and storage capacity, leads to the need of building scalable and flexible systems. That ease their expansion in an incremental way and guarantee their long term growth without degrading performance.

Although the performance of processors has increased steadily in the last years, there is a limit imposed by thermal and power supply related issues. To break this limit, a solution for providing higher computational power is to coordinate multiple processors in a system to concurrently perform the computational tasks, dividing the tasks into subtasks, each one being solved in a different processor. These systems are widely known as parallel computers.

During the last decades different parallel computer architectures have emerged. Basically, there are two fundamental types of parallel computers, either one with multiple processors sharing all the memory, known as a *shared memory multiprocessor*, or a set of processors, each one with its own private memory, interconnected through a network. These systems are known as *multicomputers*. Next, we briefly describe each system:

- Shared memory multiprocessor.
  The shared memory multiprocessor is a natural extension of the conventional computer where the processor accesses to its own memory, but now multiple processors are connected to multiple memory modules through an interconnection network and support a single address space through-out the system. This means that any processor can access to any memory location without the need of copying data from one memory to another. We can classify multiprocessors in two classes depending on how the memory is shared:

  – Multiprocessors with a centralized memory or multiprocessors with Uniform Memory Access (UMA), also known as Symmetric MultiProcessor (SMP). In this type of systems, the access time is uniform for every memory module from any processor. Figure 1.2(a) shows a UMA multiprocessor system. As an example of commercial UMA machines is the Sun Fire 15000 system with 106 UltraSparc III processors [89].

  - Multiprocessors with Distributed Shared Memory (DSM), also known as multiprocessors with Non-Uniform Memory Access (NUMA). The memory is shared but distributed among the processors, therefore the access time to memory is non-uniform (smaller to the local memory, and larger to non-local or remote memories). To reduce the effects of non-uniform memory access, caches are often used, which introduce the cache coherence problem. If cache coherence is guaranteed, the system is referred to as Cache Coherent Non-Uniform Memory Access (CC-NUMA). Figure 1.2(b) shows a NUMA multiprocessor system. Some examples are the BBN Butterfly [20], the Cray T3D [22], and the SGI Origin 2000 [60], which use NUMA, non-coherent cache NUMA, and CC-NUMA, respectively.

- Distributed memory multiprocessors or multicomputers.
  A multicomputer consists of independent processors with their local memories connected via an interconnection network as Figure 1.2(c). In this architecture, each processor has its own memory address space. That is, each processor is able to address only its local memory space. Interprocessor communication is achieved by sending explicit messages from each computer to another using a message-passing library such as MPI (Message-Passing Interface) [68]. Message-Passing multicomputers physically scale better than shared memory multiprocessors. Moreover, these systems can also support shared-memory applications by providing a certain software layer.

Although these systems are very powerful, they have also some disadvantages, being the most important one their high costs (due to their reduced volume of sales). Additionally, although they are helpful when running highly parallel applications, a high number of applications are not easily parallelizable or even do not require this parallelization. Thus, only a reduced set of applications take advantage of these systems. In addition to this, they present a high cost per node, as they are made up of expensive specialized components. Another drawback is the cost of maintenance, because highly qualified personnel is needed.

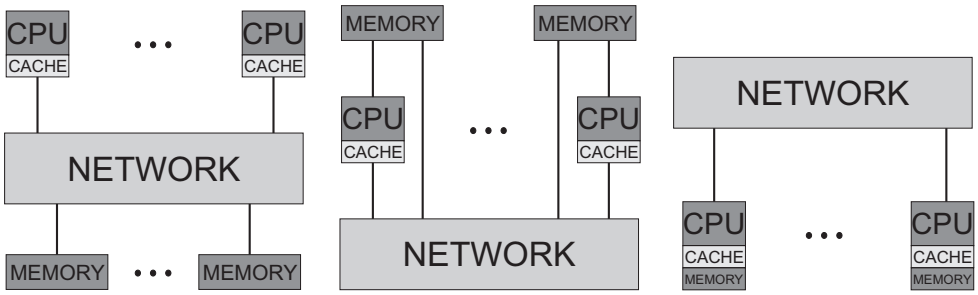


Figure 1.2: Examples of parallel architectures: (a) multiprocessor system with uniform memory access (UMA), (b) multiprocessor system with non-uniform memory access (NUMA), and (c) multicomputer system.

Nowadays, the excellent cost/performance ratio of Personal Computers (PCs) has allowed them to become a common element in many environments where high-performance computing is required. Many of the most powerful machines are built from PCs as can be observed in the Top500 supercomputer list.

In particular, the cluster architecture has become very popular in the last 10 years, because it offers the best cost/performance ratio for building low-cost supercomputers or high-performance servers that respond to new service demands. In these systems, standard off-the-shelf computer nodes are interconnected by a high-performance network, also standard technologies in most cases that allows communication among them, leading to a low-cost multicomputer.

Figure 1.3 shows the evolution of the architectures used to build HPC systems. In particular, more than 80 percent of the HPC systems referenced at the Top500 list are currently built using the cluster architecture. In these systems, interconnection networks take on a very important role in the overall system performance. Therefore, designing efficient high-performance interconnection networks becomes a critical issue to exploit the performance of parallel computers.

The lack of standards for these specialized interconnection networks and/or the advances in technology have given rise to important developments in the field of interconnection networks over the last decade. As a result, a significant number of

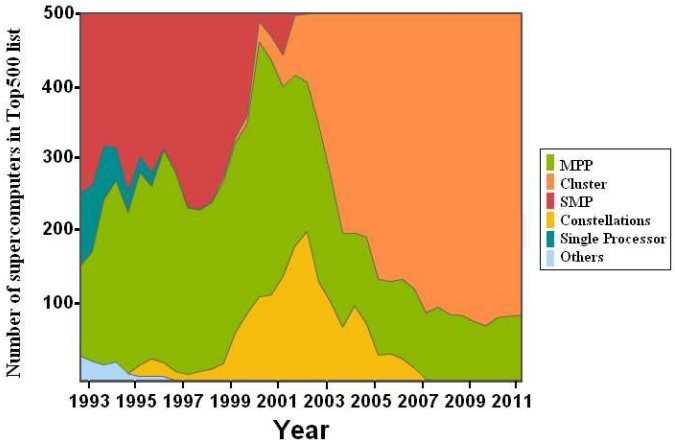
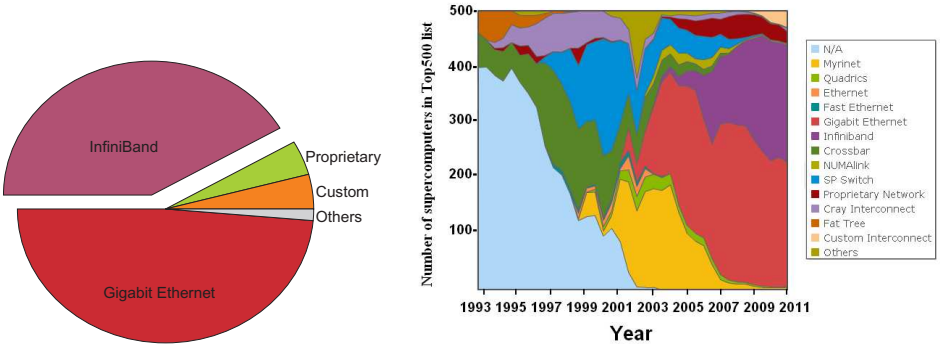


Figure 1.3: Evolution of the high-performance computing architectures.

interconnects such as InfiniBand [48], Myrinet [69], RapidIO [80], and Quadrics [79] have emerged. The main contribution of these network technologies is based on the replacement of the oldest bus-based interconnections, by a high-performance interconnection network that uses point-to-point links and high-speed switches between processors and Input/Output (I/O) devices.

As Figure 1.4 shows, clusters using standard interconnection networks as InfiniBand are becoming increasingly popular. In particular, 5 of the top 10 at the Top500 list, and a total of 209 from the 500 HPC systems are using InfiniBand as the technology to build their interconnection networks (November 2011 on [93]).

The growth of parallel computers based on these high-performance networks has increased the interest and effort of the research community in developing new techniques to achieve the maximum performance from these networks. In particular, the development of new techniques for efficient routing to reduce packet latency and increase network throughput. However, high utilization rates of the network could lead to an important problem in interconnection networks. The problem is that packets compete for shared resources (links and switches), causing a significant impact in their latency. If this contention is not effectively controlled, it is possible the network reaches an early saturation point. This network status can cause what is known



(a) Nowadays (b) Evolution

Figure 1.4: Interconnection networks used in clusters.

as *network congestion* and indicates that all or a part of the network has exceeded the maximum utilization of this resources, resulting in a degradation of the network performance.

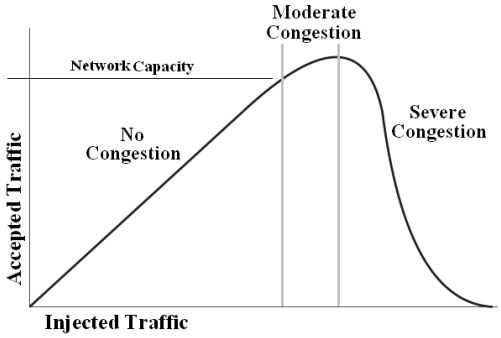


Figure 1.5: Accepted vs. injected traffic.

As shown in Figure 1.5, if the injected traffic into the network exceeds the network capacity, packets are stopped into the network due to the contention created by the demand for resources. If this situation persists, it can cause a severe congestion [87], [88], resulting in a drastic reduction in network performance. The consequences of congestion are even more serious in networks that do not allow packets to be discarded (lossless networks) when a congestion situation is detected. This is the case

of networks that make up the majority of high-performance parallel computers. The traditional solution to the congestion phenomenon has been to design the network using a higher number of resources than the strictly necessary (i.e. over designing the network) so that there is no need, in practice, to compete for those resources, thus avoiding packet contention. However, this practice is not efficient both in terms of cost and power. Therefore, the design of oversized interconnection networks is not an efficient way to build real computer systems, where cost and power issues are ones of the main concerns, as a consequence, congestion control is a key issue that requires cost-effective solutions.

Some standard interconnection networks, such as InfiniBand, establish the basis in their specifications to deal with the congestion management. Although specs have to be respected, they also leave some parts free to vendor criteria. The challenge for the researchers is to propose new techniques that, although consistent with the specification of the network standard used, cover the gaps left by the specs or contribute to their improvement. Due to the fact that there are currently few proposals for congestion management, there is a great interest in this area to develop those aspects that were left to the developers.

1.3 Objectives

The main objective of this thesis focuses on the development of new Congestion Management Mechanisms (CMMs) based on the design of new strategies for i) the early detection of congestion, and ii) new corrective actions that stop the growth of the congestion and offset its effects, allowing an equitable distribution of network resources usage. Additionally, the goal is also to develop new techniques that are difficult to implement in the current network technologies with minimal hardware support requirements. In particular, new proposals have been developed, analyzed and tested for lossless multistage interconnection networks. This overall objective can be divided into the following sub-objectives:

- Analyzing the pros and cons of previous proposals for multistage interconnec-

tion networks, broken down into the two main congestion techniques:

  - Congestion Detection techniques.
  - Congestion Correction techniques.

- Proposing new techniques for congestion detection to identify correctly the flows responsible for the congestion.
- Proposing effective congestion correction techniques, in accordance with the new proposed techniques for congestion detection, to reduce the effects of congestion and maximizing the network performance in any environment.
- Evaluating and analyzing on the whole of the new strategies for congestion detection and congestion correction under different working conditions (network configurations, traffic loads, congestion degrees), comparing the achieved results to the ones obtained by the previous proposals.
- Analyzing the impact of the different strategies proposed in the approaches, in order to find out their contributions. In particular, analyzing the influence of each strategy as if it is working alone.
- Drawing some conclusions regarding to the most appropriate mechanism to implement, depending on the network conditions and characteristics.

1.4 Organization

The thesis is organized as follows. First, a background on interconnection networks is presented in *Chapter 2*. Next, the congestion problem and the current approaches for congestion management, with an overview of the state of the art, are introduced in *Chapter 3*.

Chapters 4 and *5* describe and analyze some new congestion detection and congestion correction techniques, respectively. Next, *Chapters 6* describes the proposed congestion management mechanisms based on the techniques previously analyzed.

In *Chapter 7* the simulation model and the evaluation methodology are presented. Next, *Chapter 8* shows all the analysis and results achieved from the performance evaluation.

Chapter 9 summarizes the contributions, conclusions, scientific publications, and the future work.

Finally, the references used throughout this thesis are cited in the *Bibliography Section*.

Chapter 2

INTERCONNECTION NETWORKS

This chapter presents the structural aspects that define the interconnection networks and determine their functionality. It also gives an overview of the different systems using such networks in order to establish a definition of them and justify their use.

2.1 Overview

The interconnection network can be defined as the physical system composed of a series of elements (links and routers) that specifically interconnected, allows communication between the nodes of a system. In this sense, the network can be seen as an intelligent system [27] that allows rapid data communication between their components. Nowadays, high-performance networks are used in different systems, such as:

- PC Clusters: Systems built of a set of Personal Computers (PCs) interconnected through a high-performance network either standard (e.g., InfiniBand) or proprietary (e.g., Myrinet). Clusters emerged as a low-cost alternative to multiprocessor systems. They were initially designed as a platform for the execution of parallel applications, but then they have been used to meet the

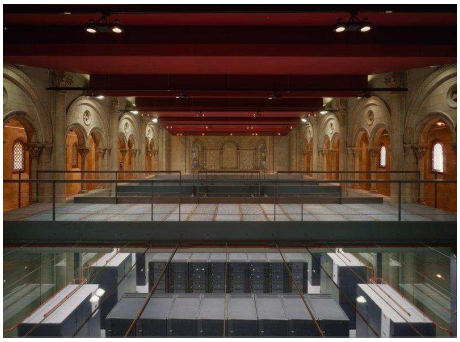


Figure 2.1: The Mare Nostrum PC cluster. 2560 dual processing nodes BladeCenter JS21, 2,5 GHz. Myrinet. Barcelona.

needs of other applications and services. In particular, nowadays they allow to implement storage area networks and WWW servers, where the volume of user access and the sophistication of the offered services require a bounded response time, high computing power and high storage capacity. Companies like Google, Hotmail, Yahoo, etc, run their WWW servers on personal computer clusters in order to provide the service demanded by millions of users. An example of a Spanish PC cluster is the Mare Nostrum IBM cluster [66] (Figure 2.1), with 2560 dual processing nodes connected with a Myrinet network. The Mare Nostrum IBM is ranked among the 300 most powerful supercomputers according to the Top500 list [93] (November 2011).

- Massive Parallel Processors (MPPs): In these systems, the network must be able to effectively manage the volume of communications that the application demands, and provide a reduced latency in order to maximize the use of processors and other network resources. To this end, network design is customized to accomplish the overall system requirements. The compute-intensive applications that require high computing power and accuracy in calculations justify the use of such parallel computers and require a continuous development in technology and research related to this development. An example of these systems is the Earth Simulator [35] with 640 vector processors (each one with 8 pipelines) shown in Figure 2.2, and the BlueGene/L [13],[90] with 65536

processing nodes (each one with 2 processors).

Figure 2.2: The Earth Simulator massive parallel processors. 640 vector processors (8 pipelines each one). 640x640 switch crossbar. 16 GB/s of Bandwidth.

Figure 2.3: Voltaire IP Router module. High performance IP Connectivity. Grid Switch ISR 6000. InfiniBand Compliant.

- IP Routers: The steady growth in the number of Internet users causes an exponential increase in bandwidth demand [7], [72], [8]. The technological limitations of the switching elements in such networks do not properly help such growth. As a consequence, a mismatch occurs between the bandwidth demanded by users and that offered by the IP routers. This bottleneck has been mitigated by a notable increase in the number of ports in this type of compo-

nent. Currently, the most viable option to build these switching elements is the network structure made up of several stages. This makes high-performance networks an interesting alternative as a communication component of these devices. An example of IP Routers is the Voltaire IP Router Module, shown in Figure 2.3.

- Networks-on-Chip (NoCs): Traditionally, internal connections into the chip have been designed with dedicated point-to-point links. For large designs, this has several limitations from a physical design point of view. The long wires occupy much of the area of the chip, interconnects dominate both performance and dynamic power dissipation, and, as signal is propagated across the chip through the wires, it requires multiple clock cycles. NoCs can reduce the complexity of designing wires for predictable speed, power, etc., thanks to their controlled structure. From a system design point of view, with the advent of multi-core processor systems, an interconnection network is the natural and effective architectural choice. NoCs can provide separation between computation and communication. IBM's CoreConnect [19] is an example of NoC.

As described above, a high-performance network is essential in legacy environments. In PC clusters, it is necessary to provide a high-performance network that enables reliable communication between processors and devices, and among devices. For MPPs, those networks should provide low latency connectivity and high bandwidth to allow high levels of concurrency. For IP routers, such networks must allow them to reach the demanded bandwidth. Finally, interconnection networks into the chip (NoC) provide a reduction in the use of silicon space being a key element to efficiently connect the increasingly number of cores, memories and processing components in general built-in into the chips.

2.2 Features of an Interconnection Network

2.2.1 Requirements

The interconnect system should establish communications between nodes, regardless of the topological design. This way, it is not necessary to make any changes to the applications each time the topology of the system is changed, or when the application is migrated to another parallel computer. Therefore, the communication system must abstract the physical details of the communication model. While the application is running, each node may need to send a packet to any destination, so the interconnection network must provide a mechanism capable of moving the packet between nodes until reaching the final destination. This mechanism will select one or more possible paths to establish the logical connection between the origin-destination pair of nodes. There is a set of requirements that can be commonly applied in the design of an interconnection network to achieve a high system performance:

- The network should provide a low latency. Latency depends both on the interconnection design and the volume of network traffic. Therefore, packets should always take the path involving the shortest time to arrive to the destination.

- The network should be able to operate near the maximum value of throughput, which depends on the available bandwidth. The throughput determines the amount of traffic per time unit that the network can handle without causing congestion, that is, the maximum traffic that can be delivered by the network per time unit.

- The routing mechanism used in the network should adapt to traffic conditions and should exploit the maximum bandwidth and connectivity degree offered by the links. This feature gives robustness to the network, as it provides fault tolerance and the ability to avoid congested areas.

- There is a number of undesirable phenomena that can appear in the interconnection network. They should be avoided because they can either cause a significant degradation in network performance or can lead to a network fault.

These phenomena occur through the use of inefficient or poorly designed routing techniques [33].

Deadlock: it occurs when some packets cannot advance toward their destination because the buffers requested by them are full.

Livelock: a packet is traveling around its destination, but never reaches it.

Starvation: if traffic is intense, a packet is permanently stopped and the resources requested are always granted to other packets, also requesting them.

2.2.2 Parameters

In addition, it is possible to classify the main parameters involved in the definition of an interconnection network as follows:

- Static parameters:

  Network components: Basically, interconnection networks are built with a set of switches connected by links that allow the exchange of messages between end-nodes.

  Topology: Describes the physical structure (shared-medium networks, point-to-point direct or indirect networks) and the interconnection pattern among the network components.

- Dynamic parameters:

  Flow Control: Establishes a dialogue between senders and receivers, allowing and stopping the forwarding of information, thus avoiding the buffer overflowing.

  Switching Technique: Defines the mechanism that manages the progress of the messages from one node/switch to the next, that is, how and when network resources are allocated to the requesting messages (store-and-forward, virtual cut-through, wormhole).

Routing Algorithm: Determines the strategy to select the path followed by a message from a source node to its destination (deterministic, adaptive, hybrid).

2.2.3 Network Components

Before describing some more complex structures, it is interesting to focus on the different types of data units that can be found in a network. Figure 2.4 shows the different data units that can be identified. The data to be transmitted is organized into different units. The message is the largest unit. It is the application data unit, and it is processed between two different end-nodes by exchanging it across the network. Before being transmitted, messages are often decomposed into smaller data units, referred to as packets, according to the Maximum Transfer Unit (MTU) allowed by the network.

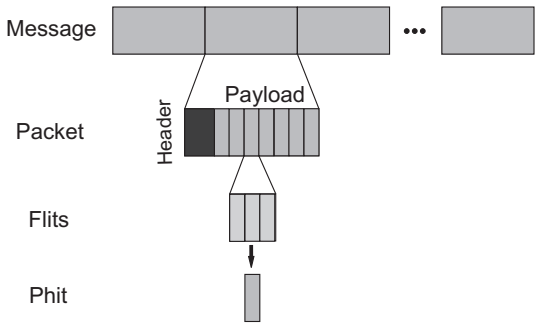


Figure 2.4: Data units.

The structure of a packet is composed of a header that contains the routing and the control information used to drive the packet from its source to its destination, and a payload that contains the application data. Sometimes a tail is included to indicate the end of the packet. Packets are composed of flits (flow control units)[1]. Flits are the minimum data that can be controlled over a link by the flow control mechanism, that is, between two adjacent nodes/switches. To transmit a single flit, multiple link

[1]Depending on the applied switching technique, the packet can be composed of either a single flit (Store-and-forward and Virtual cut-through) or several flits (Wormhole)

cycles may be used. The piece of data transmitted in a single cycle over the link is referred to as phit (physical unit).

The network components can be identified as switches and links.

- Switches: A switch can be defined as the network component able to interconnect devices (even other switches). The main task of a switch is to forward the packets arriving on its input links to the corresponding output link. Figure 2.5 shows a generic diagram of a switch. As it can be seen, there is a set of buffers attached to each input and output port. Each received packet is allocated to an input buffer until, once it is routed, it can be transmitted to the output port requested by the packet. Firstly, the routing unit will determine the appropriate output port that drives the packet towards its final destination. When there is enough buffer space at the output port, the packet will compete for the access to the crossbar. Secondly, the arbiter takes care of resolving all the potential requests to the crossbar according to the current status of the resources of the switch. Thirdly, the crossbar is configured to connect the selected input link to the corresponding output link. Finally, the queued packet at an input buffer receives a grant to traverse the crossbar, and it will be transmitted to the output buffer.

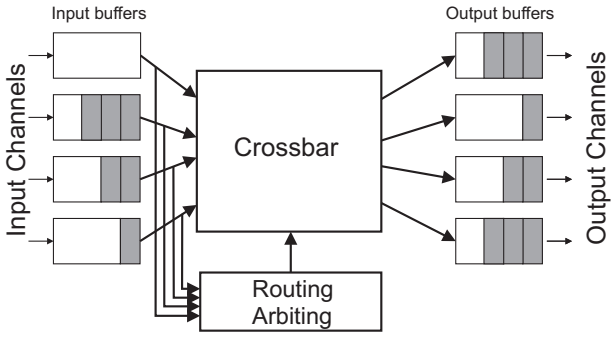


Figure 2.5: A generic diagram of a switch.

Notice that there are some variations of this basic structure with buffers only at input or output ports.

- Links: Links are the network components used to physically connect end-nodes and switches and the latter ones among them. They can be classified based on the direction of the data through the link.

  Unidirectional links: When two network components are connected using this kind of link, the information can only flow in one direction. Figure 2.6 shows a unidirectional link. To allow a node be connected to the rest of nodes, the network has to be wrapped. This condition doubles the average distance traveled by a packet. Alternatively, two unidirectional links can be used (each one in one direction) to connect a pair of nodes.

Figure 2.6: Unidirectional link.

  Bidirectional half-duplex links: These links allow both sides of the link to transmit data when it is available. However, transmission cannot be done simultaneously. So, an arbitration is needed to allow both sides to transmit in a fair way. Figure 2.7 shows a bidirectional half-duplex link.

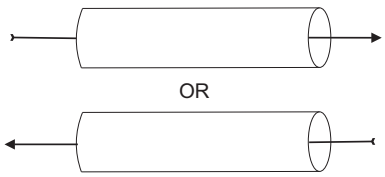


Figure 2.7: Bidirectional half-duplex link.

  Bidirectional full-duplex links: These links are halved, allocating half of the bandwidth to each direction. Thus, both sides of the link can transmit simultaneously. However, if only one side is transmitting, the other 50% of the bandwith is unused. Figure 2.8 shows a bidirectional full-duplex link. Notice that there is a proposal to use the 100% of the bandwith in each direction by applying simultaneous bidirectional signalling [59]. Alternatively, bidirectional full-duplex communication can be provided by using two unidirectional links

(each one in one direction)

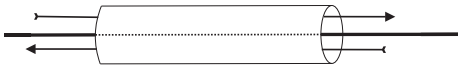


Figure 2.8: Bidirectional full-duplex link.

2.2.4 Topologies

An important parameter that influences the performance of the system is the spatial arrangement of the nodes into the network. The set of nodes and links are connected in a particular way, thus defining a topological structure or interconnection pattern. There are different types of topologies, each one with different characteristics, advantages and disadvantages [2]. Among the existing classifications of topologies, a very common and accepted one defines three important categories: *shared-medium networks*, *direct networks*, and *indirect networks* [33]:

- Shared-medium Networks: They are built with a single interconnection element that directly connects all the end-nodes. They constitute a well-established topology. These interconnection networks were used in the first parallel computers but soon fell into disuse because of their low performance. Their limited bandwidth restricts their use to multiprocessors with a low number of nodes. Figure 2.9 shows two typical examples of such networks.

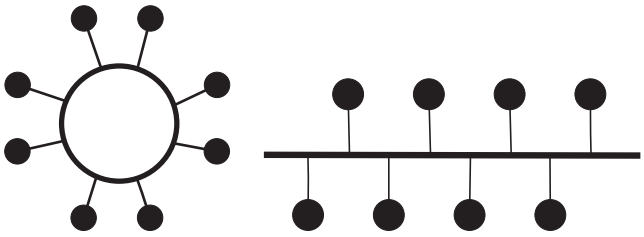
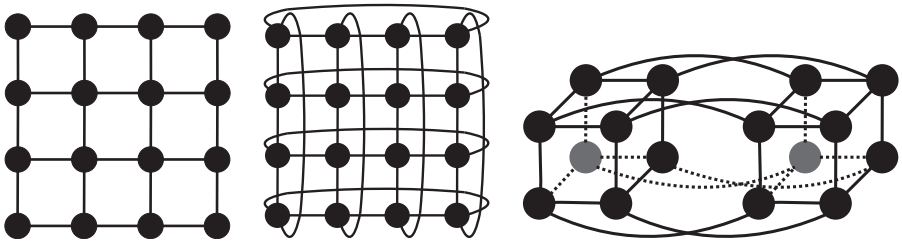


Figure 2.9: Shared-Medium networks.

- Direct Networks: The direct network or point-to-point network is a popular network architecture, designed in a regular and well-defined pattern that scales

well to a large number of processors. They are built of a set of nodes that include a router to connect to other nodes. Each router has several ports which allow for multiple links between nodes, forming topologies rich in resources, but which are also complex and expensive. Examples of direct networks are mesh topologies (Intel Paragon [34] and the MIT J-Machine [71] using 2D and 3D mesh, respectively), tori (Intel/CMU iWARP and Cray T3D using bidirectional 2D and 3D, respectively) and hypercube (intel iPSC), each one with several variants but following a common pattern. Figure 2.10 shows some examples of these topologies.



(a) Mesh (b) Tori (c) Hypercube

Figure 2.10: Direct networks.

- Indirect Networks: The indirect, or switch-based networks do not provide direct connection among any end-node. Instead, the communication between any two end-nodes has to be carried through several point-to-point connected switches. Each end-node has an adapter that connects to a network switch and each switch has several bidirectional ports. In the case of an indirect network with N end-nodes, the ideal topology that achieves the best performance, should be the one that connects all the nodes through a single switch with NxN ports. This network, known as *crossbar*, offers a non-blocking connection between all the nodes in the network. Figure 2.11 shows a crossbar network of NxN elements.

  When the number of end-nodes connected to the network grows significantly, this type of topology is not feasible due to technological constraints. This encourages the use of Multistage Interconnection Networks (MINs) as a cheaper

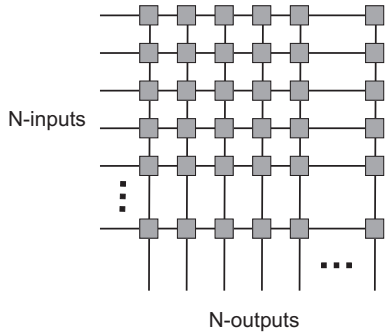


Figure 2.11: Crossbar network of NxN End-nodes.

alternative to the crossbar. These networks connect input devices to output devices through a number of switch stages, where each switch is a crossbar network. In this kind of network, not all the switches have a processing node connected to them, but they are used as an intermediate step to reach the next processing node or switch. There are a large number of proposals regarding the topological organization, however the most interesting, from a practical point of view, are those that use the same kind of switches arranged in stages and are suitable for constructing parallel computers with hundreds of processors. They have been used in some commercial machines. Figure 2.12 shows a generic MIN, in which the InterStage Connection (ISC) defines the connection pattern between each stage of switches. A network of $M = k^n$ nodes are arranged as n stages with M/k switches with k-input/output channels each stage.

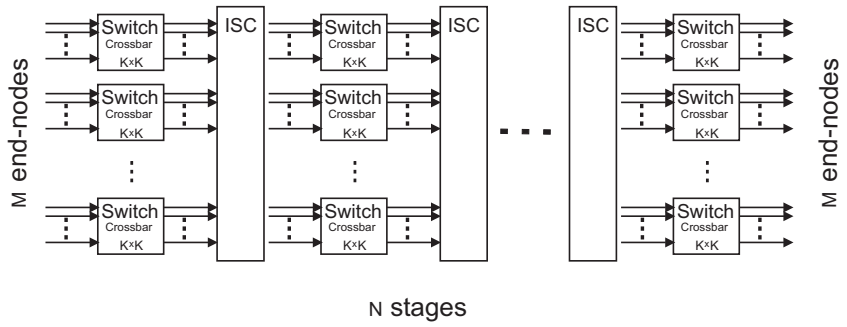
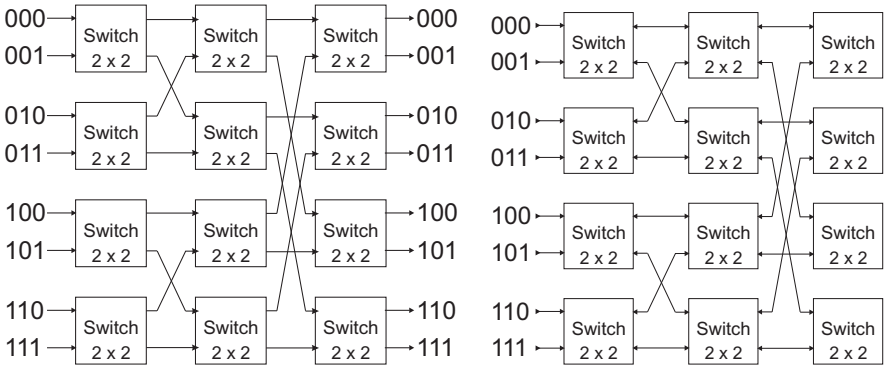


Figure 2.12: A generic MIN.

The network performance is strongly influenced by the pattern that connect the different stages. Among the more common designs of multistage interconnection networks, are the Omega, Baseline, Butterfly, Perfect-Shuffle, Closs and Benes networks. Each of these networks has a different pattern of connection links but are topologically and functionally equivalent. As an example, Figures 2.13 (a) and (b) show the implementation of a Butterfly network 2-ary 3-fly with unidirectional and bidirectional links, respectively.



(a) Unidirectional Links (b) Bidirectional Links

Figure 2.13: Butterfly networks.

Although the network structure is the same in both configurations, the difference lies in the number of switches that a packet has to cross between each origin-destination pair. So, in the case of using unidirectional links, that is Figure 2.13 (a), the number of switches to cross is three regardless of the origin-destination pair. However, for the bidirectional network, that is Figure 2.13 (b), the number of hops varies from two (one switch) for end-nodes connected to the same switch and six (five switches) for the end-nodes farther away. Alternatively, the switch interconnection pattern could be similar to that used in direct networks, such as meshes and tori. In this case, switches use a subset of their links to connect to other switches, whereas the remaining links can be used to attach one or more end-nodes. This shows that there is no difference between direct and indirect networks. Indeed, a direct network can be consid-

ered equivalent to an indirect network with a single end-node attached to every switch.

Finally, *Direct* and *Indirect Networks* share some common topological metrics that define them and, as such, are worthy of mention.

- Degree: Number of ports of each switch.
- Diameter: The maximum distance between any pair of end-nodes.
- Regularity: All the switches have the same degree.
- Symmetry: The network looks like the same from any switch/link.
- Connectivity: The minimum number of switches/links necessary to disconnect the network.

2.2.5 Flow Control

Each switch is constituted by a set of ports that are used to send and receive the packets flowing through the network. Each port has some buffers to store the received data until it is forwarded to the next switch. By using the flow control mechanism, the sender is warned about the availability of space within the receiver. Based on this information, the sender does not overwhelms the destination buffers [23]. The most commonly used mechanisms for this purpose can be divided into two groups: *Credit-based* and *Status-based* flow control.

- Credit-based mechanisms give a number of credits to each switch to send packets. The number of credits will depend on the available buffer space in the neighboring node. Every time the sender node transmits a piece of data, it consumes one credit. Eventually, when all the credits are consumed, the sender stops sending packets. Once the space in the receiver is released, new credits are sent to the sender and it can continue transmitting data.

- The status-based mechanisms are based on marks. The receiver detects that its input buffer is close to overflowing because the stored information exceeds a certain threshold. As a consequence, it sends a signal to the sender to stop packet transmission. Similarly, when there is again available enough space in the receiver (the stored information comes below certain threshold), it sends another signal to the sender to reactivate data transmission. This technique is also known as called Stop&Go [14].

2.2.6 Switching Technique

Another important interconnection network design parameter is the switching technique. It determines how the packets advance through the switches along the network. The switching technique applied has a considerable impact on the switch architecture and therefore on the performance achieved by the interconnection network. The most common switching techniques used in modern interconnection networks can be grouped in three classes: store&forward, virtual cut-through and wormhole.

- Store & Forward: This technique determines that the packet passing through the network has to be fully received and stored at the input buffer of the switch before it can be routed and sent to the corresponding output port. Figure 2.14 shows how this technique works.

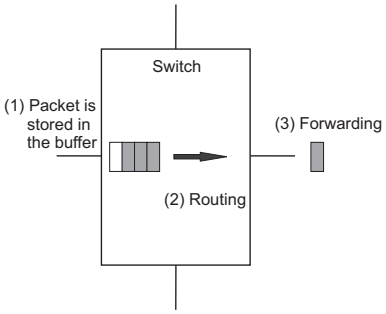


Figure 2.14: Store & Forward switching technique.

The links are only used at the time the information is sent from one node to the next, avoiding the blocking of the link for longer than that strictly necessary. In

this switching technique, network latency is proportional to the packet length and the number of hops. Indeed, the buffer size imposes certain restrictions on the maximum packet size that can be injected, due to the fact that the buffers have to store the entire packet.

- Virtual cut-through: In this technique, if the required output channel is free, the packet is transmitted to the next switch as soon as the packet header is received, which can be performed without receiving the full packet [54]. However, the buffer has still to have enough buffer space to store the whole packet in case the output channel is busy. For this reason, in the event of occupation of the output channel, the mechanism behaves in the same way as *Store & Forward*. However, the influence of the distance to the destination node on network latency is considerably lower than in the previous technique as, if the output port is free, the packet is rapidly forwarded according so its intermediate storage at the switch. Figure 2.15 shows how this technique works.

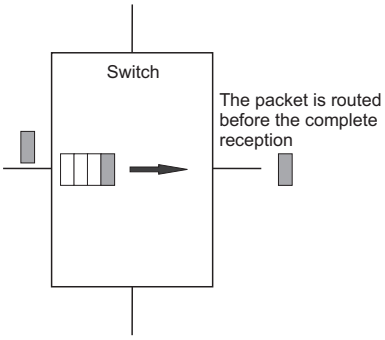


Figure 2.15: Virtual Cut-Through switching technique.

- Wormhole: As in the virtual cut-through technique, Wormhole can forward the packet before complete reception. However, the buffer size does not determine the packet size. Therefore, the messages do not need to be split into packets. Buffer size is smaller and can hold only a small part of the packet [26]. Figure 2.16 shows how this technique works. Buffer are reserved as the packet header advances to its destination. In case a packet cannot advance because the re-

quired switch port is busy, data (referred to as flits) stay at the reserved buffers. The tail flit frees the reserved resources.

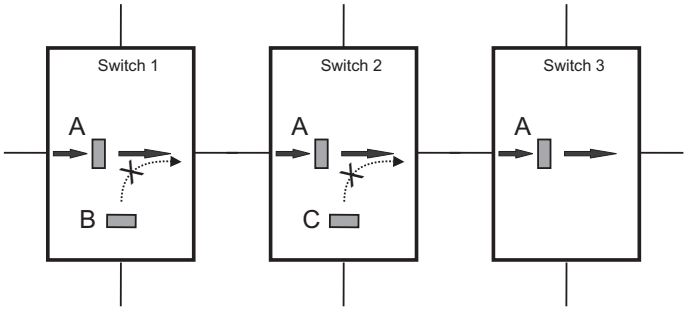


Figure 2.16: Wormhole switching technique.

This switching technique is suitable for systems with small size requirements such as NOCs. The need for temporary storage in wormhole networks is minimal, because the buffers should only be able to store a part of a packet. This technique, like virtual cut-through, is able to significantly reduce the latency of messages. However, it is more deadlock-prone as when the message is stopped into the network, it holds the possession of the reserved resources along the path, involving several consecutive nodes.

2.2.7 Routing Algorithm

The routing algorithm establishes the path followed by each message or packet. It is the responsible mechanism for driving packets from the source along all the selected paths to reach the destination. Along the journey, packets should traverse several intermediate nodes, therefore the routing algorithm has to recognize the network topology in order to select the appropriate path. The main objective of the routing algorithm is to select a path that reduces network latency, whilst also avoiding congested routes [44]. Many properties of the interconnection networks (connectivity, adaptivity, deadlock and livelock freedom, and fault tolerance) are a direct consequence of the routing algorithm used. In general, a routing algorithm should satisfy the following properties: simplicity, speed, robustness, connectivity, maximizing the

use of links and optimality. Next, a taxonomy of routing algorithms [33] is shown in Figure 2.17.

From this taxonomy, we can highlight the *adaptive routing* algorithms, whose main feature is their ability to adapt the route according to the network status. They can change the route based on the traffic conditions on the network, seeking to avoid congestion situations. On the opposite side, we have the *deterministic routing* algorithms. In this case, the route a packet will follow only depends on their origin and destination nodes and it will not be changed, regardless of the traffic conditions.

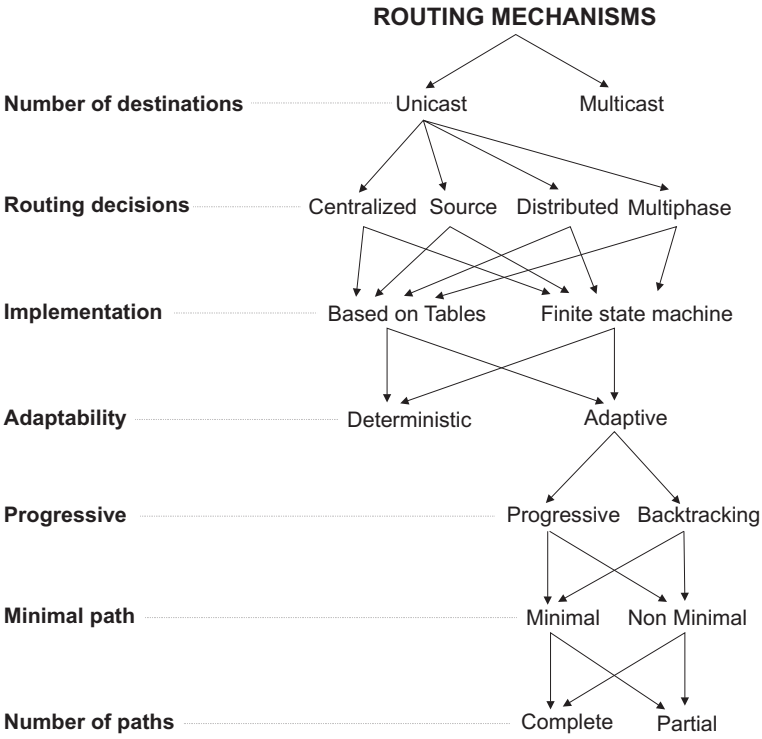


Figure 2.17: A taxonomy of the routing algorithms

Chapter 3

CONGESTION

This chapter describes and analyzes the effect of the congestion problem in interconnection networks, and examines the process of creating a congestion tree. Subsequently, some of the current congestion management mechanism are presented.

3.1 The Problem

Currently, due to the increasingly number of nodes in the systems, the interconnection network has an increasingly greater influence on the overall system performance and, in particular, on the behavior of the applications that are running on them. Ideally, the interconnection network should maintain the maximum throughput once the saturation point is reached in order to maximize the system performance [96]. The maximum theoretical throughput depends on both the bisection bandwidth of the network topology and the applied communication pattern. Figure 3.1 shows the ideal behavior for an interconnection network. In this ideal network, buffers are supposed to have an infinite capacity.

As it can be seen in Figure 3.1, as the value of the injected traffic increases, the system responds with the same increased value for the accepted traffic. When the injected traffic reaches the value “y”, which identifies the saturation point and represents the maximum traffic value accepted by the interconnection network, all the

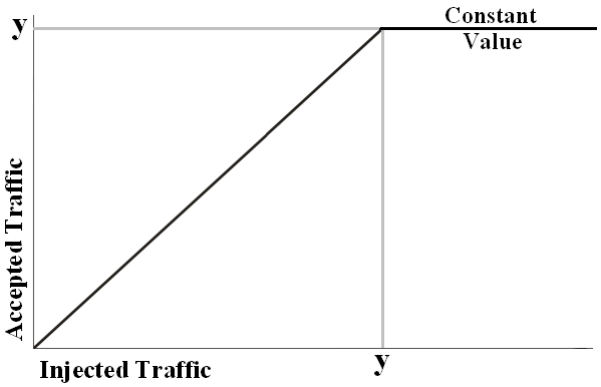


Figure 3.1: Ideal performance for an interconnection network.

additional injected traffic must wait at the (*infinite*) buffers until packets arrive to their destinations. Larger values of injected traffic would maintain a constant value for the accepted traffic. However, indeed, buffer capacity is limited. As a consequence, when a buffer becomes full, packets requesting it will have to remain stopped in their corresponding buffers, preventing, in turn, the advance of other packets.

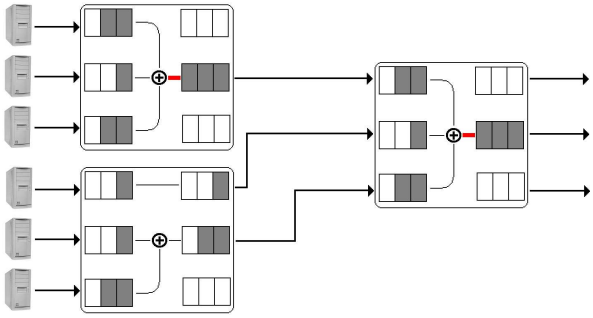


Figure 3.2: Congestion situations in a network.

Figure 3.2 shows a situation in which multiple packets stored at the input buffers of the switches compete for crossing the crossbars towards their corresponding output channels. As the volume of communication increases by the demand of end-nodes, it will also increase the demand on the network resources, and competition to access the same resource [10], usually a link, will occur. This status is known as contention.

In this case, the routing unit will select a packet between the contenders to cross the crossbar, whereas the others will have to wait at input buffers. Packets blocked at the buffer headers prevent the remaining packets of the buffer from being routed, increasing their latency and decreasing their throughput. Thus, the tasks that are waiting for those messages will suffer a delay. This situation is known as Head-of-Line (HOL) blocking, and if it continues for long, the packet accumulation at the buffers can grow excessively and even reach the saturation point. This new situation is known as congestion and causes unacceptable delays and throughput drops, mainly due to the exponential increase of the latency for the packets waiting at the buffers.

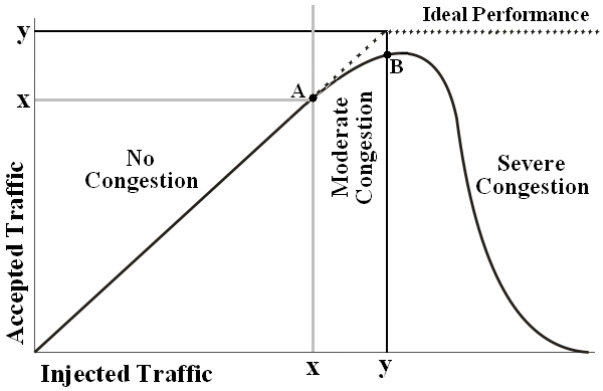


Figure 3.3: Real performance for an interconnection network.

Figure 3.3 shows the real performance for an interconnection network. While the value of the injected traffic into the network is less than the value “x”, the network will be capable of accepting and processing such injected traffic (A point). As the network load continues to increase, packets will start to be accumulated at the buffers giving rise to the congestion phenomenon. Therefore, latency of this packets will increase, and accepted traffic will start to fall behind the injected traffic. This throughput penalization with respect to the ideal throughput is due to the inefficiencies introduced by the applied routing scheme, switching technique, and flow control mechanism. Eventually, if point B is reached, the queues of the nodes will be full. In this situation, if the injected traffic value continues growing, the accepted traffic will drop. Values for injected traffic less than “x” do not generate any kind of conges-

tion because the network is able to accept and manage all the injected traffic. Values within the range [x, y], will create a moderate congestion while the buffers occupancy is growing, but the network continues to be able to manage it. However, values greater than "y" will cause severe congestion into the network, causing, in turn, a drastic reduction in network performance. Basically, the congestion problem occurs when the number of packets that are transmitted through the network exceeds the maximum network management capacity. The objective of congestion management is to maintain the number of packets into the network below the maximum level for which performance start to fall dramatically. In Figure 3.3, the ideal value for the injected traffic is "x", because packets do not suffer from contention and therefore do not increase their latency. However, the value for injected traffic should not reach the maximum value "y" in any situation.

It should be noted, though, that congestion may appear even in the case where the network is being used under its maximum theoretical utilization value. This is mainly due to the imbalance of traffic caused by the routing strategy applied or the traffic pattern injected into the network. Although adaptive routing [30], [57], [43], [92], [85] and load balancing techniques [85], [41] can help to reduce these situations, none of them can completely avoid performance degradation when the network experiences congested situations. In this sense, it is necessary to apply specific techniques to monitor and solve a congestion situation, avoiding the exponential increase in latency and keeping it bounded even with high levels of traffic load.

A simple analysis allow us to illustrate how a congestion situation could negatively impact network performance. The congestion process starts in the network due to the existence of contention for accessing a particular resource (normally a link). Figure 3.4 shows a set of packet flows coming from different origins and crossing the output link L_{yz} and, therefore, sharing its bandwidth.

Assuming that all packet flows are injecting at the maximum rate allowed by the bandwidth of the links (BW_{link}), the link L_{yz} will have its bandwidth used at 100%. As a consequence, it can be deduced that flows Y1, Y2, and the link L_{xy} are contributing with packets at a maximum rate of BW_{link}/3 (33%). In turn, the

link L_{xy} evenly shares its injection rate (BW_{link}/3) between the flows X1 and X2, which will contribute with a maximum injection rate of BW_{link}/6 (16%) each one, because they have to evenly share the L_{xy} bandwidth. Although there is a congestion situation, this is not detrimental to the overall system performance because all packet flows have the same destination and the bandwidth provided by the destination link is fully utilized. All the flows are contributing to the congestion and there are no other affected flows with different destinations. In this case, the flow control mechanism itself will be enough to manage the packet advance and to evenly utilize the available bandwidth while the packet flows are injecting at the maximum rate that the network can accept.

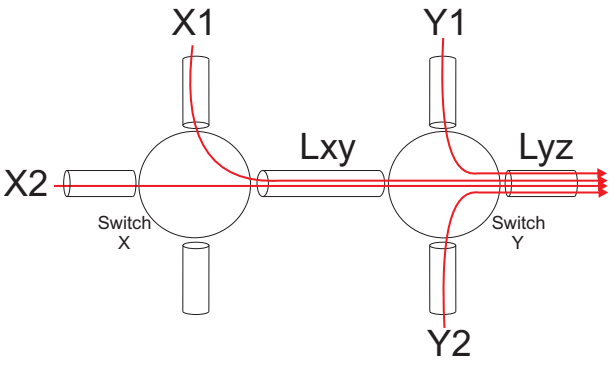


Figure 3.4: A congestion situation not affecting other flows.

Indeed, the real congestion problem arises when a new packet flow, which is not responsible for the congestion and with a different destination, is affected by the arisen congestion situation. In Figure 3.5, a new packet flow X3 has been added, sharing the link L_{xy}, but it does not cross the shared output link L_{yz}.

Given this new situation, the bandwidth of the link L_{xy} should be proportionately allocated between the three competing flows, so the maximum accepted rate of each flow (X1, X2, X3) should be BW_{link}/9 (11 %) in order to acomplish the available bandwidth limit imposed by switch *Y*. However, the flow X3 can suffer from HOL blocking if flows X1 and X2 have packets stopped at the switch *X*, thus hindering the normal progress of the flow X3. Notice that packets will be stopped at the output

ports of the switch *X* due to the fact that the aggregated bandwidth initially requested by the three competing flows would be three times higher than that provided by the link L_{xy}. In turn, the link L_{xy} will be idle at least $2BW_{link}/3$ (66%) of its time wasting its bandwidth, which corresponds to the periods in which the flows Y1 and, Y2 are transmitting to the output link L_{yz}.

As the flow X3 does not require to cross the congested output link L_{yz}, it could harness the idle bandwidth of the L_{xy} link by increasing its injection rate from $BW_{link}/9$ (11%) up to $7BW_{link}/9$ (77%), and reducing its packet latency. This way, an improvement in overall system performance could be achieved. Unfortunately, unless the flows X1 and X2 adjust their injection rate accordingly, the HOL-blocking effect will prevent flow X3 from harnessing the idle bandwidth of the L_{xy} link.

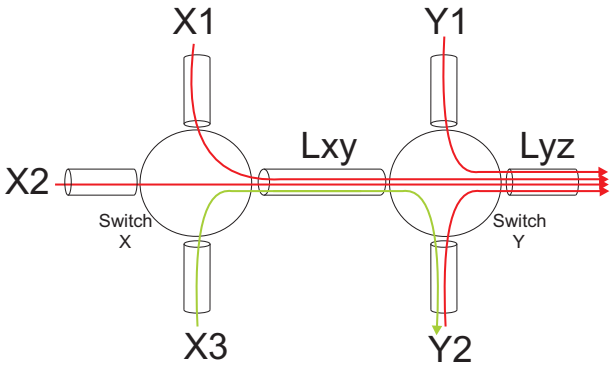


Figure 3.5: A congestion situation affecting other flows.

Traditionally, congestion in high-performance networks has been attacked by over designing the network. The basic idea is to dedicate a larger number of resources (switches and links) than strictly necessary. As shown in Figure 3.6, the saturation point would be increased to higher values if the network were over designed. However, this technique is not totally effective because if the injected traffic rate continues to rise, a value at which the system is saturated will be eventually reached, thus increasing the latency and causing a reduction in network performance.

There are some factors that make oversized systems unfeasible.

- The relative cost of the network components regarding the overall system cost.

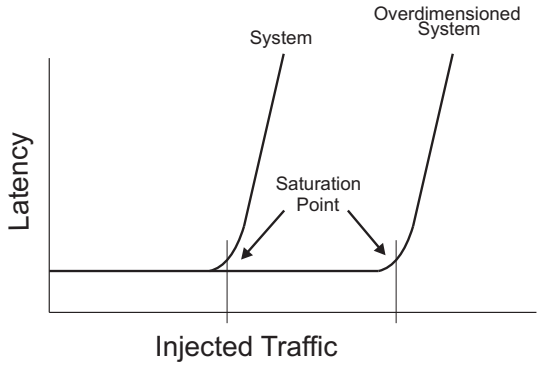


Figure 3.6: Oversizing the network.

Although the popularity of interconnection networks is increasing (InfiniBand, Myrinet, Quadrics, etc), their components (links, switches, network interfaces) are very expensive when compared to the processors used.

- Power consumption is becoming increasingly important. As the Very Large Scale Integration (VLSI) technology advances and link speed increases, interconnects are becoming to consume a growing fraction of the total system power [84]. Taking this into account, there are two ways of reducing network power consumption:

  a) Reducing the number of links in the network, using the remaining links more efficiently. It causes that the remaining links have to endure more traffic load and therefore it could create a congestion situation.

  b) Using some frequency/voltage scaling techniques to reduce link power consumption. As a consequence, links may temporarily have a lower bandwidth and therefore it could create a congestion situation.

Consequently, cost and power consumption constraints require efficient utilization of network resources. Building oversized networks does not appear to be an acceptable solution, as the cost and power consumption are high. Therefore, effective and efficient congestion management mechanisms are required to avoid congestion situations.

3.2 The Congestion Process

In order to gain a more comprehensible insight into the congestion management strategies analyzed in the next sections, an in-depth analysis of the process of creating a congestion tree, and how it spreads along the network from the root to the leaves is carried out in what follows.

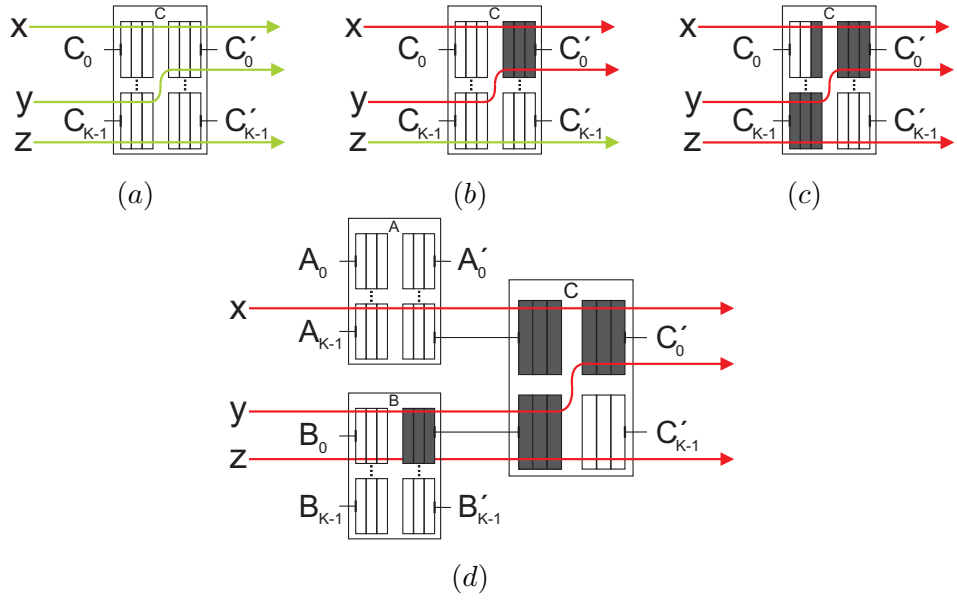


Figure 3.7: The congestion tree creation process.

Figure 3.7 shows the process of creating a congestion tree step by step. In Figure 3.7 (a), an initial situation for the three different flows *x*, *y*, and *z* crossing the switch *C* without contention is shown. Packets belonging to the flow *x* cross the switch from the input channel C_0 to the output channel C'_0. The flow *y* also advances from C_{k-1} toward C'_0, and finally the flow *z* from C_{k-1} toward C'_{k-1}. Notice that, in order to prevent switches from becoming a bottleneck, the internal bandwidth of the switch crossbar is usually higher than the channel bandwidth. As the bandwidth speedup increases the switch complexity, often a maximum speedup of two is used [42]. Now, let us assume that the overall input traffic rate for the combined flows *x* and *y* (*x*+*y*) is greater than the bandwidth of the output channel C'_0. In this situation,

packets belonging to flows *x* and *y* will have to compete for the output channel C_0' and, as a result of that, the output buffer will start to accumulate packets as Figure 3.7 (b) shows. Notice that, with a speedup equal to 1, packets would start to accumulate at their input buffers instead of at their output buffers. If this situation remains, and flows *x*, *y*, and *z* continue injecting traffic into the switch, packets may begin to accumulate at the input buffers, spreading the congestion along the switch input channels. Figure 3.7 (c) shows a new possible step in the congestion creation process, in which, due to the fact that incoming packets are stopped at the input buffer of the channel C_{k-1}, the head-of-line blocking phenomenon could appear. As a consequence, the advance of packets belonging to the flow *z* and directed toward the non-congested channel C_{k-1}' would be delayed, causing the degradation of the switch performance. If this situation persists, congestion will be spread along the previous switches, stopping and accumulating packets at the output buffers of those switches. This situation provokes a new congested output channel at the previous switch and starts a new congestion process. Figure 3.7 (d) shows how the congestion has been spread along the congestion tree toward its leaves, affecting several switches. In order to stop the propagation of congestion without delay, packet flows provoking congestion have to be correctly identified at the switch where contention starts.

Following this analysis, where an initial contention on a switch triggers the creation of a congestion tree, we can identify and classify the different flows that can be involved in the congestion tree.

Figure 3.8 shows a part of an interconnection network where a set of switches are connected by a set of links. As it can be seen, there is a congestion situation where two kinds of flows can be identified; namely the red flows, which will be referred to as “Hot-Flows”, and the green flows, that we will referred to as “Cold-Flows”. We define *hot-flows* as those flows which generate congestion and expand the saturation tree quickly. On the other hand, *cold-flows* are those flows providing packets towards other destinations that are not already generating congestion on their own. Rather, they are suffering the consequence of the congestion situation caused by *hot-flows*. This congestion may be originated either by packets destined to the same end-node

or by packets destined to different end-nodes but crossing some shared links along their paths that become congested.

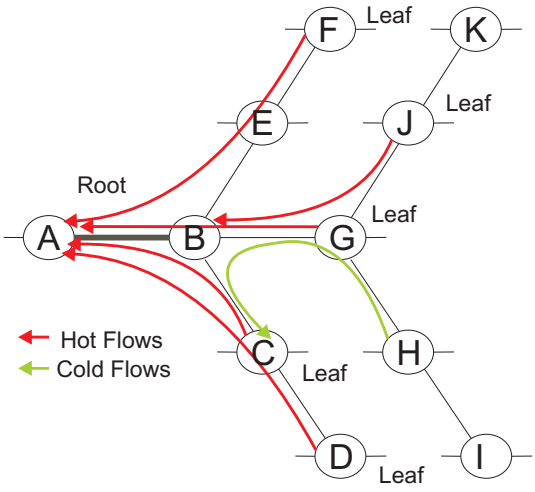


Figure 3.8: Flow identification.

In Figure 3.8, the *hot-flows* converge in the link connecting the switches A and B creating what is known as *the root of the congestion tree*, that is, the point where congestion is originated. This congestion would not be a problem, as seen before in Figure 3.4, if all packet flows were destined to the same end-node and did not affect other flows. The back pressure action exerted by the flow control of the network would automatically limit the injection rate for each packet flow. On the contrary, in this situation there are packet flows whose final destination is not beyond the root of the congestion tree but they share some links belonging to the congestion tree. This is the case for the green flow in Figure 3.8. When crossing the link between nodes G and B, packets belonging to this flow will suffer an increase in latency, even despite the fact that they may be sharing underutilized links.

As it can be seen in Figure 3.8, the key lies in the correct identification and classification of the flows in order to apply corrective actions. In this way, the corrective actions will be applied only over the *hot-flows*.

3.3 Basic Features of a Good CMM

The applied techniques in congestion management must meet a set of requirements to ensure the functionality and the performance of the network. Basically, a congestion management mechanism should satisfy three conditions [10]: robustness; not penalizing network behavior in the absence of congestion; and finally, not generating new problems.

Firstly, a congestion management mechanism is a robust mechanism when it works properly regardless of network parameters and traffic features, such as the network load, the packet size, network topology, etc. This represents an additional difficulty for the correct behavior of the mechanism, since the restrictions that are appropriate to avoid saturation for a given traffic pattern can be excessive for other destination distribution, thus unnecessarily increasing latencies before saturation, or what is worse, may be unable to avoid congestion.

Secondly, the congestion management mechanism should not penalize the network performance when it is not saturated. Namely, when the traffic load has not yet exceeded the saturation point, which is the most frequent situation [77], the congestion management mechanism should not restrict packet injection, nor introduce unnecessary overhead.

Thirdly, the restrictions applied by the mechanism to reduce congestion should not create new problems. Many mechanisms require an additional monitoring system that increases the complexity of the system because they either require new signals [56], [86], [91] or need to send additional information that increases the traffic load, worsening the congestion situation [17], [61].

Finally, congestion management mechanism should be fair regarding to all network nodes and the time to react should be properly bounded to avoid late responses in solving the congestion problem. Notice that, if these conditions are not taken into account, some nodes could start to apply strict corrective actions before others have detected the congestion problem, thus reducing network load. As a result, those nodes will not trigger their congestion control actions. This could result in starvation in some nodes while others continue to inject their packets into the network without

detecting the congestion problem.

3.4 Congestion Control Strategies

As analyzed in the previous sections, the ability to manage a high number of packets without causing a large increase in latency is a critical issue for high-performance networks. Moreover, if the applications that are being run have fine granularity, this problem is even more critical because of the high communication bandwidth requirements.

A possible solution to the congestion problem could be to discard the congested packets, radically eliminating the problem. This solution is applied to lossy networks that are commonly used in communication systems. However, in this thesis, we present our work for lossless networks, which are commonly used in HPC systems, NoCs, and Clusters of PCs. In these networks, it is unacceptable to drop packets as the packet latency would significantly increase due to packet retransmissions. Therefore, due to the characteristics of the networks as well as the systems where they are applied, a congestion management mechanism is required in order to prevent system performance degradation.

Congestion management in lossless networks has been widely studied over the years [75], generating a lot of research and proposals. Techniques that attempt to manage and solve the congestion problems can be divided mainly into two groups, proactive strategies and reactive strategies. The first group of strategies is based on avoiding the congestion, whereas the second group includes mechanisms based on detecting and recovering from congestion.

3.4.1 Proactive Strategies

Proactive strategies are intended to ensure the absence of congestion by a prior knowledge of the resources required during the execution of the applications. These strategies try to keep the network performance below the saturation point, so that congestion never occurs. These strategies are also known as *congestion prevention strate-*

gies.

As an example, in a storage system, one way to avoid congestion is by distributing the data through different devices, also providing different routes to reach them. This way, the possible contention caused is minimized.

These proactive strategies have been applied by software [97], [12] or hardware [95]. The main problem of these strategies is their implementation. It is not always possible to implement them. Generally, they have been designed for a specific environment, or they require additional network components that oversize the network and thus increases the implementation cost.

Other proactive strategies require the reservation of network resources in advance. That is, the entire data path is reserved before transmission takes place. This requires a previous knowledge of the necessary resources for each transmission, and the reservation of the entire path. The main drawback of this strategy is that this knowledge is not always available in all the possible environments. Moreover, this reservation of network resources adds a considerable delay and may involve some packet overhead in the network.

Thus, proactive strategies are only used in protocols aimed to provide Quality of Service (QoS) as in Asynchronous Transfer Mode (ATM) [78], and not for high-performance interconnection networks.

3.4.2 Reactive Strategies

Reactive strategies are also known as congestion recovery strategies, and they are based on a closed-loop control model that reacts after congestion has begun. They detect and solve the congestion on fly, at the moment it takes place [46], [4], [83]. As a result, sources reduce or even stop the packet injection rate depending on the congestion level. This strategy is usually based on three basic steps: detection, notification, and correction. During the detection phase, some dynamic parameters of the network are evaluated to determine the onset of congestion. This congestion detection can be carried out by different methods.

A first proposal consists in detecting congestion when packets remain blocked

in the network longer than a predefined threshold. It is noteworthy that the blocking threshold depends on the packet length [56], [55], [47], [86]. This option works at the connection level between the source-destination pair, testing the status of the links.

Another option is based on measuring certain individual resources such as the switch buffer occupancy [31], [48], [62], [76], [91], [94], [82]. If the occupancy at any buffer exceeds a predefined threshold, then packets crossing the switch will be marked. As a consequence, switches are able to detect and identify packets which are supposedly contributing to congestion.

Finally, congestion can be detected by monitoring the number of pending memory requests [83].

After detecting congestion, the source hosts that are injecting too much traffic have to be warned in order to evenly reduce their injection rate. For this purpose, several techniques can be applied. The first one can be carried out by means of broadcast messages [86], [94], [91]. This solution does not guarantee that notified sources are only those that are injecting traffic to the congested links. Therefore, hosts non-responsible for congestion will receive warning packets and could reduce their injection rate and, therefore, cause a decrement in network performance. Moreover, broadcasting control packets waste network bandwidth, thus penalizing network throughput even more.

Other proposals notify those sources directly connected to the switch where the congestion is detected [24], [63], [11], [9]. This option does not need to transmit additional information beyond the switch. However, it could also penalize the local sources which are not involved in the congestion.

Finally, another option is based on notifying those sources sending packets toward the congested area [55], [48], [82], [76], which results in a better use of the available bandwidth. Generally, this option applies what is known as Explicit Congestion Notification (ECN) to warn the origin hosts. This notification strategy presents a number of advantages. In particular, ECN has greater simplicity and ease of implementation in current commercial network technologies. As the proposed congestion management mechanism in this thesis use this option, we will analyze it more in

depth in the next section.

Once the sources receive the congestion notification, they apply certain actions in order to eliminate the congestion. The most applied one is the injection rate limitation (also known as message throttling) [24]. This option can be carried out by reducing or by completely stopping the injection rate. This reduction can be achieved by reducing the number of injection channels [9], or by inserting waiting intervals in-between consecutive packet transmissions [56], [55], [48], [82], [76].

Other mechanisms try to solve the problem by temporarily separating the flows responsible from the flows non-responsible for congestion in order to remove the head-of-line blocking effect [31]. To do so, it is necessary to incorporate a set of additional buffers. When the mechanism detects packets stopping the normal advance of other packets into the network, the additional buffers will harbor those packets causing the head-of-line blocking phenomenon. Later, these packets may continue their journey to their destinations through some "slow roads" or, alternatively, be reinstated again if congestion disappears. This proposal is interesting but it would be desirable to test it under heavy congestion situations, because the number of buffers is finite. So, when those buffers are full, the congestion process will not be stopped. Moreover, there are some proposals that try to resolve head-of-line blocking either at the switch level [6], [58], [86] or at the network level [25], [62].

3.5 Congestion Management Based on ECN

Basically, the congestion management mechanisms based on the ECN use a congestion detection strategy based on marking packets in transit, usually when a predefined threshold is exceeded. This packet marking action is carried out at the switches of the interconnection network.

The marked packet will continue its travel toward the destination node, carrying out the congestion detection information. When this packet reaches its final destination, the ECN technique takes advantage of the ACKnowledgment packets (ACK) sent back to the source to carry out the congestion detection information to the origins hosts. As a result of receiving marked ACK packets, those origin hosts will

apply some corrective actions, normally based on limiting the injection rate into the network. Figure 3.9 shows how the ECN technique works.

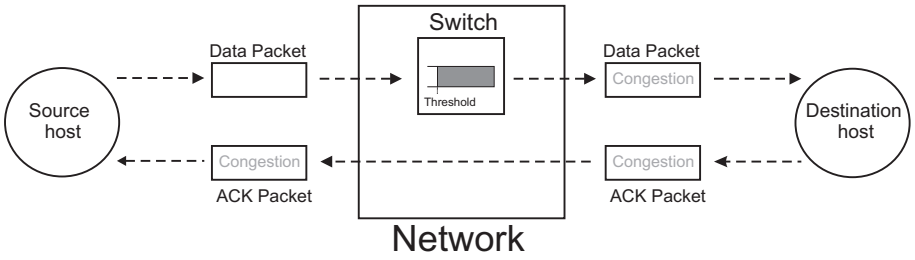


Figure 3.9: The explicit congestion notification technique.

There are some ECN proposals in lossless interconnection networks. Next, we present the current proposals and their main characteristics for congestion management mechanism in clusters.

3.5.1 Current ECN Proposals for Clusters

InfiniBand

InfiniBand [48] is a clear example of a successful interconnection network for clusters. The InfiniBand Trade Association is comprised of leading enterprise IT vendors. It found its niche in data centers and high-performance computing systems which require high bandwidth and low latency, but it did not appear to solve any problem that had not been solved previously by other network technologies. It emerged as a standard, easing its commercialization because of its reduction in cost, thus allowing the expansion to new and wider sectors. InfiniBand defines a System Area Network (SAN) in which a set of processing nodes and I/O units are connected to the network fabric through point to point links using channel adapters. The network is divided into hierarchical subnets. Each subnet exchanges packets solely by traversing switches. Links provide bidirectional and high-speed connection between two ports. Figure 3.10 shows an example of an InfiniBand network.

Although InfiniBand defines in its specifications an optional congestion management mechanism, it leaves open some aspects about it. The congestion management

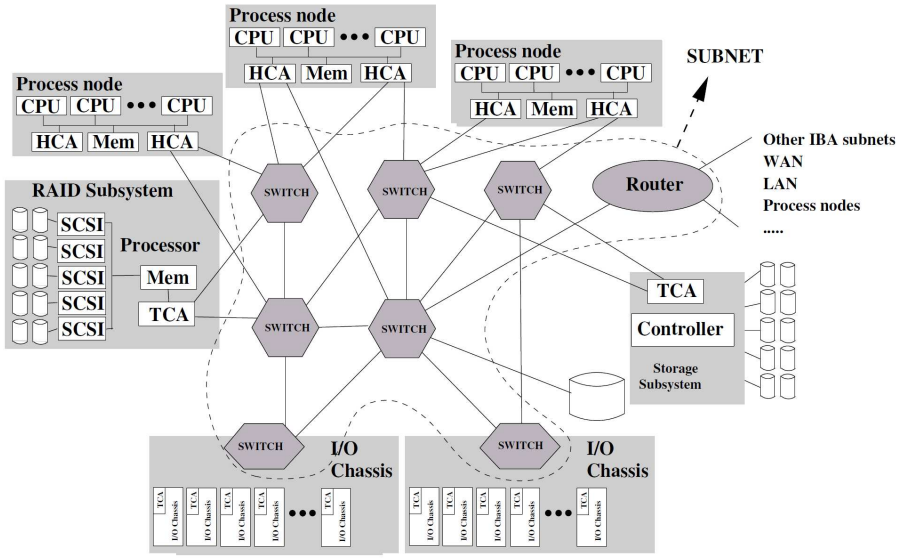


Figure 3.10: An example of an InfiniBand subnet.

mechanism is implemented with a management agent called the congestion control agent. It is based on ECN, and therefore defines a three-step process to detect congestion and to adjust the packet injection.

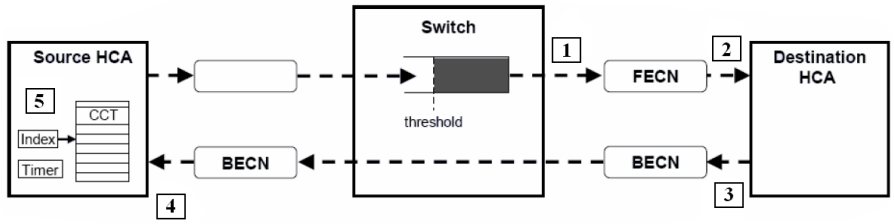


Figure 3.11: The process of congestion detection and notification in InfiniBand networks.

Figure 3.11 shows the process to detect and warn about a congestion situation in InfiniBand networks. The InfiniBand specs propose a buffer threshold which activates the packet marking strategy. The threshold value is established based on a value scale from 0 up to 15. The value 0 indicates that no packets have to be marked,

so no congestion will be detected at all, and the value 15 indicates a very aggressive marking strategy. All other values, between 1 and 14, define a uniform distribution in which the value of 1 indicates a low probability of marking and the value of 14 indicates a very high probability of marking packets. The exact meaning of the threshold values (0..15) is decided by the switch manufacturer criteria.

When a switch detects that the occupancy in a buffer goes beyond the predefined threshold, it reports this congestion situation by marking the packets that are stored in the buffer. To do this, a bit called Forward Explicit Congestion Notification (FECN) bit, in the base transport header of the packet, is set. When this packet reaches the destination node, the FECN bit value is copied to the ACK packet that is returned to the source. To do this, a bit called Backward Explicit Congestion Notification (BECN) bit, in the header of the ACK packet, is used. Once the ACK packet reaches the source node, the BECN bit is analyzed. If it is set (BECN=1), the congestion management mechanism will apply corrective actions in order to reduce the packet injection rate. InfiniBand specs propose to adjust the injection rate by using a table called Congestion Control Table (CCT) allocated in the Host Channel Adapter (HCA) of the source hosts, as shown in Figure 3.11, which is based on the Waiting Interval Insertion technique. The content of each position in this table, pointed out by a certain index value, represents a time value and it is the elapsed time between consecutive packet injections (inter-packet delay). A higher index value represents a greater elapsed time. Thus, the greater the number of marked packets received, the greater the increase in the value of the index of the table.

Eventually, if no more marked packets are received, which could be indicative of having ceased the congestion, then the packet injection rate should recover the initial values. This is carried out through a timer which triggers a signal to decrease the index of the control table and therefore to increase the injection rate. This injection rate recovery is commonly gradual. The adjustment of the different parameters of the control mechanism (index, table content, threshold, etc.) is a responsibility of the congestion control manager.

Some congestion management mechanisms based on the InfiniBand specs have

been proposed to solve the congestion problem. In particular, we will show Renato's Proposal and Pfister's Implementation. Next, we describe these congestion management mechanisms in terms of the packet marking techniques and the congestion correction strategies applied by them, such as they were proposed.

Renato's Proposal

This congestion management mechanism [82] is an end-to-end congestion management scheme for InfiniBand that consists of an ECN packet marking mechanism combined with a source response mechanism that applies injection rate control combined with a window-based limitation. In particular, it applies a kind of input packet marking strategy to warn about a congestion situation. To do this, switches need to have buffers at the input channels.

The proposed packet marking mechanism operates in three steps. First, a switch input buffer triggers packet marking each time it becomes full. Second, any output link that is requested for at least one packet stored in such a full input buffer is classified as a congested link. Third, all packets stored in any input buffer at this switch that are destined to any congested output link will be marked. It should be noted that, in order to implement these steps, an expensive scan of all input buffers in a switch, even when only one becomes full, is necessary. To this end, two counters are defined for each output link. The first counter C_1 records the current number of packets in the switch that are waiting for that output link; C_1 is incremented and decremented as packets enter and leave the switch. The second counter C_2 records the number of subsequent packets that need to be marked when transmitted on the output link. Counter C_2 is initialized to zero. Whenever a buffer becomes full, the value C_1 is copied to counter C_2. Then, the output port starts marking the next transmitted packets, decrementing C_2 at each transmission, until it reaches the value zero. Notice that this action of copying the value from C_1 to the C_2 is performed each time a packet arrives at this input buffer and the buffer is full. As can be seen, this counter strategy is not an effective solution because it may mark different set of packets, since packets can be transmitted out of order.

Next, in response to the reception of a marked packet, the corrective actions applied by the origin hosts consist in limiting the packet injection by using a Window-Based technique combined with a Waiting Interval Insertion technique. They propose a Window-Based technique based on a Static Window (SW) size equal to one for any situation because a larger value completely saturates the network. In addition, an injection rate control based on inserting waiting slots is used. For rate control, they evaluate different functions to adjust the injection rate: Additive Increase Multiplicative Decrease (AIMD), Fast Increase Multiplicative Decrease (FIMD), and Linear Inter-Packet Delay (LIPD). As a result of their study, the AIMD reduction function has the worst performance on all ranges achieving up to 10% lower utilization than the best functions FIMD and LIPD. So, they propose two novel source response functions FIMD and LIPD for dynamic and static traffic patterns, respectively. In particular, the FIMD function uses a constant value (m=2) to reduce packet injection by dividing the injection rate, and the same constant value to increase the injection rate based on an exponential function. On the other hand, the LIPD function applies a reduction in packet injection based on the inter-packet delay design feature of InfiniBand. The inter-packet delay represents the idle period length that is inserted between the injection of consecutive packets of a flow, expressed in units of packet transmission. Basically, it increases the inter-packet delay by one each time a marked ACK packet arrives at an origin host.

Pfister's Implementation

The general congestion management mechanism proposed for InfiniBand defines a quite general approach for congestion management. The proposal lacked both guidance for setting its parameters and demonstration of its effectiveness, because values for thresholds and other variables are left open to the vendor criteria. It was not even demonstrated that there were any parameter settings that worked at all, avoiding instability or oscillations. Pfister's Implementation [76] is targeted to this scenario, reporting an extensive evaluation of the InfiniBand proposal under different scenarios and congestion control parameters. That implementation presents the first guidance

for setting the parameters for InfiniBand and demonstrates that this is the first effective solution to hot-spot contention. Pfister's Implementation applies an output packet marking strategy for packet marking. Switches need to have queues associated to the output channels in order to define a threshold and to apply its packet marking strategy. Unlike the Renato's proposal, this approach does not use any window-based technique to manage congestion. However, after receiving marked packets at origin hosts, sources reduce the injection rate by applying a Waiting Interval Insertion technique. Each time a marked packet is received, the inter-packet delay value is enlarged by inserting a new waiting slot which corresponds to increase the index into the CCT. In particular, the waiting slot size applied depends on the Hot-Spot Degree (HSD) that indicates the number of sources contributing to the hot-spot traffic, and the number of network ports. Notice that this HSD depends on the traffic pattern applied.

3.5.2 Main weakness of current proposals

Once the current congestion management mechanisms for multistage interconnections networks have been presented, it is interesting to highlight some weaknesses that they present in both the *Packet Marking strategies* and in the *Corrective Actions* applied, in order to justify the work done in this thesis, which will be described and evaluated along the next chapters.

- As it has been shown, the applied *packet marking strategies* vary depending on the place where packets are marked. Basically, there are two strategies, that is, marking packets either at input (Renato's proposal) or output buffers (Pfister's implementation), carrying out these actions by setting just one bit in the packet headers. To this end, a threshold either at input or output buffers is predefined in order to detect and mark packets provoking congestion. Given that both packet marking strategies dedicate only one bit to mark packets in transit, the congestion mechanisms are not able to detect different levels of congestion. Further, it is not possible to guarantee that all marked packets are responsible for congestion.

- In its turn, the *corrective actions* applied by the current proposals present some drawbacks. On one hand, applying a Window-Based strategy, as Renato's proposal does, seems simple to implement and appropriate to palliate the congestion. However, if the window has a fixed value, as Renato's proposal does, it will be difficult to adjust the window with the correct value for any traffic condition in the network. Moreover, initializing the window with the value of one, as Renato's proposal does, seems to cause a negative impact on the network throughput (this would be the case, for example, when only one host sends packets towards a single destination host).

  On the other hand, by applying an injection reduction technique based on an index into the table which is incremented by a constant value depending on the Hot-Spot Degree that indicates the number of sources contributing to the hot-spot traffic, as Pfister's implementation does, can not be considered a generalized congestion management mechanism because it depends on the traffic load applied.

As a result of these drawbacks detected, it is necessary to conduct a thorough analysis of new mechanisms for congestion detection by marking packets in transit and exerting congestion correction in a more precise and effective way.

Chapter 4

CONGESTION DETECTION TECHNIQUES

This chapter describes and analyzes in depth different packet marking strategies proposed up to now in the literature together with two new packet marking approaches proposed by us, in order to ascertain the pros and cons of each congestion detection technique.

4.1 Introduction

As described in the previous chapter, there are different proposals for congestion detection. That is, detecting congestion when packets stay in the buffers longer than a defined threshold, monitoring the occupation level in the buffers at the switches, or depending on the number of pending packets. As a result of the congestion detection, packets in transit are marked in order to carry out the congestion information toward the origin hosts.

In this thesis, we have chosen the mechanism based on monitoring the buffer occupancy at the switches due to this implementation easiness. We will analyze the different packet marking strategies in order to find out their advantages and disadvantages.

The packet marking strategies vary depending on the place where packets are marked. Basically, packets in transit are marked by setting one or two bits in the packet header. For this purpose, unused bits in the packet header can be dedicated to carry the congestion information. To this end, a threshold at input or/and output buffer of the switches has to be predefined in order to detect an excessive accumulation of packets in those buffers.

4.2 Packet Marking Strategies

We present and analyze in depth different packet marking strategies proposed up to now in the literature together with two new packet marking approaches proposed by us. In particular, five different packet marking strategies. Firstly, the Input Packet Marking (IPM) strategy based on marking packets at input buffers of the switch, and its variant, the Renato's packet marking strategy (RPM). Secondly, the Output Packet Marking (OPM) strategy based on marking packets at output buffers of the switch. Thirdly, the Marking and Validation Packet Marking (MVPM) strategy based on marking packets at input buffer and a subsequent validation at output buffers of the switch. Finally, the Input-Output Packet Marking (IOPM) strategy based on marking packets both at input and output buffers of the switch.

4.2.1 The Input Packet Marking Strategies

Basically, if the applied strategy is based on marking packets at input buffers, switches will mark packets once a packet arrives at an input buffer and the buffer occupancy surpasses a predefined threshold. We will refer to this strategy as Input Packet Marking strategy (IPM).

Following the congestion process analyzed in Figure 3.7, we can observe that for this packet marking strategy, the congestion tree has to grow at least to the point of reaching the input buffers placed at the same switch where congestion starts, as it is shown in Figure 4.1, before the packets involved in the congestion tree are detected and marked. When a situation similar to the one shown in Figure 4.1 is reached,

any incoming packet belonging to flows *y* and *z* will be marked regardless of its destination. That is, both packets forwarded toward the congested link C'_0 or the non-congested link C'_{k-1} will be marked.

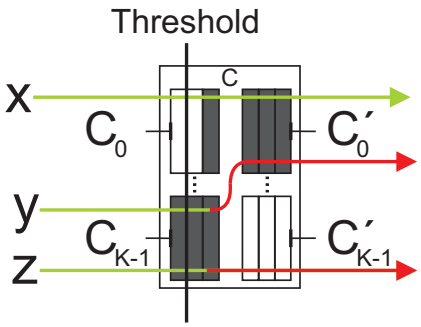


Figure 4.1: The input packet marking strategy.

The variant applied by Renato's Strategy (Renato's Packet Marking RPM) slightly differs from the original IPM strategy. As described in the previous chapter, RPM also marks packets at input buffers but instead of marking packets when they arrive at a full input buffer, it uses two counters to record the incoming packets to an input buffer and later to mark packets when they are crossing the switch toward a supposed congested output link. Again, to detect congestion when applying the RPM strategy, the congestion tree has to grow at least to any input buffer of a switch.

To apply any of these two strategies, just one of the unused bits in the packet header should be dedicated to this purpose.

Analyzing in depth how the IPM and RPM strategies work, it can be noticed that applying any of these two packet marking strategies at input buffers, the congestion management mechanisms will suffer a delay in detecting the beginning of a congestion situation. That is, taking into account how the congestion tree is created (see section 3.2), output buffers have to be filled before any input buffer starts to accumulate packets. Later on, if the occupancy at the input buffer surpasses the predefined threshold, packets will be marked. Therefore, the congestion detection process suffers a delay based on the time needed, firstly, to fill any output buffer, and secondly, the time to accumulate enough packets at any input buffer to surpass the predefined

threshold.

As a consequence of this delay and in order to reduce the time required to detect congestion, a new strategy based on marking packets at output buffers seems more appropriated.

4.2.2 The Output Packet Marking Strategy

If the applied strategy is based on marking packets at output buffers, switches will mark packets once a packet arrives at an output buffer and the buffer occupancy surpasses the predefined threshold. We will refer to this strategy as Output Packet Marking strategy (OPM).

In this case, following the same example shown in Figure 3.7, congestion will be detected when the occupancy of any output buffer exceeds the predefined threshold. As a result, all the packets belonging to the flows *x* and *y*, in Figure 4.2 and addressed to the congested output link C'_0 will be marked. Notice that with this strategy, packets belonging to the flow *z* are not being marked.

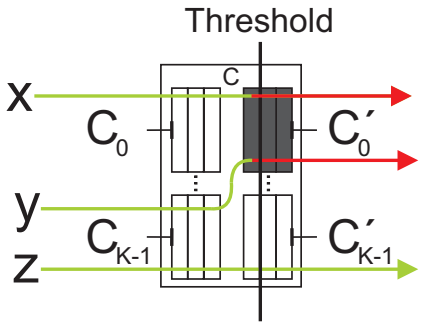


Figure 4.2: The output packet marking strategy.

Again, to apply this packet marking strategy, one of the unused bits in the packet header should be dedicated to this purpose.

4.2.3 Weaknesses of the Input and Output Packet Marking Strategies

As described above, both packet marking strategies, that is IPM and OPM, dedicate just one bit from the set of unused bits in the packet header to carry out the congestion

information. Although the strategies detect congestion when it appears by setting the congestion bit, it is not enough to correctly detect the flows responsible for congestion. Therefore, these strategies are not able to correctly classify flows as either *cold* or *hot-flows*. As a result, wrong marking actions could be carried out by incorrectly marking packets belonging to flows non-responsible for congestion.

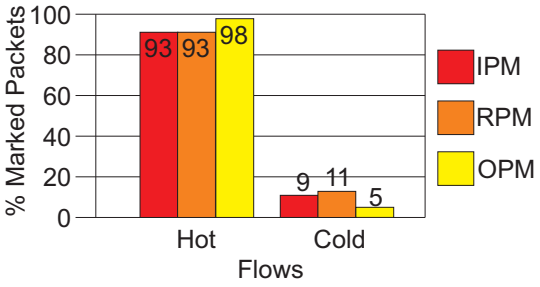


Figure 4.3: Percentage of marked packets for the IPM, RPM, and OPM strategies.

To quantitatively show the lack of accuracy of these mechanisms to detect *hot-flows* correctly, Figure 4.3 shows the percentages of marked packets by the three packet marking strategies presented in previous sections, that is IPM, RPM, and OPM. Additionally, values for both types of flows, *cold* and *hot-flows*, have been separately represented. Notice that, although those results have been obtained for a Perfect-Shuffle network 4-ary 5-fly with a traffic load based on a hot-spot traffic and with a medium injection rate (see section 7.2), similar results have been achieved for the other network configurations and traffic loads.

As can be seen, the three strategies effectively detect and mark most of the hot-packets, (values between 93% for IPM and RPM, and 98% for OPM). In particular, the OPM strategy achieves better results as an indicative that the detection of hot-packets at the ouput buffers is a better option. However, analyzing the values for marking cold-packets, graph shows values between 5% up to 11% of cold-packets. This means that the three strategies do not correctly classify both types of packets and, as a consequence of that, wrong corrective actions could be taken over flows non-responsible for congestion. Notice that, as the number of cold-packets is far larger than the number of hot-packets in our simulations, a small difference in the per-

centage value of marked cold-packets represents a great absolute amount of wrongly marked packets, which could penalize the applications that are sending or waiting for those packets.

An interesting analysis can be obtained if results from the IPM and RPM strategies are compared in Figure 4.3. In particular, both strategies achieve the same values for marking hot-packets, but a reduced value is achieved for cold-packets if the IPM strategy is applied instead of applying the RPM. It is due to the fact that RPM applies a variant of the IPM, which do not ensure that the packets crossing toward the output channels of the switch, and being marked, are those packets detected at the input buffers and classified as responsible for congestion. This behavior is due to the defined protocol for the counters (as described in section 3.5.1).

As a result, dedicating just one bit to carry out the congestion information shows some weaknesses. Firstly, both IPM and OPM strategies are not able to correctly identify the flows truly responsible for congestion. This, in turn, could affect other flows or even other switches not involved in the initial congestion problem. Secondly, they are not able to detect whether the congestion is or is not affecting other flows non-responsible for congestion. Thirdly, they are not able to handle different levels of congestion severity.

Therefore, although the aim of a well-structured packet marking strategy is to detect and mark packets belonging to flows responsible for congestion, the key to reach good results is to avoid marking packets belonging to non-responsible flows.

Next, a study of a new packet marking strategy based on a combination of both IPM and OPM strategies has to be analyzed in order to try to avoid the detected weaknesses.

4.2.4 The Input-Output Packet Marking Strategy

Next, we propose a new packet marking strategy, referred to as Input-Output Packet marking (IOPM), that combines packet marking at both input and output buffers, in such a way that packets can be indistinctly marked at input or output buffers. To this end, switches must have buffers at both input and output channels to establish

the corresponding thresholds. Moreover, two bits are dedicated in the packet header, the Marking Bit_in (MBin) and the Marking Bit_out (MBout). To implement the mechanism in a standard interconnect, we can use any of the header bits usually reserved by the specs for vendor applications, as seen in Figure 4.4.

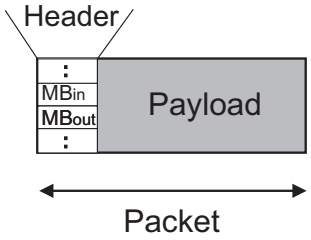


Figure 4.4: The marking bits in the packet header used by the IOPM strategy.

To properly appreciate how this strategy works, and following the example shown in Figure 3.7, let us review in detail how the packet marking process is carried out. Figure 4.5 shows a general scenario where some flows inject packets to the same target, leading to a congestion situation.

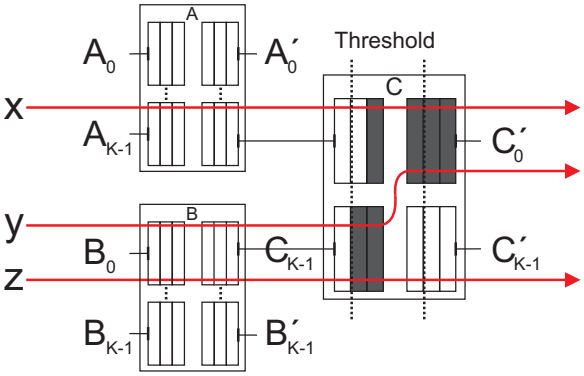


Figure 4.5: Packet marking process carried out by the IOPM strategy.

Let us assume that hosts *x* and *y* inject traffic toward the output channel host C'_0. Additionally, the host *z* injects traffic toward the output link C'_{k-1}. In this scenario, if the injection rate *x*+*y* surpasses the bandwidth of C'_0, packets arriving to the switch *C*, but not immediately transferred to the output link C'_0, will begin to accumulate at

the output buffer[1]. These are the packets really provoking congestion and the output link becomes the root of the congestion tree. In order to solve the problem, once it is detected, the injection rate of these flows should be reduced. Otherwise, if the situation goes on, packets will begin to accumulate at the input buffers of the switch *C*. Some of the packets accumulated at the input buffer C_{k-1} will be destined to the congested link (*y* flow), that is, *hot-flows*. However, others will be destined to other links (*z* flow). These *cold-flows* can suffer head-of-line blocking due to the First-In First-Out (FIFO) policy applied by buffers. In this scenario, the IOPM strategy works as follows:

1. Packets arriving at an input buffer are marked if the number of stored packets in the buffer exceeds a predefined threshold. This is performed by activating the Marking Bit_in (MBin=1) in the packet header.

2. In the same way, when a packet is forwarded through a saturated output link, we proceed to mark it by activating the Marking Bit_out (MBout=1). We assume that an output link is saturated when the number of packets stored in its buffer exceeds the predefined threshold. Again, only new incoming packets to the output buffer are marked rather than marking the ones stored at the congested output buffer.

Notice that both marking actions can be carried out several times on the same packet during its journey from the origin to its destination, but, on the contrary, it will never be unmarked. Notice that, by dedicating two bits in the packet header to carry the congestion information and, therefore, to warn the origin hosts about the network status, a more effective four-level scheme of corrective actions can be carried out at the source nodes. The possible values of the bits MBin and MBout are the ones shown in Table 4.1.

The advantage of applying the IOPM strategy relies on its capacity to detect in advance and to discriminate flows responsible for congestion from those other that

[1]This occurs due to the speedup provided by switches. Notice that, in order to prevent switches from creating a bottleneck, the internal bandwidth of the switch crossbar is usually higher than the channel bandwidth.

Ack bits		
MBin	**MBout**	**Meaning**
0	**0**	No Actions
0	**1**	Marked at output buffer
1	**0**	Marked at input buffer
1	**1**	Marked at both

Table 4.1: Possible bit combinations when applying the IOPM strategy

simply are affected by it, and identify different levels of contribution to congestion by classifying the network flows in three different types of flow:

- *Hot-Flows*: flows truly responsible for congestion.
- *Cold-Flows*: flows non-responsible for congestion.
- *Warm-Flows*: flows that were cold-flows at the beginning, but as long as congestion has spread along the network, they are becoming *hot-flows*, but at the moment contributing with a moderate degree to congestion.

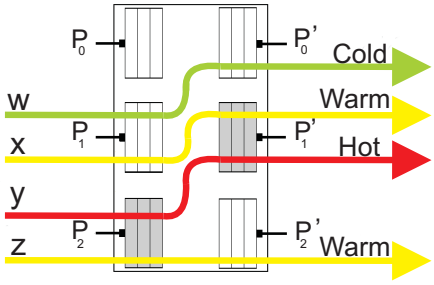


Figure 4.6: Flow classification when applying the IOPM strategy.

Figure 4.6 shows a possible situation where these three types of flows can be identified when applying the IOPM strategy.

1. Flow *w* is crossing the switch without contention at any buffer. Therefore, no marking actions will be carried out on its packets. So, this flow will be classified as a *cold-flow*.

2. Flow *x* is identified as a *warm-flow* as it is crossing a congested output buffer but not a congested input one.

   This flow is also contributing to the traffic at such output link. In order to reduce the injection rate of the flow *x* and avoid to accumulate packets at the input buffer, which would affect the flow *w*, those packets belonging to flow *x* will be marked at output buffer setting the MBout bit and their flow will be classified as a *warm-flow*.

3. Flow *y* is crossing both input and output congested buffers. Therefore, both MBin and MBout bits will be set as indicative that this flow is a truly responsible flow for congestion and then it is classified as a *hot-flow*.

4. Flow *z* is crossing the congested input buffer but it is not traversing any congested output buffer. This situation indicates that congestion has spread into the input buffer and packets belonging to flow *z* are suffering head-of-line blocking at input buffer due to the high injection rate of the *hot-flow y*. Although the flow *z* is not directly responsible for the initial congestion at the output buffer, this flow will be classified as a *warm-flow* because it is contributing in more extent to spread the congestion tree to the previous switch. So, packets will set only the MBin bit.

As a result of this flow classification, source hosts can apply different levels of corrective actions over their respective flows. Notice that the most important key in this new packet marking mechanism is to correctly detect congestion at the switch where the saturation starts by a correct classification of the flows, in order both to prevent the spread of congestion along the previous switches, which would create a new root of congestion, and to avoid applying premature corrective actions that could penalize performance. These advantages are not found in any previous proposal.

Again, a comparative analysis of the percentage of the marked packets (*cold* and hot-packets), will venture the performance of the new packet marking strategy with respect to the previous analyzed strategies. Figure 4.7 shows the some values previously presented in Figure 4.3 plus the ones achieved by the new packet marking

strategy.

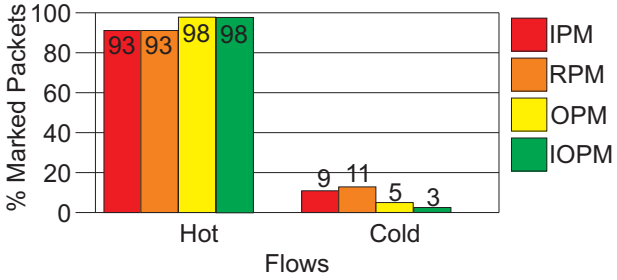


Figure 4.7: Percentage of marked packets for the IPM, RPM, OPM, and IOPM strategies.

As can be observed, the IOPM strategy achieves a value of 98% of marking hot-packets, which matches the value reached by the OPM strategy. However, the percentage value for marking cold-packets is reduced to a 3%. This redution is relevant when compared to the rest of proposals. This confirms that by using a packet marking strategy based on two marking bits, a better response can be obtained.

Notice that, in interconnection networks with no switch speedup, input buffers will fill before output buffers do and the IOPM strategy will be able to detect congestion sooner than OPM does.

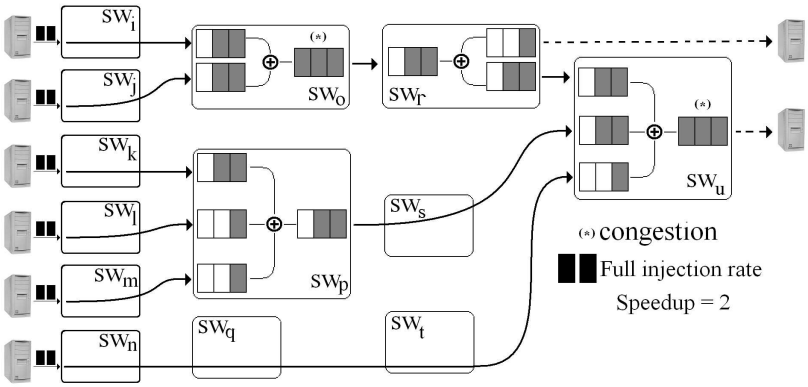


Figure 4.8: Early detection of the congestion roots.

As it can be seen in Figure 4.8, two potential situations of congestion start. One

occurs at the switch SW_o where both incoming flows converge at the output buffer and are later split in switch SW_r. The second one, at switch SW_u, which is connected to a destination host. As congestion trees usually grow from the root toward the leaves (source hosts) [42], a packet marking strategy such as IOPM, which is able to set the MBin and MBout bits depending on the input and output buffers, respectively, not only allow us to detect congestion sooner than the previous strategies at the last stage, but also at the intermediate stages before the hot-packets reach the last stages, whatever the speedup value is defined at the switches.

4.2.5 The Marking and Validation Packet Marking Strategy

As it has been seen, the Input-Output Packet Marking Strategy presents a better congestion detection mechanism with regard the previous strategies. However, it still penalizes a significant percentage of cold-packets (3%) which would suffer the application of corrective actions, which in turn could affect the network throughput.

In order to remove this drawback, a new packet marking strategy, referred to as Marking and Validation Packet Marking (MVPM), is proposed and analyzed. This packet marking strategy combines a packet marking at input and output buffers, in such a way that packets are marked at input buffers and validated at output buffers under certain conditions. To this end, switches also have to have buffers at both input and output channels to establish thresholds. Moreover, two bits are dedicated in the packet header, the Marking Bit (MB) and the Validation Bit (VB). To implement the mechanism in a standard interconnect, we can use any of the header bits usually reserved by the specs for vendor applications. The MVPM strategy works as follows:

1. Packets arriving at an input buffer are marked if the number of stored packets in the buffer exceeds a predefined threshold. This is performed by activating the Marking Bit (MB=1) in the packet header. Notice that only newly incoming packets to the input buffer are marked rather than marking those packets which are already stored at the congested buffer.

2. When a marked packet (MB=1) is forwarded through a saturated output link, even in a different switch, we proceed to validate it by activating a second bit

in the packet header, the Validation Bit (VB=1). We assume that an output link is saturated when the number of packets stored in its buffer exceeds the predefined threshold. Again, only new incoming packets to the output buffer are validated rather than marking the ones stored at the congested output buffer.

Notice that both marking and validation actions can be carried out several times on the same packet during its journey from its origin to the destination, but, on the contrary, it will never be unmarked. Moreover, a packet cannot be validated (VB=1) if it has not been previously marked (MB=0). As a result, the possible values of the bits MB and VB are the ones shown in Table 4.2. Note that the combination MB=0 and VB=1 is not possible in this packet marking strategy.

MB	VB	Meaning
0	**0**	No Actions
0	**1**	Not Possible
1	**0**	Marked
1	**1**	Marked&Validated

Table 4.2: Possible bit combinations when applying the MVPM strategy.

As an example, following the congestion process shown in Figure 3.7 (c), and defining thresholds in both input and output buffers, packets crossing the switch *C* (in Figure 3.7 (c)) would get the values for MB and VB as shown in Figure 4.9.

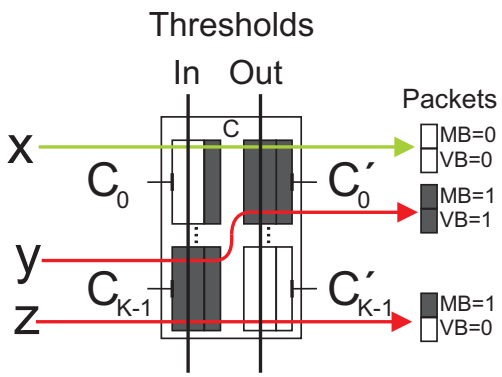


Figure 4.9: Examples of marking packets by the MVPM strategy.

The advantage of applying the MVPM strategy with respect the IOPM strategy consists in its capacity to correctly discriminate flows responsible for congestion from those other that simply are affected by it. Using the same types of flows (*Hot-Flows*, *Cold-Flows*, and *Warm-Flows*) described in the previous strategy, next we show how the MVPM identifies and classifies the different types of flows.

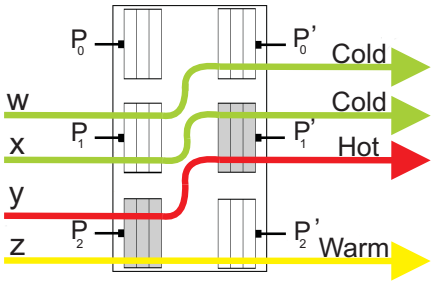


Figure 4.10: Flow classification when applying the MVPM strategy.

Figure 4.10 shows all the possible situations where these three types of flows can be identified when applying the MVPM strategy.

1. Flow *w* is crossing the switch without contention at any buffer. Therefore, no marking actions will be carried out on its packets. So, this flow will be classified as a *cold-flow*.

2. Flow *x* is also identified as a *cold-flow* as it is crossing a congested output buffer but not a congested input one. So, although this flow is also contributing to the traffic at such output link, congestion has not yet reached the input buffer, and then it is not affecting other flows that could share the input buffer such as the flow *w*. This situation can occur when the total injection rate *x*+*y* surpasses the accepted traffic at the output link P_1', but the injection rate *x* is not high enough to accumulate packets at the input buffer, thus affecting the flow *w*. Therefore, those packets belonging to flow *x* will not be marked, avoiding premature corrective actions that are not yet necessary. Notice that this behavior avoids unnecessarily penalizing the flow that is sending only a few packets to the congested link.

3. Flow *y* is crossing both input and output congested buffers. Therefore, both MV and VB bits will be set as indicative that this flow is a *hot-flow*.

4. Flow *z* is crossing the congested input buffer but it is not traversing any congested output buffer. This situation indicates that congestion has spread into the input buffer and packets belonging to flow *z* are suffering head-of-line blocking at input buffer due to the high injection rate of the *hot-flow y*. Although the flow *z* is not directly responsible for the initial congestion at the output buffer, this flow will be classified as a *warm-flow* because it is contributing to more extent to spread the congestion tree to the previous switch. So, packets will set only the MB bit. This is achieved by avoiding the labeling of certain flows to be classified as cold-flows by not contributing to the spread of congestion.

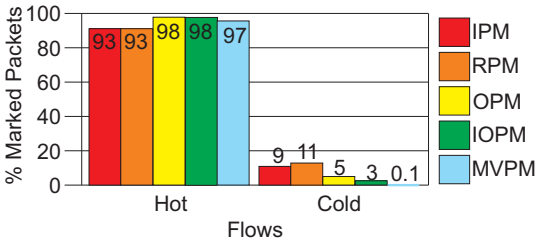


Figure 4.11: Percentage of marked packets for the IPM, RPM, OPM, IOPM, and MVPM strategies.

Again, an analysis of the percentage of cold and hot-packets marked by the MVPM strategy is necessary to evaluate the performance of this packet marking strategy. Figure 4.11 shows the values previously presented in Figure 4.7 plus the ones achieved by the new strategy.

As can be seen, MVPM identifies the 97% of hot-packets. Although this value is slightly lower than the OPM and the IOPM (because the MVPM strategy defines a dependency between both marking and validation actions), MVPM provides the minimum value of cold-packets wrongly marked, that is, about 0.1%. This low value is due to the application of a marking strategy based on a dependency between the two marking actions, which is able to correctly identify the responsible flows for conges-

tion. In particular, the cold packets wrongly marked as a hot packets are those that have been marked, firstly, as warm packets at an input buffer of a switch and later they are validated when crossing a temporaly congested output buffer of other different switch, which is congested just for a while, but it is not enough to accumulate packets at the input buffers of the same switch as an indication of a real congestion in this part of the network. So, although the network is getting congested in different switches at the same time, those packets still belong to a cold-flows. This occurs only during the onset of congestion at different points in the network.

Notice that we only represent *cold* and *hot-flows* but there are also warm-flows. Therefore, if we also paid attention to those flows, the percentage of marked packets for *hot-flows* (that is, hot+warm) will reach a value near 99%. Additionally, notice that none packet marking strategy will reach the value of 100% of marking hot-packets because any marking mechanism based on a recovery strategy needs time to reach the established threshold before marking packets.

4.2.6 Comparing the different packet marking strategies

Finally, in order to check the packet marking actions taken by the different strategies during all the simulation time, Figure 4.12 shows the packet marking performance of the described strategies, that is, IPM, RPM, OPM, IOPM, and MVPM, when a congestion situation is created in a network. The top graph shows the performance of a *Bidirectional MIN Perfect-Shuffle 4-ary 5-fly* network configuration when applying a synthetic traffic with a medium rate and a packet size of 256 bytes. It represents the evolution of latency when a congestion situation appears and no congestion management mechanism is applied. The network configurations, traffic patterns, and injection rates used in all simulations in this thesis are defined in the Chapter 7 (Evaluation Models). Graphs shown in Figure 4.12 represent the packet marking actions carried out by the different packet marking strategies during the simulation. As it can be seen, the IPM and RPM packet marking strategies achieve similar performance. Although RPM is a variant of the IPM strategy and it has a different procedure to mark packets, it does not significantly differs from the IPM strategy.

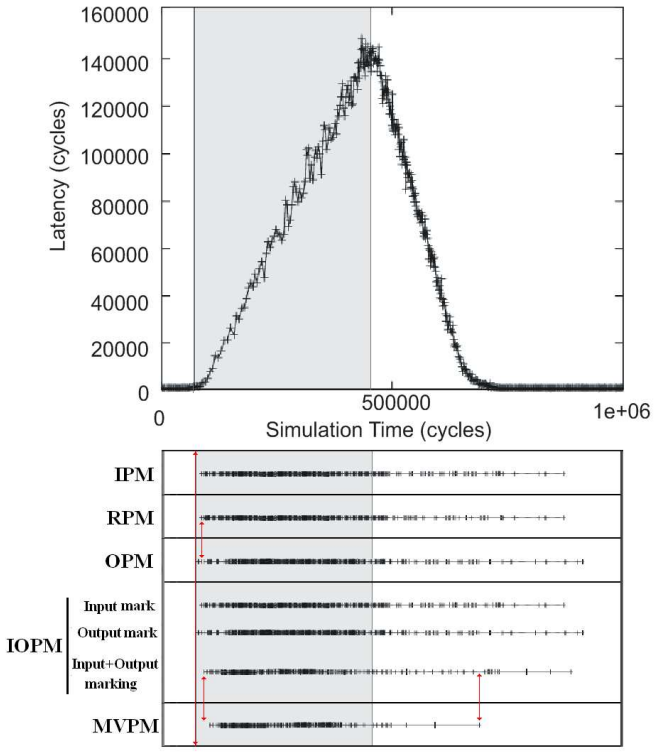


Figure 4.12: Comparing the packet marking actions throughout the simulation period.

Secondly, the OPM strategy detects congestion and, therefore, starts marking packets in transit sooner than IPM and RPM do. Thirdly, when considering packet marking at input and output buffers separately, the IOPM strategy presents similar results to that achieved by the IPM and OPM strategies, respectively. Finally, the MVPM strategy delays the validation actions with respect OPM, but concentrates most of the packet marking actions during the period where congestion arises (shown as the shaded area). Additionally, if we take into account the global results for IOPM, that is *Input+Output marking*, we detect that IOPM does more marking actions than the MVPM strategy. Although IOPM starts marking packets in advance, the marking actions are long-lasting, much more than with the MVPM strategy. This is because, as IOPM does not have any dependency between input and output marking actions, both actions can be carried out in any order and in different switches along the flow

path. This situation causes negative effects in performance because some packets are marked by chance in different switches along the path to reach their destination, and as a consequence, origins will applied corrective actions although they are not needed because there is not a real congestion. This fact can be observed comparing graphs IOPM and MVPM in Figure 4.12.

4.3 Parameters initialization: Buffer Threshold

The correct initialization of the parameters in a packet marking strategy is key to detect and mark packets in transit which are responsible for congestion. In particular, the parameter to initialize is the *Buffer Threshold*, which defines the limit beyond which the packets in transit have to be marked. Notice that a wrong value of threshold could either wrongly mark packets even when no congestion arises, or not detect congestion although a congestion situation is starting.

The threshold values are critical because they define the limit at which incoming packets must be marked. This implies that a high threshold value will produce a low rate of packet marking and therefore a possible congestion situation into the network. On the contrary, a small value of threshold could mark a large number of packets and, therefore, the application of corrective actions might reduce the packet injection rate in excess and consequently would result in a loss of network performance.

The initialization of the buffer threshold for the analyzed strategies is different in each one. That is, while RPM marks packets when the input buffers are full (threshold value=buffer size), the InfiniBand specs leaves the configuration to the vendor's criteria. Based on these specs, and on their own evaluations, the OPM strategy applied by Pfister defines a threshold value at the output buffer about 90% of the input buffer size.

On the other hand, for the IOPM and MVPM strategies, we define that the input and output buffer thresholds should be obtained under a traffic condition in which traffic is uniformly distributed to all destination as the contrary to the existence of hot-spot situations. These threshold values should be chosen in such a way that packets are allowed to cross the switches without being marked, regardless of the network

load. Although the network traffic is close to saturation point, if destination distribution is uniform, the network itself will limit the entrance of the packets at origin hosts. To define the buffer thresholds, we have calculated the average buffer occupancy for a uniform traffic distribution with an injection rate near the saturation point. The analysis was carried out with different packet sizes and network configurations.

The occupation at both input and output buffers was traced separately. In particular, we have obtained an occupation, on average, of about 66% and 33% of the input and output buffer capacity, respectively, for all the packet sizes and network configurations analyzed, as it is described in detail in section 8.2.2. Therefore, we have assumed threshold values of 66% and 33% of the input and output buffer sizes, respectively, for the IOPM and MVPM strategies

Chapter 5

CONGESTION CORRECTION TECHNIQUES

This chapter describes and analyzes in depth the different congestion correction techniques used together with their main definition parameters in order to ascertain the pros and cons of each mechanism. In addition, the chapter describes the parameter initialization strategies proposed in this thesis.

5.1 Introduction

The congestion correction techniques will be applied when origin hosts receive an ACK packet warning about a congestion situation. If this situation occurs, congestion management mechanisms generally apply a combination of several basic techniques to evenly reduce congestion. One of the most effective procedures to avoid congestion consists in reducing the injection rate of the responsible flows.

Among the commonly applied techniques, we can refer to Window-Based techniques and Waiting Interval Insertion techniques. Additionally, when congestion vanishes, two basic techniques can be applied, progressive or immediate injection recovery.

5.2 The Congestion Correction Techniques

5.2.1 Window-Based technique

Basically, this technique defines a window size to limit the maximum number of outstanding packets per flow. The value of the window size depends on vendor criteria. Notice that the outstanding packets are those sent packets that have not been acknowledged yet.

The window-based technique can be based either on a Static Window (SW) or a Dynamic Window (DW). In particular, this strategy is mainly used by TCP [3], which applies a dynamic window, but other techniques use a static one.

If the window-based technique is based on a static window, the window size value will be kept fixed during all the time. To this end, the congestion management mechanism just controls the maximum number of outstanding packets per flow. Therefore, applying a window-based strategy based on a SW seems appropriate to palliate the congestion and simple to implement. However, the chosen value for the window size may not be the most appropriate value for any traffic condition in the network. Moreover, initializing the window with a value of one, as Renato's proposal does, may negatively impact over the network throughput. Notice that, this would be the case, for example, when only one host sends packets towards a single destination host.

In order to validate by simulation that applying a static window technique is not a good solution to manage congestion, next we show network performance results when applying a static window technique to palliate the effects of a simulated congestion situation. No other corrective strategy has been applied, only a fixed window size with different values during all the simulation.

Figure 5.1 shows the influence of the window size for a *Bidirectional Perfect-Shuffle MIN 4-ary 5-fly* network configuration when applying a synthetic traffic with a high injection rate and a packet size of 256 bytes[1]. In particular, Figure 5.1 shows latency versus simulation time for a fixed-size window with the values SW=1, 2, and

[1]Notice that the network configurations, traffic patterns and injection rates used in all simulations in this thesis are defined in Chapter 7

4. The shaded area indicates the congestion period.

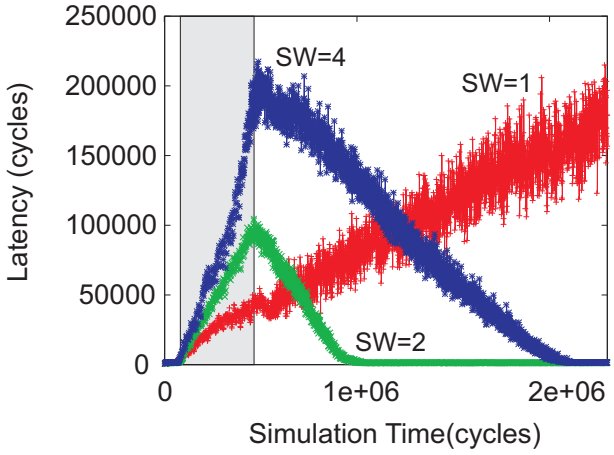


Figure 5.1: Latency for cold-flows with different SW sizes in a Bidirectional MIN Perfect-Shuffle 4-ary 5-fly with a high injection rate and a fixed packet size=256 bytes.

We can observe that, if the window size is equal to 1, the graph SW=1 presents the lowest values for latency during the congestion period. However, once the congestion situation decreases, the network latency would never recover its (low values) because a window size of 1 does not allow all accumulated packets at origin hosts during the congestion period to be consumed. As we can see, a window size of 2 (SW=2) provides the best performance in this case. Notice that, when the network is not saturated, a larger window size does not have any effect since the ACKs arrive at the origin before more than 2 packets can be injected into the network. However, as the network becomes saturated, packet contention arises and packets stay longer in the switch queues, increasing the Round-Trip Time (RTT). In this situation, a larger window size allows more packets to be in transit through the network, thus leading to an even more congested situation. In Figure 5.1, the graph with the window size equal to 4 (SW=4) confirms this behavior.

As a result of that, we can assure that applying a static window with a fixed value, in any situation, is not a good solution to palliate the congestion effects. Moreover,

applying a SW with a window size of one, as Renato's proposal does, the network performance can suffer a negative impact in latency depending on network configuration and injected traffic.

However, if a dynamic window is applied, the window size value can vary depending on network behavior. Initially, the window size will start with the maximum value defined for the target network configuration. When congestion appears, the window size will be progressively reduced until the minimum value of one. Later on, when congestion vanishes, the window size will gradually recover its initial value. Therefore, by applying a dynamic window strategy, the window size will vary between the values 1 and the maximum window size defined at configuration time.

5.2.2 The Waiting Interval Insertion Technique

The Waiting Interval Insertion technique allows to reduce the injection rate in a progressive way by injecting Waiting Slots (WS) between two consecutive packets. The size of a waiting slot depends on the vendor criteria, and it is defined at the network configuration time. Depending on the severity of congestion, the elapsed time between the injection of two consecutive packets will be increased or decreased. As long as the congestion ceases, the waiting interval between packet injection will be decreased until disappearing.

Basically, the technique works as follows. When an origin host receives a marked ACK packet, it waits during a waiting slot before injecting a new packet into the network. If more marked ACK packets are received, then more waiting slots will be inserted, thus enlarging the waiting interval. It should be noted that the injection of new packets is forbidden along the waiting interval. Later, when unmarked ACK packets are received, the number of waiting slots between packet injections will be decreased following the defined recovering method.

5.3 Parameters Initialization

The correct initialization of the parameters in a congestion correction technique is key to recover from a congestion situation and, therefore, to achieve the best network performance.

The set of parameters to initialize includes: the *Window Size*, which limits the maximum number of outstanding packets per flow and the *Waiting Interval Calculation*, which defines the duration of the waiting interval and the method used to calculate it.

The Renato's mechanism initializes the window size with a fixed value of one for all network configurations and traffic load and defines its waiting interval as the sum of waiting slots in which each waiting slot corresponds to the packet transmission time. On the other hand, the Pfister's implementation defines a waiting slot based on the hot-spot degree which indicates the number of sources contributing to the hot-spot traffic.

Notice that both the definition of the window size in the Renato's proposal, and the initialization of the waiting slots in the Pfister's implementation are not a generalized procedure that can be applied to any network configuration or traffic load. On one hand, Renato applies a fixed value for the window size regardless of the network configurations, and, on the other hand Pfister, defines a waiting slot based on the hot-spot degree what is not known in a real situation.

On the contrary, due to the fact that in our proposals the initial values of these parameters depend on the network configuration, their adjustment is easy and simple. In what follows, we explain the strategy followed to correctly define the initial values for these parameters in the congestion management mechanisms proposed in this thesis.

5.3.1 The Window Size

A fixed value of one for the window size seems appropriate to palliate the congestion in a general scenario, but it may negatively impact network throughput. Moreover, if a severe congestion appears and packets are stopped at origin hosts, the network

would not be able to consume all the waiting packets at origins when congestion disappear due to the fact that the static window may have fixed a value smaller than the optimum, which depends on the network configuration and maximizes the throughput network. Therefore, it is necessary to identify the optimum value for the window, referred to as ω, regardless of what window-based technique is applied: static or dynamic window.

In order to maximize network throughput, the window size can be calculated as the number of packets from a flow that can be injected into the network until receiving the first ACK at the origin host in the absence of contention in the network. So, it can be established according to the minimum RTT of the packets, which depends on the network depth or number of stages. Hence, the minimum RTT can be calculated as the elapsed time between the sending of a data packet and the reception of its corresponding ACK packet at the origin host. Hence, the minimum RTT comprises the time (t_{data}) required by the data packet to reach and be delivered at the destination and the time (t_{ack}) required by the ACK packet to reach and be delivered at the source host:

$$RTT_{min} = t_{data} + t_{ack} \tag{5.1}$$

Let us assume a simple approach to calculate the network base latency in networks under *cut-through switching* as follows:

$$t_{vct} = h * t_{hop} + (\frac{L}{B}) \tag{5.2}$$

where h defines the number of network hops, t_{hop} includes the $t_{routing}$ (time needed to take a decision by the routing unit and to configure the switch), the t_{switch} (time needed to cross the switch), and t_{link} (time needed to cross a link), L is the packet length, and B defines the channel bandwidth. Thus, the t_{data} and t_{ack} can be defined as:

$$t_{data} = h * t_{hop} + (\frac{L_{data}}{B})$$
$$t_{ack} = h * t_{hop} + (\frac{L_{ack}}{B})$$

where L_{data} and L_{ack} define the data and ACK packet lengths, respectively. Therefore, the minimum RTT can be calculated as:

$$RTT_{min} = 2 * h * t_{hop} + (\frac{L_{data}+L_{ack}}{B})$$

The optimum value ω will be imposed by the maximum number of packets that can be sent by the source during the time interval defined by the RTT_{min}. As each packet needs $\frac{L_{data}}{B}$ to be injected, ω is given by:

$$\omega = \frac{RTT_{min}}{\frac{L_{data}}{B}} \tag{5.3}$$

Replacing RTT_{min} by its calculated value, ω can be obtained as:

$$\omega = \frac{2 * h * t_{hop} * B + L_{data} + L_{ack}}{L_{data}} \tag{5.4}$$

Analyzing the Formula (5.4), it can be observed that the value of the expression: $(2 * h * t_{hop} * B + L_{ack})$ results in a constant K, only depending on the network configuration. As a result, the value of ω will be bounded by the maximum and minimum allowed data packet sizes:

$$\frac{K + Min(L_{data})}{Min(L_{data})} \geq \omega \geq \frac{K + Max(L_{data})}{Max(L_{data})} \tag{5.5}$$

where $K = (2 * h * t_{hop} * B + L_{ack})$ and $L_{data} = L_{head}{}^{2} + L_{payload}$

[2]Control bytes in the header of all data packets

Next, in order to calculate as an example the optimum value (ω) of the window size, let us assume a 4-ary 5-fly (512 hosts and 640 switches) bidirectional network topology where the number of stages is 5, imposing a h value equal to 9. Also, let us assume the following fixed values: $t_{hop} = 3$ cycles, $B = 1$ byte/cycle, $L_{ack} = 22$ bytes, and $L_{head} = 22$ bytes. According to the Formula (5.5) ω will vary between two limits depending on $L_{payload}$. That is, if $L_{payload}$ tends toward ∞, then $\omega \approx 1$, but if $L_{payload}$ tends toward zero, then $\omega \approx \frac{98}{22} = 4,45$.

So, the optimal ω value will be bounded by the interval [1 , 4.45]. Now, assuming in particular a fixed data payload equal to $256\ bytes$, we conclude that the optimum ω value will be:

$$\omega = \frac{K + (L_{head} + L_{payload})}{(L_{head} + L_{payload})} = \frac{76 + (22 + 256)}{22 + 256} = \frac{354}{278} \approx 1.28 \qquad (5.6)$$

The calculated value *1.28* in Formula (5.6) means that if the window size is initialized with a value equal to 1, as Renato's proposal would do, the congestion management mechanism would restrict too much the packet injection at origins, while the network is able to manage a larger volume of packets, at least a 28% more. Therefore, limiting the window size to one penalizes network throughput. However, this value also points that by choosing a fixed value of window size equal to 2 would causes congestion because origins hosts are allowed to inject more packets (72%) than the network can manage. So, the window-based strategy based on a dynamic window, which depends on the network configuration and traffic features, is presented as the best solution because it allows us to achieve, on average, the optimum value for the window size.

5.3.2 The Waiting Interval Calculation

A wrong selection of both the duration of a waiting slot and the total number of waiting slots (*number_WS*) inserted between consecutive packets could cause undesirable effects, such as oscillations, unstable performance, etc.

The previous proposals define both parameters depending on either the Hot-Spot Degree (*HSD*), that is, the number of sources that contribute to the hot-spot, as the Pfister's implementation does, or a constant value that is used to regulate the injection rate, as Renato's proposal does. However, in both cases, parameters initialization is not a generic procedure, because it depends either on the defined traffic pattern, that is the number of hosts injecting hot-spot traffic, or is a constant value, which could not be useful under other network configurations or traffic loads. So, in order to define a general strategy able to be applied in every network and traffic conditions, it is recommended that both parameters, the number of waiting slots and their duration, are based on the network configuration parameters.

For our case study, the *k-ary n-fly* multistage interconnection network, the parameters *k* and *n* define the network structure, namely the size (radix) and number of stages (network depth) through which switches are arranged. Therefore, it is interesting to find a relationship between these parameters and those other that define the waiting interval calculation.

In particular, the switch radix value *(k)* could be used to calculate the *number_WS* value because it defines the maximum number of hosts connected to a switch. So, in a congestion situation, this value can be used to evenly reduce the injection rate in a fair proportion. Additionally, the duration of the waiting slots could be based on the RTT_{min} as it is the minimum time needed to send a packet plus to receive its ACK.

As a result, the waiting interval is calculated as follows. In the absence of congestion, the $number\_WS$ value should be equal to 0. When the first ACK packet notifying congestion arrives, the $number\_WS$ is set to the value of one. If more marked ACK packets are received, then the $number\_WS$ value will be increased by a constant factor equal to the switch radix (*k*). In particular, the current number of waiting slots will be multiplied by the *k* value every time a new marked ACK packet is received, in order to obtain the new number of waiting slots ($number\_WS_{new}$), as shown in Formula (5.7). As a consequence, the waiting interval is calculated by (5.8).

$$number\_WS_{new} := k * number\_WS_{current} \tag{5.7}$$

$$WaitingInterval := number\_WS_{current} * RTT \tag{5.8}$$

Combining both Formulas (5.7) and (5.8) with the reception of some marked packets produces the results shown in Table 5.1.

Number of marked ACK packets received after Dynamic Window=1	**number_WS**	**WaitingInterval**
0	0	0
1	k^0	$k^0 * RTT$
2	k^1	$k^1 * RTT$
3	k^2	$k^2 * RTT$
...	...	...

Table 5.1: Injection rate reduction procedure for the k-ary n-fly network

On the other hand, the maximum number of waiting slots inserted between consecutive packets should be bounded in order to prevent packets from being stopped excessively at origin hosts while network throughput is being penalized. To this end, the network parameter *n*, that identifies the number of stages in the network, could be used to limit the maximum number of times the *number_WS* value can be increased.

To justify the use of both *k* and *n* as parameters on the waiting interval calculation procedure, let us assume a generic interconnection network *k-ary n-fly* based on only one switch with *k* ports as the one shown in Figure 5.2 (*k-ary 1-fly*), allowing a total k hosts to be interconnected.

Figure 5.2: Generic interconnection network, k-ary 1-fly.

We define h_i as the host i connected to the network, being the radix value k the total number of hosts injecting packets into the network. Let us divide the set K of hosts into two groups. The set A represents the group of hosts injecting 100% uniform traffic, and the set B represents the group of hosts injecting 100% hot-spot traffic to a single destination host. So, the set K can be defined as $K = A \cup B$, where $A \cap B = \phi$. If k, a, and b represent the cardinal of the sets K, A, and B, respectively, we can state that $k = a + b$.

Assuming that the network is working at the maximum injection rate, let us consider that each host is able to inject and receive at a maximum rate of x bytes/cycle. As hosts belonging to the set A inject traffic into the network evenly distributed between all the destinations, the traffic destined to a specific destination host h_i from a host belonging to the set A will have a maximum rate of:

$$r_{h_i}^{A} = \frac{x}{k}\ bytes/cycle, where; 0 \leq i \leq k-1 \tag{5.9}$$

On the other hand, hosts belonging to the set B inject traffic only to the *hot-spot* with the maximum rate of:

$$r_{HS}^{B} = x \text{ bytes/cycle.}$$

Consequently, total injection rate toward the *hot-spot*, r_{HS}^{K}, can be calculated as:

$$r_{HS}^{K} = a * r_{HS}^{A} + b * r_{HS}^{B} = (k-b) * \tfrac{x}{k} + b * x =$$

$$= (1 + b - \frac{b}{k}) * x \; bytes/cycle \tag{5.10}$$

Notice that, if the number of hosts injecting towards the *hot-spot* is greater than zero ($b > 0$), the injected traffic rate is greater than the maximum traffic rate that can be accepted by the *hot-spot*, that is, x bytes/cycle. Therefore, all the injected traffic to the *hot-spot* cannot be accepted and, as a consequence, a congestion process starts. To solve that situation, it is necessary to apply a proportional reduction of the injection rate from every host injecting traffic to the congested area in such a way that the effective traffic arrival rate towards the *hot-spot* does not exceed the value of x bytes/cycle. The reduction function f can be derived from:

$$r_{HS}^{K} * f(k, b, x) = x \text{ bytes/cycle}$$

where r_{HS}^{K} indicates the injected traffic towards the *hot-spot* and $f(k, b, x)$ is the reduction function that has to be applied to evenly reduce the injection rate. Therefore,

$$f(k, b, x) = \frac{x}{(1 + b - \frac{b}{k}) * x} = \frac{k}{b * (k - 1) + k} \tag{5.11}$$

Taking into account the range of b values ($0 \leq b \leq k$), the expression (5.11) can have the following border values: $\left[1, \frac{1}{k}\right]$. This result means that the reduction degree in the injected traffic depends on the number of hosts b injecting toward the *hot-spot*. Although this parameter cannot be predicted, this function shows the maximum reduction function that would be needed in the worst case (all the origin hosts, *k*, injecting traffic to a single destination) in order to remove the congestion situation, while still keeping the network busy.

On the other hand, real MINs are built with more than one stage due to the limited number of ports of current switches. This way, for the generic case of *k-ary n-fly*,

made up of *n*-stages of switches with a *k*-radix, a total number of k^n hosts can be interconnected. Therefore, its injection rate reduction function $f(k^n, b, x)$ should have the following border values: $\left[1, \frac{1}{k^n}\right]$.

As the congestion root can be placed at any stage of the network and the parameter b cannot be predicted, a fair reduction on the injection rate will require a reduction factor of $(\frac{1}{k})$ to be applied each time a validated ACK packet is received.

Based on an exhaustive study, we can affirm that by applying this reduction function $(\frac{1}{k})$ n-times at most, regardless of the b value (number of hosts injecting toward the hot-spot), it is enough to stop the worst congestion situation in time, while still keeping the links busy. Notice that the worst case would be when the number of hosts b injecting traffic to the hot-spot is k^n. So, applying the reduction function $(\frac{1}{k})$ more than *n* times is unnecessary and will provoke a decrease in network performance, as it is shown in the Performance Evaluation Section.

We have shown how the parameters k and n, established according to the network configuration, are used to efficiently calculate the waiting interval. As it can be seen, the mechanism is simple, easy to implement, and only depends on the network configuration.

Chapter 6

PROPOSED CONGESTION MANAGEMENT MECHANISMS

This chapter describes new congestion management mechanisms that combine more refined congestion detection techniques, able to discriminate flows responsible for congestion from those other that simply are affected by it, with the application of an efficient set of corrective actions aimed to limit the injection rate of those flows involved in the congestion tree according to their contribution degree to the congestion.

6.1 Introduction

As a result of the analysis carried out in previous chapters about the different congestion detection and congestion correction techniques, next we propose a set of combinations of the analyzed strategies that will be evaluated later in the Performance Evaluation Chapter.

The new proposals for congestion management consist of end-to-end congestion management mechanisms based on the use of explicit congestion notification. Unlike other approaches [76], [82], these new mechanisms are not targeted for a particular network technology, but for MINs in general. However, the proposed mechanisms could easily be applied to current standard interconnects, such as InfiniBand network.

In particular, two mechanisms will be described.

The first mechanism, known as Marking and Validation Congestion Management (MVCM) [36], [39], is based on applying the MVPM strategy to classify the packet flows and, therefore, to mark those packets responsible for congestion.

The second mechanism, know as Input-Output Congestion Management (IOCM) [38], is based on applying the IOPM strategy to detect congestion and to mark packets responsible for congestion.

In what follows, we describe both congestion management mechanisms, paying special attention to the set of corrective actions applied as well as the procedure to combine these corrective actions with the different packet marking strategies.

6.2 The Marking and Validation Congestion Management Mechanism (MVCM)

The main goal of this new congestion management mechanism is to properly identify the flows responsible for congestion, in order to apply packet injection limitation only at the source nodes that are actually causing congestion. Furthermore, packet injection limitation is applied with the proper intensity in accordance with the detected congestion degree in the network, thus minimizing the negative effects over the flows non-responsible for congestion. As a consequence, the available network resources are evenly distributed among the devices that demand them, improving network throughput.

6.2.1 Congestion Detection Technique

The congestion detection technique applied by the MVCM mechanism is based on the MVPM strategy. Remind that this strategy warns about congestion using two bits of the packet header: Marking Bit (MB) and Validation Bit (VB). MB is set when the number of stored packets in the input buffer of the switch surpasses a threshold. After setting MB, VB is also set if the packet arrives to an output buffer occupied more than a threshold. Although this strategy, as it has been previously shown in

Figures 4.11 and 4.12, is the slowest to start marking packets, due to the dependency between the marking and validation actions. Notice that by using two bits for packet marking, the analyzed results show that it accurately classifies the different types of flows as hot, warm, and cold.

6.2.2 Congestion Correction Technique

Following the classification shown in Table 4.2, next we present in Table 6.1 the actions assigned to each possible combination of bits. As it can be seen, *Moderate* actions will be applied over *warm-flows*, whereas *Aggressive* actions will be applied over *hot-flows*. We will refer to as *Aggressive* the actions performed on those flows whose ACK packets have both bits set. On flows where packets have only the MB bit set, we take only *Moderate* actions in order to prevent congestion expanding to the previous switch.

The following section details the processes of reducing and recovering the injection rate when a congestion situation is detected into the network.

MB	**VB**	**Meaning**	**Type of Flow**	**Action**
0	**0**	No Congestion	*Cold-Flow*	No Actions
0	**1**	Not possible		
1	**0**	Marking	*Warm-Flow*	Moderate
1	**1**	Marking&Validation	*Hot-Flow*	Aggressive

Table 6.1: Actions applied by MVCM depending on the flows classification.

One of the most effective procedures to avoid congestion into a network consists of reducing the injection rate of the responsible flows. To this end, we propose, based on the three-level packet marking scheme applied, a two level scheme of corrective actions at the origin hosts, that combines in an effective way window-based techniques with the insertion of waiting intervals.

- The first level is based on adjusting the packet injection rate by using the *Window-Based mechanism*. It is based on the idea of limiting, for each flow,

the number of outstanding packets into the network. In this case, the window mechanism is based on a dynamic window where the window size is not fixed, allowing fluctuation between a defined maximum value (DWmax) and the minimum value of one.

- If congestion persists after the dynamic window size reaches the minimum value of one, a second level of measures will start to be applied in order to reduce the injection rate even more by introducing a delay between the injection of two consecutive packets. This second level of corrective actions is based on the *Waiting Interval Insertion technique*.

In order to carry out the different corrective actions, we rely on the reception of ACK packets, which allow us to correctly identify the flows and to estimate the degree of congestion. Table 6.2 shows the corrective actions applied according to the congestion level identified by the values of the MB and VB bits received on ACK packets.

MB	**VB**	**Meaning**	**Type of Flow**	**Action**	**Strategy Applied**
0	**0**	No Congestion	*Cold-Flow*	No Actions	None
0	**1**	Not possible			
1	**0**	Marking	*Warm-Flow*	Moderate	**DW**
1	**1**	Marking&Validation	*Hot-Flow*	Aggressive	**DW+WS**

Table 6.2: Corrective actions applied by MVCM depending on the flows classification.

Next, the corrective actions to reduce the injection of new packets and to recover the initial injection rate are described in detail.

Reducing the Injection Rate

There are two types of corrective actions to apply.

1. **Moderate actions**. They are applied to *warm-flows*. In this case, only the dynamic window size is modified. The window size is initialized with the maximum value of DW_{max} and it is reduced by subtracting one per each marked (MB=1,VB=0) ACK packet received. If the window size reaches its minimum value (one) and more marked (MB=1,VB=0) ACK packets arrive, the mechanism will keep the window size equal to one and no additional actions will be taken. This situation will remain until a non-marked ACK packet arrives (MB=0,VB=0). Then, the window size will be increased by adding one per each non-marked ACK packet received until the maximum value of DW_{max} is restored. This way, the injection rate for *warm-flows* will be decreased during the strictly necessary period of time, thus stopping the spreading of congestion tree. In summary, the dynamic window size will be adjusted into the interval [1..DW_{max}].

2. **Aggressive actions**. They will be applied when validated ACK packets (MB=1, VB=1) are received at origins. That is, they correspond to packets belonging to flows truly responsible for congestion (*hot-flows*). At the beginning, the mechanism reacts by reducing the dynamic window size as moderate actions do. If the dynamic window size reduction is not enough to stop the congestion, and more validated packets (MB=1,VB=1) continue to be received, stricter actions will be applied intended to reduce the injection rate even more. This second level of actions starts when the dynamic window size becomes equal to one. Then, waiting slots will be inserted between consecutive injected packets in such a way that every received ACK packet with both MB and VB set will increase the number of waiting slots (multiplying the current value by the switch radix K). In any event, the injection rate has to be reduced until a minimum value is reached depending on the network topology. When this value is reached, the injection rate is not reduced further, regardless of continuing to receive more marked packets. Keeping this minimum injection rate is important to prevent undue message throttling. On the other hand, the reception of ACK packets with MB=0 and VB=0 will reduce the number of waiting slots as

explained below. The duration of each waiting slot is assumed as the minimum time needed to send a packet plus to receive its ACK, that is, the minimum Round-Trip Time delay (RTT). Notice that the minimum RTT is computed in the absence of network traffic when only one packet is traversing the network.

Notice that both situations, *moderate* and *aggressive* actions, can occur for a given flow at the same time. That is, packets belonging to *warm-flows* or *hot-flows* can arrive during a congestion process indistinctly. In that situation, the mechanism will combine the actions previously described.

Recovering the Injection Rate

The strategy to recover the injection rate must meet the tradeoff between achieving a fast response time when congestion has finished and avoiding injecting too much traffic when the network is still congested.

Basically, the reception of non-marked ACK packets at the origin host (MB=0, VB=0) will allow the recovery of the initial values of the parameters for the congestion control mechanism, that is, the full injection rate. The recovery period will depend on the value the dynamic window has reached and the waiting interval applied because of the waiting slot insertion. The process of recovering the full injection rate is described as follows:

1. When the first non-marked ACK packet is received, if the total waiting interval applied to that flow is greater than zero, it will be immediately eliminated, thus allowing for a fast recovery, but keeping the window size equal to one. On the other hand, if the waiting interval is equal to zero, then the recovery mechanism will act over the dynamic window, as referred below.

2. If more non-marked ACK packets arrive, the dynamic window will recover the initial value, that is, DW_{max}, by adding one for each non-marked ACK packet received. As a result, after receiving DW_{max} consecutive packets with MB=0 and VB=0, the full injection rate will be available (i.e., one packet for

removing WS and DW_{max}-1 packets for restoring the dynamic window size at its maximum value).

3. In order to speed-up even more the removal of the corrective actions when congestion is no longer detected, all parameters (Dynamic Window and Waiting Interval) will be immediately set to their initial values if an origin host injects a packet into an empty injection queue. The reason is that, if a host temporarily stops injecting packets, then it no longer contributes to congestion. Moreover, if no data packets are injected, then no ACK packets will be received. Hence, although this host is not generating congestion, it will not be able to quickly recover the initial values of the parameters, which may penalize the network throughput. Consequently, when this situation occurs, the parameters controlling the corrective actions applied for this flow will be reset by setting DW = DW_{max} and WS = 0. With this actions, we get an immediate recovery of the injection rate.

Therefore, as can be seen, the proposed mechanism takes corrective actions immediately against serious situations that could cause congestion in the network. However, if congestion takes place only during a brief period of time, recovery is also very fast. As a consequence, the mechanism should not penalize the network performance in the absence of congestion.

6.3 The Input-Output Congestion Management Mechanism

As it has been seen, the Marking and Validation Congestion Management mechanism presents a well-structured set of techniques for congestion detection, based on the MVPM strategy, and a congestion correction technique, based on a DW combined with WS. However, because the MVPM strategy presents a dependency between marking and validation actions, it may introduce a certain delay in detecting and applying corrective actions when a congestion situation arises. In order to provide an early congestion detection and analyze how it could affect the network performance, a new congestion management mechanism, referred to as Input-Output Congestion

Management (IOCM) mechanism and based on the IOPM packet marking strategy, which decouples both marking actions, is presented.

6.3.1 Congestion Detection Technique

The IOPM strategy applied, as was proposed in section 4.2.4, enables a four-level scheme of marking actions. According to it, we can define the following levels of corrective actions. Table 6.3 shows the four combinations.

MBin	MBout	Meaning	Type of Flow	Action
0	**0**	No Congestion	*Cold-Flow*	No Actions
0	**1**	Marking at output buffer	*Warm-Flow*	Moderate
1	**0**	Marking at input buffer	*Warm-Flow*	Moderate
1	**1**	Marking at both	*Hot-Flow*	Aggressive

Table 6.3: Actions applied by IOCM depending on the flows classification.

As can be observed, the bits combination, MBin=0 MBout=1, corresponds to packets which have crossed a congested output buffer but did not cross any congested input buffer. This means that the referred packets are not contributing at the moment to spread congestion. As a consequence, in order to balance the corrective actions applied, we classified these flows as *Warm-Flow* and impose moderate actions to carry out over the injection rate of these flows.

6.3.2 Congestion Correction Technique

The IOCM mechanism will basically maintain the same scheme of corrective actions applied by MVCM, but it will be extended according to the new capabilities offered by the applied packet marking strategy.

Table 6.4 presents the corrective actions applied by IOCM depending on the flows classification.

As it is shown, if an origin host receives an ACK packet marked only either at input (MBin=1 and MBout=0) or output (MBin=0 and MBout=1) buffers, corrective

MBin	MBout	Meaning	Type of Flow	Action	Strategy
0	**0**	No Congestion	*Cold-Flow*	No Actions	None
0	**1**	Marked at out-buffer	*Warm-Flow*	Moderate	**DW**
1	**0**	Marked at in-buffer	*Warm-Flow*	Moderate	**DW**
1	**1**	Marked at both	*Hot-Flow*	Aggressive	**DW+WS**

Table 6.4: Corrective actions applied by IOCM depending on the flows classification.

actions will be applied only over dynamic window in order to stop the onset of congestion. But, if a marked ACK packet is received with the value of MBin=MBout=1, then aggressive actions will be applied, that is, dynamic window plus waiting slots, in the same way as they were applied in MVCM.

Referring back to the example given in Figure 4.5, according to the IOPM strategy, both hosts (*x* and *y*) would receive the ACK packet with MBin=0,MBout=1 and MBin=1,MBout=1, respectively, which means that both hosts have to reduce their injection rate. Host *x* will reduce the size of the dynamic window, whereas host *y* will act over dynamic window and later increase the waiting slots. Notice that, although host *x* will apply corrective actions, it will not actually affect its injection traffic rate if this rate is not enough at the moment to contribute to spread the congestion. In contrast, by applying the MVCM mechanism, corrective actions on flow *y* will be only applied after surpassing threshold at the input buffer and being validated at the output buffer.

The early detection of that situation allows us to anticipate the application of corrective actions, moderate initially, on those flows that could later become responsible for congestion. This would be the case of flow *y*, which will fill the output buffer before starting to fill the input buffer.

6.4 Avoiding the Head-of-Line Blocking at Origins

Applying the FIFO technique at the injecting queues of the origin host, the HOL blocking phenomenon could appear if a packet is stopped at the head of the queue

avoiding in this way the advance of the rest of the packets at the queue. This could aggravate the network performance if a queue is shared with packets targeted to different destinations. In order to eliminate this HOL blocking phenomenon at the origins and correctly evaluate the proposed congestion management mechanisms, first, we assume a Full Virtual Output Queue (FVOQ) technique at the NICs of the source host. Next, in order to improve the scalability of the mechanisms we propose and evaluate some alternative structures, that is a Partial Virtual Output Queue (PVOQ) and a Shared-Buffer (SB) techniques.

6.4.1 Full Virtual Output Queue Technique

If packet injection were carried out through a simple output queue, as a consequence of the application of corrective actions when congestion is detected packets belonging to *hot-flows* stopped at the head of the buffer due to the reduction in the injection rate would prevent the advance of the packets belonging to *cold-flows*. To solve the HOL blocking problem referred above, the most effective solution is to provide a separate output queue for each packet flow, that is, applying a FVOQ technique.

The FVOQ implementation defines a number of queues equal to the number of destination hosts and with a queue size that depends on the maximum value for the DW size. Each origin host generates packets, classifying and storing them in a different queue depending on their destination host. The number of packets that can be stored in a queue is equal to the DW size, thus limiting the outstanding packets per flow. Each time an ACK packet arrives at the origin host, the first buffer in the corresponding queue is released, allowing to store another data packet into the queue. Notice that the queues are under the FIFO policy. Figure 6.1 shows the FVOQ structure.

The main drawback of the FVOQ implementation is that is not scalable. Despite the fact that FVOQ completely eliminates the HOL blocking phenomenon at the source nodes, the required storage space depends on the number of destinations, increasing proportionally with the number of hosts. The solution could be acceptable for small system sizes, but in systems with thousands of nodes the memory require-

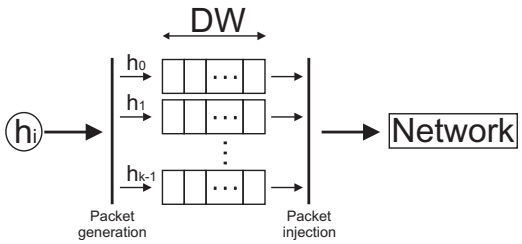


Figure 6.1: The full virtual output queuing technique.

ments would be quite significant.

Therefore, the FVOQ technique should be replaced by an alternative mechanism able to significantly reduce the storage requirements (preferably, its size should be independent of the number of hosts) without penalizing network performance. Notice that using a buffer organization different from FVOQ would also mean that every flow no longer has a dedicated queue to be stored, which could provoke to some extent the appearance of the undesirable HOL blocking effect. Two alternative approaches will be considered. One of them will consist of the use of a PVOQ technique, whereas the second one will involve the use of a SB technique. Both approaches exhibit some pros and cons, which we attempt to analyze in more detail in what follows.

6.4.2 Partial Virtual Output Queue Technique

The PVOQ technique consists in using a number of queues at the network interface of the source hosts smaller than the number of destination hosts. In particular, the number of queues could range from, at least, two queues up to the number of hosts minus one. Therefore, the storage space could be significantly reduced, becoming independent of the number of hosts, considerably improving the scalability of the mechanism. However, when using a number of bounded queues, packets addressed to different destination hosts will have to share the same queue, maybe causing HOL blocking, which could affect network performance, in particular the latency of the cold-flows. Notice that the first packet from the head of the queue is always chosen to be injected into the network. Therefore, the key issue is to reduce the number of queues as much as possible without affecting network performance too much.

The method used for mapping packet flows to queues is based on the destination address. Some bits of the destination address indicate the queue to place the packet. In particular, we use *module mapping* [70], where the queue selected to map the flow is obtained by applying the module operation to its destination address. As an example, for a network configuration of 512 destinations and 16 queues, the four least significant bits of the destination identifier (9 bits) indicate the queue to map the flow.

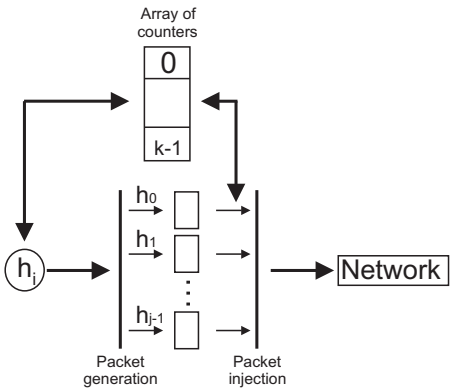


Figure 6.2: The partial virtual output queuing technique with an array of counters.

Additionally, in order to bound the number of outstanding packets when a DW mechanism is applied, as it is the case of MVCM and IOCM, the queue size is imposed by the DWmax value. However, notice that it is not required to store the whole packet until its corresponding ACK has been received (remember that we assume a lossless network). It is enough to remember how many packets of the flow have been injected. Thus, in order to reduce the amount of storage resources size even more, an array of counters can be used to control the number of outstanding packets. The counter range is imposed by the DWmax value in application of the corrective measures. This way, just one buffer can be dedicated per queue which will store the packet that could not be injected into the network due to the maximum counter value was achieved. We assume this implementation in the PVOQ technique as shown in Figure 6.2. Notice that the number of destination hosts (k) is larger than the number of dedicated queues (j). For each flow, its associated counter is increased each time

a packet belonging to it is injected into the network from the queue, whereas it is decremented when the ACK packet is received. Notice that the size of the counter array ($k * round[\log_2(DW_{max} + 1)]\ \ bits$) is not significant with regard to the space occupied by a single packet.

6.4.3 Shared-Buffer Technique

The SB technique is an alternative solution to reduce the storage requirements at the network interfaces of the origin hosts. It is based on dedicating a fixed buffer size regardless of the system size and traffic load, so also contributing to improve the scalability of the congestion management mechanism. It works as follows.

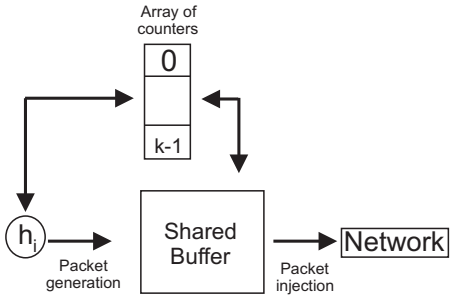


Figure 6.3: The shared buffer technique with an array of counters.

The origin host generates packets and stores them in a Shared-Buffer (SB) provided there is free space. The mechanism chooses the most appropriate packet from the SB to be injected into the network depending on the time when the packets were generated and the number of outstanding packets per origin-destination pair. Due to the fact that the SB is limited, its size has to be defined large enough to ensure that, in a congested situation, the SB has enough free space to store those packets whose injection rate is not affected by the corrective actions applied by the congestion management mechanism. This way, *cold-flows* can continue to inject packets into the network while packets destined to a congested area remain stored in the buffer due to the applied corrective actions. Moreover, as the corrective actions applied are based on a DW, it is also needed to control the number of outstanding packets per origin-destination pair. Therefore, a counter per destination host is required in order to track

the number of outstanding packets belonging to every flow. Figure 6.3 shows this implementation.

Once a packet is injected into the network, its occupied space in the SB is immediately released, remaining available for any other generated packet from any flow. As in the case of the PVOQ technique, if the buffer size is not large enough, the injection of packets belonging to *cold-flows* could be delayed. This would cause an increase in their latency. Again, the key issue is to minimize the buffer size without negatively impacting network performance.

Chapter 7

EVALUATION MODEL

This chapter describes the methodology used to evaluate all the contributions made in this thesis. Firstly, we describe the different methods that can be used to evaluate a system. Secondly, the assessment method used and the required parameters are explained. Finally, the simulated topologies and the applied traffic patterns are described.

7.1 Evaluation Methods

The evaluation of a system can be accomplished in several ways [52]:

- Measuring a real system: This method provides real results from the evaluated system. However, it requires the real system to be available, which in most cases is expensive, or simply not available. This method is not used in this thesis.

- Evaluating an analytical model: This is the cheapest alternative and provides results in a short time. However, these results are less accurate than other methods. Therefore, this model is not used in this thesis because we need a high level of detail in the model to obtain highly accurate results.

- Evaluating a simulated system: This is the most common process to evaluate

a system. Simulation tools provide great versatility and are ease to use. Using simulation tools may exhibit benefits but also inconveniences. Next, a list of benefits are enunciated.

- Most complex real-world systems with stochastic elements cannot be accurately described by a mathematical model and be so evaluated analytically. Thus, simulation is often the only possible way of investigation.
- Simulation allows to estimate the performance of an existing system under the own projected set of operating conditions.
- Alternative proposed system designs (or alternative operating policies for a single system) can be compared via simulation to see which of them better meets a specified requirement.
- In a simulation we can maintain much better control over experimental conditions than would generally not be possible when experimenting with the real system.
- Simulation allows to study in a short time the system running for a long time (e.g., an economic system), or alternatively a detailed study of a system running in a short period of time.

On the contrary, the problem of this method is the large required time to develop the simulation model, which must be a "valid" representation of the system under study for providing confidence simulation results, and also the time needed to run simulations and obtain results with regard to real systems.

In this thesis we have obtained all the results by applying the *simulated system method* as it is versatile, simple to use, and allows many simulations to be run in parallel, thus speeding up the research.

7.2 Simulation Tool

To perform these simulations, we have used one of the general-purpose interconnection-networks simulators developed by the GAP research group and appropriately up-

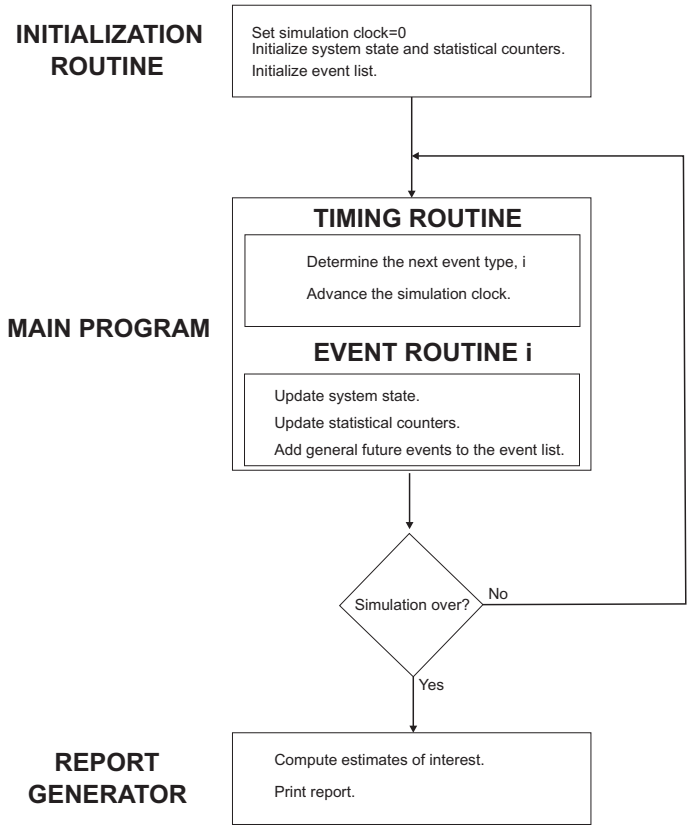


Figure 7.1: The structure of the discrete-event simulation model used.

graded with new functionalities, in order to evaluate the different proposals.

Figure 7.1 shows the basic structure of the discrete event-driven simulation model applied by the simulation tool used in our study. The simulation begins with the initialization routine, where the simulation clock is set to zero, the system state and the statistical counters are reset, and the event list is initialized. Then, the timing routine is called to determine the next event to process. As an example, we consider the next one as the event *i*. The simulation clock advances to the time in which the event *i* will occur. Then, the event routine *i* is called, where typically three types of activities occur: (1) updating the system state to account for the fact that an event of type *i* has occurred, (2) gathering information about system performance by updating the

statistical counters, and (3) computing the future events due to the accomplishment of the current event (this information is added to the event list). After processing the event, the simulator checks if the simulation process has finished (reaching a time stamp or a certain number of processed messages). If so, the statistics are dumped and the process finishes.

7.3 Simulator Features

There is a set of parameters (Input Parameters) to be defined in the interconnection networks simulator used in this thesis. These parameters enable the simulation of different network configurations under different workloads. As a result of the simulation process, a set of measures (Output Parameters) is obtained.

7.3.1 Input Parameters

- Design parameters of the network: The **network topology** used (Multistage Interconnection Networks), the **switching technique** applied (Virtual Cut-Through), **links** (Unidirectional or Bidirectional), the **routing algorithm** applied (Deterministic), the **number of virtual channels** defined (one), the **input and output buffer sizes**, and the **delay times** (Routing, Switch, and Link).

- Traffic load parameters: The **distribution of packets destinations** (Uniform, Hot-Spot), the **packet size** (fixed or variable), and the **generation rate**.

Some of these parameters have a specific value depending on the evaluation conditions. Therefore, their values will be indicated for each of the cases of study analyzed in Chapter 8.

7.3.2 Output Parameters

There are a large number of output results that the simulator offers us to assess the analyzed mechanisms. From the set of offered results, we use the following ones, and classify them in primary or secondary depending on their relevance:

- Primary: The **number of packets** (the number of generated, injected, and received packets, and the packets pending at origins or stopped in the network), the **throughput network** (amount of information delivered by the network per time unit), **average latency** (it is interesting to know the average time needed to cross the interconnection network from the origin to the destination node), **average latency from generation time** (it informs about the average time needed from the time the packet is generated at the origin node until it completely arrives at the destination node).

  Notice that all the results shown in this thesis are based on latency from generation time because working with network latency could be misleading. Let us take for example a network where a set of hosts restrict too much the injection rate of some flows as a result of a congestion notification. Then, the global network latency obtained will be reduced, but maybe the throughput would be also reduced due to the fact that network is becoming idle. That is, too much restriction on the injection rate could cause idle channels. On the contrary, latency from generation considers the extra time suffered by packets due to a congested network. In other words, a high value of the latency from generation would mean that larger buffers are needed to avoid a buffer overflow, and therefore that the application crashes. Anyway, the latency values should not be analyzed alone, but in combination with throughput.

- Secondary: The average occupancy queues (To detect a possible congestion situation).

7.4 Traffic Load Models

Once the simulation system has been developed, we have to define the traffic load which has to be applied in order to get results with different situations of traffic. There are different models of traffic load:

- Traces: In a real system, we can easily get trace files. Basically, these files contain information from the traffic generated by real applications, that is, in-

formation about the origin and destination nodes, the time when the packet was generated and the size of packets. Next, these files are loaded into the simulation system to simulate a real traffic load. The problem with this method is that it is dependent on many factors so that, when the trace files are applied to a different system, it could not reflect the real network performance.

- Synthetic Traffic: This is the most used traffic load in system evaluations with simulator tools because it allows to evaluate the network in the most generic way. Traffic flowing through the network is generated from a predefined traffic pattern. We evaluate the complete range of traffic injection rate, from low levels up to saturation point. The typical traffic distributions used are: uniform and hot-spot. Each one has a different distribution of packet's destination. Next, we describe them:

  – For *uniform traffic*, each source end node sends packets to all the destinations with the same probability. This pattern of communication is the most widely used in other jobs [23], [18], [15], [32], [1], [40].

  – For *hot-spot*, a percentage of the source end nodes inject traffic to the same destination, whereas the rest of origins inject traffic to random destinations (i.e.uniform traffic). This traffic pattern allows to model the situation when one or more end nodes (a disk server, for instance) are frequently accessed by the remaining origins.

  In this type of traffic load, the time to inject a packet and its size have to be defined in order to obtain different injection rates. The synthetic traffic is easy to generate, but could not fully model the traffic load in a real system. So, in order to get valid results, a large number of simulations has to be run and analyzed.

To evaluate the proposed mechanisms in this thesis, we will use mainly synthetic traffic. However, in a few cases we will apply different traces provided from real systems in order to validate their performances.

7.5 Experiments

In this section we describe the selected parameters for the simulator, the type of traffic load applied and the experiments run.

- Performance indices: The most important measures to evaluate the network performance are *simulation time*, *number of injected packets per origin node*, *latency*, and *throughput*. *Latency* informs about the time needed to deliver a packet at the destination node, whereas *throughput* reports about the accepted traffic. *Simulation time* and *latency* are measured in cycles, whereas *throughput* is measured in bytes/cycle/sw. Notice that, sw refers to the switches with source hosts connected to them (i.e. the switches of the first stage). It is interesting to know the *number of injected packets per origin node* in order not to create an unfair traffic load that could provide inequitable results.

- Evaluated topologies: The presented results in this thesis have been obtained by simulating the following topologies:

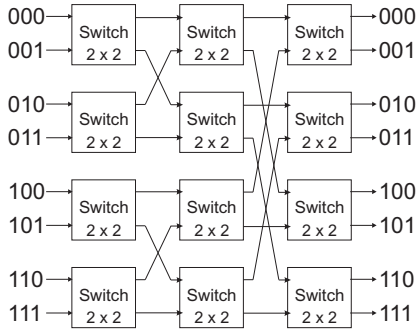
  Unidirectional Multistage Interconnection Networks (UMIN)

  - Butterfly network 4-ary 4-fly (256 hosts)
  - Butterfly network 8-ary 3-fly (512 hosts)

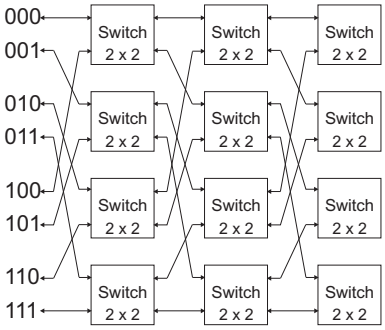
  Bidirectional Multistage Interconnection Networks (BMIN)

  - Perfect-Shuffle network 4-ary 3-fly (64 hosts)
  - Perfect-Shuffle network 4-ary 5-fly (512 hosts)
  - Perfect-Shuffle network 8-ary 3-fly) (512 hosts)

  Figure 7.2 shows two examples of networks used in our study, but with reduced dimensions in order to fit the diagram in this space. In particular, Figure 7.2(a) presents a Unidirectional Butterfly 2-ary 3-fly network, and Figure 7.2(b) shows a Bidirectional Perfect-Shuffle 2-ary 3-fly network.



(a) Unidirectional Butterfly 2-ary 3-fly



(b) Bidirectional Perfect-Shuffle 2-ary 3-fly

Figure 7.2: Samples of used networks.

The interest in analyzing topologies with a different number of stages, *n*, and different values of radix, *k*, is that, by varying these parameters, the performance of the proposed mechanisms for the congestion management could change. So, in order to ensure the proposed mechanisms work correctly under different scenarios, it is necessary to simulate different configuration networks.

- Evaluated Packet Sizes: Simulations have been carried out with different data packet sizes. Depending on the network configuration, we have considered both fixed payload of 256, 512, and 1024 bytes and variable payloads of 64 up to 512 bytes. In all the configurations, 22 bytes for the header have been assumed. Moreover, the ACK packet size is 22 bytes for all the cases. These packet sizes have been assumed following some standard specs as InfiniBand.

- Switch design parameters: For the delay times (Routing, Switch, and Link) a value of one cycle for each parameter has been assumed. The input and output ports of the switch have 1kB buffers associated. In particular this buffer size is assumed when the simulated packet size used is 256 bytes. For the rest of packet sizes, that is, 512 and 1024 bytes, a buffer size of 2kB and 4kB have been assumed for the fixed packet size respectively, and a buffer size of 2kB for the variable packet size.

- Traffic load: As it is commented before, both a synthetic traffic pattern and

a traffic load based on traces are used to evaluate the proposed mechanisms. In particular, for the synthetic traffic two destination distributions have been applied. Notice that, depending on the study case, these distributions have been applied either in isolation or combined in a certain ratio, as will be shown in Chapter 8.

- *Uniform distribution*: In this type of distribution, the destination host is chosen randomly from all hosts on the network.
- *Hot-spot distribution*: In this type of distribution, the destination host is always the same host for all the generated packets.

Basically, the traffic load used in this thesis is composed of two different types of flows, that is, flows injecting uniform traffic and flows injecting hot-spot traffic. In particular, all network configurations dedicate the 80% of all origin hosts to inject uniform traffic while the remaining 20% inject hot-spot traffic. Hosts injecting hot-spot traffic have been uniformly chosen between the total set of origin hosts. It first generates packets according to a uniform distribution of destinations, and when the network becomes stable, then it creates a hot-spot into the network.

Additionally, a traffic load based on traces have been applied. In our case of study, we have worked with traces from message passing interfaces (MPI) that allows computers to communicate with one another in clusters and supercomputers. In particular, the traces used in our study are from the applications DL_POLY [29] and CPMD [21]. DL_POLY is a general purpose software for the study of the classical molecular dynamics process, and CPMD is a software which implements the density functional theory, particularly designed for molecular dynamics. Both traces have been simultaneously run in order to stress the traffic network even more. The process followed was, firstly, to inject the traces from the CPMD application and to keep injecting packets from this pattern throughout the simulation time. Secondly, when the network status became stable, the traces from the DL_POLY application were then injected.

From that moment, the network performance is traced.

7.6 Parameters of Analysis

We have based our analysis on two basic performance parameters, namely *latency* and *throughput*. In particular, these parameters have been analyzed under certain conditions, that is, under network saturation and congestion. This allow us to find out if the Congestion Management Mechanisms (CMMs) analyzed are able to: *i*) improve the network performance under a congestion situation, and *ii*) they do not affect network performance in the absence of congestion.

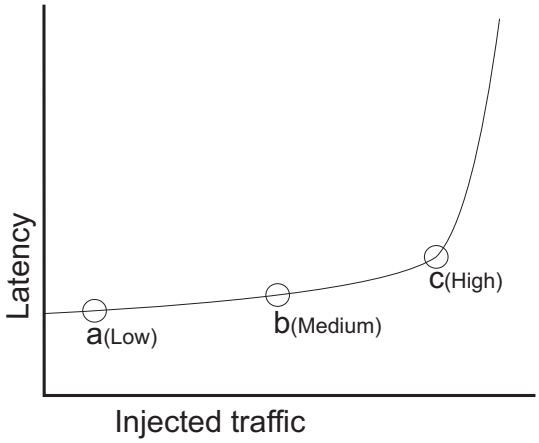


Figure 7.3: Analysis points depending of the traffic load.

In order to analyze the behavior of the system, we have defined three points of analysis such as shown in Figure 7.3. They represent the achieved latency for a given injected traffic into a network. These three points (*a*,*b*,*c*) allow us to analyze the network performance with a low, medium, and high traffic load, respectively. In particular, the point, *c* presents the network performance when the network is near the saturation point but not in the saturation situation yet. In this three point of analysis, we take samples of the system and represent the latency and throughput along the simulation time.

To obtain valid results, it is necessary to define a transitional period of simulation in which the network performance is unstable. Once the system reaches a stable state, we can take samples for the analysis. This period of instability depends on network size and injected traffic. For each case of study the duration of the transition period is described in the Performance Evaluation Section.

Chapter 8

PERFORMANCE EVALUATION

In this chapter, we evaluate in depth all the congestion management mechanisms proposed in this thesis. In addition, a comparative analysis with respect to the previous proposals is made.

8.1 Introduction

After proposing two new a new congestion management mechanisms, two basic analyses have to be carried out to validate the new proposal. The main goal is to check if the mechanisms are able to effectively reduce the negative effects caused by a congestion situation. Firstly, the performance of the new proposal with different network configurations and under different traffic loads has to be compared with the results when no congestion management mechanism is applied. The second step is to compare the mechanism performance to the ones achieved by previous proposals. This second analysis is intended to verify whether the proposed mechanisms are able to improve the results achieved by the latter ones.

Following this evaluation method, in this section, first, we show in detail a complete analysis of the MVCM mechanism with different network configurations and under different traffic loads. Next, a comparative analysis between the MVCM performance and the previous proposals, (Pfister's implementation, and Renato's pro-

posal) is carried out. Additionally, the IOCM proposal has been added to this analysis in order to find out which packet marking technique achieves better results in different environments, under the same conditions (network configuration and traffic load). In particular, this comparative study focuses on a global analysis of the congestion management mechanisms such as they were originally proposed.

Next, in order to explain the behavior of these mechanisms, a more detailed analysis of the influence of the different strategies applied for packet marking, congestion detection, and congestion correction is presented. In particular, this analysis presents the results of the impact of applying the different packet marking schemes (IPM, RPM, OPM, IOPM, and MVPM), the window-based schemes applied (dynamic and static window), the corrective actions scheme defined, the effect of applying a WS limitation, and the impact of applying a recovery scheme (progressive or immediate recovery), such as they were defined in Chapters 4 and 5.

Finally, an analysis of the three techniques to eliminate the HOL-blocking phenomenon by classifying generated packets at origins, that is, Full Virtual Output Queue, Partial Virtual Output Queue, and Shared-Buffer, such as proposed in Chapter 6, is presented. Additionally, a study about the storage requirements of each alternative is done.

8.2 The MVCM Proposal

8.2.1 Performed Analysis

In this study, a large set of results from several network configurations and different traffic rates have been evaluated and analyzed. In particular, performance results for the *Unidirectional MINs (UMINs) Butterfly 4-ary 4-fly* and *8-ary 3-fly*, and for the *Bidirectional MINs (BMINs) Perfect-Shuffle 4-ary 3-fly*, *4-ary 5-fly*, and *8-ary 3-fly* are shown. Notice that these network configurations have been chosen in order to find out if the MVCM mechanism behaves in the same way regardless of the switch radix and the number of network stages. Additionally, all these network configurations have been evaluated with *Low*, *Medium*, and *High* injection rates.

The traffic patterns applied are based on synthetic traffic. Basically, this traffic load is composed of two different types of flows, that is, flows injecting uniform traffic and flows injecting hot-spot traffic. Table 8.1 shows the synthetic traffic pattern applied to all these network configurations. In particular, in all network configurations an 80% of all origin hosts inject uniform traffic while the remaining 20% inject hot-spot traffic. Hosts injecting hot-spot traffic have been uniformly chosen among the total set of origin hosts. As shown in Figure 8.1, it first generates packets according to a uniform distribution of destinations, and when the network becomes stable, then it creates a hot-spot into the network.

Network	**Traffic pattern**
#Sources	**Destination**
80%	Uniform
20%	Stop-HotSpot-Stop

Table 8.1: The synthetic traffic pattern applied in the MVCM analysis

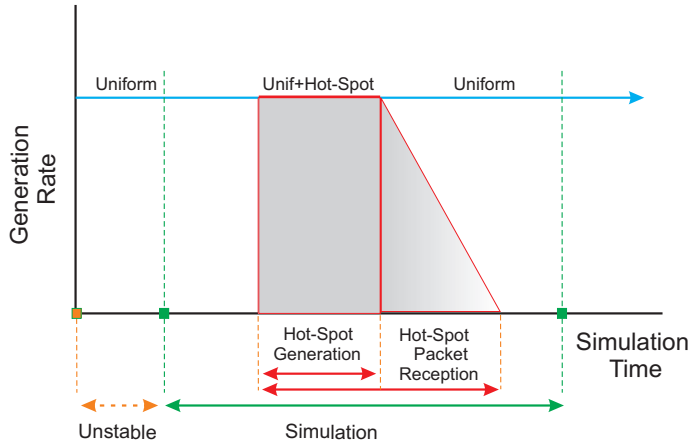


Figure 8.1: Graphic diagram of the applied traffic pattern, based on a hot-spot traffic.

Basically, hosts that send uniform traffic continue injecting packets during the whole simulation, and hosts that generate hot-spot traffic remain inactive until the first 50,000 packets have been completely received at the destination hosts, which

makes the network performance stable. Next, they start generating a set of packets from each origin host with the same injection rate as that of the other hosts, but addressed to a single destination host (the hot-spot). Later, when each host has completed the generation of the hot-spot packets, they stop generating new ones. Notice that simulation finishes after all the packets belonging to the hot-spot traffic are completely received.

In order to analyze the real effect of the mechanism over the different types of flows in the network, both *cold* and *hot-flows* have been separately traced in all simulations.

8.2.2 Parameter Initialization

First of all, to evaluate the MVCM proposal, it is necessary to initialize previously its configuration parameters with the correct values. As described in sections 4.3 and 5.3, the correct initialization of those parameters is key to achieve a good mechanism performance and, therefore, to reach the best results. In what follows, we describe the process followed to initialize correctly the different parameters of the mechanism, that is, the buffer thresholds, the window size, and the waiting interval calculation values.

- Buffer Threshold: To define the values for the input and output buffer thresholds, we have calculated the average buffer occupancy under the worst working conditions under a well-balanced traffic, that is, when a uniform traffic distribution with an injection rate near the saturation point is applied. With this traffic load, we try to stress the network by reaching performance values close to the saturation point, thus obtaining the maximum values for the buffer occupancy that can be achieved with a uniform traffic load. To obtain valid samples for the analysis, it has been necessary to start the simulation and to wait for a stable period. After this transitional period, the simulator takes some samples. Figure 8.2 shows the results of our simulations.

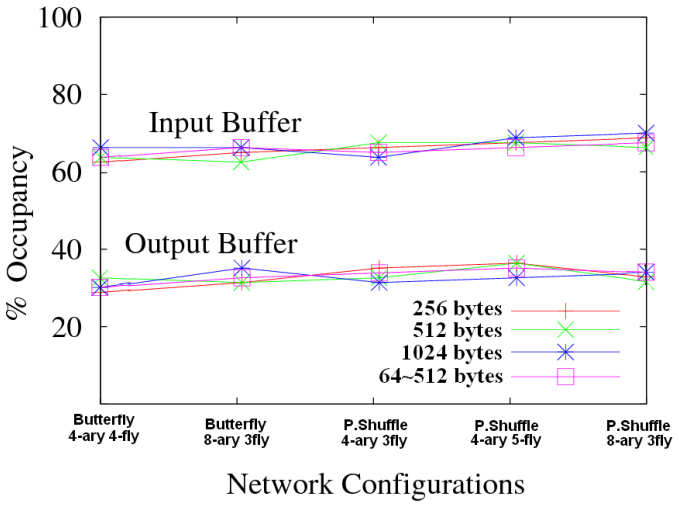


Figure 8.2: The average occupancy for the input and output buffers for different network configurations and packet lengths.

The analysis was carried out with all the network configurations evaluated in this thesis, that is the *UMINs Butterfly 4-ary 4-fly* and *8-ary 3-fly*, and the *BMINs Perfect-Shuffle 4-ary 3-fly, 4-ary 5-fly* and *8-ary 3-fly*. Additionally, different packet sizes have been tested, that is, fixed packed sizes of 256, 512, and 1024 bytes, and variable packet sizes between 64 and 512 bytes. In particular, these packet sizes have been chosen with the purpose of comparing the results with those achieved by the current proposals for InfiniBand. For these packet sizes, the buffers were defined with a size of 1, 2, and 4 kB for the different fixed sizes of 256, 512, and 1024 bytes, respectively, and 2 kB for the variable size, in order to keep constant the ratio between the buffer size and the packet length. Notice that the occupancy at both input and output buffers were separately traced. In particular, the occupancy for every input and output buffer has been checked every 1,000 cycles during all the simulation, thus obtaining an average result at the end of the simulation process.

As it can be seen in Figure 8.2, we have obtained, on average, an occupancy of about 66% and 33% of the input and output buffer capacity respectively, for all

the packet sizes and network configurations analyzed. These results allow us to choose the initial values for the thresholds which are the responsible for detecting a congestion situation. So, as a result of that, we have assumed the values of 66% and 33% of the buffer capacity for the input and output buffers, respectively, in all the simulations conducted in this thesis. Notice that, although the obtained results for buffer threshold are similar in every configuration network analized, it will always depend on the buffer and packet size assumed. However, the most important thing is that the process applied to calculate the buffer threshold is simple and effective.

- Window Size: In order to initialize the dynamic window value for the different network configurations analyzed in this thesis, let us apply the theoretical study shown in section 5.3.1 to the largest and smallest network configurations (in number of stages) simulated, that is, the *BMIN 4-ary 5-fly*, and the *UMIN 8-ary 3-fly* configurations, respectively.

  For the *BMIN 4-ary 5-fly*, the calculated dynamic window values for the smallest (256 bytes) and the largest (1024 bytes) packet sizes by applying the formula (5.6) are:

$$\omega_{256} = \frac{K+(L_{head}+L_{payload})}{(L_{head}+L_{payload})} = \frac{76+(22+256)}{22+256} = \frac{354}{278} \approx 1.28$$

$$\omega_{1024} = \frac{K+(L_{head}+L_{payload})}{(L_{head}+L_{payload})} = \frac{76+(22+1024)}{22+1024} = \frac{1122}{1046} \approx 1.07$$

  where the value of the K constant is equal to $(2 * h * t_{hop} * B + L_{ack})$ (see section 5.3.1).

  In order to check this theoretical values, we have obtained some simulation results. Let us assume a uniform traffic distribution destination with an injection rate near the saturation point and with the two different fixed packet sizes. In these scenarios, we have obtained window sizes with values of at least $\omega'_{256} \approx 1.4$ and $\omega'_{1024} \approx 1.27$ packets for packets of 256 and 1024 bytes, respectively. These values have been obtained by calculating the average number of outstanding packets per flow during all the simulation time. Notice that

a large number of samples have been taken during all the simulation time in order to calculate the average value.

As the calculated values in the absence of contention are $\omega_{256} = 1.28$ and $\omega_{1024} = 1.07$ packets, and the ones achieved by simulation are $\omega_{256}^{'} = 1.4$ and $\omega_{1024}^{'} = 1.27$ packets, respectively, the correct window size to be chosen should be $\omega = 2$ in both cases, which is the nearest integer value which does not penalize network throughput as a smaller value would do. Notice that the $\omega^{'}$ values are slightly larger than the theoretical values ω because some contention appears in the network. Larger values for the window size are not useful when the network is not congested, but allow to inject more packets into the network when congestion appears and packet latency increases, worsening the congestion situation.

Next, we analyze the dynamic window size for the *UMIN 8-ary 3-fly*, with the same conditions described in section 5.3.1 and evaluating both packet sizes, that is, a fixed data payload equal to 256 and 1024 bytes. We obtain that the optimum ω values are:

$$\omega_{256} = \frac{K+(L_{head}+L_{payload})}{(L_{head}+L_{payload})} = \frac{40+(22+256)}{22+256} = \frac{318}{278} \approx 1.14$$

$$\omega_{1024} = \frac{K+(L_{head}+L_{payload})}{(L_{head}+L_{payload})} = \frac{40+(22+1024)}{22+1024} = \frac{1086}{1046} \approx 1.04$$

Again, let us assume a uniform traffic destination distribution with an injection rate near the saturation point in both cases. With this configuration, we have obtained a window size at least with a value of $\omega_{256}^{'} \approx 1.2$ and $\omega_{1024}^{'} \approx 1.09$ packets for the data payload equal to 256 and 1024 bytes, respectively. Therefore, the correct window size to be chosen will be again $\omega = 2$ in both cases.

As a result of all of that, we will apply a dynamic window with a value of two in all the network configurations evaluated in this thesis, as a smaller or larger value will penalize over or under the packet injection.

- Waiting Interval Calculation: As a result of the analysis shown in section 5.3.2, where it was studied that the parameters *k* and *n* from the network configuration (*k-ary n-fly*) can be used to efficiently calculate the waiting interval in multistage interconnection networks, we will apply the proposed method for all the network configurations evaluated in this thesis. Remember that, the *k* parameter indicates the value to reduce the injection rate by dividing the current injection rate each time a validated ACK packet is received, whereas the *n* parameter limits the maximum number of reductions that can be applied.

8.2.3 Evaluation Results

In order to check if the MVCM proposal is a proper alternative to reduce the undesirable effects caused by a congestion situation, first we analyze in depth how the congestion management mechanism reacts against a congestion situation created in a *BMIN Perfect-Shuffle 4-ary 5-fly* network. Different injection rates have been checked, (low, medium, and high injection rate) when applying the synthetic traffic described in Table 8.1. In particular, we analyze this network configuration because its size is big enough to analyze all the effects caused by a congestion situation. Notice that, although other network configurations are evaluated in order to confirm the mechanism performance, this network configuration will be the one used to analyze and compare later the MVCM mechanism with respect to the other current proposals in the same working environment.

Next, we will also analyze other network configurations as *BMINs Perfect-Shuffle 4-ary 3-fly* and *8-ary 3-fly* and the *UMINs Butterfly 4-ary 4-fly* and *8-ary 3-fly*, also with different injection rates, in order to validate that the proposed mechanism gets good results regardless of the network configuration and traffic load. We separately analyze latency and throughput results for *cold* and *hot-flows*. Notice that, instead of using network latencies, we calculated latencies from generation time in all the simulation results presented in this thesis. The reason is that, although real applications would stop generating packets when packets cannot be injected into the network as a consequence of congestion, working with network latency could be misleading. Let

us take, for example, a network where a set of hosts restrict too much the injection rate of some flows as a result of a congestion notification. Then, the global network latency obtained will be reduced, but maybe the throughput would be also reduced due to the fact that network is becoming idle. That is, too much restrictions on the injection rate could cause idle channels. On the contrary, latency from generation time considers the extra time suffered by packets due to a congested network. In other words, a high value of latency from generation would mean that larger buffers are needed to avoid a buffer overflow, and therefore that the application may crash. Anyway, the latency values should not be analyzed alone, but in combination with throughput values.

Rate	**UMINs Butterfly**		**BMINs Perfect-Shuffle**			
	4-ary 4-fly	**8-ary 3-fly**	**4-ary 3-fly**	**4-ary 5-fly**	**8-ary 3-fly**	
Low	0.25	0.6	0.25	0.25	0.5	(1)
	16	39	4	32	32	(2)
Medium	0.45	1.1	0.45	0.53	0.9	(1)
	29	69	7.2	68	58	(2)
High	0.6	1.6	0.62	0.76	1.5	(1)
	38	102	9.6	98	96	(2)

Table 8.2: Traffic rates applied.
(1) bytes/cycle/injecting_sw and (2) bytes/cycle.

Table 8.2 presents the different injection rates applied to every case of study. Notice that values in rows (1) from Table 8.2 indicate the injection rate in *bytes/ cycle/ injecting sw*[1], whereas values in rows (2) indicate the corresponding rate in *bytes/cycle*.

In order to choose each of the traffic rates: low, medium, and high for a specific network configuration, a graph of latency versus traffic has been obtained. Figure 8.3 shows the traffic evolution on the network from low load till the saturation point when

[1] *sw* refers to the switches with source hosts connected to them, that is, the switches of the first network stage

the MVCM mechanism is applied for the *BMIN Perfect-Shuffle 4-ary 5-fly*. Additionally, curve *no CMM* plots the network performance when no congestion management mechanism is applied. The curve *MVCM* identifies three points of analysis, namely a, b, and c, which correspond to low, $medium$, and $high$ injection rates, respectively. In particular, the values of the injection rate are 32, 68, and 98 *bytes/cycle*, which correspond to a 0.25, 0.53, and 0.76 *bytes/cycle/injecting_sw*. This network performance has been obtained by applying the traffic pattern shown in Table 8.1 in each simulation point. Notice that this pattern combines uniform traffic and traffic addressed to a hot-spot. This fact explains the latency increase even when only a low rate of traffic is applied.
The rest of the values for the other network configurations, shown in Table 8.2, have been chosen following the same method.

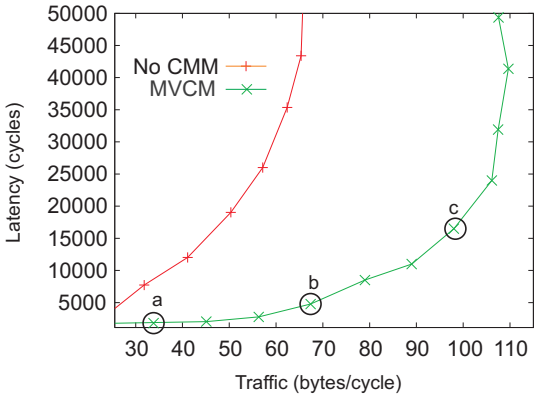


Figure 8.3: Points of analysis in a BMIN Perfect-Shuffle 4-ary 5-fly when applying the synthetic traffic traffic described in Table 8.1.

The method followed in the evaluation of the MVCM mechanism starts with the study of the *BMIN Perfect-Shuffle 4-ary 5-fly* with a medium injection rate, which could represent the normal load of an interconnection network, and with a fixed packet size of 256 bytes. Next, these results are compared to the ones obtained with the same network configuration but with low and high injection rates, in order to check if the MVCM mechanism is able to palliate congestion with different injection

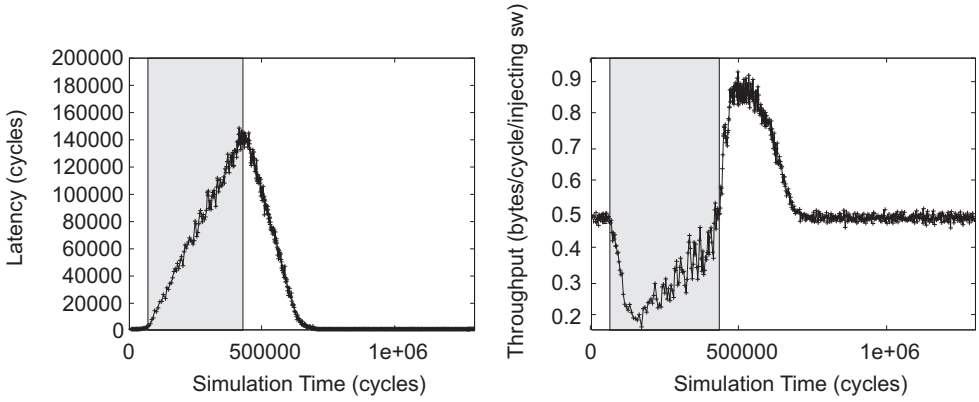
rates. Additionally, the same network configuration and traffic rates are evaluated with a variable packet size, between 64 and 512 bytes, in order to validate that the mechanism also functions with variable packet sizes. The MVCM has also been evaluated with fixed packet sizes of 512 and 1024 bytes. However, performance results for these values are not shown because they are similar to those obtained for the fixed packet size of 256 bytes.

Finally, in order to generalize these results to other multistage interconnection networks, the MVCM is evaluated with other network configurations which vary the parameters k and n (in networks *k-ary n-fly*), in order to find out if MVCM is able to also react efficiently, regardless of the size of the switch radix (k) and the number of network stages (n).

BMIN Perfect-Shuffle 4-ary 5-fly

1. ***Medium Injection Rate, Fixed Packet Size***

   Figures 8.4 and 8.5 show how the performance of the *cold* and *hot-flows*, respectively, are affected when a congestion situation appears and no congestion management mechanism is applied to solve it. Notice that in all figures, the shaded area approximately indicates the period of time during which the origin hosts that generate the traffic destined to the hot-spot are injecting packets in order to provoke a congestion situation.

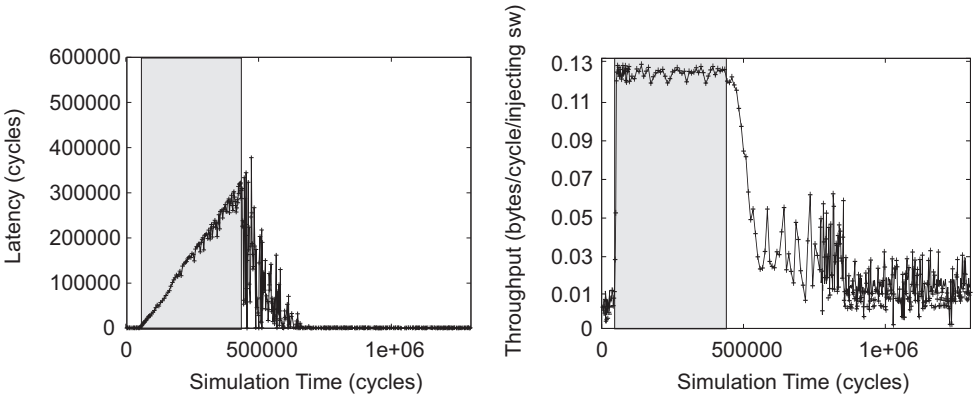


(a) Latency (b) Throughput

Figure 8.4: (a) Latency and (b) throughput for cold-flows in a BMIN Perfect-Shuffle 4-ary 5-fly without any CMM when applying a medium injection rate and a fixed packet size=256 bytes.

In particular, Figures 8.4(a) and 8.5(a) represent latencies from generation measured in clock *cycles*, whereas Figures 8.4(b) and 8.5(b) represent throughput measured in *bytes/cycle/injecting_sw*.

As can be seen in Figure 8.4(a), latency for *cold-flows* increases up to more than 140,000 cycles during the congestion period, thus penalizing the normal traffic for *cold-flows*. In the same way, Figure 8.4(b) confirms this trend by showing the throughput drop suffered by these flows within this congestion period. This reduction appears because traffic addressed to the hot-spot allows

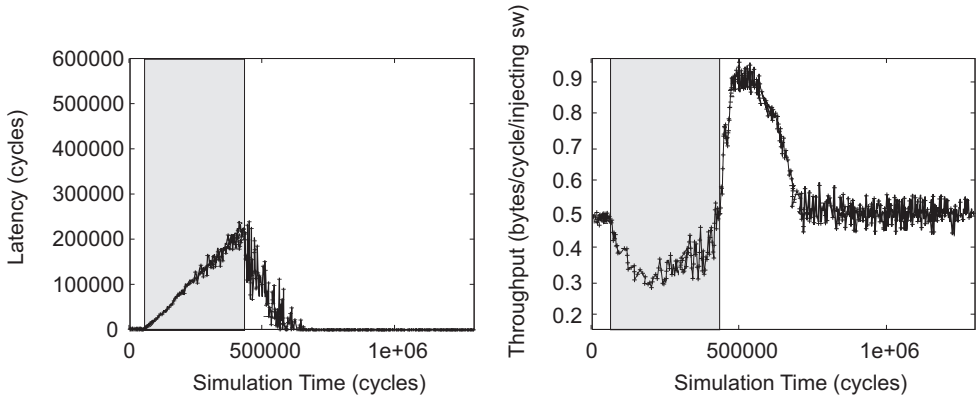


(a) Latency (b) Throughput

Figure 8.5: (a) Latency and (b) throughput for hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly without any CMM when applying a medium injection rate and a fixed packet size=256 bytes.

lower throughput than uniform traffic distribution does, due to the higher contention for the channels. Therefore, when the hot-spot traffic starts, all the network throughput falls down till the maximum level allowed by the *hot-flows* (hot-spot traffic). This is due to the *cold-flows* are suffering HOL blocking provoked by the *hot-flows* that prevents the network bandwidth from being efficiently exploited. Figure 8.5(b) shows how *hot-flows* increase their throughput during the congestion period until approx the level where the throughput of *cold-flow* drops. Also, as Figure 8.4(a) shows during this process, *hot-flows* increase progressively their latency. Notice that *cold-flows* traffic is about 80% of the generated traffic, so this decrease represents a very substantial fall of the traffic that is not compensated by the increase in *hot-flows* traffic. Therefore, the injected traffic is not fairly balanced.

This effect can be observed in Figures 8.6(a) and 8.6(b) which present latency and throughput, respectively, for *cold+hot flows* when no CMM is applied. As it can be seen, although the percentage of *hot-flows* is about 20%, it provokes a significant reduction in network throughput and an increase in network latency.

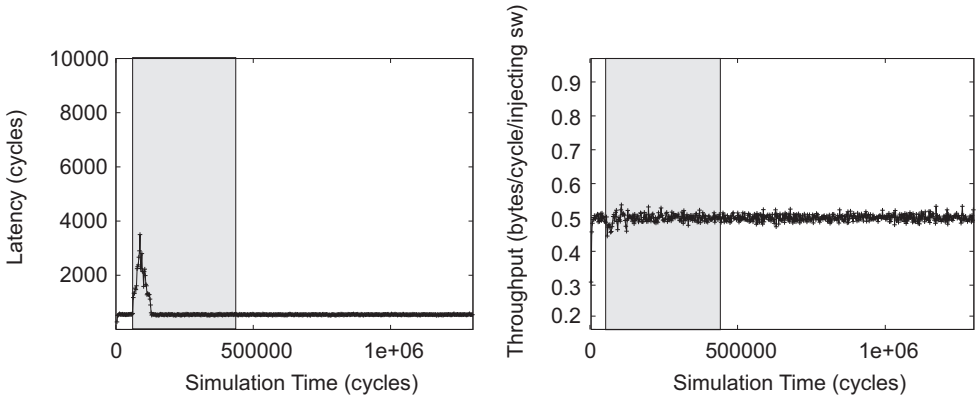


(a) Latency (b) Throughput

Figure 8.6: (a) Latency and (b) throughput for cold+hot flows in a BMIN Perfect-Shuffle 4-ary 5-fly without any CMM when applying a medium injection rate and a fixed packet size=256 bytes.

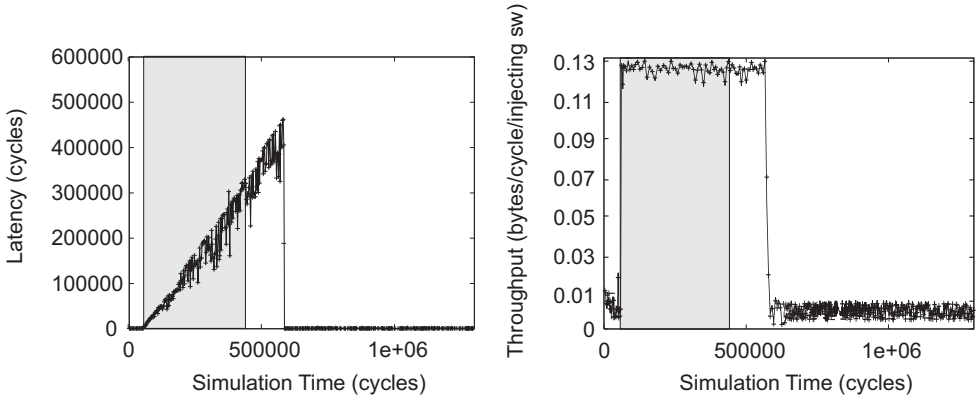
Next, Figures 8.7 and 8.8 show how the MVCM mechanism is able to reduce the negative effects caused by this congestion situation. In particular, Figures 8.7(a) and 8.7(b) represent latency and throughput, respectively, for *cold-flows*, whereas Figures 8.8(a) and 8.8(b) represent latency and throughput, respectively, for *hot-flows* when the MVCM mechanism is applied. As can be seen in Figure 8.7(a), the maximum value for latency has been reduced to less than 4,000 cycles approximately. It represents an important reduction when compared to the maximum value of 140,000 cycles achieved in Figure 8.4(a).

Notice that in Figures 8.4(a) and 8.7(a) the scales of y-axis are different in order to better compare the maximum values of latency achieved. In the same way, the MVCM mechanism reacts avoiding the sharp drop which appeared on Figure 8.4(b), and providing a sustained performance level reducing the throughput degradation, as presented in Figure 8.7(b). This result is positive as it is indicative that HOL blocking is avoided and packets belonging to *cold-flows* are barely suffering the congestion problems.



(a) Latency (b) Throughput

Figure 8.7: (a) Latency and (b) throughput for cold-flows in a BMIN Perfect-Shuffle 4-ary 5-fly when applying the MVCM with a medium injection rate and a fixed packet size=256 bytes.

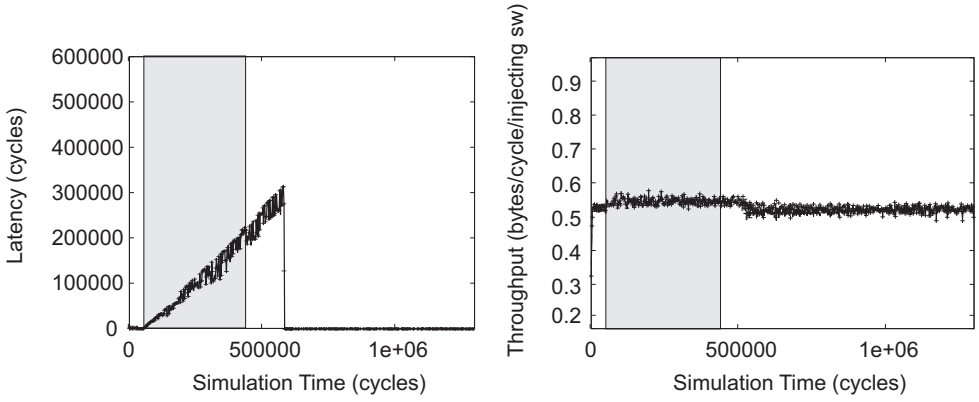


(a) Latency (b) Throughput

Figure 8.8: (a) Latency and (b) throughput for hot-flows in a BIMN Perfect-Shuffle 4-ary 5-fly when applying the MVCM with a medium injection rate and a fixed packet size=256 bytes.

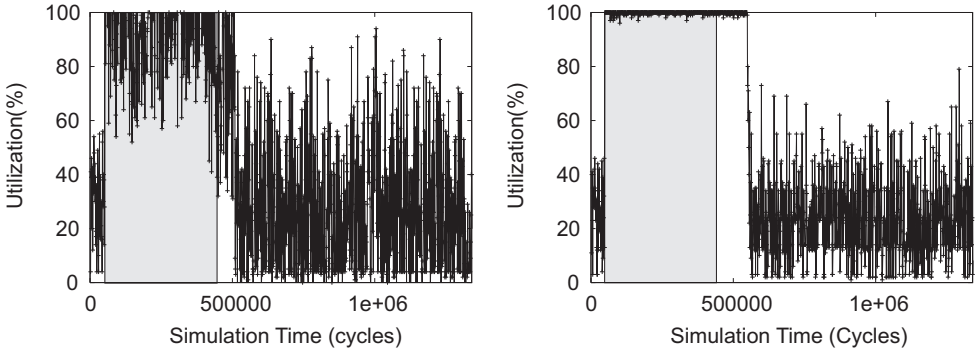
In addition, when comparing the *hot-flow* performance between Figures 8.5 and 8.8, we can observe that, although packets belonging to *hot-flows* suffer an increase in their latency up to 450,000 cycles in Figure 8.8(a) with regard

to Figure 8.5(a), again this result is positive because the MVCM mechanism is able to fairly manage the network bandwidth due to the fact that oscillations are reduced when the congestion period ends, as can be observed in Figures 8.8(a) and 8.8(b).



(a) Latency (b) Throughput

Figure 8.9: (a) Latency and (b) throughput for cold+hot flows in a BMIN Perfect-Shuffle 4-ary 5-fly when applying the MVCM with a medium injection rate and a fixed packet size=256 bytes.



(a) No CMM (b) MVCM

Figure 8.10: Percentage of the utilization of the link connected to the hot-spot destination in a BMIN Perfect-Shuffle 4-ary 5-fly when applying a medium injection rate, a fixed packet size=256 bytes, and with (a) no CMM and (b) MVCM.

Notice that these results are expected because the total amount of bandwidth demanded by the origin hosts cannot be provided by the hot-spot, so the throughput is reduced and as a consequence, the latency of the involved packets is increased.

In the same way, Figures 8.9(a) and 8.9(b) present latency and throughput, respectively, for *cold+hot flows* when the MVCM mechanism is applied. As it can be seen, Figure 8.9(a) presents a steady increase in the global latency. However, when all hot-spot packets are consumed, latency is suddenly reduced unlike what Figure 8.6(a) shows when no CMM is applied. Additionally, Figure 8.9(b) shows that there are no longer throughput drops due to congestion.

Next, in order to validate the fairness of the strategy applied to reduce the congestion, it is mandatory to verify that MVCM does not reduce the injection rate of flows injecting packets toward the hot-spot in excess. To this end, Figure 8.10 shows the percentage of utilization of the link connected to the hot-spot for the analyzed network. Figures 8.10(a) and 8.10(b) present results when no corrective actions are taken and when the MVCM corrective actions are applied over the responsible flows, respectively. Notice that during the congestion period, Figure 8.10(b) shows that the analyzed link is busy at 100 per cent when the MVCM proposal is applied. On the contrary, when no CMM is applied, the link bandwidth is not utilized at 100 percent in a sustained way due to the fact that *hot-flows* also suffer the consequences of the congestion caused by themselves. Moreover, when congestion disappears, the utilization values present less oscillations than when no congestion management is applied in Figure 8.10(a). So, although corrective actions are stopping packets belonging to flows responsible for congestion at origins to provide enough bandwidth for the flows non-responsible for congestion, those flows continue to inject enough packets into the network to keep busy the link connected to the hot-spot. Therefore, the MVCM proposal not only benefits latency and throughput for *cold-flows*, as it has been seen in previous figures, but also improves the *hot-flows* performance by keeping the link connected to the hot-spot busy dur-

ing all the simulation time. A higher injection of packets belonging to *hot-flows* would create head-of-line blocking into the network and, therefore, the latency for *cold-flows* would increase.

As a result, it is verified that the set of corrective actions applied by MVCM over the flows responsible for congestion not only benefits *cold-flows*, but it does not penalize the accepted traffic rate for *hot-flows*, rather it improves it.

2. ***Low and High Injection Rate, Fixed Packet Size***
   Although a medium injection rate seems to be the normal traffic load for a well-designed interconnection network, it is interesting to test how the MVCM behaves when a lower or a higher injection rate is applied in the same network configuration and with the same traffic pattern.

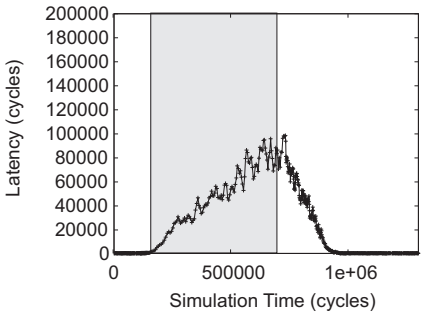
   Next, graphs in Figure 8.11 show the network performance when applying the same traffic pattern as the one applied in the medium injection rate, but now with low and high injection rates. In particular, Figures 8.11(a_1) to (d_1) present results for low injection rates, whereas Figures 8.11(a_2) to (d_2) present results for high injection rate. Moreover, Figures 8.11(a_1) and (a_2) represent latency for *cold-flows* when no congestion management mechanism is applied, whereas Figures 8.11(b_1) and (b_2) present latency for *cold-flows* when MVCM is applied. As can be observed, again, the values of latency have been drastically reduced when MVCM is applied, allowing the packet advance through the network with almost no contention.

   In particular, when applying a low injection rate, for *cold-flows* maximum values of latency up to 100,000 cycles have been reduced to a maximum value of 2,000 cycles whereas values up to 300,000 cycles have been reduced to 4,000 cycles for a high injection rate.
   Comparing Figures 8.11(a_1) and (a_2) to the one in Figure 8.4(a), we observe some differences in performance due to the different injection rate applied. In particular, when a lower injection rate is used (Figure 8.11(a_1)), the period generating the hot-spot traffic is larger than the one in Figure 8.4(a). Addition-

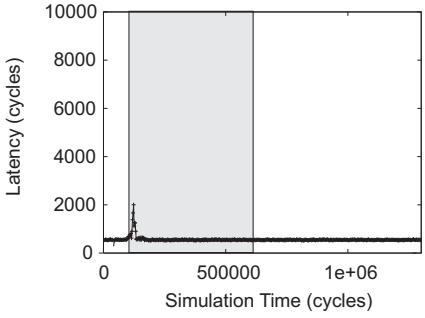
ally, Figure 8.11(a_2) presents the shortest injection period for hot-spot traffic because the injection rate is higher than the previous ones.

On the other hand, Figures 8.11(c_1) and (c_2) represent latency for *hot-flows* performance when no congestion management mechanism is applied, whereas Figures 8.11(d_1) and (d_2) present latency for *hot-flows* when MVCM is applied. Again, although packet latency is increased because *hot-flows* have to stop packets at origins due to the corrective actions applied over the flows responsible for congestion, once more the MVCM mechanism improves the *hot-flow* performance as it eliminates oscillations when the congestion period is finished. As can be observed, the MVCM mechanism immediately reacts against a congestion situation when a lower or a higher injection rate are applied as it did with a medium injection rate (see Figure 8.8(a)).

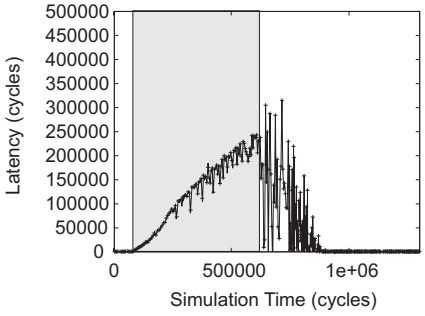
Based on our extensive study, we can conclude that MVCM works well with a fixed packet size of 256 bytes regardless of the traffic injection rate of the nodes (graphs shown before) and also for packet sizes of 512 and 1024 bytes (graphs not shown because the achieved results are similar). Next, to complete the study, it is also interesting to analyze the same network configuration but with a variable packet size. In particular, the size of the packets will vary between 64 and 512 bytes.



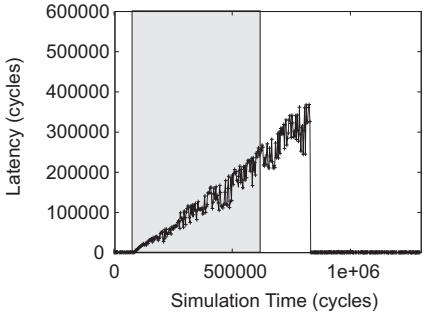
(a_1) Low inject, Cold-flows, no CMM



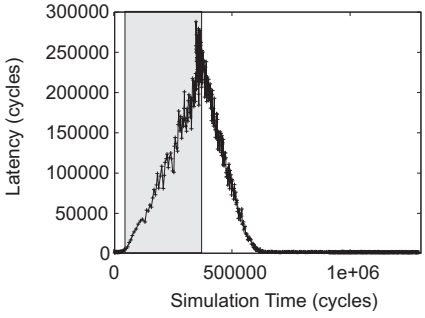
(b_1) Low inject, Cold-flows, MVCM



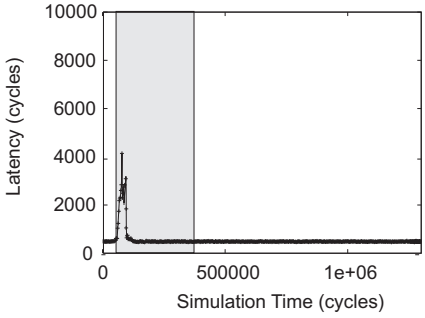
(c_1) Low inject, Hot-flows, no CMM



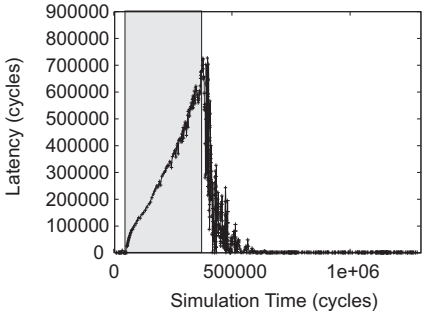
(d_1) Low inject, Hot-flows, MVCM



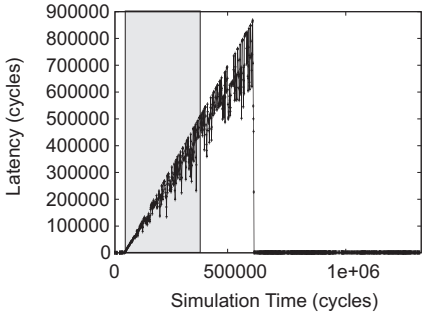
(a_2) High inject, Cold-flows, no CMM



(b_2) High inject, Cold-flows, MVCM



(c_2) High inject, Hot-flows, no CMM

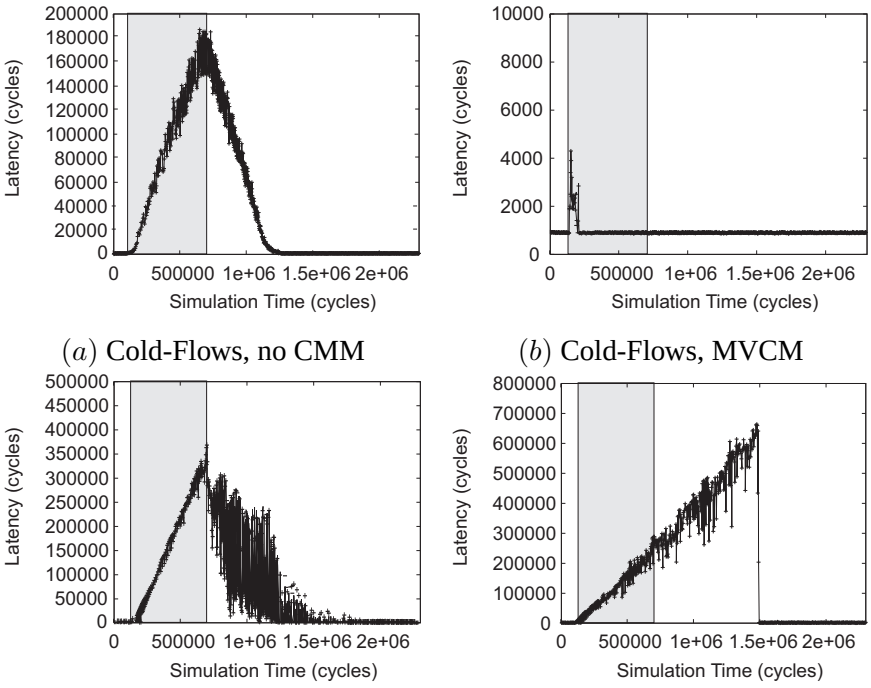


(d_2) High inject, Hot-flows, MVCM

Figure 8.11: Latency for $(a_i)(b_i)$ cold and $(c_i)(d_i)$ hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying (x_1) low and (x_2) high injection rate and a fixed packet size=256 bytes.

3. ***Variable Packet Size***

   Figures 8.12, 8.13(a_1) to (d_1), and 8.13(a_2) to (d_2) present the network performance for the *BMIN Perfect-Shuffle 4-ary 5-fly* when applying medium, low, and high injection rates respectively, with a variable packet size of 64/512 bytes. Notice that the traffic pattern applied is based on distribution of a 50% of 64-byte packets and a 50% of 512-byte packets.



(a) Cold-Flows, no CMM

(b) Cold-Flows, MVCM

(c) Hot-Flows, no CMM

(d) Hot-Flows, MVCM

Figure 8.12: Latency for $(a)(b)$ cold and $(c)(d)$ hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with $(a)(c)$ no CMM and $(b)(d)$ MVCM when applying a medium injection rate and variable packet size 64/512 bytes.

As can be seen, although *cold-flows* experience a greater increase in their latency in Figures 8.12(a), 8.13(a_1) and 8.13(a_2) when no CMM is applied, than for a fixed packet size of 256 bytes due to the congestion process, MVCM continues to reduce the maximum values of latency, thus achieving good results

(Figures 8.12(b), 8.13(b_1) and 8.13(b_2), respectively) by applying its corrective actions, that is, reducing latency for *cold-flows* but allowing the maximum injection rate for *hot-flows* in order to avoid penalizing them in excess. Comparing Figures 8.12(d), 8.13(d_1), and 8.13(d_2), with respect to Figures 8.12(c), 8.13(c_1), and 8.13(c_2), respectively, we can observe that MVCM reduces the oscillations and manages to achieve the best performance for *hot-flows* by maintaining a constant injection of packets belonging to *hot-flows* until the congestion period is totally consumed.

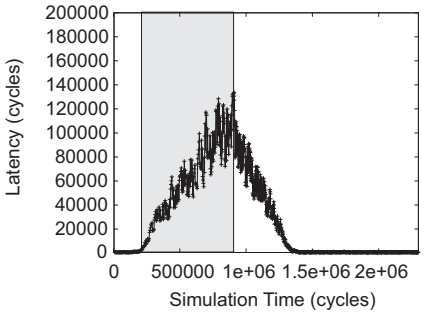
Comparing these results with the ones achieved when applying a fixed packet size of 256 bytes, it can be noticed that although applying a proportionally traffic pattern composed of packets of 64 and 512 bytes, we obtain a small penalization in latency due to the fact that the average packet size is slightly larger (288 bytes) than with a fixed packet size (256 bytes).

After analyzing the MVCM performance for the *BMIN Perfect-Shuffle 4-ary 5-fly*, it is necessary to check if the mechanism behaves in the same way when the network configuration changes. For this study, first we have analyzed the *BMIN Perfect-Shuffle 4-ary 3-fly*, which reduces the number of stages in the interconnection network with respect the *BMIN Perfect-Shuffle 4-ary 5-fly*, and next the *BMIN Perfect-Shuffle 8-ary 3-fly*, which increases the size of the switch radix but maintains the number of stages with respect the *BMIN Perfect-Shuffle 4-ary 3-fly*. These network configurations have been evaluated with low, medium, and high injection rates, and with different packet sizes (fixed and variable) as in the previous analyses. Next, we present results for medium traffic rate with a fixed packet size of 256 bytes. The rest of the combinations behave as expected, achieving similar results and leading to the same conclusions.

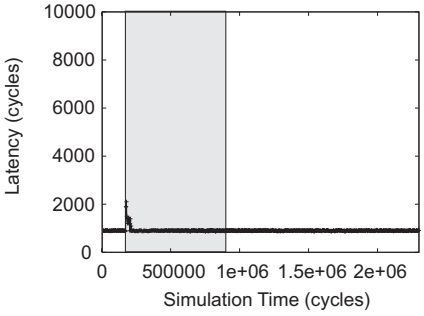
Following the same process of study applied in the previous analysis, Figures 8.14, 8.15(a_1) to (d_1), and 8.15(a_2) to (d_2) represent the results of the network performance when medium, low, and high injection rates are applied in a *BMIN Perfect-Shuffle 4-ary 3-fly*, respectively. In the same way, Figures 8.16, 8.17(a_1) to (d_1), and 8.17(a_2) to (d_2) represent the network performance

for a *BMIN Perfect-Shuffle 8-ary 3-fly* when medium, low and high injection rates are applied, respectively. In particular, the values of the injection rates of 7.2 bytes/cycle, which corresponds to 0.45 bytes/cycle/sw, and 58 bytes/cycle, which corresponds to 0.9 bytes/cycle/sw, have been applied to these network configurations, respectively. All results shown below have been obtained by applying the traffic pattern described in Table 8.1 with a fixed packet size of 256 bytes.

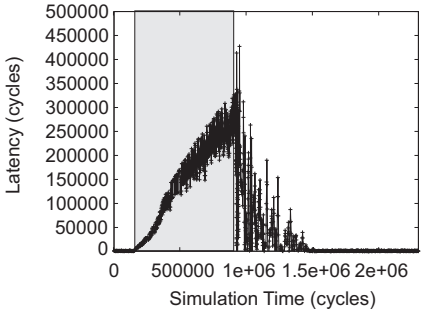
As can be observed, the MVCM mechanism behaves as expected. On the one hand, it reduces latency and palliates the throughput drop for *cold-flows* caused by the congestion situation. On the other hand, although *hot-flows* experience a slight increase in their latency because latencies are calculated since generation time, their performance experience an improvement by eliminating oscillations when congestion ends. Additionally, although the size of the switch radix and the number of stages have changed for these network configurations with respect to the previous one, again the MVCM mechanism behaves as expected.



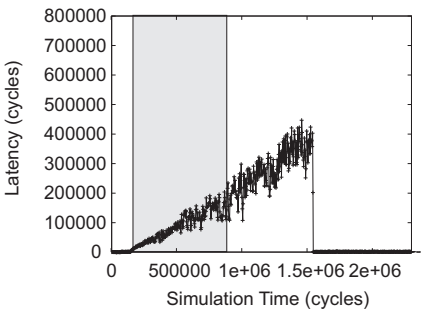
(a_1) Low inject, Cold-flows, no CMM



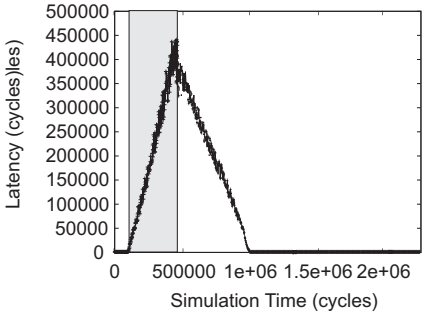
(b_1) Low inject, Cold-flows, MVCM



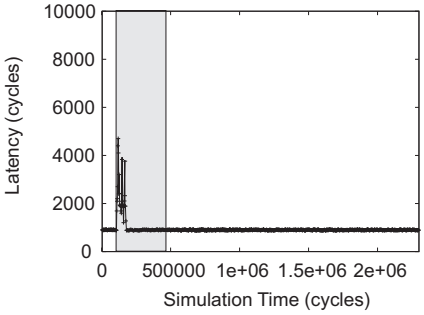
(c_1) Low inject, Hot-flows, no CMM



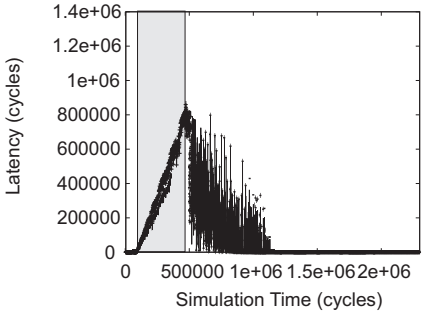
(d_1) Low inject, Hot-flows, MVCM



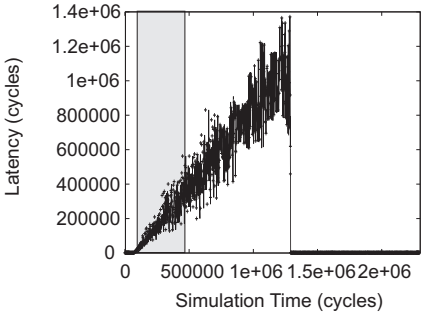
(a_2) High inject, Cold-flows, no CMM



(b_2) High inject, Cold-flows, MVCM



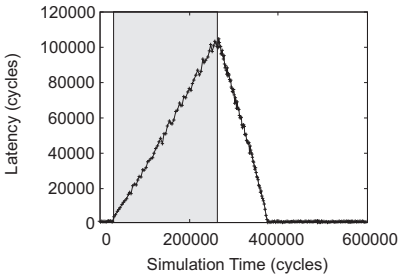
(c_2) High inject, Hot-flows, no CMM



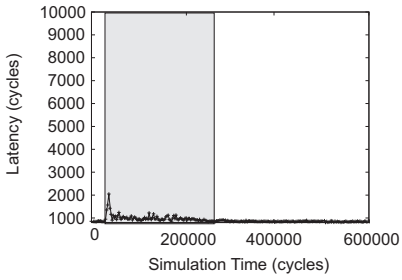
(d_2) High inject, Hot-flows, MVCM

Figure 8.13: Latency for $(a_i)(b_i)$ cold and $(c_i)(d_i)$ hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying (x_1) low and (x_2) high injection rate and variable packet size 64/512 bytes.

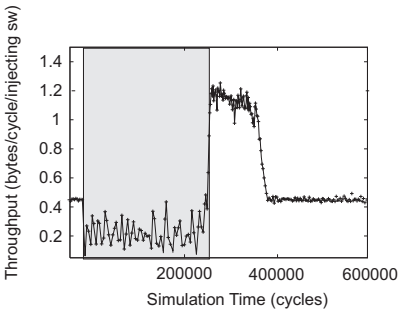
BMIN Perfect-Shuffle 4-ary 3-fly



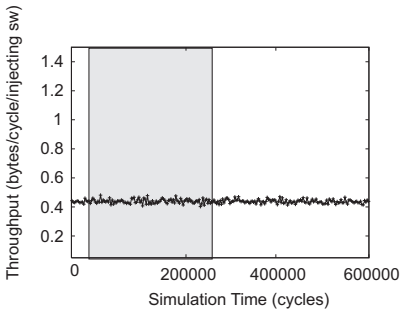
(a_1) Latency, Cold-flows, no CMM



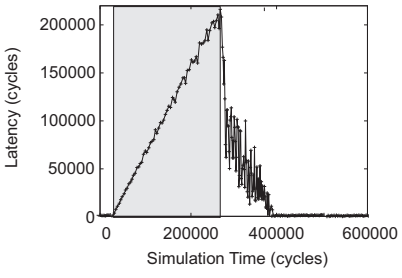
(b_1) Latency, Cold-flows, MVCM



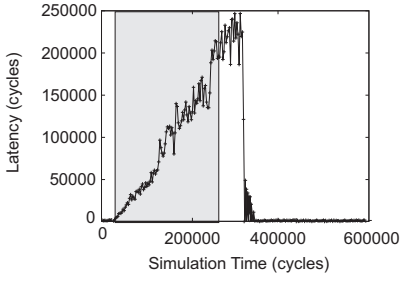
(c_1) Throughput, Cold-flows, no CMM



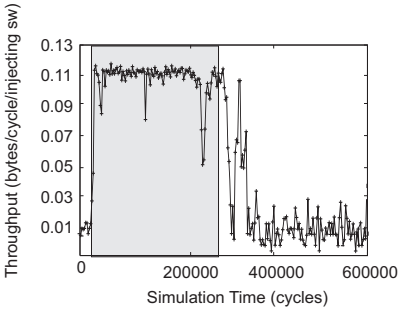
(d_1) Throughput, Cold-flows, MVCM



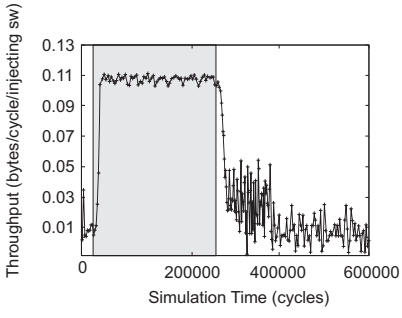
(a_2) Latency, Hot-flows, no CMM



(b_2) Latency, Hot-flows, MVCM

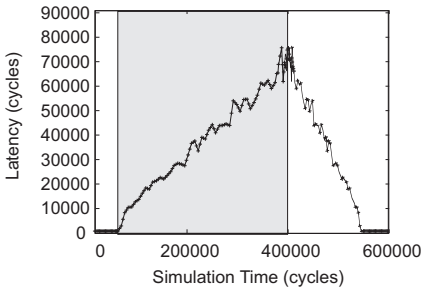


(c_2) Throughput, Hot-flows, no CMM

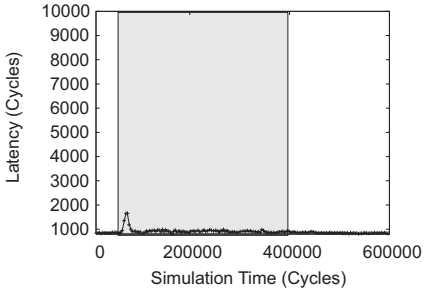


(d_2) Throughput, Hot-flows, MVCM

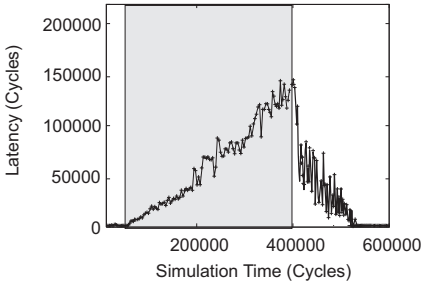
Figure 8.14: $(a_i)(b_i)$ Latency and $(c_i)(d_i)$ throughput for (x_1) cold and (x_2) hot-flows in a BMIN Perfect-Shuffle 4-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying medium injection rate and a fixed packet size=256 bytes.



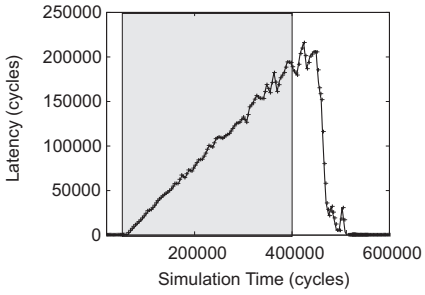
(a_1) Low inject, Cold-flows, no CMM



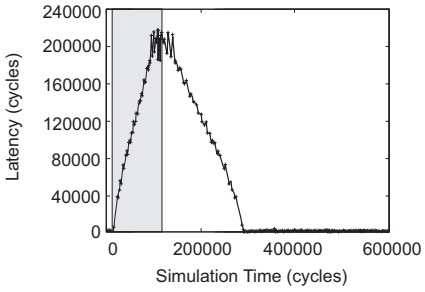
(b_1) Low inject, Cold-flows, MVCM



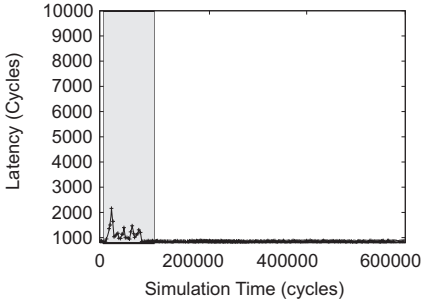
(c_1) Low inject, Hot-flows, no CMM



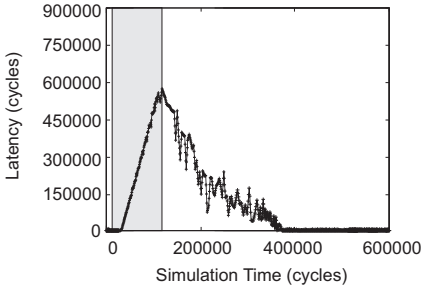
(d_1) Low inject, Hot-flows, MVCM



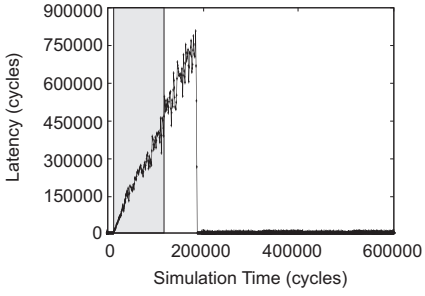
(a_2) High inject, Cold-flows, no CMM



(b_2) High inject, Cold-flows, MVCM



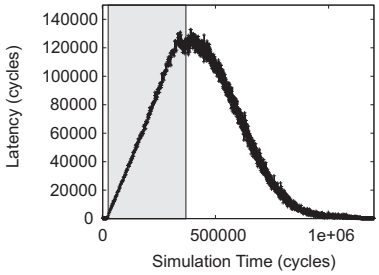
(c_2) High inject, Hot-flows, no CMM



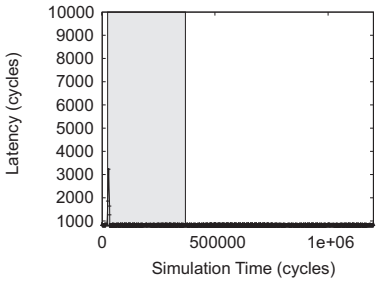
(d_2) High inject, Hot-flows, MVCM

Figure 8.15: Latency for $(a_i)(b_i)$ cold and $(c_i)(d_i)$ hot-flows in a BMIN Perfect-Shuffle 4-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying (x_1) low and (x_2) high injection rate and a fixed packet size=256 bytes.

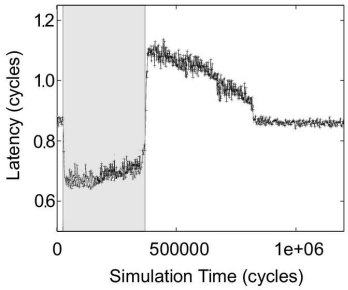
BMIN Perfect-Shuffle 8-ary 3-fly



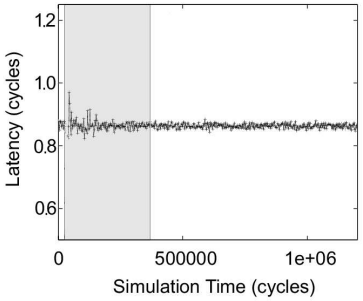
(a_1) Latency, Cold-flows, no CMM



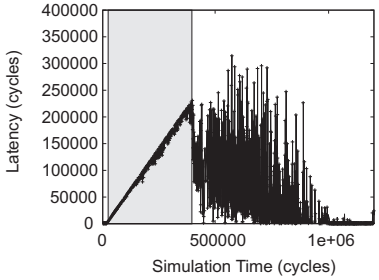
(b_1) Latency, Cold-flows, MVCM



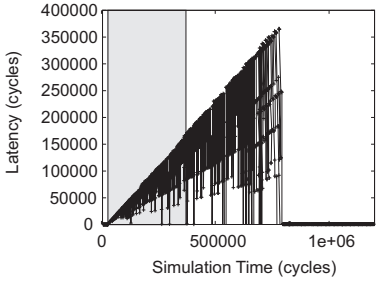
(c_1) Throughput, Cold-flows, no CMM



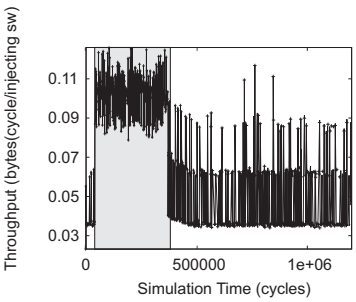
(d_1) Throughput, Cold-flows, MVCM



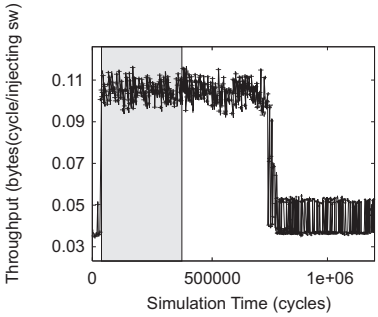
(a_2) Latency, Hot-flows, no CMM



(b_2) Latency, Hot-flows, MVCM

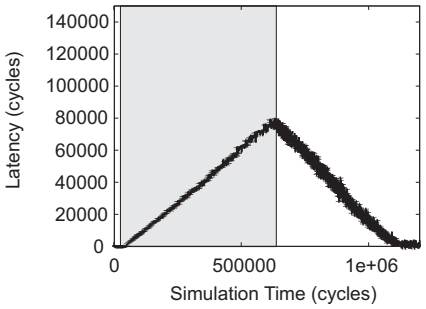


(c_2) Throughput, Hot-flows, no CMM

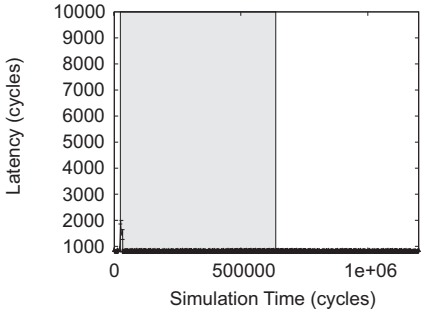


(d_2) Throughput, Hot-flows, MVCM

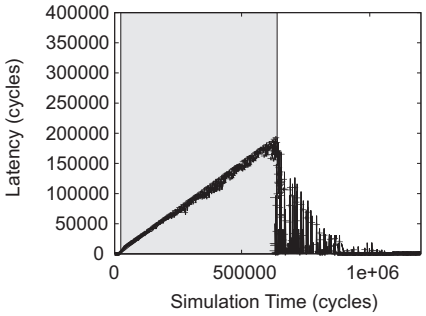
Figure 8.16: $(a_i)(b_i)$ Latency and $(c_i)(d_i)$ throughput for (x_1) cold and (x_2) hot-flows in a BMIN Perfect-Shuffle 8-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying medium injection rate and a fixed packet size=256 bytes.



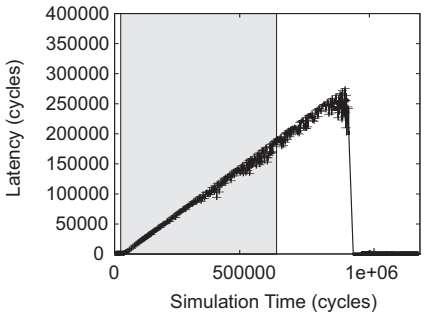
(a_1) Low inject, Cold-flows, no CMM



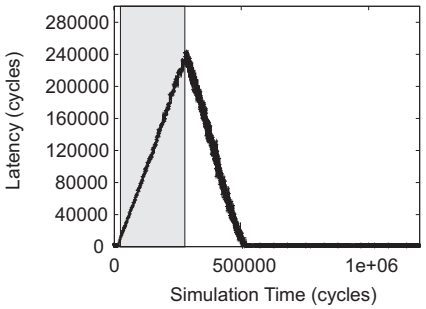
(b_1) Low inject, Cold-flows, MVCM



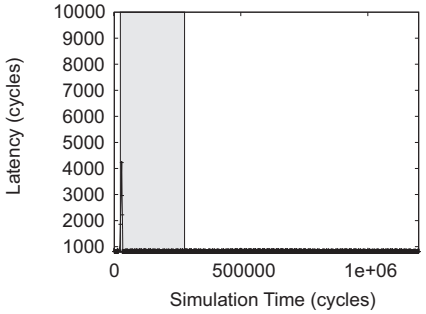
(c_1) Low inject, Hot-flows, no CMM



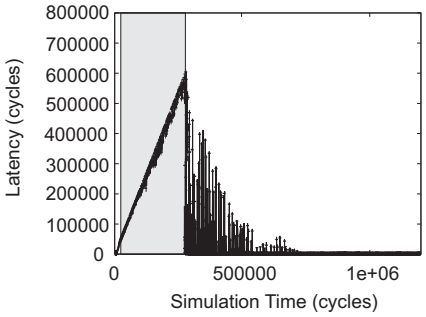
(d_1) Low inject, Hot-flows, MVCM



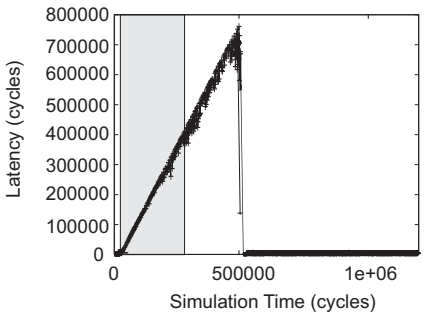
(a_2) High inject, Cold-flows, no CMM



(b_2) High inject, Cold-flows, MVCM



(c_2) High inject, Hot-flows, no CMM



(d_2) High inject, Hot-flows, MVCM

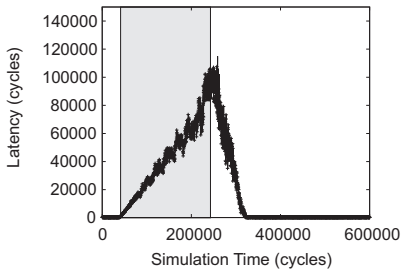
Figure 8.17: Latency for $(a_i)(b_i)$ cold and $(c_i)(d_i)$ hot-flows in a BMIN Perfect-Shuffle 8-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying (x_1) low and (x_2) high injection rate and a fixed packet size=256 bytes.

After analyzing the network performance when applying the MVCM mechanism in a set of bidirectional networks, next, we present some more results for other configurations of multistage interconnection networks in order to validate the MVCM mechanism. In particular, two unidirectional networks have been analyzed, that is, the *UMINs Butterfly 4-ary 4-fly and 8-ary 3-fly*. Once more, these network configurations have been simulated and evaluated under the same traffic conditions applied in the previous analysis, that is, under the synthetic traffic described in Table 8.1 with low, medium, and high injection rates, and with a fixed and variable packet size. Again, as in all cases, the achieved results showed a similar qualitative behavior as the one shown in previous analyzed networks. Next, we only show results for both network configurations when applying a medium injection rate and a fixed packet size of 256 bytes.

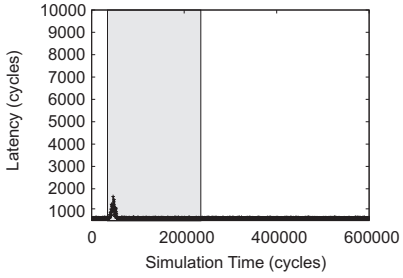
Results for the UMIN *Butterfly 4-ary 4-fly* network are presented in Figure 8.18, whereas Figure 8.19 presents results for the UMIN *Butterfly 8-ary 3-fly* network. As it can be observed, again the MVCM mechanism reacts against the congestion situation in time, that is, reducing the negative effects of congestion over the *cold-flows* and providing a good network performance for both *cold* and *hot-flows* for the two analyzed network configurations.

As it can be seen in this analysis, the MVCM mechanism is an efficient congestion management mechanism, achieving good results with respect to the same situation without applying any mechanism to manage congestion. Additionally, results show that MVCM behaves well for multistage interconnections networks regardless to the network configuration and traffic load.

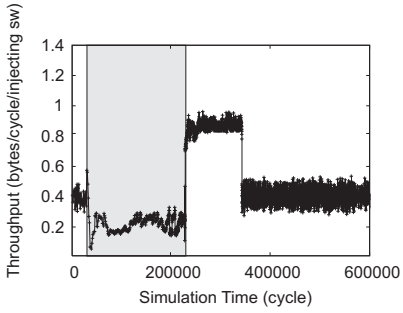
UMIN Butterfly 4-ary 4-fly



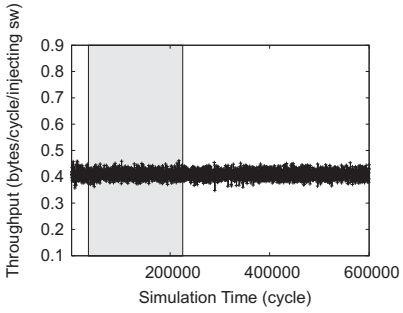
(a_1) Latency, Cold-flows, no CMM



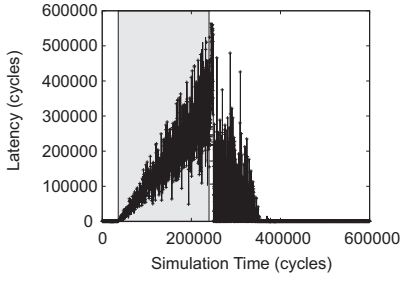
(b_1) Latency, Cold-flows, MVCM



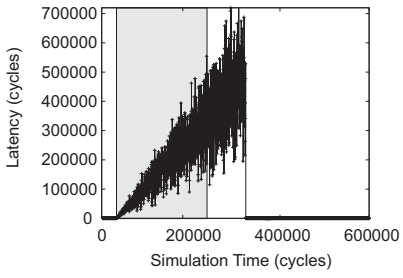
(c_1) Throughput, Cold-flows, no CMM



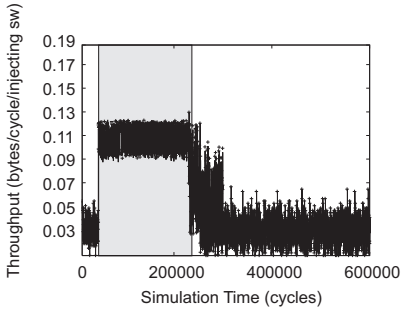
(d_1) Throughput, Cold-flows, MVCM



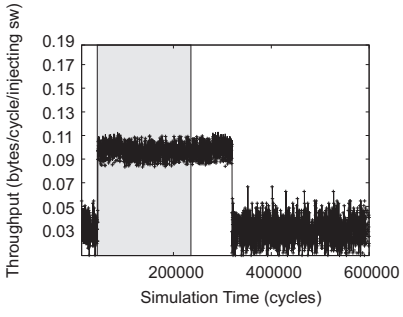
(a_2) Latency, Hot-flows, no CMM



(b_2) Latency, Hot-flows, MVCM



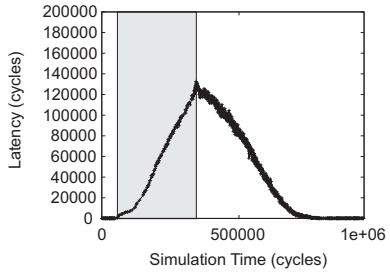
(c_2) Throughput, Hot-flows, no CMM



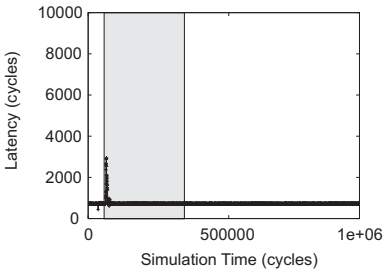
(d_2) Throughput, Hot-flows, MVCM

Figure 8.18: $(a_i)(b_i)$ Latency and $(c_i)(d_i)$ throughput for (x_1) cold and (x_2) hot-flows in a UMIN Butterfly 8-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying medium injection rate and a fixed packet size=256 bytes.

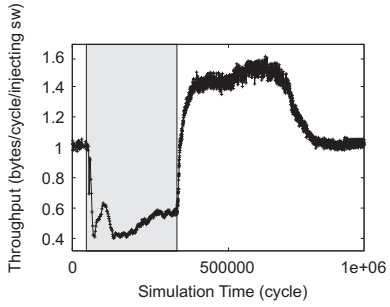
UMIN Butterfly 8-ary 3-fly



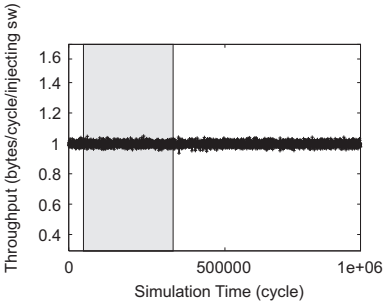
(a_1) Latency, Cold-flows, no CMM



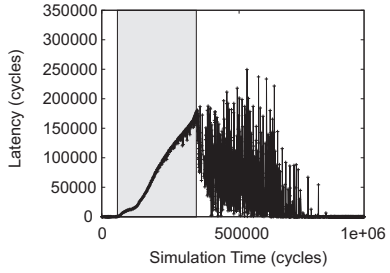
(b_1) Latency, Cold-flows, MVCM



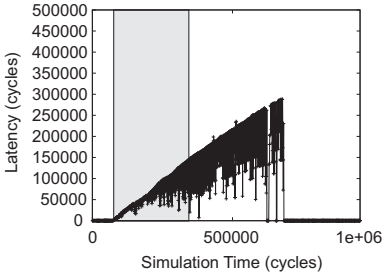
(c_1) Throughput, Cold-flows, no CMM



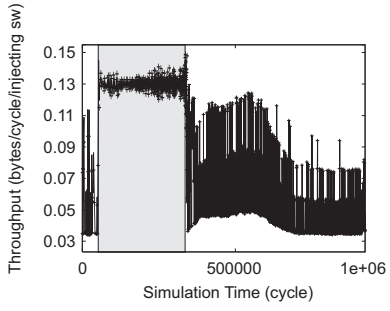
(d_1) Throughput, Cold-flows, MVCM



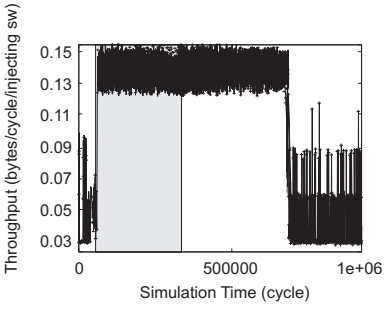
(a_2) Latency, Hot-flows, no CMM



(b_2) Latency, Hot-flows, MVCM



(c_2) Throughput, Hot-flows, no CMM



(d_2) Throughput, Hot-flows, MVCM

Figure 8.19: $(a_i)(b_i)$ Latency and $(c_i)(d_i)$ throughput for (x_1) cold and (x_2) hot-flows in a UMIN Butterfly 8-ary 3-fly with $(a_i)(c_i)$ no CMM and $(b_i)(d_i)$ MVCM when applying medium injection rate and a fixed packet size=256 bytes.

8.3 Comparing Proposals

In order to evaluate whether the new mechanisms improve the network performance in a congestion situation with regard to the current proposals (Renato and Pfister), next, we show a comparative analysis of the congestion management mechanisms as they were originally proposed under the same network configurations and traffic loads. Additionally, we also compare MVCM with respect to the IOCM mechanism. In particular, to compare them we show results for two types of traffic loads. Firstly, a synthetic traffic based on the one shown in Table 8.1 with three different injection rates, that is, low, medium, and high injection rate, and secondly, a traffic based on real traces.

8.3.1 Performance Results for Synthetic Traffic

First of all, a global comparison between the four proposals as they were originally proposed is undertaken. Figure 8.20 shows the average packet latency versus traffic for the four analyzed mechanisms, that is, Renato, Pfister, IOCM, and MVCM in a *BMIN Perfect-Shuffle 4-ary 5-fly* with a fixed packet size of 256 bytes.

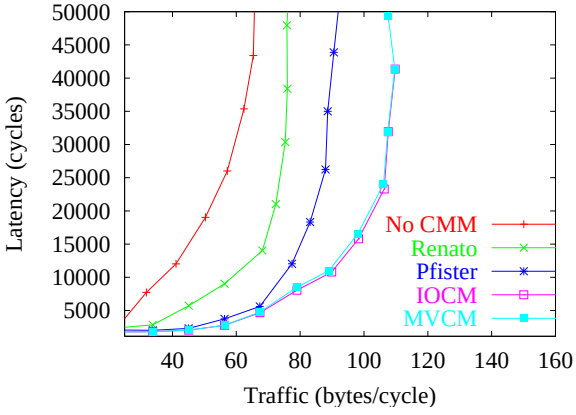


Figure 8.20: Latency vs. traffic for the analyzed CMMs in a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256 bytes.

The four mechanisms significantly improve the results for traffic and latency with

respect to the one achieved when no congestion management mechanism is applied. Among them, as it can be observed, Renato's proposal shows the worst performance. It is because this proposal defines a set of corrective actions based on a static window with a fixed size equal to one in any situation. So, in the absence of congestion, it cannot take advantage of the free network bandwidth because packets have to wait at origins while channels could be idle (i.e. a single origin host injecting packets to a single destination). Moreover, its packet marking strategy based on marking packets only at input buffers seems insufficient to correctly identify the flows responsible for congestion in all situations.

In the same way, although Pfister's implementation achieves better results than Renato's proposal, it does not present the best performance as it also uses a simple packet marking technique based on marking packets only at output buffers with a subsequent simple recovery strategy. Marking packets only at output buffers also appears insufficient to correctly identify the flows responsible for congestion in all situations. Indeed, as this proposal uses only a waiting slots insertion technique in a progressive way, if a hard congestion situation is suddenly reached and, therefore, many waiting slots have been inserted, it cannot recover the initial parameters in a short time while the network is becoming idle. Additionally, since this proposal does not use any window control, it allows packets to be injected without any limit while no congestion is detected. So, it works well with low injection rates, but with high injection rates if marked ACK packets begin to arrive to the origin hosts, they immediately insert waiting slots which can excessively limit the accepted traffic, leading to a decrease of network performance.

On the contrary, both MVCM and IOCM mechanisms present the best performance since they combine a better packet marking technique with a set of effective corrective actions based on a dynamic window and waiting slots insertion. As it was described in previous sections, the dynamic window strategy carries out a first injection control if the congestion situation is not so severe, but if this situation persists, it starts inserting waiting slots between newly generated packets. Additionally, as the window-based technique applied by IOCM and MVCM limits the outstanding

packets per flow in any situation, even without congestion, the technique is indeed preventing a congestion situation before it appears. Notice that the slight improvement shown by IOCM with respect to MVCM is due to the packet marking applied, which allows to mark packets in advance as it was analyzed in section 4 (Figure 4.12). A more detailed analysis about the influence of the applied packet marking technique will be carried out in section 8.5. Anyway, results in Figure 8.20 clarify that, with low injection rates, any of the three mechanisms are able to palliate the effects of congestion, but, with medium and high injection rates, both Renato's and Pfister's mechanisms are not able to achieve the network performance obtained when applying the IOCM or the MVCM mechanisms due to the better corrective actions defined and the improved packet marking scheme applied.

Next, in order to confirm these results, an analysis in depth of how the latency and throughput of the *cold* and *hot-flows* is affected in the presence of congestion when the four mechanisms are applied for the same traffic conditions and with the same network configuration is presented in Figures 8.21 to 8.24. In particular, Figures 8.21 and 8.22 show results for medium injection rate, whereas Figures 8.23 and 8.24 show the performance of each flow type for low and high injection rates, respectively, such as it was described in Table 8.2. Figure 8.21(a_1) shows latency for cold-flow packets when no congestion management mechanism is applied, whereas Figures 8.21(a_2), 8.21(a_3), 8.21(a_4), and 8.21(a_5) show results for *cold-flow* packets when the Renato's, Pfister's, IOCM, and MVCM mechanisms are applied, respectively. As it can be observed, the four mechanisms are able to drastically reduce the maximum values of latency from the 140,000 cycles achieved in Figure 8.21(a_1) until less than 10,000 cycles. But, comparing the four graphs (a_2), (a_3), (a_4), and (a_5), both MVCM and IOCM mechanisms achieve values for latency three and two times lower than the ones obtained with the other mechanisms, that is Renato's and Pfister's, respectively. Additionally, notice that not only are reduced the values for latency, but also the length of the time period required to consume the set of *cold-flow* packets affected by the congestion is shorter when applying any of the two new mechanisms (MVCM or IOCM) than that required by the other ones. To confirm these

results, Figures 8.21(b_1), 8.21(b_2), 8.21(b_3), 8.21(b_4), and 8.21(b_5), which represent throughput when no CMM, Renato's, Pfister's, IOCM, and MVCM mechanisms are applied, respectively, show to what extend the corresponding mechanisms are able to mitigate the throughput drop. Again, both IOCM and MVCM clearly present a more even throughput performance by eliminating almost all oscillations observed when the Renato's and Pfister's mechanisms are applied. Therefore, both IOCM and MVCM mechanisms significantly reduce the negative effects of congestion over the *cold-flows*.

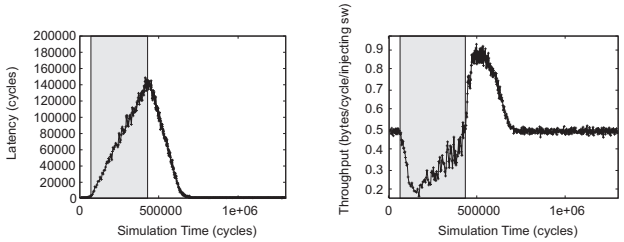
On the other hand, Figures 8.22(a_1) to (a_5) and 8.22(b_1) to (b_5) present latency and throughput for *hot-flows*, respectively. In particular, Figures 8.22(a_1) and (b_1), 8.22(a_2) and (b_2), 8.22(a_3) and (b_3), 8.22(a_4) and (b_4), and 8.22(a_5) and (b_5) present results when no CMM, Renato's, Pfister's, IOCM, and MVCM mechanisms are applied, respectively. As it can be observed, both IOCM and MVCM latency results present a shorter and steadier incline with less oscillations. Renato's results for latency show the longest period to recover the initial situation, that is, the network status before the congestion appears, due to the fact that the application of a static window stops *hot-flow* packet injection for too long. In the same way, Pfister's graph shows a long period to recover the initial network state because many gaps of injection packets appear during the simulation time as a result of the application of the waiting slots insertion technique without any limitation, unlike both IOCM and MVCM do. This negative effect, produced by not applying a correct limitation in the waiting slots technique, will be analyzed in depth in section 8.4.4.
To corroborate these results, we can observe in Figures 8.22(b_1) to (b_5) that the plots that correspond to IOCM and MVCM (b_4) and (b_5) present more stable values during the congestion period, and, when congestion ends, oscillations hardly appear.

Finally, Figures 8.23 and 8.24 show the latency results for low and high injection rates, respectively. In particular, graphs (a_1) to (a_5) represent latency for *cold-flows*, whereas graphs (b_1) to (b_5) represent latency for *hot-flows*.

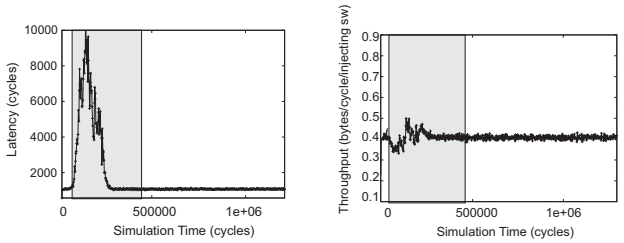
Analyzing the results in Figure 8.23, we detect that for low injection rate the four mechanisms are able to manage congestion, achieving similar results, as expected.

The four mechanisms use some typical well-known techniques to detect and manage congestion, so with low congestion any mechanism is able to detect and manage it in time. On the contrary, when a high injection rate is applied, as shown in Figure 8.24, we observe that, again, both IOCM and MVCM mechanisms are the ones able to manage congestion in the best way. Moreover, we can state that, when a high congestion degree is suddenly reached, the corrective actions applied by both IOCM and MVCM, that is, reducing the dynamic window and later inserting waiting slots, reach the maximum capacity to manage congestion in a short time.

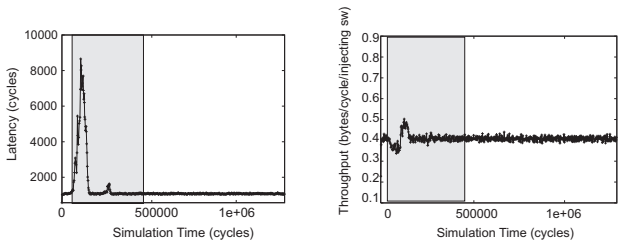
On the other hand, Renato's proposal is not able to manage congestion at all with a high injection rate because it applies a static window with a fixed value of one. So, when corrective actions start to be applied as a consequence of the detected congestion, the mechanism is never able to recover the initial values. This will be analyzed in depth in section 8.4.2. Likewise, Pfister's implementation reduces the maximum value of latency for *cold-flows*, but does not achieve the IOCM and MVCM results neither for latency nor throughput (value not shown in Figure). As a result, we can conclude that both IOCM and MVCM mechanisms improve the performance achieved by previous proposals under the same network configuration and traffic rates.



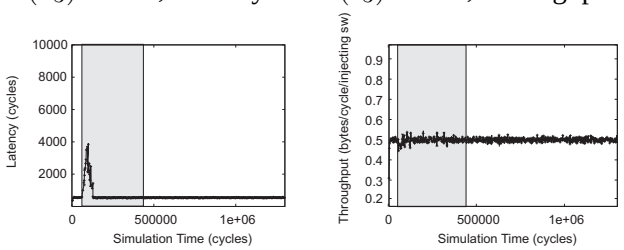
(a_1) No CMM, Latency (b_1) No CMM, Throughput



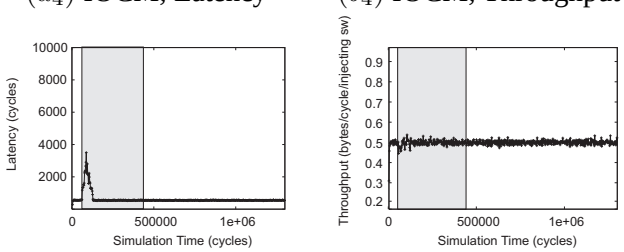
(a_2) Renato, Latency (b_2) Renato, Throughput



(a_3) Pfister, Latency (b_3) Pfister, Throughput

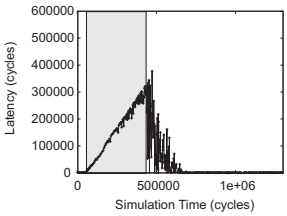


(a_4) IOCM, Latency (b_4) IOCM, Throughput

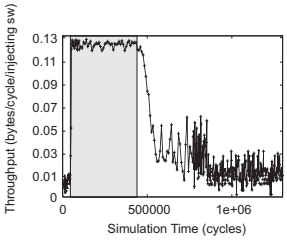


(a_5) MVCM, Latency (b_5) MVCM, Throughput

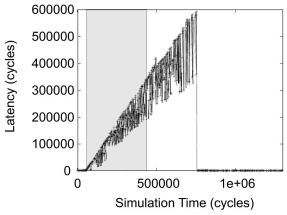
Figure 8.21: (a_i) Latency and (b_i) throughput for *cold-flows* in a BMIN Perfect-Shuffle 4-ary 5-fly with (x_1) no CMM, (x_2) Renato, (x_3) Pfister, (x_4) IOCM, and (x_5) MVCM when applying medium injection rate and fixed packet size=256 bytes.



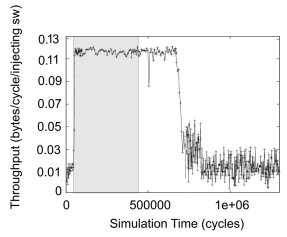
(a_1) No CMM, Latency



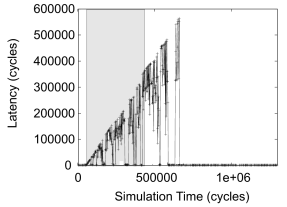
(b_1) No CMM, Throughput



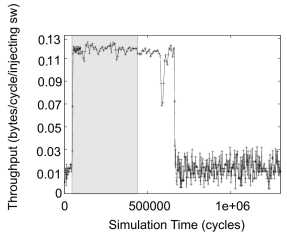
(a_2) Renato, Latency



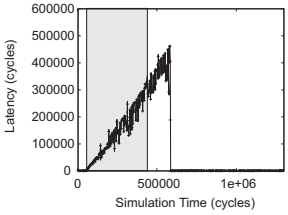
(b_2) Renato, Throughput



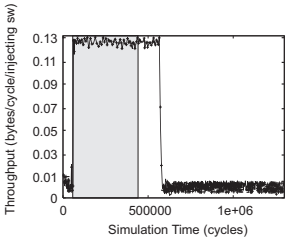
(a_3) Pfister, Latency



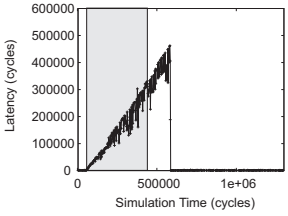
(b_3) Pfister, Throughput



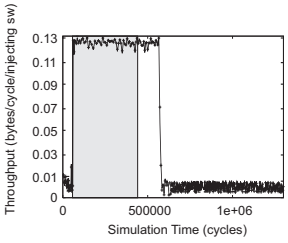
(a_4) IOCM, Latency



(b_4) IOCM, Throughput

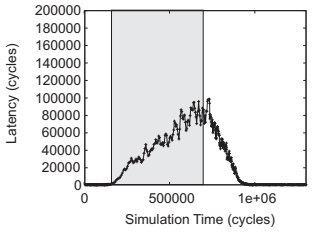


(a_5) MVCM, Latency

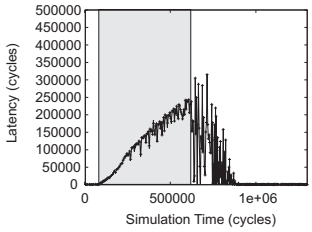


(b_5) MVCM, Throughput

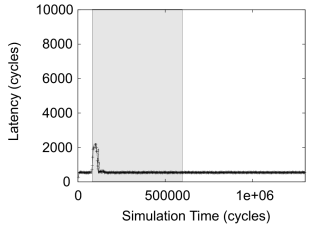
Figure 8.22: (a_i) Latency and (b_i) throughput for hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with (x_1) no CMM, (x_2) Renato, (x_3) Pfister, (x_4) IOCM, and (x_5) MVCM when applying medium injection rate and fixed packet size=256 bytes.



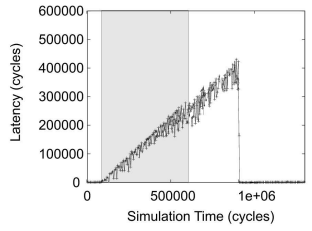
(a_1) No CMM, Cold-Flows



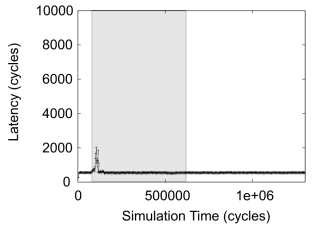
(b_1) No CMM, Hot-Flows



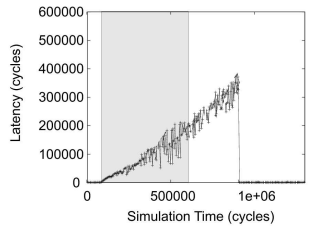
(a_2) Renato, Cold-Flows



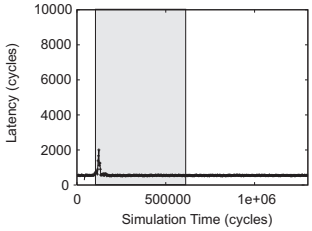
(b_2) Renato, Hot-Flows



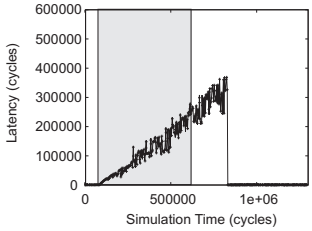
(a_3) Pfister, Cold-Flows



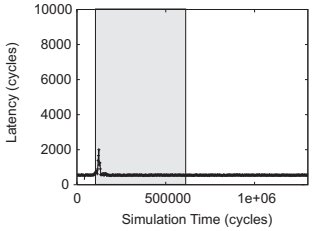
(b_3) Pfister, Hot-Flows



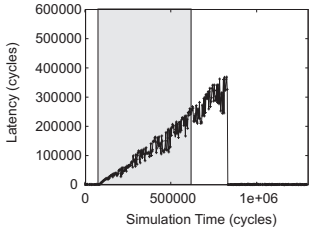
(a_4) IOCM, Cold-Flows



(b_4) IOCM, Hot-Flows

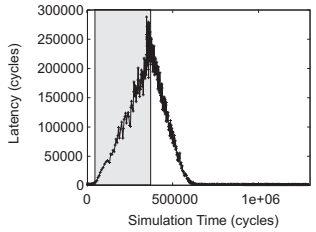


(a_5) MVCM, Cold-Flows

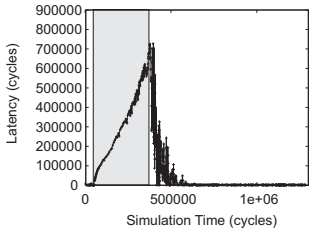


(b_5) MVCM, Hot-Flows

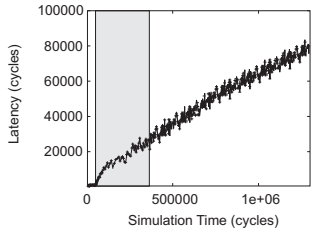
Figure 8.23: Latency for (a_i) cold-flows and (b_i) hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with (x_1) no CMM, (x_2) Renato, (x_3) Pfister, (x_4) IOCM, and (x_5) MVCM when applying low injection rate and fixed packet size=256 bytes.



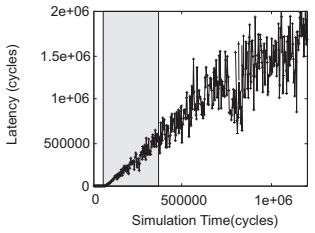
(a_1) No CMM, Cold-Flows



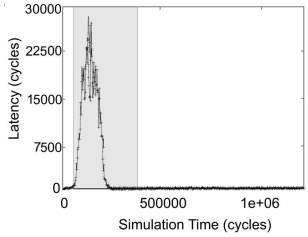
(b_1) No CMM, Hot-Flows



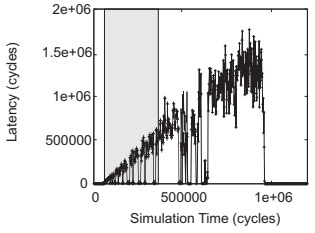
(a_2) Renato, Cold-Flows



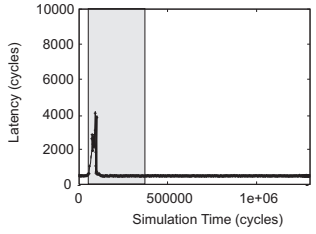
(b_2) Renato, Hot-Flows



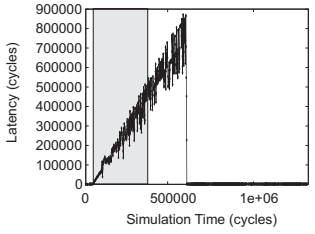
(a_3) Pfister, Cold-Flows



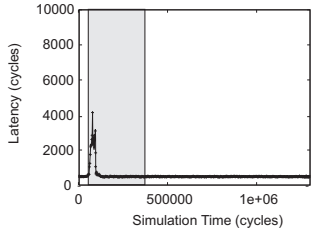
(b_3) Pfister, Hot-Flows



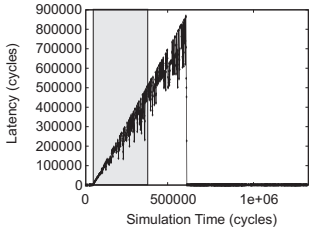
(a_4) IOCM, Cold-Flows



(b_4) IOCM, Hot-Flows



(a_5) MVCM, Cold-Flows



(b_5) MVCM, Hot-Flows

Figure 8.24: Latency for (a_i) cold-flows and (b_i) hot-flows in a BMIN Perfect-Shuffle 4-ary 5-fly with (x_1) no CMM, (x_2) Renato, (x_3) Pfister, (x_4) IOCM, and (x_5) MVCM when applying high injection rate and fixed packet size=256 bytes.

8.3.2 Performance Results for Traces

In previous sections, the newly proposed congestion management mechanisms have been evaluated with several network configurations and under different traffic loads. These traffic loads have been specifically defined to test the new proposals under different possible traffic conditions which create different levels of congestion. As analyzed, both IOCM and MVCM mechanisms react efficiently against congestion, thus presenting good performance results. However, all these results have been obtained by simulating synthetic traffic. Therefore, it is interesting to test the performance of the new mechanism under a traffic load based on real traces.

Simulating real traces involves, firstly, getting the trace files from a real application, and secondly, adapting those files to the simulation environment which evaluates the mechanism. Notice that, although a traffic based on traces comes from a real application, normally the trace files have to be adapted to the simulation environment. That is, fitting the number of hosts and adapting the compression ratio to our needs. Therefore, when applying the resulting files to the simulation scenario, we are not exactly replicating the original behavior.

In our case of study, we have worked with traces from message passing interface (MPI) that allows nodes to communicate in clusters and supercomputers. In particular, the traces used in our study come from the applications DL_POLY [29] and CPMD [21]. DL_POLY is a general purpose software for the study of the classical molecular dynamics process, and CPMD is a software which implements the density functional theory, particularly designed for molecular dynamics. Both traces have been simultaneously run in order to stress the traffic network even more. The process followed was, firstly, to inject the traces from the CPMD application and to keep injecting packets from this pattern throughout the simulation time. Secondly, when the network status became stable, the traces from the DL_POLY application were then injected. From that moment, the network performance is traced till all packets from the DL_POLY files are injected and consumed, as shown in Figure 8.25. Notice that in this case, we globally analyze the flows in the network, instead of analyzing *cold* and *hot-flows* in a separate way. This reproduces a more real scenario in which two

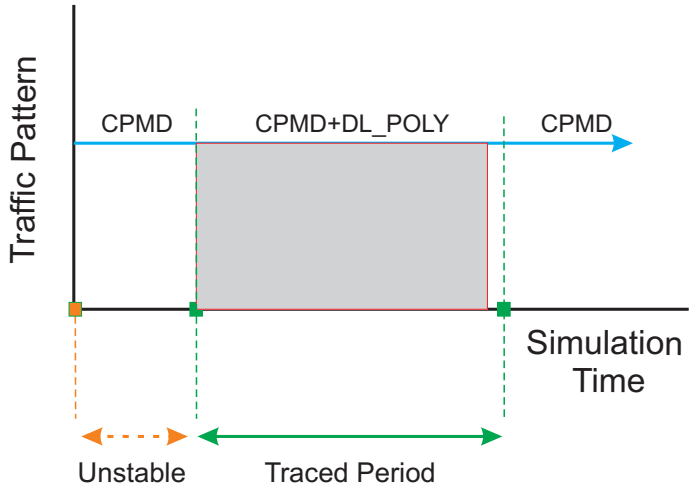
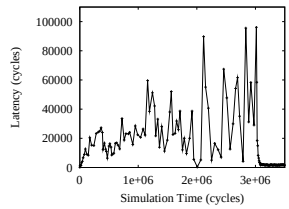


Figure 8.25: Graphic diagram of the traffic pattern based on traces applied.

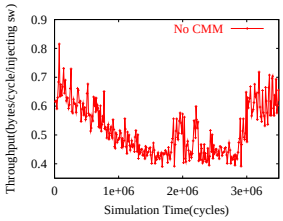
or more applications interact. All the analysis is performed in a *BMIN Perfect-Shuffle 4-ary 3-fly*.

Figure 8.26 presents the network performance when applying the traffic based on traces in each analyzed mechanism. In particular, Figure 8.26 (a_1) to (a_5) shows latency, whereas Figure 8.26(b_1) to (b_5) shows network throughput. As it can be observed, both IOCM and MVCM mechanisms obtain the same results and present the lowest values for latency with respect to the other proposals. In particular, maximum latency value up to 100,000 cycles in Figure 8.26(a_1) is reduced to less than 5,000 cycles in Figures 8.26(a_4) and 8.26(a_5). In particular, it achieves values around 1,000 cycles on average. On the contrary, the other two proposals reduce latency to values around 4,000 cycles on average, reaching peak values near 10,000 cycles. Additionally, as it can be seen, both IOCM and MVCM present less oscillations than the other proposals. These results are corroborated in Figure 8.26(b_1) to (b_5), which shows that, although the four proposals recover the throughput drop caused by congestion, both IOCM and MVCM achieve the best performance.

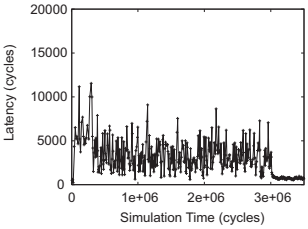
As analyzed, both IOCM and MVCM mechanisms are also able to manage real congestion in a general scenario as that recreated by applying traffic based on traces.



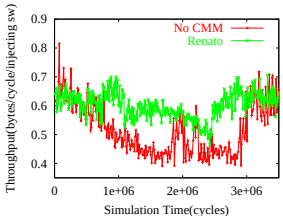
(a_1) No CMM, Latency



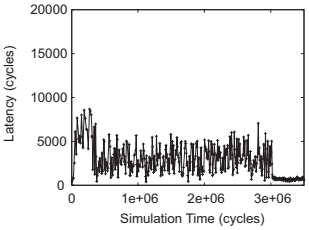
(b_1) No CMM, Throughput



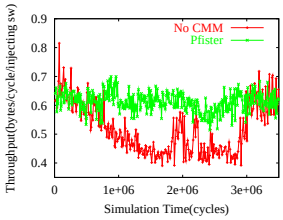
(a_2) Renato, Latency



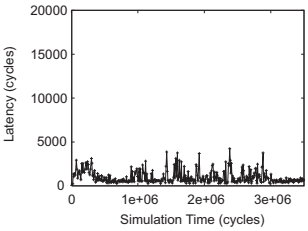
(b_2) Renato, Throughput



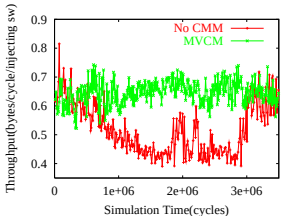
(a_3) Pfister, Latency



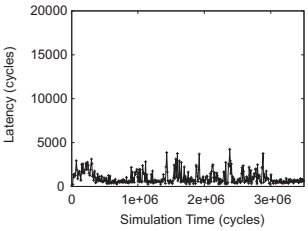
(b_3) Pfister, Throughput



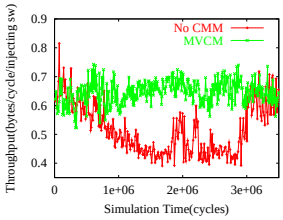
(a_4) IOCM, Latency



(b_4) IOCM, Throughput



(a_5) MVCM, Latency



(b_5) MVCM, Throughput

Figure 8.26: (a_i) Latency and (b_i) throughput when applying traffic based on traces in a BMIN Perfect-Shuffle 4-ary 3-fly.

8.4 Analysis of the Influence of the Techniques Applied by the Evaluated CMMs

In previous sections, the four congestion management mechanisms have been globally evaluated. The network performance have been compared such as they have been originally proposed. Results show the global performance of each mechanism, but they do not separately analyze the impact of the different techniques applied by those mechanisms, that is, the packet marking mechanism and the corrective actions scheme. Therefore, we consider convenient to carry out an exhaustive study [37] analyzing to what extent the performance exhibited by those mechanisms is due to the packet marking strategy or, conversely, it is due to the corrective actions applied.

In the following sections, we present an analysis in depth of the different schemes applied by the evaluated mechanisms in order to assess the impact of each technique on the final performance. In particular, the techniques analyzed are: the different packet marking schemes to detect a congestion situation, the window management schemes, the corrective actions applied to reduce the congestion effects, the waiting slots insertion limitation, and finally the recovery strategies used when congestion ends.

The results presented in the next sections have been achieved for a synthetic traffic pattern (as described in Tables 8.1 and 8.2) in a *BMIN Perfect-Shuffle 4-ary 5-fly* with a fixed packet size of 256 bytes.

8.4.1 Impact of the Packet Marking Scheme

First, we are going to analyze the influence of the packet marking scheme as it is the first mechanism that takes place to detect packet congestion. The five packet marking schemes described at section 4.2 have been separately analyzed, that is, marking packets only at input buffers, referred to as *IPM*, the Renato's packet marking variant, referred to as *RPM*, the Pfister's packet marking based on marking packets only at output buffers, referred to as *OPM*, the input-output packet marking, referred to as *IOPM*, and finally the marking and validating packet marking, referred to as *MVPM*.

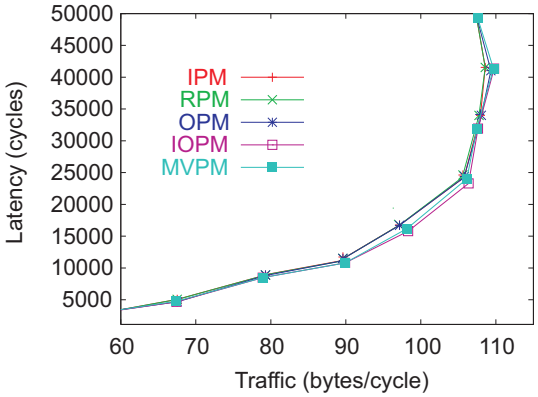


Figure 8.27: Impact of the different packet marking schemes applied in a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256 bytes.

Notice that both *IPM* and *RPM* detect congestion at input buffers but they work in a different way. That is, *IPM* just marks new incoming packets when the occupancy overflows the defined threshold, whereas *RPM* works in a more elaborated way, as described in section 3.5.1.

Figure 8.27 shows a performance comparison of the five different marking schemes, but applying the same corrective strategy in all cases. This way, we can evaluate the impact of the packet marking scheme under the same conditions. In particular, the corrective actions scheme applied is the one used in the MVCM and IOPM mechanisms, since it achieved the best results. Only the packet marking scheme has been varied in order to detect whether the packet marking strategy is or is not the key in the behavior achieved by a congestion management mechanism. Table 8.3 shows how the strategy of corrective actions has been adapted to the different mechanisms based on their packet marking strategy.

As it can be seen, all the packet marking schemes provide similar results. Only with medium and high injection rates, both *IPM* and *RPM*, based on marking packets only at input buffers, present slightly higher values for latency. This is due to the fact that packets are only marked at input buffers after the output buffers are full, so this small delay slightly aggravates the congestion situation before the congestion management mechanism starts to apply the corresponding corrective actions.

Scheme	Marking Actions		Corrective Actions
	Input-Buffer	Output-Buffer	
IPM	1	Not used	**DW + WS**
RPM	1	Not used	**DW + WS**
OPM	Not used	1	**DW + WS**
IOPM	1	0	**DW**
	0	1	**DW**
	1	1	**DW + WS**
MVPM	1	0	**DW**
	1	1	**DW + WS**

Table 8.3: Adapting the strategy of corrective actions to the different packet marking schemes.

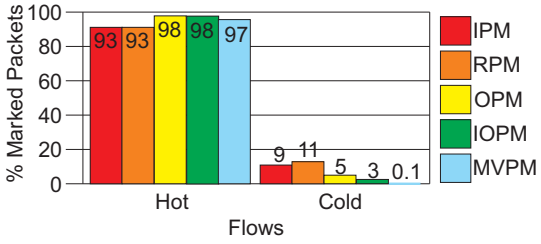


Figure 8.28: Percentage of marked packets for the IPM, RPM, OPM, IOPM, and MVPM strategies.

These results are justified in Figure 8.28, which shows the percentage of marked packets for the different packet marking strategies. As it can be seen, both *IPM* and *RPM* are the ones that achieves the lowest marking values for marking truly hot packets. On the contrary, they have a high marking percentage of cold packets with regard to the other strategies. Anyway, after analyzing the results shown in Figure 8.27, it can be concluded that the impact of the packet marking scheme is not the most important factor determining the global performance of the mechanism. This first conclusion is very important because it determines that if the corrective actions scheme is well-designed, the mechanism will be able to react against a congestion

situation in time, regardless of the packet marking scheme applied.

8.4.2 Impact of the Window Management Scheme

Two different window-based schemes are used in the current congestion management mechanisms for MINs, that is, either a static or a dynamic window. In order to decide which is the better scheme to apply, we will analyze the impact of the window management scheme under the same working conditions. Notice that, in order to correctly evaluate the impact of each scheme, no other corrective actions will be applied to manage congestion, only a window-based technique will be used. In all cases, the MVPM packet marking scheme has been applied, as it was one of the which that showed lower latencies, and it is the native marking scheme described in the MVCM mechanism.

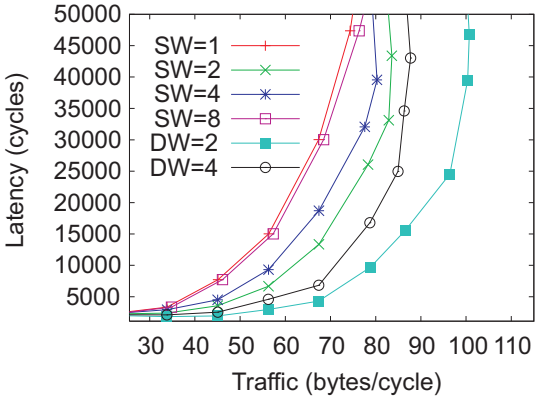


Figure 8.29: Impact of the window management scheme in a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256 bytes.

Figure 8.29 shows the results for the two window management schemes with different window sizes. In particular, curves *SW=1,2,4, and 8* show network performance with a static window scheme for sizes of 1, 2, 4, and 8 packets, respectively, whereas curves *DW=2 and 4* correspond to the application of a dynamic window scheme with maximum values for the dynamic window size of 2 and 4, respectively. As can be observed, both schemes work correctly with low injection rates. In such a

situation, although a congestion situation suddenly appears, the window-based mechanism is enough to palliate the low-congestion effects regardless of the window management scheme applied and the window size defined. However, with medium and high injection rates, the worst results are globally obtained if the congestion management mechanism is based on a static window. In particular, a fixed value of one (*SW=1*) restricts too much the injection rate, preventing pending traffic from being injected, although channels are idle and the network is not congested yet. Additionally, although the size of the static window increases, it will never achieve similar results to those obtained with a dynamic window.
An interesting fact is that the best results with a static window are achieved with a *SW=2*. Higher values decrease network performance until that achieved by *SW=8*. From this window size upwards, larger values of static window do not change results (curves not shown). Notice that a static window will not reduce injection when a congestion situation is detected, therefore network will reach saturation and network performance will decrease. Additionally, with a larger window size, the problem gets worse.

On the other hand, the best results are achieved when applying a dynamic window technique with an initial value of two (*DW=2*). If a larger value is used, we continue to obtain better results than with a static window scheme, but worse than with a *DW=2*. This is because with a value of four (*DW=4*) if congestion appears, as packet latencies increase, too many packets continue to be injected while ACKs are not received, which will take more time to reach the origin hosts as a consequence of the arisen congestion. In this situation, a larger dynamic window value will not be useful when the network begins to be congested. So, as can be seen, the best value for this network is two (*DW=2*).

As shown in both schemes (dynamic or static window), a window size of two is the best option. This value depends on the RTT of the network, as previously calculated in section 5.3.1. In the analyzed network, when network performance is near the saturation point, the RTT is equal to the time needed to inject 1.4 packets, approximately, for a fixed packet size of 256 bytes. This is the ideal size of the

window regardless of the window scheme chosen, as it is the minimum size that is big enough to avoid bubbles in packet sending. Therefore, a value of 2 will be chosen as it is the nearest integer. This justifies why a SW=2 or a DW=2 achieve the best results in their particular window schemes and corroborates the theoretical window size value obtained in section 5.3.1.

Next, in order to confirm why a static window with a fixed value of one (as Renato's mechanism) is a bad option to manage congestion, we represent in Figure 8.30 latency versus simulation time for *cold-flows* when applying only a static window scheme to palliate the effects of a congestion situation. No other corrective actions have been used, only a fixed-size window with different values (SW=1, 2, and 4) along the entire simulation. Figure 8.30 shows the performance for a *BMIN Perfect-Shuffle 4-ary 5-fly* when applying the synthetic traffic pattern shown in Table 8.1 with a high injection rate and a packet size of 256 bytes. Notice that the gray band represents the time interval while packets toward the hot-spot are generated.

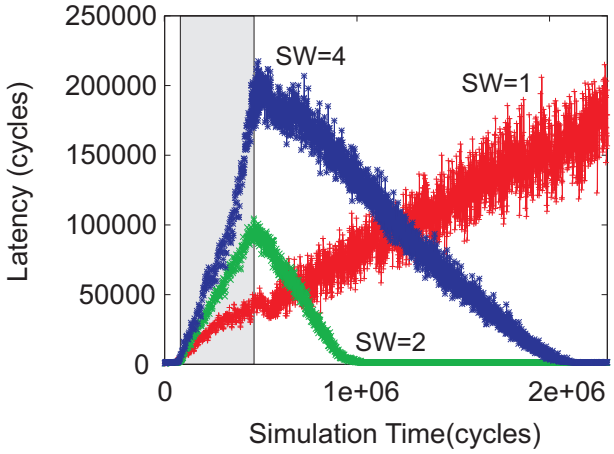


Figure 8.30: Latency since generation time for cold-flows with different SW sizes in a BMIN Perfect-Shuffle 4-ary 5-fly with a high injection rate and a fixed packet size=256 bytes.

We can observe that, if the window size is equal to 1 (graph SW=1) lowest values of latency are achieved during the congestion period. However, once the congestion

situation decreases, packet latencies continue increasing because a window size of 1 does not allow all packets accumulated at origin hosts during the congestion period to be consumed later. Note that latencies are calculated since generation. As we can see, a window size of 2 (SW=2) provides the best performance. Notice that when the network is not saturated, a larger window size provides no advantages since the ACKs arrive at the origins much before 2 packets can be injected into the network. However, as the network becomes saturated, packet contention arises and packets stay longer in the switch queues, increasing the RTT and delaying ACKs. In this situation, a larger window size allows more packets to be in transit through the network, thus leading to an even more congested situation. In Figure 8.30, the graph with the window size equal to 4 (SW=4) confirms this behavior.

As a result of that, we can conclude that instead of applying a static window with a fixed value of one in any situation, as Renato's proposal does, the window mechanism has to be initialized with the correct value based on the RTT of the network.

8.4.3 Impact of the Corrective Actions Scheme

Both IOCM and MVCM mechanisms apply a set of corrective actions based on a combination of a window-based mechanism and the waiting slots insertion. On the contrary, the other current proposals only use one of these strategies. Therefore, it is interesting to find out which strategy is the best to get good results. To this end, we present in Figure 8.31 results for three different options analyzed. The *+DW+WS* option identifies the original combination of corrective actions applied by both IOCM and MVCM mechanisms which use a combination of both methods, that is, dynamic window and waiting slots. The *+DW-WS* represents the use of a dynamic window as a corrective action with the corresponding window size based on the RTT but without any waiting slots insertion technique. Finally, the *-DW+WS* option identifies the use of the waiting slots insertion without any window strategy. Results shown in Figure 8.31 correspond to network performance when applying the three different options with the same packet marking strategy (MVPM), in a *BMIN Perfect-Shuffle 4-ary 5-fly*.

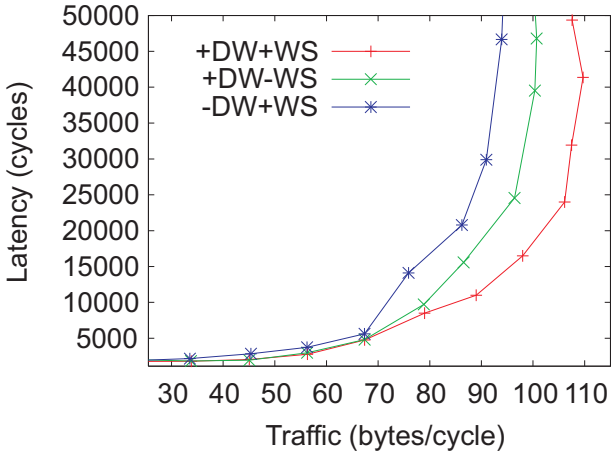
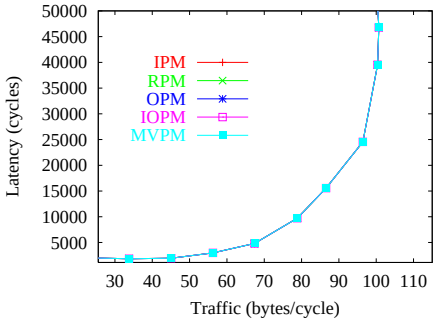


Figure 8.31: Impact of the corrective actions scheme for a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256.

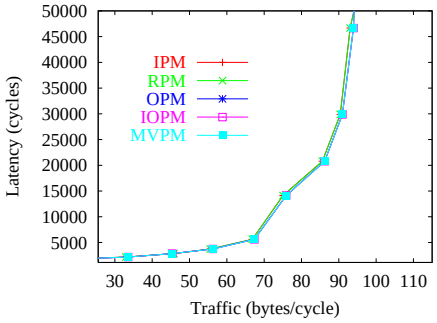
As can be seen, with low and even medium injection rates, any of the two options (+DW+WS and +DW-WS) are able to manage congestion in a good way. However, although the option -DW+WS seems to present similar results, it presents a significant increase in latency with respect to the two other combinations.

On the other hand, with high injection rates the best results are achieved when both techniques (dynamic window and waiting slots) are jointly applied instead of working alone.

On the contrary, when applying only the waiting slot technique without any window-based mechanism, we get the worst results. This is because hosts inject packets without any limitation until congestion is detected. So, when congestion is detected and origins are warned about this problem, traffic network is so collapsed that the congestion management mechanism is not able to react in time. It is already too late to react. On the other hand, when applying a dynamic window, at least this scheme limits the injection rate to the maximum value defined by the dynamic window size and according to the network configuration. So, if the value is correctly calculated, that is, based on the RTT, hosts only proportionally inject the number of packets per flow that the network can deliver in the best case. So, if more packets are generated,



(a) With only a DW technique



(b) With only a WS technique

Figure 8.32: Impact of applying corrective actions based on (a) a DW or (b) a WS technique for a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256 bytes.

they will be stopped at origins instead of making worse the congestion situation in the network.
It is an interesting conclusion because it confirms that the window-based technique correctly sized is key to manage congestion.

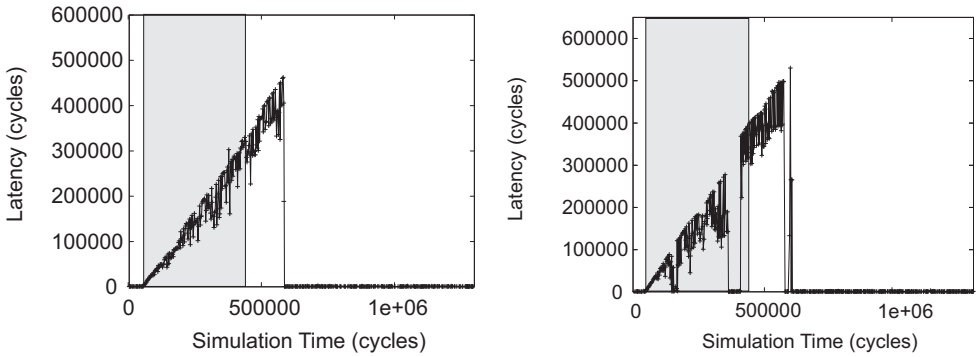
Next, in order to check what happens when applying only corrective actions based on a DW or WS with the different packet marking mechanisms, Figure 8.32 shows network performances when applying (a) only corrective actions based on a DW and (b) corrective actions only based on inserting WS. Both Figures 8.32(a) and 8.32(b) show the network performances when applying the different packet marking mechanisms (IPM, OPM, RPM, IOPM, and MVPM.)

As it can be seen, both figures show similar results to that previously presented in Figure 8.31. A simple window-based mechanism based on a DW, provides better results than a technique based only on inserting WS. These results corroborate the idea that the set of corrective actions defined is the most important strategy in a congestion management mechanism from the overall performance point of view. Anyway, an interesting conclusion is that the corrective action scheme is the one which contributes to achieve a good global throughput by a CMM, whereas the packet marking scheme improves the results by reducing the number of cold packets marked as it was observed in Figure 8.28.

8.4.4 Impact of the Waiting Slot Insertion Limitation

In this section, we check whether the waiting slot insertion limitation technique imposed by both the IOCM and MVCM proposals contribute to avoid penalizing the *hot-flow* performance, as it was analyzed in section 5.3.2. To this end, we analyze what happens if no limitation on inserting waiting slots is defined. As was shown in section 4.4.3, it is mandatory to limit the number of inserted waiting slots according to the number of network stages in a multistage interconnection network. Applying the waiting interval calculation without any limit causes a reduction in the packet injection rate in excess, stopping packets at origins beyond it is required, while the network is becoming idle. To justify this behavior, let us analyze what happens when no limitation in the waiting slots insertion technique is defined in the original MVCM mechanism.

Figure 8.33 presents the average latency of packets belonging to the *hot-flows* when the MVCM mechanism is applied in a *BMIN Perfect-Shuffle 4-ary 5-fly* for a medium injection rate (a) with a limitation in the waiting slots insertion method and (b) without limitation.



(a) WS insertion limitation (b) No WS insertion limitation

Figure 8.33: Analysis of the impact of the waiting slots insertion limitation over hot-flows for a BMIN Perfect-Shuffle 4-ary 5-fly when applying a medium injection rate.

As can be observed, some gaps appear as a consequence of increasing excessively

(more than n-times) the number of waiting slots. Notice that, for this network configuration (4-ary 5-fly), the value is $n = 5$. This is because, despite the fact that there is available enough network bandwidth, too much packets belonging to *hot-flows* are stopped at the origin hosts because the waiting interval applied is too long. Notice that this only occurs if MVCM inserts waiting slots without any limitation.

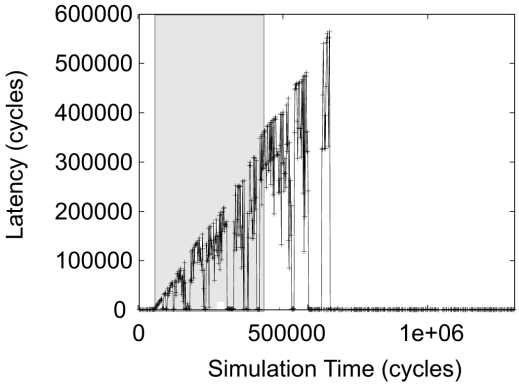


Figure 8.34: Latency for hot-flows when applying the Pfister's implementation in a BMIN Perfect-Shuffle 4-ary 5-fly when applying a medium injection rate.

Additionally, we present in Figure 8.34 the *hot-flow* performance when applying Pfister's implementation on the same network configuration and traffic load. Analyzing Figure 8.34, we can observe that Pfister's implementation suffers from this problem because, unlike MVCM, this mechanism applies the waiting interval calculation strategy without imposing any limitation in the waiting slots insertion as does occur with MVCM. Therefore, with a suddenly high congestion, origins could receive a large number of marked ACK packets in a short period of time and, as a consequence, they would introduce too much waiting slots, extending the waiting period beyond it is necessary, that is, more than the maximum number of stages in the network. Comparing Figures 8.34 and 8.33(b), it can be observed that there are some differences. First of all, with MVCM the injected hot packets are consumed sooner than when applying the Pfister's implementation. Additionally, the maximum reached values of latency with MVCM are lower than with Pfister's implementation. Secondly, as MVCM applies a DW technique combined with the WS technique, al-

though no limitation is applied for the WS insertion technique, a more steady and constant consumption of hot packets is presented.

This analysis corroborates that the insertion of waiting slots is a good strategy to palliate congestion, but the key of this strategy lies in properly limiting the number of waiting slots insertion, as theoretically described in section 5.3.2.

8.4.5 Impact of the Injection Rate Recovery Scheme

As seen in previous sections, the set of corrective actions applied by both the IOCM and MVCM mechanisms is well-defined and, therefore, it is an efficient strategy to react against the negative effects caused by a congestion situation, regardless of the network and traffic conditions. But now, it is interesting to check if the recovery scheme applied is also an efficient strategy when congestion disappears. As described in section 6.2.2, the MVCM mechanism applies what we call an *Immediate Recovery* scheme. Basically, this scheme is intended to recover the initial values for the dynamic window and waiting interval as soon as the congestion is reduced even although no ACKs are received. In particular, both parameters will be immediately set to their initial values if an origin host injects a packet into an empty injection queue. Other proposals use what we call the *Progressive Recovery* scheme. Basically, it only depends on the ACK packet reception or on a timer which times out and then the initial values are recovered. Notice that the *Immediate Recovery* technique is complementary and not replaces the *Progressive Recovery*, that is, *Immediate Recovery* is equal to *Progressive Recovery* plus *Reset Parameters*.

We have analyzed both options in order to decide which one works better when recovering from a congestion. To this end, two configurations have been evaluated, the original scheme defined and applied by the two proposed mechanisms (IOCM and MVCM), called *Immediate Rec*, and the one used in most of the current proposals, referred to as *Progressive Rec*. To evaluate the impact of both recovery schemes, we have used the MVCM mechanism as originally proposed for the *Immediate Rec* option, and the MVCM mechanism but with a progressive recovery scheme based only on the ACK packet reception to recover the initial values, as the other current

proposals apply, for the *Progressive Rec* option. Figure 8.35 shows the impact of applying both schemes.

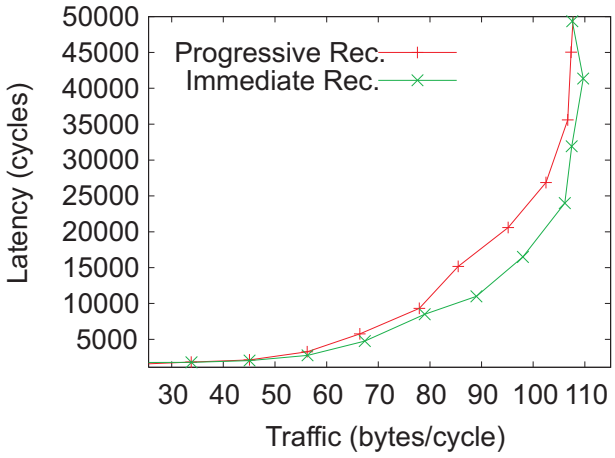


Figure 8.35: Impact of the recovery scheme for a BMIN Perfect-Shuffle 4-ary 5-fly with a fixed packet size=256 bytes.

As can be observed, for low injection rate there is no difference in their performances. In this case, the congestion is not so heavy and the reception of ACK packets, which warn of the non-congestion situation when congestion decreases, is enough to recover the initial network status. However, with a medium and high injection rate, the packet generation is so fast that the ACK packet reception rate is not enough to recover the injection rate quickly when congestion decreases, and therefore, to achieve the best performance. The scenario is as follows. There is no generated packets waiting at the injection queues to be injected, but there is still restrictions (waiting slots) that are being applied until non-marked ACK packets arrive at origins and progressively release them. However, because of the low packet generation, the number of ACK packets will be also small, so the recovery period will be unnecessarily enlarged. On the contrary, when applying the *Immediate Rec.* option, if a generated packet is stored in an empty queue, the initial parameters are immediately recovered. Notice that when network is reaching the saturation point, the performance of both options converge.

8.5 MVCM versus IOCM

Although section 8.4.1 showed that the impact of the packet marking scheme is limited in the overall behavior of a congestion management mechanism, provided that the mechanism has a well (structured) set of corrective actions, next we present a particular analysis about the two congestion management mechanism, the IOCM and the MVCM, which presented slight differences in latency in Figure 8.20 due to the packet marking schemes used in those proposals, the IOPM and the MVPM, respectively.

Figure 8.36 shows the performance of both proposals, the MVCM and the IOCM, with their corresponding packet marking schemes in a *BMIN Perfect-Shuffle 4-ary 5-fly* from low injection rate until saturation. In particular, this figure zooms on Figure 8.20 in order to show the slight improvement presented by the IOCM mechanism.

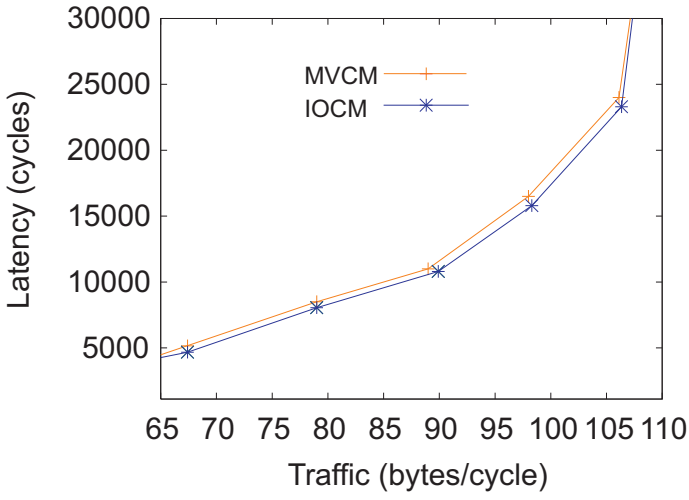


Figure 8.36: Latency vs. traffic for the MVCM and IOCM mechanisms in a BMIN Perfect-Shuffle 4-ary 5-fly when applying a fixed packet size=256 bytes.

The improvement produced by IOCM with respect to MVCM is due to the early packet marking at output buffers carried out by IOPM. This is because output buffers fill up earlier than input buffers do when congestion appears, and packets can be marked at output buffers (MBout=1) regardless of the value of MBin, as commented in section 6.3.1. This way, the IOCM mechanism can apply corrective actions slightly

earlier. On the contrary, with the MVPM scheme, packets will not be validated (VB=1) at output buffers until they have been marked (MB=1) at input buffers. So, MVCM could introduce a small delay on applying corrective actions.

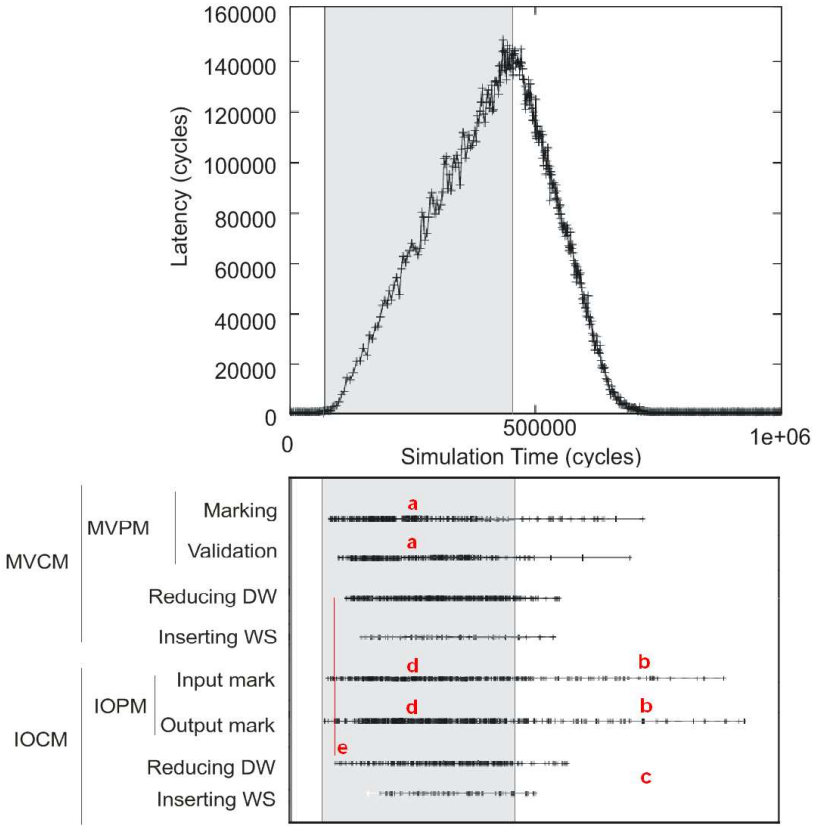


Figure 8.37: Analysis of the corrective actions applied by MVCM and IOCM in a BMIN Perfect-Shuffle 4-ary 5-fly when applying a medium injection rate and a fixed packet size=256 bytes.

To justify this small improvement, Figure 8.37 shows the corrective actions taken over the flows responsible for congestion by both MVCM and IOCM mechanisms.

In particular, the top graph shows the latency for *cold-flows* in a *BMIN Perfect-Shuffle 4-ary 5-fly* under a medium injection rate and a fixed packet size of 256 bytes without any congestion management mechanism. On the other hand, graph below

shows the packet marking process and the corrective actions carried out by both mechanisms.

It can be observed that, when applying the MVPM packet marking strategy, most of the marking and validation actions are carried out during the time the hot-spot pulse is being applied (a). On the contrary, the IOPM continues to mark packets at input and output buffers beyond the end of the pulse (b), due to the wake created by the pulse. However, as it can be seen, these marking actions do not have too much effect over the corrective actions because the first injected packet to an empty queue will recover the initial values for the mechanism parameters (c). Anyway, notice that most of the marking actions are also carried out during the hot-spot pulse (d), as expected.

As a result of the marking actions, both mechanisms apply corrective actions. However, as it can be seen, IOCM starts to apply the dynamic window reduction (e) before the MVCM does. This is because the IOCM strategy allows the value of the dynamic window to be reduced at both input and output packet marking without any delay caused by a dependency between marking actions. So, this marking independence allow us to apply corrective actions slightly sooner.

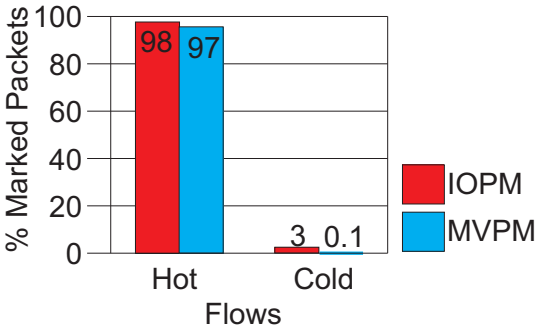
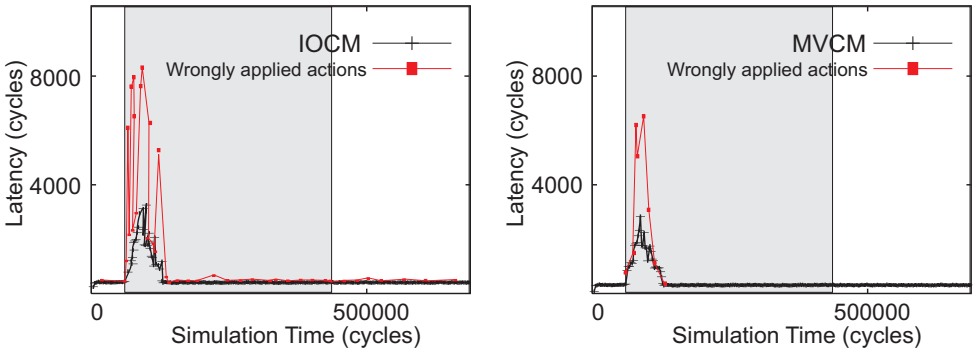


Figure 8.38: Percentage of marked packets for the IOPM, and MVPM strategies.

On the other hand, as it was analyzed in section 4.2.5, although IOPM detects and marks packets slightly sooner than MVPM does, this packet marking strategy causes a higher percentage of wrongly marked packets, as it can be seen in Figure 8.38, because its accuracy is lower than that of MVPM. As a result of wrongly marking some packets, corrective actions could be applied over their flows and, it could pe-

nalize *cold-flow* performance. In particular, if these wrongly marked packets belong to a critical application, IOPM could introduce a negative effect in the application runtime.

In order to analyze the effect of the wrong packet marking on the latency of the *cold-flows*, Figures 8.39(a) and 8.39(b) show cold packet latency when applying IOPM and MVPM, respectively. In particular, each figure presents a graph in black, which represents latency for *cold-flow*, and a set of red dots, which represent latency for packets with wrongly applied actions, as a result of wrongly marked packets. As it can be seen, Figure 8.39(a) presents a set of packets with wrongly actions at the beginning of the pulse as a result of the early packet marking at both input and output buffers. On the contrary, Figure 8.39(b) presents less wrongly actions because of its well-structured and more accurate packet marking strategy based on validating packets at output buffers after previous packet marking at input buffers.



(a) IOCM (b) MVCM

Figure 8.39: Latency for packets with wrongly applied actions when applying (a) IOCM and (b) MVCM in a BMIN Perfect-Shuffle 4-ary 5-fly with a medium injection rate and a fixed packet size=256 bytes.

8.6 Avoiding the Head-of-Line Blocking at Origins

In previous sections, it has been shown that the MVCM mechanism achieves better results than the other proposals due to the application of a thorough packet marking strategy combined with an efficient set of corrective actions. However, as it was seen in section 6.4, a FVOQ technique has been assumed at the NICs of the source hosts in all the simulations, in order to eliminate the HOL blocking phenomenon at the origins, and therefore to correctly evaluate the proposed congestion management mechanism. In spite of this, a FVOQ technique is not scalable. Therefore, in order to improve the scalability of the mechanism, two alternative structures, that is, a Partial Virtual Output Queue (PVOQ) and a Shared-Buffer (SB), were defined in section 6.4. To find out if the MVCM mechanism with the proposed structures (PVOQ or SB) obtains the same performance as the one achieved with a FVOQ, next we present the evaluation and analysis of the two alternative structures.

In what follows, first we evaluate the MVCM mechanism with the same packet marking and corrective actions scheme, given that it has been shown to be quite effective, but applying the two alternative structures. Next, we present a study of the implementation cost of both proposed structures, in order to find out if they contribute to lower the cost in a significant way with respect to the FVOQ technique.

To evaluate the new storage structures at the NICs of the source hosts, we have defined two different synthetic traffic patterns. These traffic patterns are intended to provoke network congestion with different intensity levels and, therefore, to test the new proposals under diverse traffic conditions. Table 8.4 describes the different traffic patterns applied.

Network	**Pattern I**	**Pattern II**
#Sources	**Destination**	
80%	Uniform	Unif+HS+Unif
20%	stop+HS+stop	stop+HS+stop

Table 8.4: Traffic Patterns Applied to Evaluate the PVOQ and SB Structures.

In the first traffic pattern (Pattern I), which has been used in previous analysis to

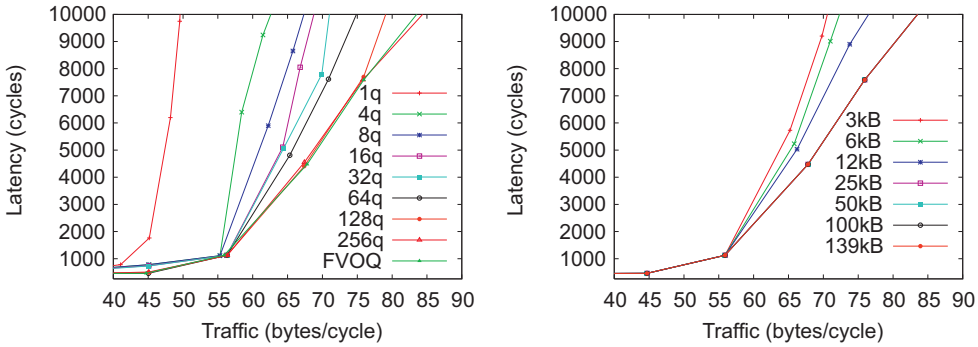
evaluate the different CMM, a set of sources generate and inject packets according to a uniform distribution of packet destinations. Then, the rest of the sources create a hot-spot in the network by injecting 1,000 packets to a single destination. This traffic pattern has been applied to obtain the results of Figures 8.40, 8.41, 8.45, and 8.46. In the second traffic pattern (Pattern II), hosts injecting packets to the hot-spot work as in the Traffic Pattern I, but hosts injecting uniform traffic start injecting uniform traffic and randomly each one injects a set of 100 packets to the same hot-spot and after that, they continue injecting uniform traffic. This second traffic pattern has been defined to stress even more the network and to evaluate the behavior of the new storage structures by generating a single bursty traffic. This traffic pattern has been applied to obtain graphs in Figures 8.42 to 8.44, and 8.47 to 8.48. Also, we show results for two configurations of packet size. In particular, a fixed packet size of 256 bytes and a variable packet size between 64 and 512 bytes. For other fixed packet sizes, that is, 512 and 1024 bytes, the results were qualitatively similar.

8.6.1 Evaluating the Scalability

Firstly, we show the evaluation results for the *BMIN Perfect-Shuffle 4-ary 5-fly* in Figures 8.40 to 8.44, and secondly, to corroborate the results, we show the analysis for the *BMIN Perfect-Shuffle 4-ary 3-fly* in Figures 8.45 to 8.48.

Curves in Figures 8.40(a) and 8.40(b) show the network performance for the PVOQ and SB techniques when using different buffer sizes. Notice that the size of the queues depends on the technique applied (see section 6.4). In Figure 8.40(a), the number of queues varies from 1 to 256 queues. For comparison purposes, curve FVOQ, which represents the original MVCM implementation with 512 queues, is included. Notice that the module mapping technique has been used to allocate a new generated packet into the available queues. So, a packet destined to the host *n* is stored in the queue *n mod q*, where *q* represents the number of queues.

On the other hand, curves in Figures 8.40(b) show the network performance when different sizes of SB are used. The values of the buffer size vary between 3kB (approximately 11 data packets for a fixed packet size of 256 bytes) and 139kB (approx-



(a) PVOQ (b) SB

Figure 8.40: Network performance with (a) PVOQ and (b) SB when applying pattern I in a BMIN Perfect-Shuffle 4-ary 5-fly and fixed packet size=256 bytes.

imately 512 data packets). Notice that to define a FVOQ, at least a 278kB of storage source would be needed:

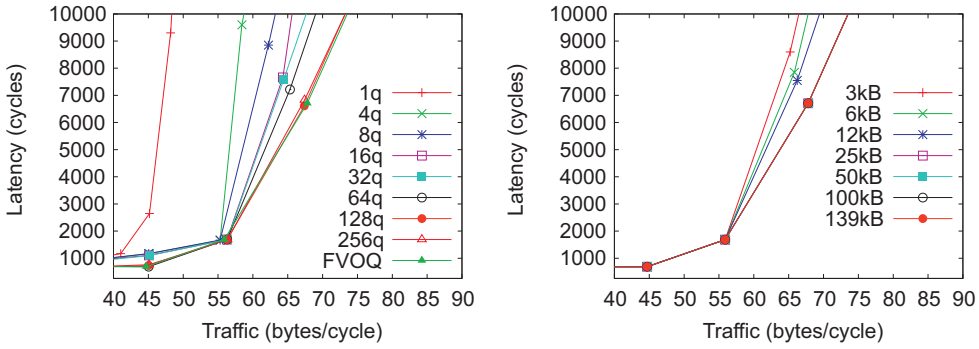
$$(DWmax * \#destinations * packetsize)$$

However, using an array of counters (see section 6.4), the total size is reduced by half (139kB), because only one buffer is required whatever the size of the dynamic window.

Comparing results from both figures, it is expected that, for an acceptable maximum latency of 5,000 cycles, (notice that this value is 10 times the average latency for a packet when network is injecting uniform traffic near the saturation point), it would be enough to dedicate a maximum SB of 25kB to achieve the same network performance as with the FVOQ scheme, whereas a PVOQ needs at least 128 queues, which correspond to 32kB (128 queues * 278 bytes). Moreover, for a latency of 10,000 cycles (20 times the average latency), a SB size of 25kB still continues to be enough, whereas PVOQ needs more than 64kB (256 queues * 278 bytes) to achieve similar results. Notice that when the PVOQ scheme is used, the number of queues needed increases as congestion increases. On the contrary, the shared-buffer size remains constant regardless of the congestion level.

In the PVOQ scheme, packets belonging to different flows share the same queue.

As known, packets belonging to flows responsible for congestion are stopped because of the injection restriction applied by the congestion management mechanism. So, once congestion appears, this fact could cause head-of-line blocking, stopping also packets belonging to *cold-flows*, which should not be affected by the injection restrictions. As traffic injection increases, more packets are affected and more queues are needed to maintain the same performance. On the other hand, it is enough to dedicate a SB size of 25kB to keep enough free buffer space to first allocate and later inject packets belonging to *cold-flows*. As a result, these packets do not suffer the head-of-line blocking phenomenon as it occurs in the PVOQ scheme. Additionally, it should be highlighted the best use of storage space compared to the PVOQ technique.
Notice that with low injection rates, both proposals, PVOQ and SB, are able to achieve the best results by dedicating a little memory space instead of applying a FVOQ.

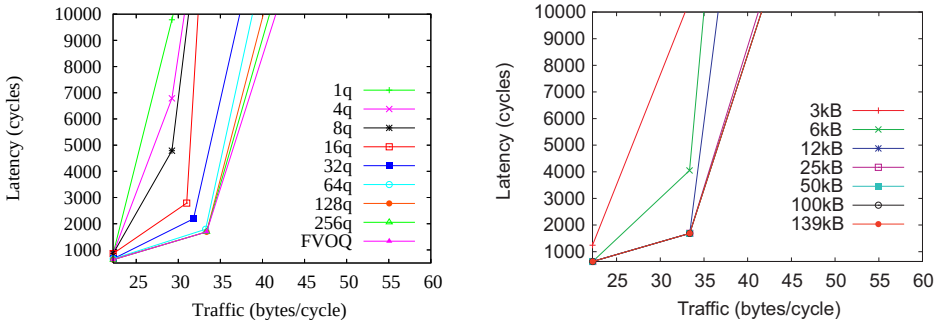


(a) PVOQ (b) SB

Figure 8.41: Network performance with (a) PVOQ and (b) SB when applying pattern I in a BMIN Perfect-Shuffle 4-ary 5-fly and variable packet size.

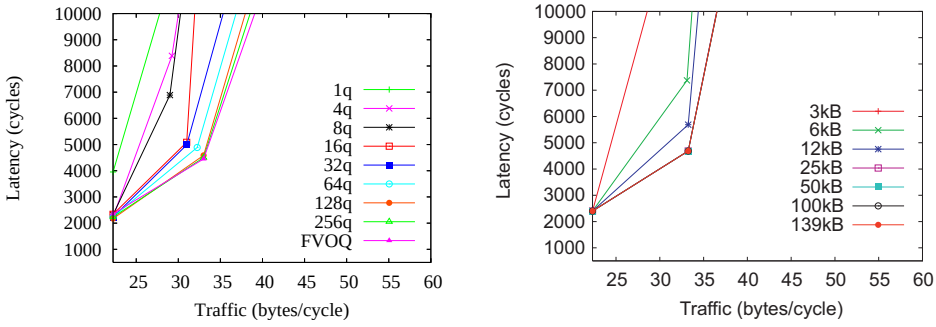
Next, Figures 8.41(a) and 8.41(b) show the performance of PVOQ and SB, respectively, when applying a variable packet size between 64 and 512 bytes. Again, the SB scheme just needs a buffer size of 25kB, whereas PVOQ needs more than 36kB (128 queues). Although we know that in both cases of study the memory size will depend on the traffic pattern and the injection rate applied, we have evaluated other traffic patterns which stress the network much more in order to deduce if there

is an approximate size which not only saves memory but also allows the best performance results to be achieved.



(a) PVOQ (b) SB

Figure 8.42: Network performance with (a) PVOQ and (b) SB when applying pattern II in a BMIN Perfect-Shuffle 4-ary 5-fly and a fixed packet size=256 bytes.



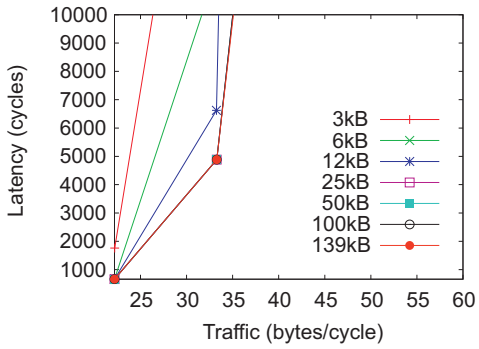
(a) PVOQ (b) SB

Figure 8.43: Network performance with (a) PVOQ and (b) SB when applying pattern II in a BMIN Perfect-Shuffle 4-ary 5-fly and variable packet size.

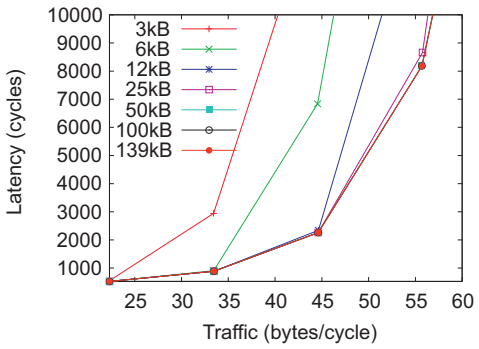
Figures 8.42 and 8.43 show the performance when Pattern II is applied with fixed and variable packet sizes, respectively. In particular, curves in Figures 8.42(a) and 8.43(a) represent performance results for the PVOQ, whereas curves in Figures 8.42(b) and 8.43(b) show the performance results for SB. Analyzing the results for both configurations of packet sizes, again, for a latency of 5,000 cycles, at least 128

queues are needed in the PVOQ scheme to roughly achieve the best performance, whereas the SB continues to need 25kB of buffer size. Notice that, again there is a particular buffer size for SB, which allows to reach the best performance for a network, regardless the traffic load and injection rate, whereas the PVOQ depends slightly on those parameters.

Now, in order to test whether the results obtained for the SB are significant or not, Figure 8.44 presents results when applying the SB scheme with a fixed packet size and with a modified Pattern II in order to vary the pressure on the shared-buffer. In particular, an increase and a decrease of 10% of the hot-spot traffic belonging to the *hot-flows* has been applied to obtain graphs (a) and (b), respectively. Once more, the SB still obtains good results with a memory size of 25kB for maximum latencies of 5,000 and 10,000 cycles.



(a) +10% in hot-spot packets



(b) -10% in hot-spot packets

Figure 8.44: Network performance when applying SB with a modified pattern II in a BMIN Perfect-Shuffle 4-ary 5-fly and fixed packet size=256 bytes.

Next, to justify that this performance is achieved in other network sizes, Figures 8.45 to 8.48 present results for the *BMIN Perfect-Shuffle 4-ary 3-fly*. In particular, Figures 8.45 and 8.46 are the result of applying the traffic pattern I with a fixed and a variable packet size, respectively. In the same way, Figures 8.47 and 8.48 are the result of applying the traffic pattern II with a fixed and a variable packet size, respectively.

As it can be analyzed, dedicating a SB size of 3kB is enough to achieve the best

results in both values of latency (5,000 and 10,000 cycles), whereas with a PVOQ more than 4kB (16 queues * 256 bytes) are needed for a fixed packed size and more than 6kB for a variable packet size.

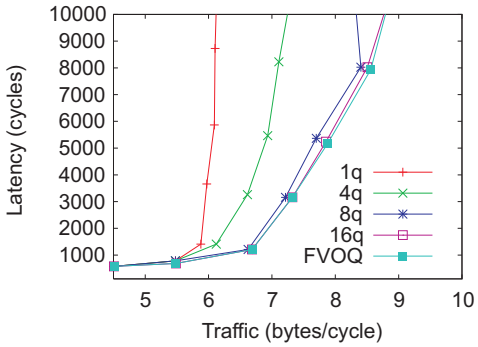
Finally, we show in Table 8.5 a summary of the memory size needed by both SB and PVOQ to achieve the same results as a FVOQ for different network configurations. Additionally, each network configuration presented in Table 8.5 shows the percentage of reduction of the memory size when applying either PVOQ or SB with respect to the size required by FVOQ. Notice that all these values are results from our simulations.

Network	**# Hosts**	**FVOQ**	**PVOQ**	**Δ%**	**SB**	**Δ%**
BMIN 4-ary 5-fly	512	256 kB	32 / 64 kB	*87 / 75%*	25 kB	*90%*
BMIN 4-ary 3-fly	64	32 kB	4 / 4 kB	*87 / 87%*	3 kB	*90%*
BMIN 8-ary 3-fly	512	256 kB	32 / 64 kB	*87 / 75%*	25 kB	*90%*
UMIN 4-ary 4-fly	256	128 kB	16 / 32 kB	*87 / 75%*	12 kB	*90%*
UMIN 8-ary 3-fly	512	256 kB	32 / 64 kB	*87 / 75%*	25 kB	*90%*

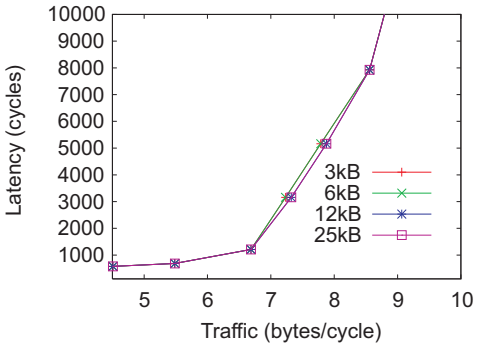
Table 8.5: Percentage of the memory reductions when applying SB or PVQ with respect to FVOQ.

As can be seen in Table 8.5, the memory reductions are significant in both schemes. Approximately, the memory dedicated by a FVOQ can be reduced between 75% and 90%, if a PVOQ or a SB is used.

In conclusion, the SB scheme is a cost-effective solution when compared to the alternative of applying FVOQ or even PVOQ in order to remove the head-of-line blocking effect at origins. This is because, if the memory is well-managed as in a SB, a reduced value of memory size can be found so that it allows good results to be achieved. Additionally, although the memory size seems to depend on the traffic pattern and the injection rate applied, we have checked that with a smaller memory for SB (25kb), the CMM obtains good results under a low, medium, and high injection rates and for different traffic patterns.

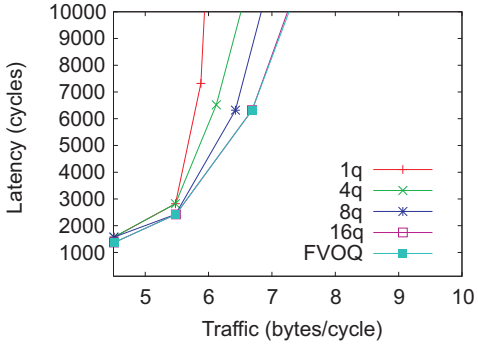


(a) PVOQ

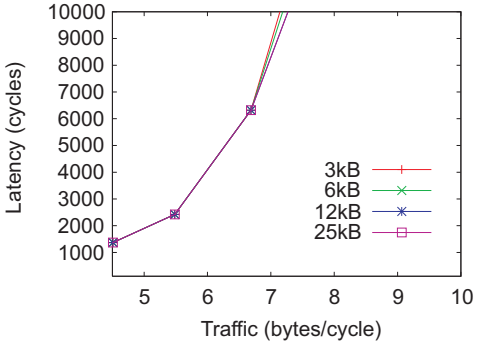


(b) SB

Figure 8.45: Network performance with (a) PVOQ and (b) SB when applying pattern I in a BMIN Perfect-Shuffle 4-ary 3-fly and a fixed packet size=256 bytes.

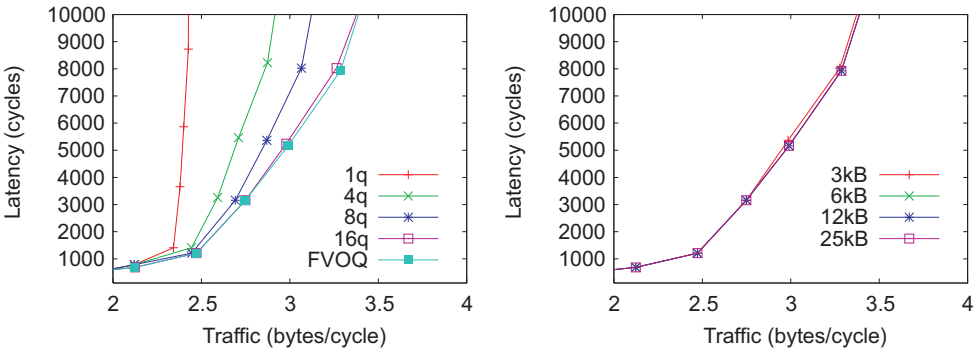


(a) PVOQ



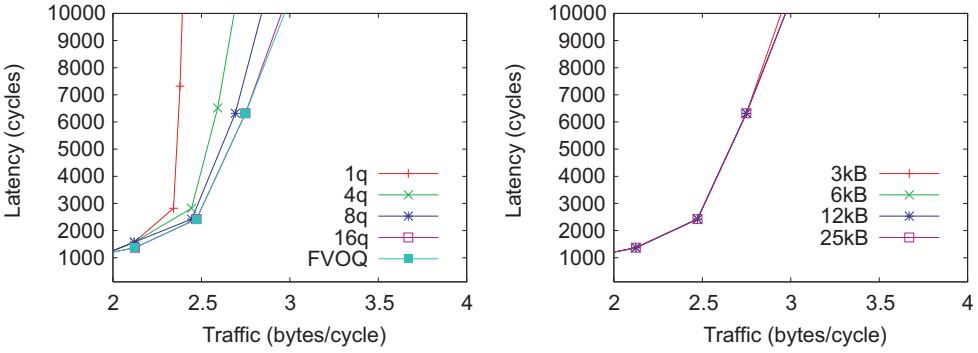
(b) SB

Figure 8.46: Network performance with (a) PVOQ and (b) SB when applying pattern I in a BMIN Perfect-Shuffle 4-ary 3-fly and variable packet size.



(a) PVOQ (b) SB

Figure 8.47: Network performance with (a) PVOQ and (b) SB when applying pattern II in a BMIN Perfect-Shuffle 4-ary 3-fly and a fixed packet size.



(a) PVOQ (b) SB

Figure 8.48: Network performance with (a) PVOQ and (b) SB when applying pattern II in a BMIN Perfect-Shuffle 4-ary 3-fly and variable packet size.

8.6.2 Chip Area Requirements

In the previous section, we have evaluated by simulation the feasibility of the alternative storage structures to improve the scalability of the MVCM mechanism. In particular, the analysis was focused on the replacement of the FVOQ buffer organization, which has been assumed in the evaluation of the MVCM mechanism, by

a PVOQ or a SB organizations. Table 8.5 presented a summary of the calculated memory sizes needed with regard to the network configuration.

In this section, we estimate the cost of the new storage schemes in the NICs of the hosts by calculating the silicon area required to implement them. The hardware description language used in the design of the memory schemes was *Verilog* using the applications DSCH [67] and Microwind [67] to compile the corresponding designs. In particular, the method applied to obtain the results was, firstly, to design the proposed memory schemes with DSCH in order to get the Verilog files, and, secondly, to import these files with Microwind to obtain the cost of implementation by calculating the silicon area required. The different schemes have been designed for a 45nm CMOS technology.

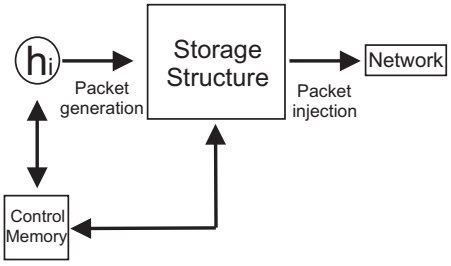


Figure 8.49: Basic diagram implemented to calculate the cost of implementing the different memory schemes (memory structure and control memory).

Figure 8.49 shows a basic diagram representing the implemented scheme. In particular, the h_i identifies any origin host in the network, the *Storage Structure* represents the different memory schemes analyzed, that is FVOQ, PVOQ and SB, and the *Control Memory* identifies the additional memory needed to control the outstanding packets, the dynamic window size, and the waiting slots per flow.

As a result of this study, the Figure 8.50 shows the implementation cost in terms of required silicon area for different network configurations. Each point represents the silicon area in mm^2 required to implement each particular storage scheme. As it can be observed, the FVOQ presents the highest cost in all cases, as it was expected. This is because its implementation cost directly depends on the number of destination

hosts in the network.

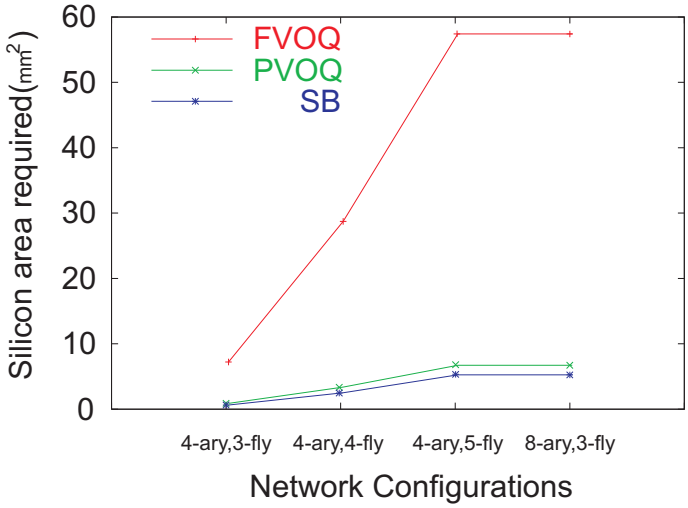


Figure 8.50: Required silicon area when applying the different memory schemes evaluated.

The differences in cost that can be observed in Figure 8.50 between a FVOQ and a PVOQ or a SB scheme are about 80 and 90% of silicon area reduction, respectively.

Network	**FVOQ** mm^2	**PVOQ** mm^2	**Δ%**	**SB** mm^2	**Δ%**
8-ary 3-fly(512hosts)	57.39	6.71	*88.3%*	5.24	*90.9%*
4-ary 5-fly(512hosts)	57.39	6.71	*88.3%*	5.24	*90.9%*
4-ary 4-fly(256hosts)	28.72	3.38	*88.2%*	2.51	*91.3%*
4-ary 3-fly(64hosts)	7.21	0.88	*87.8%*	0.63	*91.3%*

Table 8.6: Silicon area requirements for different configurations.

In particular, Table 8.6 presents the exact values of the silicon area required in each network configuration. As a result, the implementation of any of the two alternative schemes to the original FVOQ scheme is shown to be highly recommended. Anyway, as concluded in previous section, SB presents the best cost-effective alter-

native for implementing the storage resources of NICs.

Notice that, despite the network configurations are different (*8-ary 3-fly* and *4-ary 5-fly*), the obtained results are the same in both cases, as the number of hosts is equal (512 hosts). Therefore, the size of storage depends on the number of hosts and not on the network configuration.

Chapter 9

CONCLUSIONS AND FUTURE WORK

In this chapter we describe the contributions and conclusions of the work done, the publications and the future work.

9.1 Contributions and Conclusions

The growth of parallel computers based on high-performance networks has increased the interest and effort of the research community in developing new techniques to achieve the maximum performance from these networks. In particular, the development of new techniques for efficient routing to reduce packet latency and increase network throughput. However, high utilization rates of the network could result in what is known as *network congestion*, which could cause a degradation of the network performance because all or a part of the network has exceeded the maximum utilization imposed by the saturation point of the network.

The aim of this dissertation has been the development of new Congestion Management Mechanisms able to effectively detect and recover from a congestion situation. These mechanisms are based on new strategies for *i)* the early and thorough detection of congestion, and *ii)* new corrective actions that stop the growth of the sat-

uration tree and offset its effects, allowing an even distribution of network resources. Indeed, the main challenge has been to develop new strategies able to achieve good performance acting on flows responsible for congestion and avoiding penalizing non-responsible flows.

In this thesis, we aim at providing congestion management mechanisms which achieve the previous requirements. Moreover, the proposed mechanisms accomplish three additional requirements, such as their robustness, the fact that they do not penalize network behavior in the absence of congestion, and finally, the fact that they do not generate additional problems.

A part of this thesis is dedicated to describe in detail the new strategies for congestion detection and correction, which make up the new congestion management mechanisms (i.e., MVCM and IOCM). Additionally, we have evaluated those mechanisms under simulation, and compared their results to the ones achieved by the current proposals for congestion management (i.e., Renato's and Pfister's proposals). In order not to loss generality, unlike other approaches, these new mechanisms are not targeted for a particular network technology, but for the multistage interconnection networks in general. However, the proposed mechanisms could easily be applied to current standard interconnects, such as InfiniBand. Therefore we consider that this thesis achieves the previously presented objectives.

The main contributions of this dissertation are the following:

- Firstly, this thesis presents a deep analysis of the congestion process. The analysis states that every congestion process starts in a switch when some flows compete for the same output link, and the bandwidth of the corresponding output link is not enough to consume all the incoming packets. Additionally, the study states that this situation is critical only if it affects other flows not responsible for congestion. As a result of that, a suitable technique to classify packet flows into the network is needed.

- Secondly, this thesis presents a thorough study of the current techniques to detect congestion in multistage interconnection networks. We have analyzed the Input Packet Marking (IPM), the Renato's Packet Marking (RPM), and the

Output Packet Marking (OPM) techniques, highlighting their pros and cons. As a conclusion, they are not able to correctly classify flows in a congestion situation and, therefore, the marking actions may be applied over the packets non responsible for the congestion. As a consequence, two new packet marking techniques have been proposed in this thesis, namely the Input-Output Packet Marking (IOPM) and the Marking and Validation Packet Marking (MVPM) techniques, which, unlike the previous packet marking strategies, use two bits to warn about a congestion situation. These new proposals allow to correctly classify flows by identifying a four-level classification of congestion.

- Thirdly, this thesis analyses in depth the current schemes of application of corrective actions, such as the static window-based and the waiting interval techniques, which depend on a packet marking technique based on a single marking bit (i.e., IPM, RPM, and OPM). The study shows that: *i)* the static window-based technique is not an effective technique because it could cause too much restriction on the packet injection, and therefore, it have to be replaced by a dynamic window-based technique, *ii)*with a well-designed packet marking technique, that is IOPM or MVPM, based on two bits, the correction actions schemes can be enhanced by applying a combination of different corrective actions, that is, a window-based technique and a waiting interval technique. Additionally, a suitable methodology for parameters initialization is proposed.

- Fourthly, two new proposals for congestion management mechanism are presented and evaluated, that is, the Marking and Validation Congestion Management (MVCM) and the Input and Output Congestion Management (IOCM). In particular the MVCM and the IOCM mechanisms are based on the MVPM and the IOPM packet marking techniques, respectively. Additionally, both CMMs apply a well-designed set of corrective actions based on a combination of a window-based and a waiting interval technique that make packets belonging to the flows responsible of congestion wait at their source hosts, instead of remaining blocked into the network.

- Lastly, this thesis presents the results of an extensive evaluation of the current proposals for CMM, that is, Renato's and Pfister's mechanisms, together with the new proposed CMMs, that is, MVCM and IOCM. The evaluation has been carried out by comparing those CMMs under the same network configuration and traffic loads. Additionally, a study of the silicon area required by different storage structures to classify generated packets at the NICs is presented. In particular, a Partial Virtual Output Queue (PVOQ) structure and a Shared-Buffer (SB) approach are analyzed.

From the analysis and evaluation of the proposed CMMs, the following conclusions are drawn:

- Firstly, the adjustment of the parameters of the proposed mechanisms, that is the buffer threshold, the window size and the waiting interval calculation, only depends on the network configuration, therefore the mechanism is simple and easy to implement.

- Secondly, it has been shown that the MVCM proposal provides good performance for congestion management regardless of the traffic load. Moreover, MVCM not only reduces latency for *cold-flows* with regard the current proposals, but also improves the performance of *hot-flows* by avoiding oscillations in network throughput and keeping their packet injection at the maximum rate.

- Thirdly, from the comparison between MVCM and IOCM mechanisms, we conclude that, although IOCM presents a slight increase in performance for accepted traffic, due to the fact that IOCM marks packets in advance because there is no dependency between both marking actions, it wrongly marks more packets belonging to *cold-flows* than MVCM does. Therefore, depending on whether it is preferred to promote accuracy or an early packet marking, we can apply either the MVCM or the IOCM mechanisms, respectively.

- Fourthly, when compared to current proposals, it is observed that MVCM and IOCM provide the best results for congestion management because of their

more efficient recovery actions based on a dynamic window, waiting insertion technique, and immediate recovery of the initial parameters. Previous proposals, that is, Pfister's and Renato's proposals, obtain worse results because they limit the injection rate too much, specially Renato's proposal, due to the fact that a static window is applied whatever the network behavior is. On the other hand, Pfister's implementation applies a simple recovery technique based on a waiting insertion technique, and if this technique works alone, it is not able to react as fast as the combination of a dynamic window and waiting insertion technique.

- Fifthly, analyzing in isolation the design issues that make different these proposals, packet marking and recovery techniques, we observe that the packet marking election is not the most critical issue in a CMM. Even so, the strategy that is based on input buffer marking combined with output buffer validation, that is MVPM, has obtained the best results, because it is the one which wrongly marks less cold-packets in a congestion situation. Additionally, the study concludes that marking only at input buffers achieves the lower performance. Concerning recovery techniques, it is shown that applying only a static window restricts too much the injection rate and it will never achieve the results of a dynamic window does. Moreover, using a dynamic window as a preventive action combined with a waiting slots insertion technique minimizes the penalization on the flows not responsible for the congestion. Graphs reveal that even by applying a simple dynamic window technique, it is possible to achieve good results because the dynamic window better tunes injection limitation. Finally, a limitation in the waiting slot insertion and an immediate recovery of the initial values of the injection rate, when congestion is not longer detected, avoids penalizing the *hot-flows* in excess, and is much better than a progressive recovery.

- Finally, we have analyzed different alternative store structures to improve the scalability of the MVCM and IOCM mechanisms. In particular, the possibility of replacing the FVOQ at origins by either a PVOQ or a SB has been evalu-

ated. Results show that it is not necessary to dedicate a FVOQ at origin hosts to eliminate HOL blocking. Despite the fact that applying a PVOQ contributes to improve the scalability, the use of a SB is the best choice because its size is independent from the system size. Further, it does not depend on the applied traffic load. We have shown that a small size of SB is enough to maintain a similar performance level to that achieved applying a FVOQ. Additionally, a study of the silicon area requirements for the different alternative store structures shows that by applying a PVOQ or a SB instead of a FVOQ, the reductions in the silicon area requirements is about 88% and 90% of the initial silicon area required, respectively. In conclusion, SB is a cost-effective solution when compared to the alternative of applying FVOQ or PVOQ in order to remove the HOL blocking effect at origin. This is because the required memory size is reduced, it does not depend on the traffic load, and it is a fixed parameter that can be defined at network implementation. Notice that these sizes of SB and PVOQ are related to a high injection rate, but if a medium injection rate is applied (as in a normal network situation), smaller memory sizes for SB or PVOQ could be used.

9.2 Scientific Publications

Next, we list the articles published in relation to the work presented in this thesis.

Publications on National Conferences:

- "**MVCM: A Packet Marking/Validation Mechanism for Congestion Management in InfiniBand**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the CEDI 2005 - I Congreso Español de Informática, XVI Jornadas de Paralelismo*, Actas de las Jornadas de Paralelismo, pages 189-196, Granada (Spain), September 2005, Publisher: Ed. Thomson. ISBN:84-9732-430-7.

- "**Fair Congestion Management in MINs**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the XVII Jornadas de Paralelismo*, pages 217-222, Albacete (Spain), September 2006, Publisher: Universidad de Castilla-La Mancha. ISBN:84-690-0551-0.

- "**A Comparative Study of Congestion Management Mechanism for MINs**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the XIX Jornadas de Paralelismo*, pages 387-392, Castelló (Spain), September 2008, Publisher: Publicacions de la Universitat Jaume I-Castelló. ISBN:978-84-8021-676-0.

Publications on International Conferences:

- "**Congestion Management in MINs through Marked & Validated Packets**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based (PDP '07)*, pages 254-261, Naples (Italy), February 2007. Publisher: IEEE Computer Society Press. ISBN: 0-7695-2784-1.

- "**On the Influence of the Packet Marking and Injection Control Schemes in Congestion Management for MINs**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the 14th Euro-Par International Conference*,

pages 930-939, Las Palmas de G.C. (Spain), August 2008. Publisher: Lecture Notes in Computer Science, Springer-Verlag. ISBN: 978-3-540-85450-0.

- "**A Scalable and Early Congestion Management Mechanism for MINs**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. *In Proceedings of the 18th EUROMICRO International Conference on Parallel, Distributed and Network-Based (PDP '10)*, pages 43-50, Pisa (Italy), February 2010. Publisher: IEEE Computer Society Press. ISBN: 978-0-7695-3939-3.

Publication on International Journals:

- "**Progressive Congestion Management Based on Packet Marking and Validation Techniques**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. IEEE Transactions on Parallel and Distributed Systems (TPDS), September 2012.

- "**Progressive Congestion Management Based on Packet Marking and Validation Techniques**". J.Ferrer, E.Baydal, A.Robles, P.López, and J.Duato. Computing Now. IEEE computer society. March 2012. http://www.computer.org/portal/web/computingnow/0412/whatsnew/tc

9.3 Future Work

- Taking into account that both the MVCM and IOCM mechanisms achieve good performance with their original configuration, it will be interesting to analyze the performance when considering the following modifications:

    1. Applying an adaptive routing. Both CMMs apply a deterministic routing mechanism to route packets through the network. Injected packets follow the defined path at origins host regardless the network traffic and congestion. By applying adaptive routing, packets could be rerouted through alternative paths if a congestion problem appears in any part of the network.
    2. Applying the proposed CMMs in other network configurations. This thesis presents a thorough analysis of the proposed mechanism when they are applied in multistage interconnection networks. In particular over Butterfly and Perfect-Shuffle MINs. It could be interesting to find out if those mechanism also work correctly over other type of networks, and therefore they could be applied in other network environments.

- There are other current proposals for congestion management which present good results. It could be interesting to evaluate and compare their results to the ones achieved by our proposals. In particular, it would be interesting to compare MVCM and RECN [35] under the same network configuration and traffic conditions. The aim of this study is: *i)* to evaluate if the technique used in RECN to manage congestion is able to solve a heavy congestion situation as the MVCM mechanism does, and *ii)* if a combination of both MVCM and RECN mechanisms could improve the network behavior in other networks configurations and under other traffic conditions.

- As it was presented in section 5.3.1, the dynamic window size takes its value from the integer number higher but closer to the real value achieved when applying the (5.6) formula. As an example, for the BMIN 4-ary 5-fly network

we calculated a real value of 1.28, so the window size took the integer value of 2. As it can be seen, the mechanism allows to inject a 56% more packets than the network can deliver. So, it also could create congestion, which has to be solved by reducing the size of the window, at origins, after the congestion is detected.

A well-known algorithm to draw lines in a computer screen, that is, the *Bresenham's line algorithm*, could dinamically be applied to accurately calculate the window size value, and therefore to avoid creating congestion situations.

The *Bresenham's line algorithm* determines which points in an n-dimensional raster should be plotted in order to form a close approximation to a straight line between two given points. As it uses only integer addition, subtraction and bit shifting, the algorithm is easy to implement in standard computer architectures. Nowadays, the speed and simplicity of Bresenham's line algorithm means that it can be used in hardware such as plotters and graphics chips of modern graphics cards. It can also be found in many software graphics libraries, or even in either the firmware or the hardware of modern graphics cards.

With this algorithm, the new proposal could dynamically adjust and assign the value of the dynamic window size in a preventive way, instead of reacting when congestion appears as current proposals do.

Bibliography

[1] R. Alcover. ”Optimización de Redes de Interconexión para Multicomputadores mediate Técnicas de Diseo Robusto de Taguchi”. *Tesis Doctoral*.

[2] R. Alcover, P. López, J. Duato, and L. Zunica. ”Interconnections Network Design: a Statistical Analysis of Interactions between Factors”. *in Proceedings of the 4th Euromicro Workshop on PDP.*, 1996.

[3] M. Allman, V. Paxsm, and W. Stevens. ”TCP Congestion Control”. *Http://www.rfc-editor.org/rfc/rfc2581.txt*, 1999.

[4] G. Almasi and A. Gottlieb. ”Highly Parallel Computing”. *in Ed.Benjamin-Cummings Publishing Co.,Inc.*, 1994.

[5] M. Alonso, S. Coll, J. Martinez, V. Santonja, P. López, and J. Duato. ”Dynamic Power Saving in Fat-Tree Interconnection Networks Using On/Off Links”. *in Proc. on Int. Symp. on HP-PAC and IPDPS*, 2006.

[6] T. Anderson, S. Owicki, J. Saxe, and C. Thacker. ”High-Speed Switch Scheduling for Local-Area Networks”. *ACM Trans. On Computer*, 1993.

[7] A. Asthana. ”Toward a gigabit IP Router”. *in Jouenal of High Speed Networks, vol.1, no 4, pp.281-288.*, 1992.

[8] F. Baker. ”Requirements for IP version 4 routers”. *in RFC 1812.*, June 1995.

[9] E. Baydal and P. López. ”A Robust Mechanism for Congestion Control:INC”. *in Proc. 9th. International Euro-Par Conference, pp. 958-968.*, August 2003.

[10] E. Baydal, P. López, and J. Duato. ”A Family of Mechanisms for Congestion Control in Wormhole Networks”. *IEEE Trans. Parallel Distrib.Syst., vol.16 n8.*, August 2005.

[11] E. Baydal, P. López, and J. Duato. ”A Congestions Control Mechanism for Wormhole Networks”. *in Proc. 9th. Euromicro Workshop Parallel and Distributed Processing, pp. 19-26.*, February 2001.

[12] R. Bianchini, T. Leblanc, L. Kontothanassis, and M. Crovella. ”Alleviating Memory Contention in Matrix Computations on Large-Scale Shared-Memory Multiprocessors”. *in Technical Report 449, Dept.of Computer Science, Rochester University.*, April 1993.

[13] BlueGene. ”Available at:https://asc.llnl.gov/computing-resources/bluegenel/”.

[14] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, and W. Su. ”Myrinet a Gigabit per Second Local Area Network”. *in IEEE Micro, pp.29-36.*, 1995.

[15] R. Bopana and S. Chalasani. ”A Comparison of Adaptive Wormhole Routing Algorithms”. *in Proc. 20th Annu. Int. Symp. Comput. Architecture*, May 1993.

[16] L. Brakmo and L. Peterson. ”TCP Vegas: End to End Congestion Avoidance on a Global internet”. *IEEE Journal on Selected Areas in Communication, vol.13, no.8,pp.1465-1480.*, 1995.

[17] T. Callahan and S. Goldstein. ”NIFDY: a Low Overheas, High Through Network Interface”. *Proceedings of the 22th International Symposium on Computer Architecture.*, June 1995.

[18] A. Chien and J. Kim. ”Plannar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors”. *in Proc. 19th Annu. Int. Symp. Comput. Architecture*, May 1992.

[19] CoreConnect. ”https://www-01.ibm.com/chips/techlib/techlib.nsf/products/coreconnect-bus-architecture.”.

[20] J. Costanzo, L. Crowl, L. Sanchis, and M. Srinivas. ”Subgraph Isomorphism on the BBN Butterfly Multiprocessor. Butterfly Project Report 14”. *Department of Computer Science, University of Rochester, Rochester, New York, 14627-0226*, 1986.

[21] CPMD. ”Available at: http://www.cpmd.org/.”.

[22] CRAY. ”The CRAY T3E.Available at:http://www.cray-cyber.org/systems/t3e.php”.

[23] W. Dally. ”Virtual-Channel Flow Control”. *in Proceedings of the 17th international symposium on Computer Architecture,pp.60-68.*, 1990.

[24] W. Dally and H. Aoki. ”Deadlock-Free Adaptative Routing in Multicomputer Networks Using Virtual Channels”. *IEEE Transactions on Parallel and Distributed Systems, vol.4, no.4, pp.466-475.*, April 1993.

[25] W. Dally, P. Carvey, and L. Dennison. ”The Avici Terabit Switch/Router”. *in Proc. on Hot Interconnects*, 1998.

[26] W. Dally and C. Deitz. ”Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *in IEEE Transactions on Computers, vol. C-36, no.5, pp.547-553.*”, May 1987.

[27] W. Dally and B. Towles. ”Principles and Practices of Interconnection Networks”. *IEEE Computer Society Press*, 2004.

[28] S. Dcott and G.Thorson. ”The Cray TE Networking: Adaptive Routing in a High Performance 3D Torus”. *in Proc. Hot Interconnects IV.*, 1996.

[29] DLPOLY. ”Available at: http://www.cse.scitech.ac.uk/ccg/software/dlpoly/.”.

[30] J. Duato. ”A New Theory of Deadlock-Free Adaptive Routing in wormhole Networks”. *in IEEE Transactions on Parallel and Distributed Systems, vol.4,no.12,pp1320-1331.*, December 1993.

[31] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo. ”A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks”. *in Proc. on Int. Sym. on HPCA*, 2005.

[32] J. Duato and P. López. ”Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes”. *Parallel Computer Routing and Communication, K.Bolding and L.Snyder (ed.), Springer-Verlag, pp.45-59*, 1994.

[33] J. Duato, S. Yalamanchili, and L. Ni. ”Interconnection Networks an Engineering Approach”. *IEEE Computer Society*, 2003.

[34] T. Dunnigan. ”Early Experiences and Performance of the Intel Paragon.”. *Techncal ReportORNL/TM-12194, Oak Ridge National Laboratory. http://www.csm.ornl.gov/∼dunigan/paragon.*, 1994.

[35] EarthSimulation. ”Available at: http://www.jamstec.go.jp/esc/index.en.html.”.

[36] J. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato. ”Congestion Management in MINs through Marked & Validated Packets”. *in Proc. on 15th Int. Conf. Euromicro*, 2007.

[37] J. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato. ”On the Influence of the Packet Marking and Injection Control Schemes in Congestion Management for MINs”. *in Proc. on 14th Int. Conf. Euro-Par*, 2008.

[38] J. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato. ”A Scalable and Early Congestion Management Mechanism for MINs”. *in Proc. on 18th Int. Conf. Euromicro*, 2010.

[39] J. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato. ”Progressive Congestion Management Based on Packet Marking and Validation Techniques”. *In IEEE Transactions on Parallel and Distributed Systems (TPDS)*, September 2012.

[40] J. Flich. ”Mejora de las Prestaciones de las Redes de Estaciones de Trabajo con Encaminamiento Fuente”. *Tesis Doctoral*.

[41] D. Franco, I. Garces, and E. Luque. ”A New Method to Make Communication Latency Uniform: Distributed Routing Balancing”. *in Proceedings ACM International Conference on Supercomputing,pp.210-219.*, May 1999.

[42] P. Garcia, J. Flich, J. Duato, I. Johnson, F. Quiles, and F. Naven. ”Dynamic Evolution of Congestion Trees:Analysis and Impact on Switch Architecture”. *in Proc. on Int. Sym. on HiPEAC*, 2005.

[43] P. Gaughan and S. Yalamanchili. ”Adaptative Routing Protocols for Hypercube Interconnection Networks”. *in IEEE Computer, vol.26, no.5,pp12-23.*, May 1993.

[44] M. Gómez. ”An Efficient Fault-Tolerant Routing Methodology for Meshes and Toti”. *MCYT.*, 2004.

[45] M. Gómez, J. Flich, A. Robles, P. López, and J. Duato. ”VOQsw: A Methodology to Reduce HOL Blocking in InfiniBand Networks”. *in Proc. 17th. IPDPS.*, 2003.

[46] W. Ho and D. Eager. ”A Novel Strategy for Controlling Hot Spot Contention”. *in Proceedings of International Conference Parallel Processing, vol.I, pp.14-18.*, 1989.

[47] C. Hyatt and D. Agrawal. ”Congestion Control in the Wormhole-Routed Torus with Clustering and Delayed Deflection”. *Workshop on Parallel Computing, Routing, and Communications (PCRCWp7)*, June 1997.

[48] InfiniBand. ”trade association infiniband architecture. specification volumen 1. release 1.2. available at: http://www.infinibandta.org”.

[49] C. Izu and A. Arruabarrena. ”Congestion Control for a k-ary n-cube Static Network with Bimodal Traffic”. *Procedings of the 5th Australasian Conference on Parallel and Real Time Systems.*, 1998.

[50] V. Jacobson. ”Congestion Avoidance and Control”. *in Proc. ACM SIGCOMM*, 1988.

[51] R. Jain. ”A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneus Computer Networks”. *ACM Computer Communication Review, vol.19, no5, pp.56-71.*, 1989.

[52] R. Jain. ”The Art of Computer System Performance Analysis”. *New York, ED.John Wiley & Sons*, 1991.

[53] M. Katevenis, D. Serpanos, and E. Spyridakis. ”Credit-Flow Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch”. *in Proc. on Int. Symp. on HPCA*, 1998.

[54] P. Kermani and L. Kleinrock. ”Virtual Cut-Through: a New Computer Communication Switching Technique”. *in Computer Networks, vol.3. pp.267-286.*, 1979.

[55] J. Kim and Z. Liu. ”Compressionless Routing: a Framework for Adaptative and Fault-Tolerant Routing”. *in IEEE Transactions on Parallel and Distributed Systems, vol.8, n.3.*, 1997.

[56] J. Kim, Z. Liu, and A. Chien. ”Compresionless Routing: a Framework for Adaptive and Fault-Tolerant Routing”. *Proceedings of the 21st International Symposium on Computer Architecture*, April 1994.

[57] S. Konstantinidou and L. Snyder. ”Chaos Router: Architecture and Performance”. *in Proceedings 18th International Symposium on Computer Architecture, pp.79 88.*, June 1991.

[58] V. Krishnan and D. Mayhew. ”A Localized Congestion Control Mechanism for PCI Express Advanced Switching Fabrics”. *in Proc. IEEE Symp. on Hot Interconnects*, 2004.

[59] K. Lam, L. Dennison, and W. Dally. ”Simultaneous Bidirectional Signalling for IC Systems”. *In International Conference on Computer Design: VLSI in Computers and Processors*, 1990.

[60] J. Laudon and D. Lenoski. ”The SGI Origin: A ccNUMA Highly Scalable Server”. *In the 24th Annual International Symposium on Computer Architecture ACM, pages 241-251*, 1997.

[61] E. Leonardi, M. Gerla, and P. Palnati. ”Congestion Control in Asynshronous, High-Speed WormholeRouting Networks”. *IEEE Communications Magazine, pp.58-69.*, November 1996.

[62] J. Liu and K. Shin. ”Prevention of Congestion in Packet-Switched Multistage Interconnection Networks”. *in IEEE Transactions on Parallel Distributed Systems, vol.6, n.5, pp.535-541.*, May 1995.

[63] P. López and J. Duato. ”Deadlock-Free Adaptive Routing Algorithms for the 3D.torus: Limitations and Solutions”. *in Proc. Parallel Architectures and Languages Europe 93.*, June 1993.

[64] P. López, J. Martínez, and J. Duato. ”DRIL: Dynamically Reduced Message Injection Limitation Mechanism for Wormhole Networks”. *International Conference Parallel Processing.*, August 1998.

[65] P. López, J. Martínez, J. Duato, and F. Petrini. ”On the Reduction of Deadlock Frequency by Limiting Message Injection in Wormhole Networks”. *Proceedings of Parellel Computer Routing and Communication Workshop.*, June 1997.

[66] MareNostrum. ”Available at:http://www.bsc.es/”.

[67] Microwind and DSCH. ”Available at:http://www.microwind.org”.

[68] MPI. ”Message Passing Interface.Available at:http://www.mcs.anl.gov/mpi”.

[69] Myrinet. ”2000 Sereis Networking. Available at: http://www.cspi.com.”.

[70] T. Nachiondo, J. Flich, and J. Duato. ”Efficient Reduction of HOL Blocking in Multistage Networks”. *in Proceedings on International Symposium IPDPS.*, 2005.

[71] M. Noakes, D. Wallach, and W. Dally. ”The J-machine Multicomputer: and Architectural evaluation.”. *in proceedings of the 20th International Symposium Computer Architecture, pages 224-235.*, 1993.

[72] C. Partridge. ”A 50-Gb/s IP Router”. *in IEEE/ACM Transactions Networking, vol.6, pp.237-248.*, June 1998.

[73] F. Petrini and M. Vanneschi. ”Minimal Adaptative Routing with Limited Injection on Toroidal k-ary n-cubes”. *Procedings of Supercomputing.*, 1996.

[74] G. Pfister. ”High Performance Mass Storage and Parallel I/O, chapter 42: An Introduction to the InfiniBand Architecture”. *IEEE Press and Wiley Press.*, 2001.

[75] G. Pfister and V. Norton. ”Hot Spot Contention and Combining in Multistage Interconnection Networks”. *IEEE Trans. on Computers*, 1985.

[76] G. Pfister et al. ”Solving Hot Spot Contention Using Infiniband Architecture Congestion Control”. *Ion HPI-DC*, 2005.

[77] T. Pinkston and S. Warnakulasuriya. ”On Deadlocks in Interconnection Networks. *24th International Symposium on Computer Architecture.*”, June 1997.

[78] M. Prycker. ”Asynchronous Transfer Mode: Solution for Broadband ISDN”. *Ed. Prentice Hall*, 1995.

[79] Quadrics. ”QsNet. Available at: http://docs.quadrics.com.”.

[80] RapidIO. ”Available at: http://www.rapidio.org.”.

[81] J. Renato, Y. Turner, and G. Janakiraman. ”Evaluation of Congestion Detection Mechanisms for Infiniband Switches”. *on IEEE GLOBECOM*, 2002.

[82] J. Renato, Y. Turner, and G. Janakiraman. ”End-to-End Congestion Control for Infiniband”. *on IEEE INFOCOM*, 2003.

[83] S. Scott and G. Sohi. ”The Use of Feedback in Multiprocessors and Its Application to Tree Saturation Control”. *in IEEE Transactions on Parallel Distributed Systems, vol.1, n.4, pp.385-398.*, October 1990.

[84] L. Shang, L. Peh, and N. Jha. ”Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks”. *in Proc. on Int. Sym. on HPCA*, 2003.

[85] A. Singh, W. Dally, B. Towles, and A. Gupta. ”Globally Adaptative Load-Balanced Routing on Tori”. *in Computer Architecture Letters, vol.3,no.1,pp.69.*, July 2004.

[86] A. Smai and L. Thorelli. ”Global Reactive Congestion Control in Multicomputer Networks”. *Proc. on Int. Conf. on HPC*, 1998.

[87] W. Stallings. ”Data and Computer Communications”. *Prentice Hall.*, 2004.

[88] W. Stallings. ”Comunicaciones y Redes de Computadores”. *Prentice Hall*, 2008.

[89] SunFire. ”The Sun Fire 15000. http://sunstuff.org/hardware/systems/other/sunfire15000/”.

[90] I. B. Team. ”An Overeview of BlueGene/L Supercomputer.”. *in Proc. ACM Supercomputing Conference.*, 2002.

[91] M. Thottetodi, A. Lebeck, and S. Mukherjee. ”Self-Tuned Congestion Control for Multiprocessor Networks”. *in Proc. on Int. Symp. on HPCA*, 2001.

[92] M. Thottetodi, A. Lebeck, and S. Mukherjee. ”BLAM: a High-Performance Routing Algorithm for Virtual Cut-Through Networks”. *in Proceedings International Parallel and Distributed Processing Symposium.*, April 2003.

[93] TOP500. ”Available at: http://www.top500.org.”.

[94] W. Vogels et al. ”Tree-Saturation Control in the AC3 Velocity Cluster Interconnect”. *in Proc. Conference on Hot Interconnects*, 2000.

[95] M. Wang, H. Siegel, M. Nichols, and S. Abraham. ”Using a Multipath Network for Reducing the Effects of Hot Spot”. *in IEEE Transactions on Parallel and Distributed Systems.*, March 1995.

[96] C. Yang and A. Reddy. ”A Taxonomy for Congestion Control Algorithms in Packet Switching Networks”. *IEEE Networks.*, 1995.

[97] P. Yew, N. Tzeng, and D. Lawrie. ”Distributing Hot-Spot Addressing in Large-Scale Multiprocessors”. *in IEEE Transactions on Computers, vol.36, n.4, pp.388-395.*, April 1987.

“Quan naixes plores i els demés riuen...,
viu de tal manera que quan mores, els demés ploren i tu rigues”.
Anònim