Universitat Autònoma de Barcelona

Departament d'Arquitectura de

Computadors i Sistemes Operatius

Màster en

Computació d'Altes Prestacions

Predictive and Distributed Routing Balancing (PR-DRB).

High Speed Interconnection Networks.

Memoria del trabajo de ”Iniciación a la Investigación. Trabajo de Fin de Máster” del “Máster en Computació d'altes prestacions”, realizada por Carlos Núñez, bajo la dirección de Daniel Franco. Presentada en la Escuela de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

2010

Iniciación a la investigación. Trabajo de fin de máster

Máster en Computación de Altas Prestaciones

Predictive and Distributed Routing Balancing (PR-DRB).

High Speed Interconnection Networks.

Realizada por Carlos Núñez en la Escuela de Ingeniería, en el Departamento Arquitectura de Computadores y Sistemas Operativos

Dirigida por: Daniel Franco

Firmado

Director	Estudiante
Daniel Franco	**Carlos Núñez**

Acknowledgments

To Nati, because we once decided we could go anywhere we wanted, together.

To my parents, for everything ….and more

To my family, who are always by my side, even from the distance

To Dani and the CAOS research group I am in, for being my mentors and fellows

To the CAOS, for giving me the opportunity

Abstract

Current parallel applications running on clusters require the use of an interconnection network to perform communications among all computing nodes available. Imbalance of communications can produce network congestion, reducing throughput and increasing latency, degrading the overall system performance. On the other hand, parallel applications running on these networks posses representative stages which allow their characterization, as well as repetitive behavior that can be identified on the basis of this characterization. This work presents the Predictive and Distributed Routing Balancing (PR-DRB), a new method developed to gradually control network congestion, based on paths expansion, traffic distribution and effective traffic load, in order to maintain low latency values. PR-DRB monitors messages latencies on intermediate routers, makes decisions about alternative paths and record communication pattern information encountered during congestion situation. Based on the concept of applications repetitiveness, best solution recorded are re-applied when saved communication pattern re-appears. Traffic congestion experiments were conducted in order to evaluate the performance of the method, and improvements were observed.

Resum

Les aplicacions paral.leles actuals en els Clústers requereixen l'ús d'una xarxa d'interconnexió per comunicar a tots els nodes de còmput disponibles. El desequilibri en la càrrega de comunicacions pot congestionar la xarxa, incrementant la latència i disminuint el throughput, degradant el rendiment total del sistema. D'altra banda, les aplicacions paral.leles que s'executen sobre aquestes xarxes contenen etapes representatives durant la seva execució les quals permeten caracteritzar-les, a més d'extraure un comportament repetitiu que pot ser identificat en base a aquesta caracterització. Aquest treball presenta el Balanceig Predictiu de Encaminament Distribuït (PR-DRB), un nou mètode desenvolupat per controlar la congestió a la xarxa en forma gradual, basat en l'expansió de camins, la distribució de trànsit i càrrega efectiva actual; per tal de mantenir una latència baixa. PR-DRB monitoritza la latència dels missatges en els encaminadors, pren decisions sobre els camins alternatius a utilitzar i registra la informació de la congestió sobre la base del patró de comunicacions detectat, utilitzant com a concepte base la repetitivitat de les aplicacions per després tornar a aplicar la millor solució quan aquest patró es repeteixi. Experiments de trànsit amb congestió van ser portats a terme per avaluar el rendiment del mètode, els quals van mostrar la bondat del mateix.

Resumen

Las aplicaciones paralelas actuales en los Clústeres requieren el uso de una red de interconexión para comunicar a todos los nodos de cómputo disponibles. El desbalance en la carga de comunicaciones puede congestionar la red, incrementando la latencia y disminuyendo el throughput, degradando el rendimiento total del sistema. Por otro lado, las aplicaciones paralelas que corren sobre estas redes contienen etapas representativas durante su ejecución las cuales permiten caracterizarlas, además de un comportamiento repetitivo que puede ser identificado en base a dicha caracterización. Este trabajo presenta el Balanceo Predictivo de Encaminamiento Distribuido (PR-DRB), un nuevo método desarrollado para controlar la congestión en la red en forma gradual; basado en la expansión de caminos, la distribución de tráfico y carga efectiva actual; a fin de mantener una latencia baja. PR-DRB monitorea la latencia de los mensajes en los encaminadores, toma decisiones sobre los caminos alternativos a utilizar y registra la información de la congestión en base al patrón de comunicaciones detectado, usando como concepto base la repetitividad de las aplicaciones para luego volver a aplicar la mejor solución cuando dicho patrón se repita. Experimentos de tráfico con congestión fueron llevados a cabo para evaluar el rendimiento del método, los cuales mostraron la bondad del mismo.

Table of Contents

List of figures

List of equations

List of tables

Chapter 1 Introduction

1.1 Overview

The world of the High Performance Computing (HPC) is just a small part of the bigger world of the general Computing Science as we know today, but it is the one that take the most challenging problems, assume most resource demands and it's the leading sector in innovation in the field. HPC main goal is to give answers to the most challenging problems of the real world, including engineering and many other disciplines as well.

A super computer could be defined as a computer that is at the front line of current processing capacity, particularly speed of calculation. This feature also indicates that usually is the most expensive (both at initial and maintenance costs). Generally, a super computer is made by computing nodes aggregation, linked together by an interconnection network. Besides the processing capacity, a super computer generally has other components designed and built to fit the particularly purpose of the equipment such as the interconnection network and the capability of treating the system as a unique entity [1].

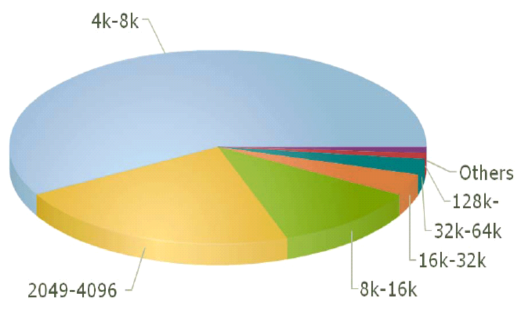
Traditional areas that require a high level of (global) computation had been categorized as "Grand Challenge Problems", and in this categorization are included the most iconic applications in science, as the space-conquer related, weather prediction, geology, decipher of genetic codes, simulation of nuclear experiments, ocean dynamics, and computing vision among many others.

Current requirements of high performance applications are beyond the strict scenario of scientific production. Nowadays the global penetration of the computing in our lives has evolved in such a way that the need of complex (or more demanding) results are in every aspect, such as internet communications, database oriented search, image processing, medical treatments, Enterprise Resource Planning (ERPs), etc. No longer a super computer capable of performing strictly millions of operations per second is the only conception of "super", but a combination of many other factors than affect the overall perception of "fast response" or "correctness and/or precision of the results", such as capacity of processing input data, communications of high amount of data, real time applications and interactions with the user.

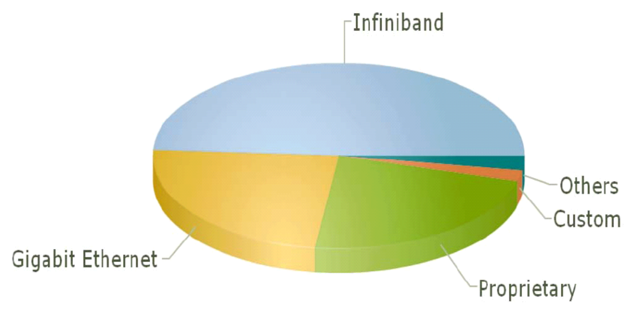
Among these new requirements of a super computer, we can mention the need of an interconnection network capable of move all the data between processing nodes to accomplish the restrictions of speed, delay and overall network performance to applications and users [2].

Combination of top tier engineering for processors, networks and I/O to fulfill the supercomputer concept, also carries other configurations that must be addressed carefully, such as the power consumption, the fault tolerance methodology, programming and operating environments and administrative tasks must also be considered.

The specialized site top500.org [3] has a ranking of HPC supercomputers maintained since 1993 and its updated every 6 months. In order to be listed, any supercomputer must perform under the Linpack benchmark, which offers a floating point operations execution speed characterization, mostly used in HPC applications. Figure 1 *(a)* shows the processing units distribution among all supercomputers evaluated at June 2010. The supercomputer with less processors available has between two and four thousands units, and we can see that some reach extreme high values such as 128 thousands. The interconnection network used to link together this volume of devices must be carefully designed, and efficiently implemented. In Figure 1 *(b)* we can see the interconnect family distribution used in the analyzed supercomputers. Infiniband is the interconnection technology most used, because it is based on High Speed Interconnection Network (HSIN) in mind.



(a)



(b)

Figure 1: Number of processors (a) and Interconnect Family (b) - June 2010

In Figure 2 we can see the evolution of interconnection networks in the last 25 years [3]. From this figure we can conclude that as the processing power is augmenting every year to meet applications requirements, the interconnection network implemented within supercomputers are evolving as well.

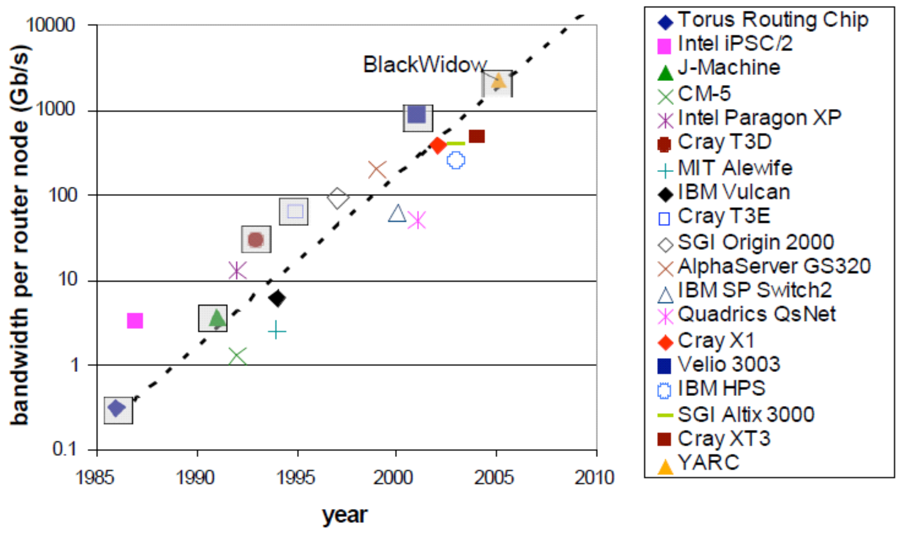


Figure 2: Interconnection network evolution.

In this work, we concentrate on the interconnection network of a high performance computer, because it is considered as one of its key features and a fundamental component. As we have seen with previous figures, supercomputers are evolving, and so are the interconnection network, so continue contributions in this area will have a positive impact over the architecture, and as a direct consequence over applications, in the future.

1.2 High Performance Computers

High performance computers are based on fundamental basic components replication, as processing units and memory to increase overall capacity and processing power. Replicated components are connected by means of an interconnection network, to execute all the task concurrently thus using resources efficiently to tackle some particular problem and expect better results and in less time. Parallel computers have been categorized by the parallelism at instruction level and data level in four main categories [4]. These categories are SISD, SIMD, MISD, MIMD and are explained below and shown in Figure 3.

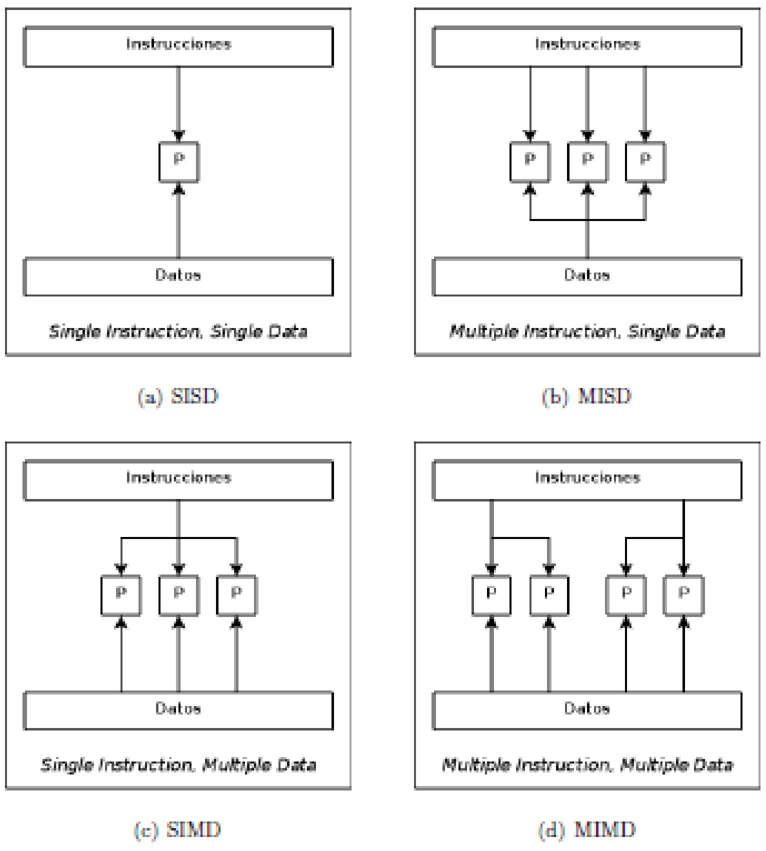

Instrucciones
P
Datos
Single Instruction, Single Data
(a) SISD
Instrucciones
P
P
P
Datos
Multiple Instruction, Single Data
(b) MISD
Instrucciones
P
P
P
Datos
Single Instruction, Multiple Data
(c) SIMD
Instrucciones
P
P
P
P
Datos
Multiple Instruction, Multiple Data
(d) MIMD

Figure 3: Parallel computers classification.

1.2.1 SISD

Single Instruction, Single Data. In this category we can find the traditional computer of the Von neumann architecture, with only one processing unit. This kind of computers have been along the evolution of traditional computers and even the latest hardware available are still basically based on it, although the advent of multi- and many- core systems have changed the situation.

1.2.2 SIMD

Single Instruction, Multiple Data. Parallel Computers with only one thread of execution working with many data flows at the same time are among this category. These kind of computers have only one control unit that governs the whole processor, and many functional units to work simultaneously with the data. They are mostly oriented to mathematics vector problems.

1.2.3 MISD

Multiple Instruction, Single Data. Even this model is theoretically valid, there is not a known physical implementation so far.

1.2.4 MIMD

Multiple Instruction, Multiple Data. Many instructions flows are executing in parallel with many data flows. There are many configurations available to this kind of computing, and is the most advanced technology that has lead the construction of many super computers currently known. Many sub categories can be obtained from the configuration and main usage of all its components, such as message passing parallel computers and shared memory super computers.

1.3 Interconnection Networks

Last sections have introduced the supercomputer concept, and can be summarized as a set of independent processing unit (or processor) linked together by some kind of interconnection network. This kind of schema provides computation power in a distributed shape to solve a particular problem. The problem to be solved most likely will require communication between all the processing units working in the problem, thus the whole system must be capable of automatic task-to-processing-units assignation/mapping, and must offer an efficient communication system to perform this work.

One of the key aspects of a high performance computer, parallel or distributed, is the interconnection between all its components. This interconnection is the one responsible for the effective parallelism achieved, because all the processors available are currently standard units working locally.

The importance of an interconnection network could be based on many items, as explained below:

- Is one of the central elements to build a supercomputer, along with the processing units. Scalability is then an important matter when building this kind of computers.
- Interconnection network features affects directly to the overall performance of the whole parallel computer system. The amount of information transmitted over a period of time is known as the

bandwidth of the network. This bandwidth of offered by each of the communication nodes. The performance of a network is measured as the amount of messages effectively sent over a period of time, known as *throughput.* Another performance concept is the *latency*, which represent the time it takes to transmit a message over the network, assuming the network is not fully connected due to monetary costs and other topological reasons. Some key factors demanded to an interconnection network is the ability to handle high values of *throughput* keeping *latency* values as low as possible.

- The vision of the interconnection network presented to the user, because it defines which model can be used based on network features. This includes practical questions, as if network resources should be handled transparently and if internal configurations available such as messages sizes, communications type (one to one, one to many, etc) should be all hidden to final users.

1.4 Interconnection network problems

Having a great number of communication nodes interacting to transmit a message from the source node towards a destination can lead to traffic unbalance, due to poor packets transmission strategies and inefficient mechanisms to prevent overflow of network resources capabilities.

Traffic unbalance can introduce network situations where the mentioned goals of a interconnection network may no be fulfilled. For example, a routing algorithm is in charge of selecting the best routes to transmit a message over the network, but even when there may be many possible alternative paths to transmit those messages, the routing algorithm may not make proper decisions and thus causing situation where a lot of messages are being sent through some particular nodes in the network, causing a *congestion situation* or *hotspot* when other portion of the network have enough resources to handle the traffic being locked due to poor decisions [5].

Congestion of messages in transit is also a known issue in interconnection networks. This situation appears when there are shared resources in the interconnection network, such as intermediate routers and links, and saturation can be reached if the situation is not controlled properly and in time. When traffic is not being properly handled by the network, then all messages start racing to obtain those resources. This race increases messages transit time, producing *high latency* values in the network in general and thus reducing the overall *system performance* [6] . The implication of these kind of congestion in networks where dropping of packets is not allowed is even more critical. And parallel computers run this kind of networks.

One solution given to these congestion problem is the over provisioning of resources in the interconnection network, avoiding the need of race condition to access to routers resources. This over-dimensioning practice is obsolete in part due the actual cost of network components, respect to processing nodes, and the power consumption associated to this practice [7].

Having mentioned this issues that lead to problematic situations in interconnection networks, we can conclude that over provisioning is not a good solution under any perspective, and with the alternatives given by technology today, efficiency can be obtained by combining different topological approaches, and improving other layers of network protocol stack, such as congestion control and routing, it is considered as an affordable technique.

1.5 Motivation

We have seen that parallel computers performs at many scenarios in current scientific and industrial applications, to solve practical problems and increase human knowledge. Here, the interconnection network pay a fundamental role in the effective performance achieved by a super computer, being part of the very essence of parallelism when acting as an efficient transfer agent of information between all processing nodes.

Carefully designed interconnection network routing algorithms are essential in the process of optimal utilization of communication resources, adapting to situations presented at every stage of processing, and improving the overall performance perception of the system.

With the idea of improving the performance of the whole system and based on the premise that the final user will be the most benefited, newer and better algorithms to properly manage the interconnection network (and others computational resources) are needed. The conception of more specialized techniques to use available resources in order to being able to solve more complicated real life problems, would make science solidly advance one step further, and then also industry will be able to transform the ideas conceived here, into specialized and of practical use.

Initial design of an interconnection network takes into consideration the bandwidth required by the applications running over the network, but a good algorithm must also consider the effect of the traffic dynamics that can lead to unexpected situations due to traffic unbalance and hotspot situations, avoiding performance degradation where possible.

Different areas of scientific real life problems are solved with parallel computer systems. These problems produce, as a consequence of its executions, different traffic load patterns. Parallel applications patterns in this kind of systems posses repetitive behavior throughout execution time [8]. This feature, repetitiveness, should be used by network designers to develop specific systems models accordingly, and try to use information about past behavior to make better decisions about future traffic conditions. This could help to improve performance in the interconnection network, through less time to adapt to unbalanced communications, avoid unnecessary congestion situations, etc.

All the ideas expressed here converge to one simple concept: performance. Routing algorithms must be designed to accomplish that goal as faithful as possible; and line up all techniques available, such as congestion control, flow control, etc; but without letting aside other considerations as power consumption, costs, efficiency and security.

1.6 Thesis goals

As expressed so far in this document, this work is concerned about high speed interconnection networks, at the level of high performance computer systems. We have seen that interconnection networks are an essential component in a parallel computer system, specifically to achieve the general goals demanded to this kind of systems. Our work is based on scientific environments where processing power and communications requirements are extreme. Also, this work concentrates on applications with repetitive behavior. By repetitive behavior we mean the repetitive traffic pattern, known as bursty traffic, and the repetitiveness in program execution flows. Therefore, the main goals of this work can be described as follows:

- Improve overall system performance
  - Perform proper traffic load distribution among all communication resources
  - Avoid unnecessary hotspot situation in the network
  - keep global latency values low
  - Improve total throughput capacity
- Study and analyze specific features of an interconnection network that affect performance
  - Study dynamic features of interconnection networks using specific computing models, aimed to identify the problems during the normal operations of the network (under traffic) and their major causes
  - Analyze existing approaches of path distribution
  - Study parallel applications patterns impact on network´s behavior, and how to establish a relationship between these applications and the routing mechanism to improve performance
- Analyze current routing models and propose new approaches, based on past implementations made at the CAOS group.

1.7 Overview of Content

This chapter has shown a general overview of the field where this work is situated, besides some fundamental and basics concepts related to the world of High Performance Computing. Along with these concepts, the main goals and motivation to start this work is expressed. With these main goals in mind, this document is then organized as follows:

Chapter 2 describes in more detail interconnection networks fundamentals. The basis of an interconnection network is explained, and a general classification of existing topologies and its features is presented. Then, congestion problem is introduced as well as some approaches of how to solve this situation, along with a comparison about congestion control techniques developed through the years.

Chapter 3 Describes the idea of dynamic balancing routing in interconnection network. Details about communication balancing and main features of algorithms based on path distribution is given. Also in this chapter, other important concepts are introduced, such as bursty traffic and parallel applications basic behavior, which are fundamental concepts for this work.

Chapter 4 presents the Predictive module proposal, based on the problematic expressed throughout previous chapters. The selected approach and methodology of the proposal is then explained.

Chapter 5 details the experimentation carried out to validate models and proposed ideas in this work. OPNET simulation tools is basically introduced, and results obtained are explained.

Chapter 6 gathers all the conclusions of this work, and presents basic ideas to work with in the future .

Chapter 2 Interconnection Networks. Concepts and Related Work

Many different systems are currently based on communicating data through interconnections networks, ranging from very specific hardware equipments such as very large scale systems integration (VLSI) to wide area networks. Some applications requiring interconnection networks are Internet switches (IP); interconnection network for multicomputers and memory, and distributed shared-memory multiprocessors; cluster of workstations; local area networks (LAN) and network for industrial applications. This chapter is focused primarily in the interconnection network used in multicomputers and distributed shared-memory multiprocessors.

Technology developed for interconnection network of multicomputers in mind, featured with high performance and reliability, was transferred to distributed shared-memory multiprocessors, thus improving scalability of shared-memory devices. Requirements of high performance and reliability was greater in shared-memory multiprocessors, then improvements in the technology were introduced pushing the development even more. This chain of events happened again when the technology was transferred to local area networks (LAN). In conclusion, advantages in the development of interconnection network for multicomputers are the basis for the development of other interconnection architectures as well.

2.1 Network design considerations

Interconnection networks are critical to the performance of modern parallel computer systems, and there are many factors than influence the choice of an appropriate interconnection network for a particular computer. These factor includes the following, according to [9]:

- *Performance requirements.* Process executing in different processors synchronize and communicate through the interconnection network, hence performance of the network is vital. Common parameters to measure performance are *latency and throughput,* where *latency* is the time elapsed between a message is generated in the source node until it reaches its destination and is delivered. Network *throughput* is the maximum amount of information delivered by the network per time unit.
- *Scalability.* Increasing the number of processors, the overall bandwidth of the system (network, I/O and CPU) increases as well. If this is not true, bottlenecks in performance appears.

- *Incremental expandability.* Parallel computer systems are purchased by parts, and the system must allow the possibility of adding more resources (network components) when needed.
- *Partitionability.* Due the nature of applications running in parallel systems, one execution must not interfere with other execution taking place at the same time, in terms of traffic generated. This is also true when security matters arises.
- *Simplicity.* Designs made simple help the user to understand the architecture and then configure the whole system to exploit performance at maximum level.
- *Distance span.* This factor is related with the distance between communication nodes. Distance variation can cause problems of electromagnetic noise, coupling, etc.
- *Physical constraints.* Packaging of components is related to distance span, and influence in other factors such as temperatures, latency due to distance variations, etc.
- *Reliability and repairability.* An interconnection network should be able to communicate in a reliable way. Besides, it should have a modular design and allow hot upgrades and repairs.
- *Expected workload.* Network design should be robust to adapt to different traffic conditions and always offer reasonable performance.
- *Cost constraints.* Trade off must always been made between all the factors mentioned and the cost of an interconnection network.
- Power consumption. Energy efficient interconnection networks has become and imperative matter in HPC world. This efficiency is related tightly with costs constraints, because it is a significant part of cost increases. Just to mention some examples, there are costs of power and cooling in all communication devices, and as organizations look forward to reduce energy use in their data centers, computing and communications systems can be a focus of the efforts to reduce power consumption.

Many parameters influence in the formal definition of an interconnection network. Some important definitions are the network topology, which is the physical interconnection scheme used to connect all nodes in the network, flow control, which handles the mechanisms to move information from one particular node toward the next in the network, and routing; in charge of find the best path, according to certain constraints, that a packet will use in its journey from source to destination.

2.2 Topology

Topological disposition of nodes in the interconnection network are known as the topology of the network. This represent the path where messages must traverse during communication. Topology is modeled as a graph, where vertex represent nodes and edges represent links between a pair of vertex. A set of parameters defines an interconnection network, as defined below:

- ***Degree:*** Number of edges connecting one node with another.
- ***Diameter:*** Maximum of all minimum distance between a pair of nodes.
- ***Average distance:*** Average distance between any pair of nodes in the network.
- ***Bisection width:*** Minimum links that must be eliminated to divide equally in two pieces the network.
- ***Symmetry:*** A network is considered symmetric if all the nodes in the network look alike from every other node.
- ***Expandability:*** A network is expandable if the procedure of expanding its size is done in a simple way.

2.2.1 Classification

Among other criteria, interconnection networks have been traditionally classified according to the operating mode (synchronous or asynchronous) and network control (centralized, decentralized or distributed). Right now, multicomputers, multiprocessors and network of workstations (NOWs) dominate the parallel computer market.

Classification scheme is shown in Figure 4, which categorized known interconnection networks into four major classes based primarily on network topology: shared medium network, direct networks, indirect networks, and hybrid networks [10].

In *shared medium*, the physical medium is shared by all communicating devices, as seen in Figure 5. Other alternative would be having point to point links directly connecting each communicating device to a small subset of other communicating devices in the network. In this case, communication between non neighboring devices requires transmitting the information through several intermediate devices. These network are known as *direct networks.* Figure 6 and Figure 7 depicts some direct networks. Figure 6 Shows an example of direct orthogonal topologies, also known as symmetric topologies network. Under this classification of direct orthogonal networks, we can find different topologies such as mesh and k-ary n-cube, being the later a special case of meshes called *closed mesh*. Meshes are currently used mostly in super-computers. Meshes are rectangular matrix shaped, in a 2D or 3D configuration, where external nodes are not interconnected. Meshes are easily expandable and messages routing is also simple. Closed meshes are network where external nodes are interconnected to opposite nodes in the network. These topologies are usually known as "k-ary n-cube", where n is related to dimension and k to the number of nodes per dimension. For example, if n=2, networks topologies are called *torus,* and if k=2, they are called *hypercubes.* Another categorization of direct network are the non orthogonal topologies. Figure 7 Depicts the tree network topology, as an example of non orthogonal networks.

Instead of directly connecting communication devices to each other, interconnection can be done by means of one or more switches, where several switches are connected to each other through point to point links, this concept is known as *indirect networks,* as shown in Figure 8. Crossbar topology of size NxM is shown in Figure 8 (a). Crossbar network allow communications between any pair of processors or memory unit simultaneously. Figure 8 (b) show a multistage network topology (MIN), where input devices are connected to output devices through series of switches organized in stages. Each switch in this configuration is a crossbar. Number of stages and connection patterns available determine the overall routing capabilities of this scheme. MINs are used when hundreds of processors are interconnected in a parallel super-computer. Figure 8 (c) shows another indirect network topology, known as fat-tree where. Fat-tree is represented in a tree structure where links weights gets thicker as they go toward the root node.

Finally, hybrid topologies are possible by combining direct and indirect approaches. This networks increment bandwidth available in respect to shared medium networks, and reduce distances between nodes in respect to to direct and indirect networks. Hybrid network topology example is given in Figure 9.

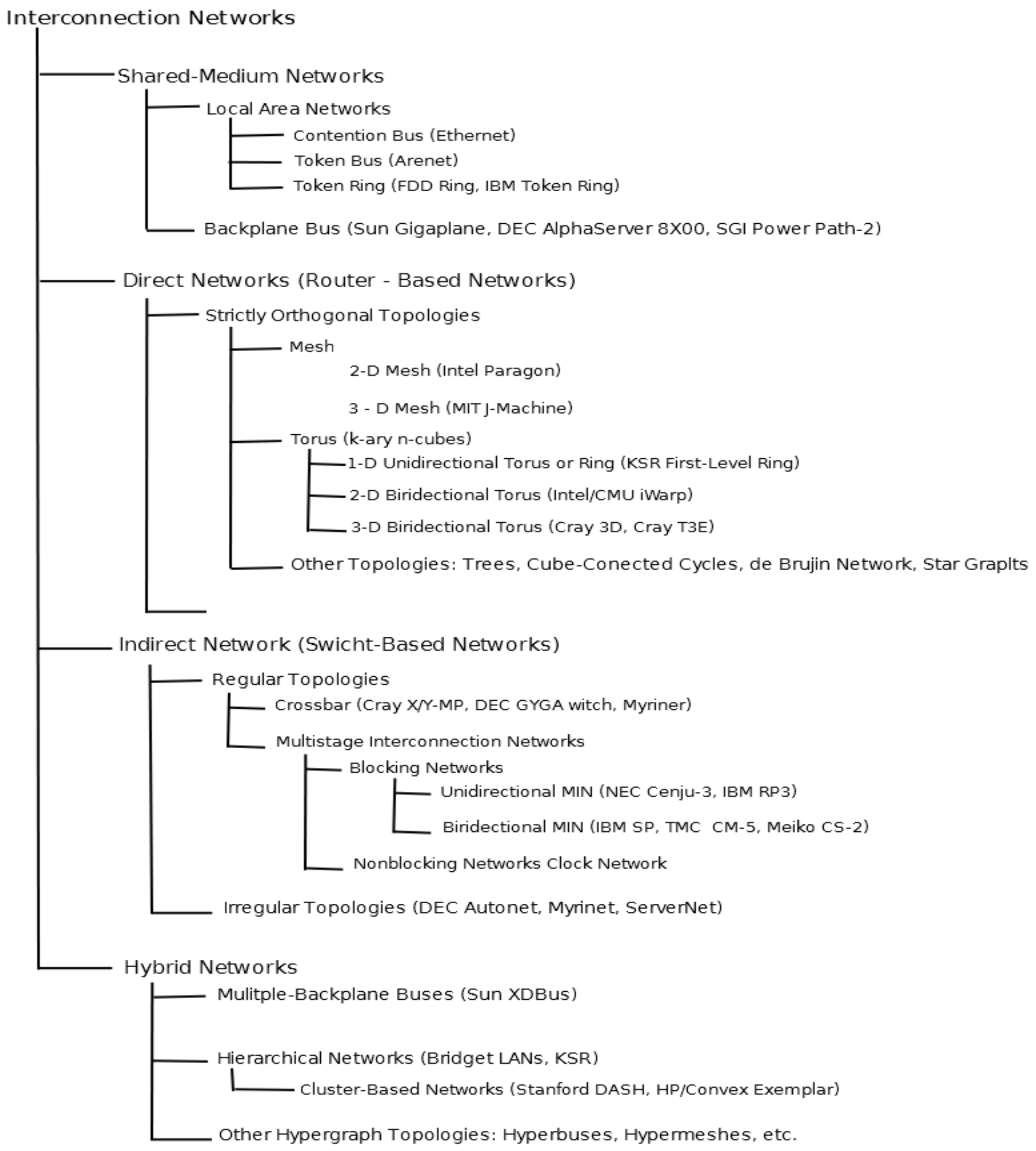
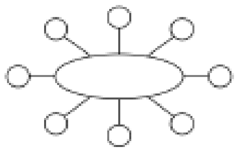

Interconnection Networks
Shared-Medium Networks
Local Area Networks
Contention Bus (Ethernet)
Token Bus (Arenet)
Token Ring (FDD Ring, IBM Token Ring)
Backplane Bus (Sun Gigaplane, DEC AlphaServer 8X00, SGI Power Path-2)
Direct Networks (Router - Based Networks)
Strictly Orthogonal Topologies
Mesh
2-D Mesh (Intel Paragon)
3 - D Mesh (MIT J-Machine)
Torus (k-ary n-cubes)
1-D Unidirectional Torus or Ring (KSR First-Level Ring)
2-D Biridectional Torus (Intel/CMU iWarp)
3-D Biridectional Torus (Cray 3D, Cray T3E)
Other Topologies: Trees, Cube-Conected Cycles, de Brujin Network, Star Graplts
Indirect Network (Swicht-Based Networks)
Regular Topologies
Crossbar (Cray X/Y-MP, DEC GYGA witch, Myriner)
Multistage Interconnection Networks
Blocking Networks
Unidirectional MIN (NEC Cenju-3, IBM RP3)
Biridectional MIN (IBM SP, TMC CM-5, Meiko CS-2)
Nonblocking Networks Clock Network
Irregular Topologies (DEC Autonet, Myrinet, ServerNet)
Hybrid Networks
Mulitple-Backplane Buses (Sun XDBus)
Hierarchical Networks (Bridget LANs, KSR)
Cluster-Based Networks (Stanford DASH, HP/Convex Exemplar)
Other Hypergraph Topologies: Hyperbuses, Hypermeshes, etc.

Figure 4: Classification of interconnection networks [10].


(a) Local area network.

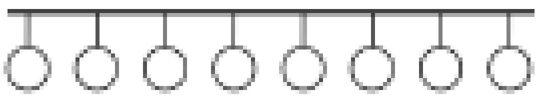

(b) Bus network.

Figure 5: Shared medium topologies.

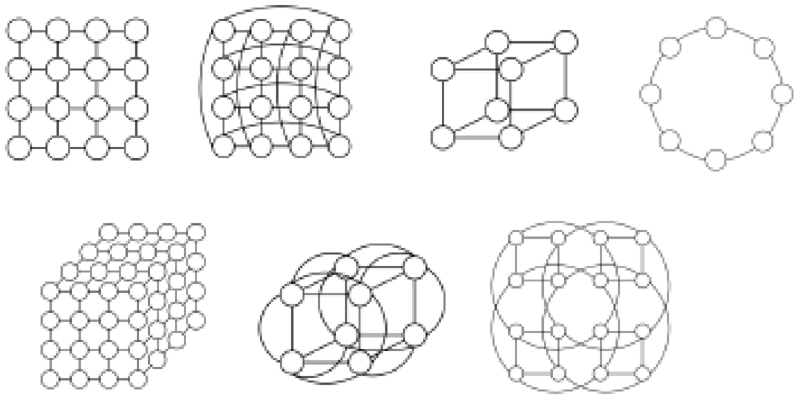


Figure 6: Direct orthogonal networks.

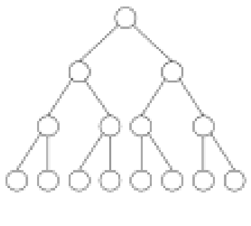


Figure 7: Direct non orthogonal network.

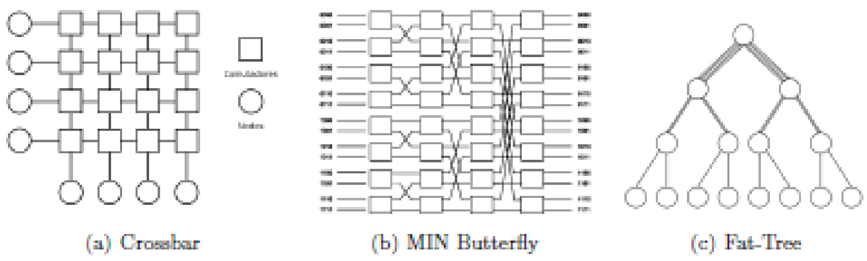


Figure 8: Indirect networks.

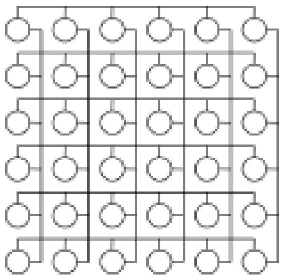


Figure 9: Two dimensional hypermesh, hybrid network topology.

2.3 Message switching layer

Many services are involved at inter-process communications, and this services provides many layers that perform some specific task such as transference of bits streams, higher protocols communications between layers, packetization, compression an more. While not standardized, three layers are the most distinguished in the operations of the interconnection network: these are the *physical layer, the switching layer* and *the routing layer.* Physical layer identifies link-level communications to transfer messages and manage physical channel and other hardware components. Switching layer utilizes the physical layer protocols in order to implement mechanisms for messages forwarding through the network. Routing decisions, selection of output channels and hence the path through the network, is the main goal of the routing layer.

Switching techniques most accepted at the high performance computing world are: *store and forward, virtual cut through and wormhole.*

Store and forward: This technique store the whole packet at the router node, and once it has all the bits of the corresponding message, then proceeds with the transference, if the output port selected is not busy, to other router in the network. One of its main disadvantages is the high latency value of a message in transit, due to the fact that it must wait for the whole message to be stored. This latency value is calculated proportionally to the size of the message and the amount of routers traversed.

Virtual cut through: This technique does not wait that the whole message is stored at internal buffers to be transmitted. The early inspection of the header enables immediate forwarding of message bytes as soon as the routing decisions have been made. If output channels are full, then this technique must have enough space to handle all the message in its internal buffers. In this case, its behavior is the same as the store and forward.

Wormhole: Instead of saving all the packets in its own internal buffers, wormhole can have its packets stored temporary in others previous routers traversed before. Because the only one that contain information about the destination node is the header, if it is blocked in one router, then all the other packets will have to wait as well, even if they have resources to move forward. Other negative item is the requirement of both input and output buffers to handle a whole message. One of its main advantages is the low latency value of packets traversing the network, and the minimum temporal store capacity required.

2.4 Flow Control

Currently communication devices have input and output buffers to store sent and receive data, so that it can process it without any loss of data. Flow control is tightly coupled with buffer management techniques, in determining how to manage the use of such buffers. With the availability of a flow control mechanism, notification of buffers space can be sent to the source node based on the remaining capacity. With this purposes in mind, two major classification exists credit based and mark based:

Credit based flow control: This mechanism assign credits to each router in the network, in order to allow message transfer to other neighbors routers. Each transmission decrement the credit count and once the router runs out of credit, then it stops transmitting.

Mark based flow control: Possible buffer overflow is detected at the receiver node, and then it proceed with the notification of the situation to the source node to stop transmitting. When the receiver has enough space to resume communication, it sends a control message informing of this situation to the sender node.

2.5 Routing

Routing algorithms establish the path followed by each message or packet. The procedures of how to determine every path is based on different network parameters. To reach certain destination, a message may traverse many "hops" or intermediate routers.

To accomplish this task, the routing unit must know the network topology and select proper path based on it. Caution must be taken into consideration when selecting those paths, to try to balance communication and not leaving zones in the network without traffic load whatsoever, while others are practically under congestion and even saturation.

Some properties found in most routing algorithm are correctness,simplicity, robustness, stability, justice and optimization [11]. Correctness and simplicity implies the algorithm have to be computationally simple, because routing decisions must be made in short periods of time. Ability to adapt to topology changes because of nodes failure, is considered robustness. Convergence to a stable situation, if an adaptive technique is used, is considered stability. The property of justice is accomplished when access to resources, such as channels and links, is given under equal conditions to all demanding nodes. Optimization can be achieved in two ways, by minimizing global latency value of the interconnection network, or by global efficiency maximization. These two goals often contradicts each other, because maximizing efficiency often means to insert more packets into the network, and this situation can lead to higher waiting times for packets, hence increasing latency, unless proper mechanism of congestion control are used.

Table 1 Presents a taxonomy for routing protocols. Routing algorithms can be classified according to several criteria as expressed in [10]. They can be first classified by the number of destinations of a message, being unicast when there is only one destination and multicast when involves collective communications. Routing algorithms can also be classified according to the place where routing decisions take place. If one central node is in charge of all decisions then is called *centralized*, *source routing* if decisions are taken at the source node before packet injection and *distributed routing* if decisions are performed while the message is traversing intermediate routers in the network. *Multiphase* is a concept that mixes both previous schemes. Implementation of routing algorithms can also be done in different ways. Most common approaches are the use of a table to look up routes to be uses *(table lookup)* or using hardware of software approaches according to a *finite state machine.* Both cases can fall into the categories of *deterministic* or *adaptive.* Deterministic algorithms always chooses the same path between a pair of source and destination. Adaptive algorithms takes into consideration the status of the network at any time, to choose the best routes based on congestion situations, channel allocations, latency values, etc.

Adaptive routing algorithms can be categorized as a *progressive* or *backtracking.* Progressive always make decisions at each routing operation, while backtracking can go back and deallocate resources previously allocated.

At lower level, routing algorithms can be classified as according to their minimality as *profitable* or *misrouting.* Profitable only supply channel that bring the packet closer to its destination. Misrouting algorithms may supply channel that send a packet away from its destination. At the lowest level can be classified according to the number of alternative paths as *completely adaptive (*also known as fully adaptive) or *partially adaptive.*

Routing Algorithms	
Number of destinations	
• **Unicast**	
• **Multicast**	
Routing decisions	
• **Centralized**	
• **Non Centralized**	• **Source routing**
	• **Distributed routing**
	• **Multiphase routing**
Implementation	
• **Table lookup**	
• **Finite state machine**	
Adaptivity	
• **Deterministic**	
• **Adaptive**	
	Progressiveness
	• **Progressive**
	• **Backtracking**
	Minimality
	• **Profitable**
	• **Misrouting**
	Number of Paths
	• **Complete**
	• **Partial**

Table 1: Taxonomy for routing protocols.

2.6 Congestion control

The correct behavior of the interconnection network can perceptibly affect the performance of the entire parallel system, and the applications running on the computer. If a node starts heavy communications procedures during parallel application execution, this request can also stress the network, because it could lead to more resources demanded to the interconnection network to satisfy application communications requirements. When this situation occurs, control must be performed and only one packet must be allowed to use the resources, such as an outport, and the remaining packets must wait until the arbitration unit allows them to move forward and use those resources. The packets waiting due to arbitration decision are in a so called *contention* state. Accumulation of high number of packets in *contention* state can lead to buffers overflows and even reach to saturation state, where no more messages are allow to enter the specified router. Under extreme saturation, latency can climb to unacceptable values. Latency behavior of packets can be represented as shown in Figure 10. The x axis is represented by the traffic load offered by the interconnection network (Offered *Load*), and is normalized and measured as [bits x seg x node]. This offered load is the amount of traffic offered by processing nodes (O*ffered Load)* that is expected to be handled by the interconnection network. When traffic is low the network can manage the received load and send all messages to destination, without bigger problems (point where *Accepted Load = Offered Load).* This situation is not the same when traffic in the network start increasing. This is produced because the communication volume demanded to the network surpasses network capacity, imposed by topological and technological constraints. If the increasing process remains a considerable time, then this situation can lead to a complete saturation of the network (point where *Accepted Load = 0.15 x Offered Load)*.

A practically linear behavior can be seen in latency values, corresponding to traffic loads not surpassing maximum capacity of the network. This latency values remain practically unchanged even under big changes in traffic loads, meaning the network can accept all the traffic produced by all processing nodes communicating. This zone of the curve delimits the *working zone* of the interconnection network, where the maximum performance may be achieved. If traffic injection keeps raising, then latency curve behavior changes considerably, even when traffic injection rates are minimal. At this point the congestion zone is perceived and the traffic load is not fully managed by the interconnection network. The high values in latency is caused by the fact that messages are waiting at internal buffers to move into the network, as mentioned before. Working under on this zone is not desirable due to unpredictable latency values. Desirable working zones are between 0 and approximately 500 [bits x seg x node], for this example.

2.7 Congestion Control Analysis

Routing algorithm proposed in this work pretend to augment working zone under the same conditions. Congestion situation can be presented even when traffic injection does not reach its maximum value, and network is far from its maximum capacity, produced by inappropriately traffic load distribution causing saturation in some portion of the network. This inappropriately traffic distribution in some links can introduce high latency compared to a situation where the entire network is saturated [12], [13].

Latency values can affect directly on applications performance, because under congestion situation they may experience high response time, timeouts, etc. leading to processing units being idle for considerably periods of time.

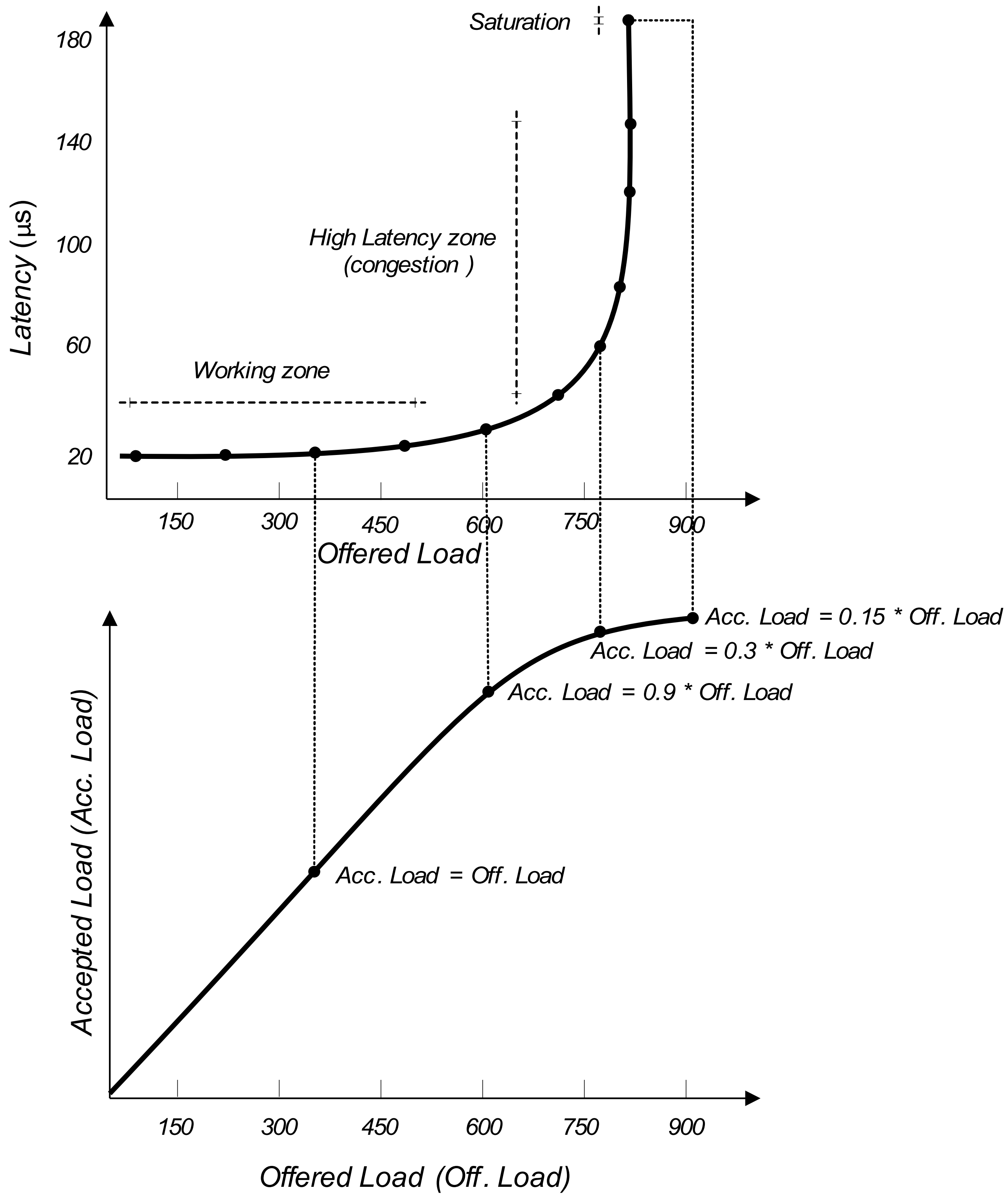


Figure 10: Offered/Accepted load (bits x seg x node)

2.7.1 Approaches to solve congestion control problems

Two major approaches exists to tackle congestion situations, preventive and reactive techniques. Each of them has its own set of advantages and disadvantages for particular congestion situations explored.

Preventive techniques involve solutions based on the concept of increasing resources available in the system (over dimension) thus avoiding congestion by allowing communications to perform without interruption.

A necessary condition for this technique to work is that application behavior is to be known beforehand, so that resources to fulfill the task can be properly estimated.

One disadvantage of this method is the inability to always determine *a priori* resources required for an application. Also, reservation of resources is not an optimal solution because it can lead to inefficient network usage and also cause congestion, caused by the process of search and reservation of resources itself, which is performed through packets injection within the network.

This solution, though simple, could be very difficult to implement due to current network features. Over dimensioning of resources is not a valid proposal based on communication devices cost, compared with the processing power. Another point to take into consideration, and now more institutions are paying attention to this subject, is power consumption. Deliberately including more resources can lead to highest power consumption, and in super-computers where there are thousand of processing nodes and no less communication devices, this can be a serious item to study.

Reactive techniques embraces mechanism to use available resources properly, adapting to adverse traffic using no more than the minimum required resources. To accomplish this task, a process of analyzing network status during communications is performed, and when network situation is approaching congestion state, notification procedures are executed in order to alert injecting nodes about the situation, and let them know that some strategy to control this problem has to be done. So, three main phases can be extracted to perform the congestion control:

- Network monitoring and detection of congestion situations
- Notification about the recently found congestion to injecting nodes
- Execute corrective procedures in order to effectively control the congestion situation

During monitoring and detection phase, some features of the interconnection network are analyzed to determine congestion situation somewhere in the network. Latency values and flow speed implicit in congestion (also known as back-pressure) are metric analyzed during this phase. Decisions can be made globally or locally. Globally wide decisions are more accurate than local decisions, but introduces higher overhead into the network as well, because of synchronization between all nodes to make those good decisions.

Notification phase can be implemented with variations according to the notification scheme used. One alternative is when there is no notification at all, in this case the node under congestion try to solve the problem by itself, adjusting internal configuration parameters as internal buffers, ports, etc. Other alternative is to notify about congestion only to closer nodes in the network, neighbors, which will try to solve the congestion situation. Notification to sender nodes is other approach, based on the idea that sender nodes caused the congestion and they should be responsible for taking proper actions to regulate traffic injection, hence solving the congestion situation detected.

One last obvious approach could be notification to all nodes in the network about congestion in a particular point, expecting that every node receiving this notification take actions to try to solve the situation unleashed previously. The problem with this technique is the overhead involved in traffic load, to communicate information about the congestion to all other nodes, and their response and actions taken to try to solve the situation.

Once congestion have been detected and notified, actions to control the situation must be executed. A technique that regulate traffic at the source node during congestion is known as *Message Throttling [14],* stopping injection during the congestion, or slowing down new packets generation. This technique, even alleviating congestion, introduces high latency valued into the network due to message generation regulation.

Another possibility is to try to organize and manage internal router buffers, re-ordering packets of different flows to avoid collision. The essence of this technique is to try to transmit packets that are not under congestion, but share resources with other that indeed are, as fast as possible. While congestion is reduced at the router applying this procedure, the problem is that congestion is not controlled at the injecting node, thus no giving a solution at the core of the problem [15],[16].

Congestion control based on path or load distribution alter original communication paths based on the status of the network, in order to avoid network areas under congestion. These techniques inject packets destined to a node through a series of alternative paths, to keep latency values low without reducing traffic load levels. Fair traffic load distribution is accomplished using this kind of algorithms because traffic is sent over idle or less burdened nodes in the network, thus avoiding congestion and as a consequence improving performance by avoiding congested spots in the network.

2.7.2 Routing and Congestion control techniques

Figure 11 and Figure 12 depicts a classification of major routing and congestion control techniques in the last years. The type of routing used and the behavior based on each phase of the mechanism (monitoring and detection, notification, and decision). Figure 11 Figure 12 and show a chronological time evolution towards adaptive or hybrid routing algorithms, some of them are AI and AL [17], RCA [18], DyAD [19], HSAM [20], PIPD [14], DEF/R [21], GOAL [22], RECN [23], GAL [24], CQR [25], INC [7], 2-Turn [2], BLAM [26], Self Tunned [27], Ping Bubble [28], Bubble R. [28], ALO [7], U-Channels [7], DRB [29], [30] , Turn Model [31].

Other algorithms widely used were the *oblivious* but because their poor latency response were mostly deprecated and replaces with algorithms that take into consideration the status of the network to make routing decisions. Between the most known *oblivious* routing techniques are Valiant [32], CHAOS routing [33], DOR [10] among others.

Buffer occupation time or message waiting time are the preferred monitoring techniques for adaptive routing strategies. In respect to the way the notification is done, algorithms performing it at link level are the most encountered, mainly because are much simpler to implement and use less resources. Finally, corrective actions of algorithms can be observed, where traffic load distribution is the preferred option because of better network utilization and better latency values. Nevertheless, message injection regulation is also a good alternative due to simplicity in its implementation and low cost.

A summary, at the bottom of Figure 12, shows global utilization results for each of the described technique in Figure 11 and Figure 12 .

✓	Supported
○	Not supported

Routing/Congestion Technique						Detection			Notification			Decision		
Name	Description	Oblivious	Adaptive	Hybrid	Year	Queue occupation	Back-pressure	Latency	Local	Link	Global	Regulation	Queue management	Adaptive
AI	Adaptive Injection (AI), utilize pipe-lined architecture and multi-layer networks. Adaptively selects a layer to inject a packet according to network status, and after injecting the packet, deterministic routing is	○	✓	○	2008	○	✓	○	✓	○	○	○	○	✓
AL	Adapive Layer Selection, similar to AI, but a packet can change layers during its delivery.	○	✓	○	2008	○	✓	○	✓	○	○	○	○	✓
RCA	Lightweight technique to improve global network balance. Instead of relying solely on local congestion information, RCA informs the routing policy of congestion in parts of the network beyond adjacent routers.	○	✓	○	2008	○	○	✓	✓	○	○	○	○	✓
DyAD	*DyAD* from **Dy**namic **A**daptive **D**eterministic switching, is based on the current network congestion. Each router in the network continuously monitors its *local* network load and makes decisions based on this information. When the network is not congested, a *DyAD* router works in a deterministic mode On the contrary, when the network becomes congested, the *DyAD* router switches back to the adaptive routing mode and thus avoids the congested links by exploiting other routing paths; this leads to higher network throughput which is highly desirable for applications implemented using the NoC approach.	○	✓	○	2004	○	○	✓	✓	○	○	○	○	✓
HSAM	Design of a MPI functionality, which provides hot-spot avoidance for different communications,without a priori knowledge of the pattern. Designed to be compatible with InfiniBand standard. Create multiple paths	○	✓	○	2008	○	○	✓	○	✓	○	○	○	✓
PIPD	For InfiniBand networks, with features such as no packet dropping, small buffer size and low latency. Improved Explicit Congestion Notification (ECN) packet marking mechanism for InfiniBand. Proposes an effective source response functionPower Increase and Power Decrease (PIPD) which adopts the rate control with a window limit to reduce congestion of multiple-class traffic in InfiniBand networks.	○	✓	○	2006	✓	○	○	○	✓	○	✓	○	○
DEF/R	Deflection routing for networks on chip. A deflection switch can be much smaller and faster than a wormhole or virtual cut-through switch. A deflection-routed network has three or- thogonal characteristics: *topology*, *routing algorithm* and *de-flection policy* . Upon contend-ing for links, packets with a lower priority will be *misrouted* to unfavored links according to a *deflection policy* .	○	○	✓	2006	○	○	✓	○	✓	○	○	○	✓
GOAL	Globally Oblivious Adaptive Locally – Provides high throughput on adversarial traffic patterns, matching or exceeding fully randomized routing and exceeding the worst-case performance of Chaos,and other algorithms. GOAL also preserves locality, and achieves global load balance by randomly choosing the direction to route in each dimension. Local load balance is then achieved by routing in the selected directions	○	○	✓	2003	✓	○	○	✓	○	○	○	○	✓
RECN	congestion management strategy for lossless multistage interconnection networks. Instead of eliminating congestion, this strategy avoids performance degradation beyond the saturation point, by using separate queues for congested flows	○	✓	○	2005	✓	○	○	○	✓	○	✓	✓	○
GAL	Globally Adaptive Load-Balance (GAL). GAL makes global routing decisions using global information. It senses global congestion using segmented injection queues to decide the directions to route in each dimension. It further load balances the network by routing in the selected directions adaptively	○	✓	○	2004	✓	○	○	✓	○	○	○	○	✓

Figure 11: Congestion control techniques classification.

✓	Supported
○	Not supported

Routing/Congestion Technique						Detection			Notification			Decision		
Name	Description	Oblivious	Adaptive	Hybrid	Year	Queue occupation	Back-pressure	Latency	Local	Link	Global	Regulation	Queue management	Adaptive
CQR	Channel Queue Routing (CQR), for load-balanced routing on k-ary n-cube interconnection networks. CQR estimates global congestion in the network from its channel queues while relying on the implicit network backpressure to transfer congestion information to these queues. It uses this estimate to decide the directions to route in each dimension. It further load balances the network by routing in the selected directions adaptively.	○	✓	○	2004	✓	○	○	✓	○	○	○	○	✓
INC	Injection and Network Congestion (INC), checks message advance speed in order to detect network congestion. Each virtual channel has an associated counter that increases each time one it is transmitted. At the end of each interval, the channel is considered congested if it is busy and the number of transmitted its is lower than a threshold *fc* (*it counter*). Once congestion is detected, the applied policies are different depending on whether the node is currently injecting messages towards the congested area	○	✓	○	2005	○	✓	○	○	✓	○	✓	○	○
BLAM	Bypass Buffers with Limited Adaptive lazyMisroutes (BLAM). BLAM achieves Chaos-like performance by allowing packets to be "lazily" misrouted outside the minimal rectangle. Lazy misrouting is critical to BLAM's performance because eager misrouting can misroute unnecessarily, thereby degrading performance.	○	○	✓	2003	○	○	✓	✓	○	○	✓	✓	○
Self-Tunned	This design is composed of two key components. First, global information about a network to obtain a timely estimate of network congestion. Then, comparison of this estimate to a threshold value is done, to determine when to throttle packet injection. A combination of these two techniques provides high performance under heavy load, does not penalize performance under light load, and gracefully adapts to changes in	○	✓	○	2001	✓	○	○	○	○	✓	✓	○	○
The adaptive bubble router	The design of a new adaptive virtual cut-through router for torus networks is Presented. This has been achieved by combining a low-cost deadlock avoidance mechanism for virtual cut-through networks, called Bubble flow control, with an adequate design of the router's arbiter.	○	✓	○	2001	○	○	✓	○	✓	○	✓	○	○
ALO	At Least One (ALO), the mechanism allows injection if, after applying the routing function, at least one virtual channel of all the useful physical channels is free or at least one useful physical channel have all its virtual channels completely free.	○	○	✓	2000	○	✓	○	✓	○	○	○	○	✓
U-Channel	Useful-Channels, detect congestion by tracking the number of busy virtual output channels. If this number exceeds a threshold, congestion is detected and message injection is completely stopped.	○	○	✓	2000	✓	○	○	✓	○	○	○	○	✓
DRB	Distributed Routing Balancing (DRB), uniformly balance communication traffic over the interconnection network, based on limited and load-controlled path expansion in order to maintain a low message latency.	○	✓	○	1999	○	○	✓	○	✓	○	○	○	✓
Turn Model	Presents a model for designing wormhole routing algorithms that are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive.	○	✓	○	1992	○	○	✓	✓	○	○	○	○	✓
Global Summary	Summary of most used approaches for each category.	○	✓	○		○	○	✓	✓	○	○	○	○	✓

Figure 12: Congestion control techniques classification (cont.).

Chapter 3 Dynamic Routing Balancing Analysis

3.1 Introduction

This chapter introduces the alternative path distribution concept and its related definitions. It is first described the situation of hotspot in the interconnection network, and its effect when is not controlled. Then hotspot situations impact over application is introduced, and also the need for correct balance of communications. A set of definitions to accomplish the balance of communications and creation of alternative paths are latter described. At the end of this chapter, the algorithm this work is based on is introduced, and further analyzed its weakness in particular situations of hotspot in the interconnection network.

3.2 HotSpot Analysis

Network behavior can be analyzed under different traffic distribution going from benign traffic [22], where most communications are performed among neighbors nodes, to adverse traffic [22], where lack of locality is found and communications are performed among distant nodes. As a definition to be used below, a channel is a set of logical nodes interconnecting process between sources and destinations, and those process communicate using only this channels abstraction. The extreme situation of adverse traffic at high load is known as **HotSpot**, where some resources on the network are overloaded while others are idle. HotSpot can produce very high latency values compared with zones in the network that have not reached load saturation state, state where most messages injected into the network do not reach its destination nodes. Many situations can contribute to hotspot appearance, such as message frequency, channels load (logical nodes interconnecting process loads between sources and destinations), message length among others. Message frequency is related to the packet generation rate, where high traffic load represents the fact that every message is injected into the network practically without interruption, one after another. Channel load is considered when many different source nodes in the network collide at some common locations. The size of the message is also a matter of hotspot analysis. For packetized systems, longer messages means more packets injected.

Latency values in hotspot situations can be viewed as a non linear curve, where two distinct zones can be easily recognized. The first zone represent a low traffic load situation, where all networks requests are completely fulfilled without resources penalization and latency is mostly linear. Changes in traffic load during this zone do not increment, or slightly change, latencies values. In this situation the network is not fully loaded and some resources may be idle.

The other working zone represents a more sensitive situation, where subtle changes in traffic load can increment latency values beyond a value than the network can tolerate. The curve in this situation is a sharp slope, and it means that network state is near saturation, and messages must wait inside routers buffers before been injected into the network or during their journey to their destination.

The transition between both zones, linear and slope, clearly shows the point where the network enters a saturation state, if the network is exposed to more traffic load injection. The point where the transition occurs is called ***threshold load***.

When the network is incapable of accepting more traffic, latency can grow uncontrollably to extremely large values. This situation is considered as undesirable because the overall performance of the network, and hence the applications using it, is diminished.

Although all the factors that can produce a hotspot situation are important, one that we take into consideration is the collision of channels in their path toward their destinations. When more nodes involved at the collision, then higher latencies values emerged, and this increase in latency is not necessary linearly with the number of nodes in the hotspot situation.

One particular and interesting effect of hotspot situation is the ***domino effect*** of latency propagation, where one particular zone of the network where the hotspot appears can cause a global effect of higher latency across the entire network, even though the whole network is not initially collapsed. This effect introduces a wrong perception of the network behavior, where it may seems that is entirely under congestion and not capable of accept any more traffic in the whole network, when there are plenty of resources available to handle the communication load efficiently. This erroneous behavior is due in part to a bad communication pattern distribution, and caused by only a small group of nodes with high traffic loads.

3.3 Interconnection network impact over applications

Applications that are executed in an interconnection network may suffer from the hotspot situations mentioned earlier, and thus considerably decrease their performance. A typical parallel program consist of many independent task running on different nodes, and make use of a message-passing strategy to synchronize and communicate their data among all participant nodes.

Total execution time could be estimated as the time it takes to an application to perform all the calculations on his own data, plus the communication time involved to pass all the information among nodes. Hence, the latency suffered by a message is critical to estimate the overall parallel application execution time , due to the fact that some nodes may wait to receive their next portion of data to work with. Another point of view to avoid consideration of the communication time, could be overlapping computation and communications, but to make this overlapping work efficiently latency values are to be known and controlled, otherwise it could lead to long waiting times.

Other factors, which may be different among applications, come into consideration when analyzing the latency impact over applications. In compute intensive applications the main goal is to distribute tasks among nodes and as a consequence minimize the overall execution time. When there is only one metric to work with, in this case the overall execution time, it is considered a ***best-effort*** strategy.

Applications task distribution is performed with the idea of processing resources maximization in mind. Under this distribution, most of the time is expected to be dedicated to perform calculations and no processor become idle for a long period of time. Execution time of any task is very difficult to estimate, due in part to functional dependencies among all task that are executing concurrently and the traffic and communications load, therefore making it hard to perform an optimal task distribution and allocation of resources. We are concerned about to the communication time, which is a complicated matter due to the latency situations under hotspot mentioned earlier.

If the network is under saturation, small changes in traffic load can produce a global state of congestion and the communication between all nodes will be penalized and waiting times could increase. As a result here, the optimal distribution of task to processing nodes would be inefficient. According to this situation, the major goal is to maintain latency values under desirable values and avoid the slope of congestion where latency behavior could be unpredictable.

3.4 Communication balancing

Previous sections have described the interconnection network behavior under different hotspot traffic load patterns. One of the main goals of an interconnection network is the ability to connect efficiently all component nodes of a supercomputer. Efficiency can be represented as some particular features as the linear interconnection cost while adding more processing nodes, linear increase of capacity or bandwidth according to the amount of nodes and a controlled/minimal latency of messages. On the other hand, one of the main feature also demanded to the network is the maximization of messages delivered to their destinations per unit time, known as the ***network throughput*** [34]**.** Latency and throughput appear to have contradictory goals, as we can see that when more messages are injected into the network, then more latency is perceived at network level. The opposite occurs when the traffic load is kept low, and latency values are small. Whit this is mind, some form of balance must be defined where latency values are kept low and uniform even under high traffic load. This balance is to be maintained under all situations and should be independent of the paths where a message is sent, the distance between source and destinations nodes or traffic load .

To avoid hotspots areas in an interconnection network the communication load must be distributed evenly among all nodes of the network, similarly to tasks distribution in compute bound traffic loads.

Communication balance across the network is close related to the concept of uniform and predictable latency and the avoidance of hotspot situations. All of these concepts are tightly coupled and a reference to one of them is also related to the other in one way or another. If there are no major variations in latency values, then it is assumed a uniform state, and this condition can lead to a reasonable prediction about the latency in all the network if the traffic load is known globally. A correct distribution of the traffic load across the entire network effectively avoid hotspot situations, because traffic is flowing through underloaded communication nodes, and the ones subject of congestion will be less loaded and as a result network resources will be better used because the network is not collapsed.

3.5 Algorithms based on path distribution

This section analyze the main features of a routing algorithms used to fulfill the goals mentioned in the previous sections, such as an uniform latency value and the balance of communications. The task of selecting the path, or paths, that a message is going to use towards its destination is responsibility of the routing algorithm. During the message traversal, it can go through many intermediate nodes depending on information about the state of the network in that particular time, in case of an adaptive routing algorithm.

A routing algorithm present many characteristic features such as simplicity, speed, robustness, connectivity, justice, optimization, etc., and its main goal is to guarantee the combination of links that represent a path to a destination node with the lowest possible latency value under certain traffic constraint. A routing algorithm must perform as quickly as possible, so it is not recommended to be very complex and always maintain simplicity. Robustness is achieved when the algorithm is capable of working with its main features under different traffic patterns and conditions, without major performance degradation. Connectivity is given by the availability of at least one valid path between any pair of nodes. Justice is given by the fact that access to resources must be equally provided to all contending flows. Optimization is the concept of commitment solution when there are two or more features that are desirable but contradictory to each other, such as minimizing latency values and increasing traffic load.

To achieve these features mentioned as desirable, our proposal is based on the balancing of the communication load in a reactive manner, that is, subject to current network state to react properly under unbalanced traffic conditions. The method creates alternative paths to the original one between any pair of nodes that are under adverse traffic conditions and latency values. The idea is to distribute the load between many individual alternative paths. These new paths uses less loaded or idle resources available to handle the communications, and they can be of minimum distance like the original path or can have more hops if necessary, always under strict control of the algorithm.

According to definitions of these kind of reactive algorithms mentioned in section 2.7.1 , there are three major phases to tackle the situation.

The first phase continuously *monitor* the network state for congestion situation due to traffic unbalance. Also in this stage the congestion is *detected* and then *notified* back to the source node about the situation encountered. With this information the source node will try to adapt the traffic injection according to the balancing policies. The second phase is actually where the balancing function is *decided* according to the congestion detected. Here, alternative paths are arranged to distribute the congested paths traffic. The third and last phase corresponds to the mechanism to control the effective path expansion, in a controlled way, using the paths created in the previous phases. With this approach the state of the network is taken into consideration constantly, and acting locally based on information of the whole network finally produces an effect in the whole network.

3.5.1 Intermediate nodes

The idea of traffic balancing in this work is based on path distribution, according to the particular topology used. The path distribution is performed for every source/destination pair involved in the network communications, if under a situation of high latency or congestion. The basic idea is based on the presence of intermediate nodes between the source and the destination, to act as bridges towards building the new paths. According to this approach each message must traverse several stages to reach its destination. The number of intermediate nodes chosen is two, and the reasons for this number will be explained later on this chapter. With two intermediate nodes the path towards the destination is then divided in three specific pieces, one from the source to the first intermediate node, then communicating the first and the second intermediates nodes, and the last from the second intermediate node to the destination. Minimum static path is used for each of the steps mentioned before.

This distribution pretends to use different alternatives to the original path established by the static routing. This strategy also tries to distribute the traffic uniformly across many alternatives in the network, even though the parallel application running over the network has a specific pattern that overloads some destination nodes in particular, as some pattern where a node is under congestion while other are with no load whatsoever or with minimum traffic. Path distribution is carried out to maintain low latency values and occupation of idle physical resources. The actual distribution is governed by the actual traffic load in the network.

The task of setting up the intermediate nodes chooses one node, from a set situated *close or centered* to the source node, for the first intermediate node. The second intermediate node takes similar approach to find its corresponding node, but related to the destination node. The selection of nodes *closed or centered* will be defined in this section. This selection of nodes always involves the source and destination nodes, and are called *supernodes*. Figure 13 shows the path expansion concept using intermediate nodes selected near the source and destination.

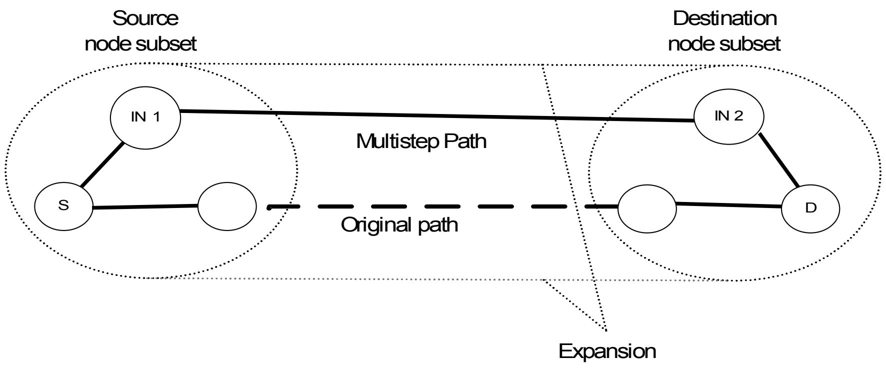


Figure 13: Path expansion.

Communication between source and destination is then carried out using those intermediate nodes, conforming multiple paths to reach the destination, then forming and *expanded path,* called MultiStepPath.

The alternative paths construction relies on the network topology itself, to choose the appropriate nodes to become the *supernodes* for a particular situation. Also relies on dynamic features of the status of the network, such as the traffic load, to select which of all available paths to use in one particular time.

Using this path distribution between a pair of source/destination nodes has two major effects over the messages involved, as we can mention below:

- New alternative paths will be less loaded than the original one, which will gradually become less loaded too, and the bandwidth sum of all alternative paths can equal the original bandwidth of the original path.
- Concurrency can be achieved by using more than one path at the time, thus incrementing the actual bandwidth available for the communication. Communication overlapping produces also a decrement of latency values in the network.

3.5.1.1 Supernodes

A supernode can be defined as a region of the interconnection network where there exists *l* adjacent nodes around a central node *n.*

Figure 14 shows the concept of a supernode structure. Here, the central node is part of the supernode. If a supernode has only one node, it is classified as a *minor canonical;* if a supernode contains all the nodes in the network, then it is known as a *mayor canonical form.* The nodes involved in the supernode are candidates to be part of the alternative path towards a destination, for a particular message in the network. We can mention that any node in the network may be part of one or more supernode.

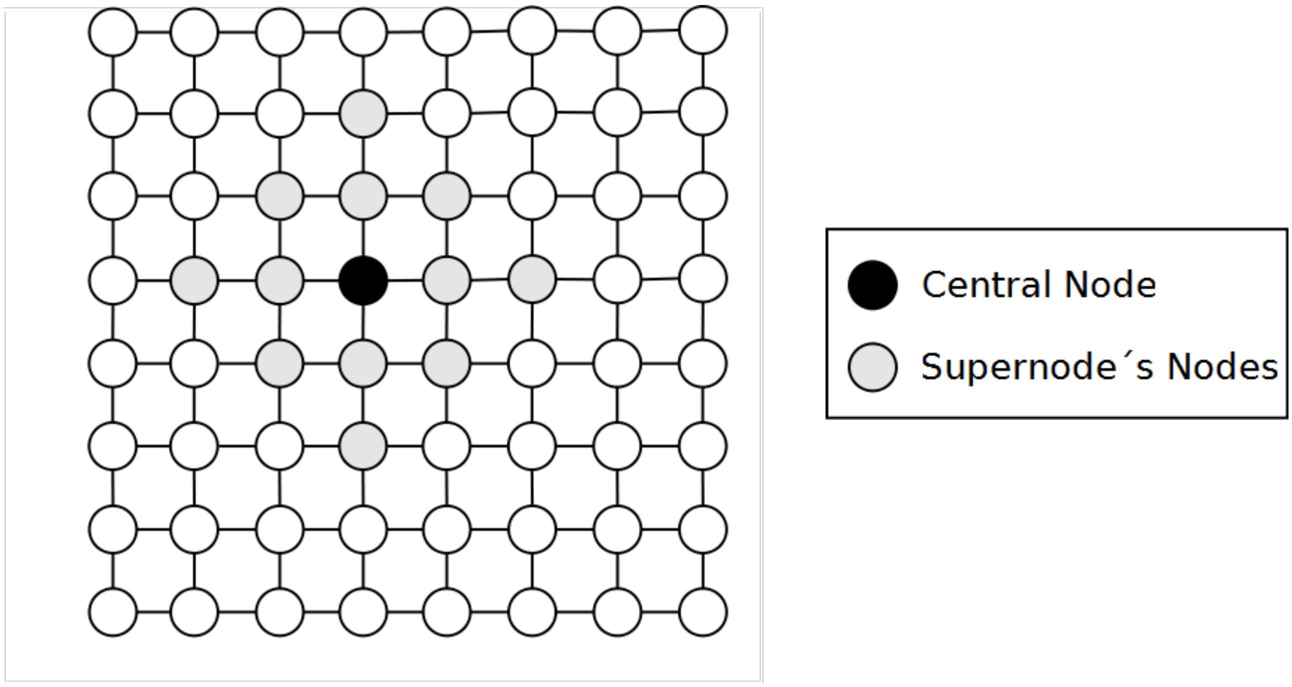


Figure 14: Supernode.

3.5.2 MultiStepPath (MSP)

A M*ultiStepPath (MSP)* is the path generated between the source and destination nodes, by linking together any two particular supernodes in the network and the source/destination ones. The MSP is basically made out of 3 steps:

- Step 1: (SN – IN1): From the source node (SN) to the first intermediate node (IN1), which is part of the ones centered among the source.
- Step 2: (IN1 – IN2): From the first intermediate node (IN1) to the second intermediate node (IN2), which is part of the ones centered among the destination.
- Step 3: (IN2 – DST): From the second intermediate node, to the destination.

Figure 15 Shows an example of a MSP in a network. It shows the source and destination pair, and two intermediate nodes. It can be seen, by this example, that a MSP can contain paths which are not of minimum distance.

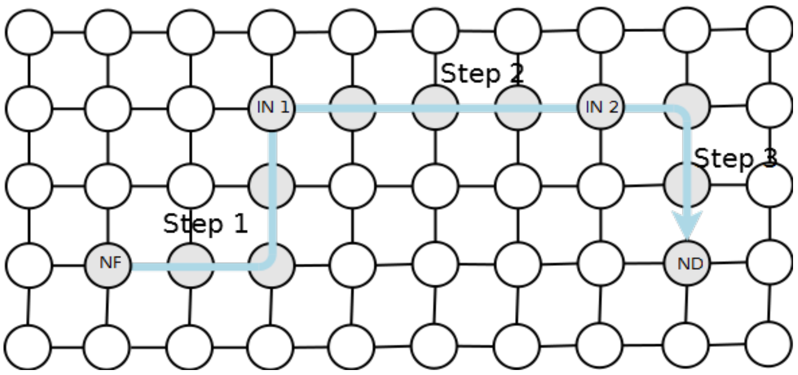


Figure 15: MultiStepPath example.

The ***length*** of a MSP can be obtained by adding each composing step length in the MSP, according to the static routing between them.

$$Lenght(MSP)=Length(SN-IN1)+Length(IN1-IN2)+Length(IN2-DST)$$
Equation 1: MultiStepPath length.

The MultiStepPath *Latency* is the accumulation of transmission time, plus waiting time of a message in every intermediate router in their way to the destination, due to contention .

$$Latency(MSP)=Transmission\,Time+\sum_{\forall\, nodes\in MSP} Waiting\,Time(node)$$
Equation 2: MultiStepPath Latency.

Latency values gives a metric that can be used to know the state of one path, and serves as a decision value when load distribution must be performed. The inverse of the latency identifies the amount of messages passing through a MSP per time slot, and it is defined as the *MultiStepPath Bandwidth.*

$$Bandwidth(MSP)=Latency(MSP)^{-1}$$
Equation 3: MultiStepPath bandwidth.

3.5.3 MetaPath

The set of all MultiStepPath that can be generated between the source supernode and the destination supernode is defined as the MetaPath, as shown in Figure 16.

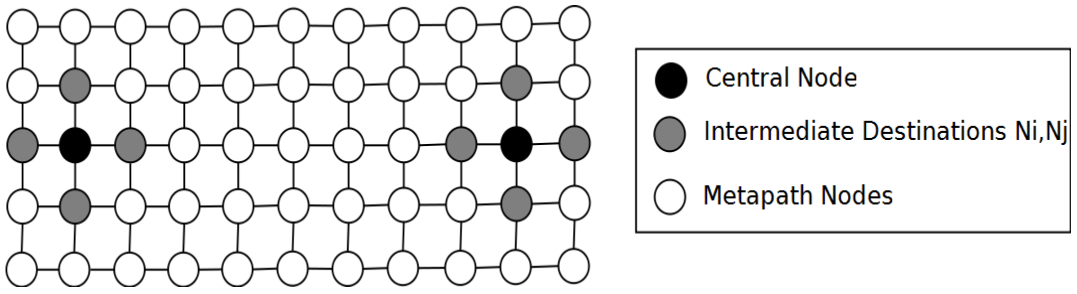


Figure 16: MetaPath.

The width of the MetaPath is based on the number of MSP available for that MetaPath, and is expressed as *j times k,* where j represents the source supernode nodes count, and *k* represents the destination supernode nodes count.

$$MetaPathWidth=s=j*k$$
Equation 4: MetaPath width.

3.5.4 MetaPath length and latency

The MetaPath length is expressed as the average length value of all its conforming MultiStepPath, as show in Equation 5.

$$\text{Length(P*)} = \left(\frac{1}{s}\right)\sum_{\forall s} length\left(MSP_s\right)$$

Equation 5: MetaPath length.

MetaPath latency is the inverse of adding up the inverse of each of all its conforming MultiStepPath latencies. These are the latency values over each message across the MultiStepPath. These inverses are in fact the capacity of the path.

$$\text{Latency (P*)} = \left(\sum_{\forall s} Latency\left(MSP_s\right)^{-1}\right)^{-1}$$

Equation 6: MetaPath latency.

3.5.5 MetaPath Advantages

So far the formal definition of the alternative paths set components have been shown. Every pair source/destination gets its own *MetaPath* according to the source/destination supernode found for that particular node. As an analogy, source supernode would be like the concept of messages scattering area and the destination supernode as the concept of messages gathering area, then the MetaPath is the zone among source and destination areas where all messages traverses. All the MultiPath concepts may be better viewed as an analogy, where the MultPath itself would be a car highway, and the scattering and gathering areas represent the entrance and exits to that highway. Greater bandwidth is then available under this scheme to applications communication than in the original, unique, path.

3.6 Parallel Applications Patterns

Programs are designed to be executed dynamically, not in a steady state. This dynamic is not uncontrolled, but it is bounded within certain limits. During their execution, programs tend to go through different phases according to specific tasks performed by the program. For example, during the beginning of a program it starts with initialization of data structures and some other initial parameters, that remains until the end of execution. Then other phases must be carried out as well, as communication of initial data to other nodes, program data distribution, computation, synchronization, and many other depending on particular application requirements. Each of these phases can account for different time of computation or execution, based on the kind of problem being solved.

Most programs tends to be written with a modular scheme in mind, where the basic program structure is based on procedures within a loop, where each procedure is also using loops and calling other procedures. Although this is not a strict concept for all kind of programs, it is for most compute bound programs. Applications designed and implemented under this scheme, presents some features such as strong periodic behavior, alternating between different portions of code [35].

Based on the repetitiveness exposed above, relevant parts of applications can be extracted and then create a signature to identify them properly during executions [8].

3.7 Bursty Traffic

Identification of relevant stages of an application could help to find situations where repetitive communications patterns alternated by computation phases can also be extracted. In high performance computing (HPC), we could find many traffic load patterns according to the paradigm or model of the applications. DRB [30] for example works well under uniform traffic load distributions, representing applications where heavy communications are done at the beginning and remains the same mostly until the end of execution. There are, though, other types of load distribution found in HPC, as the **bursty traffic** composed basically of a portion of uniform, low traffic load and other very distinct traffic pattern representing a heavy traffic load. Both low/high traffic load patterns performs in a cyclical way, leading to a distribution depicted in Figure 17. This kind of traffic load represents applications where computation stage is well differentiated from communications stage.

Figure 17 depicts two traffic load patterns examples. The first one, shown in (a), represents an application with an uniform distribution for low traffic load, and one distribution for high traffic load, the bit-reversal.

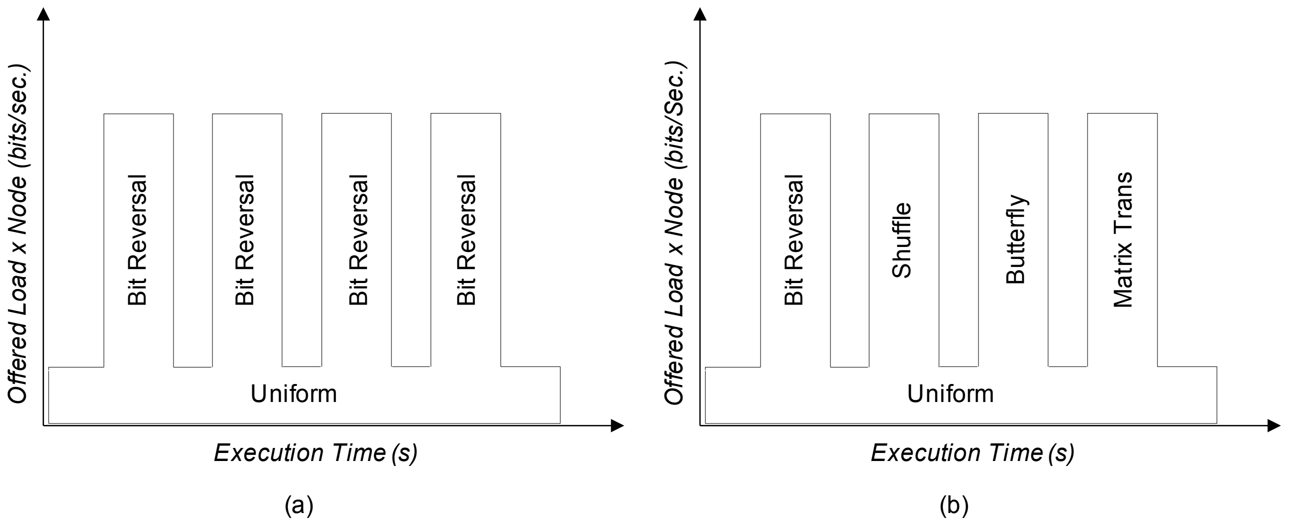


Figure 17: Bursty traffic load patterns.

This communication pattern is just called bursty traffic, and represents applications performing some calculations over a set of data for a period of time, then proceeds with communications and start the process of computations again. This process repeats until execution ends, and communications are always performed among the same set of nodes. The second figure, (b), is called bursty traffic with variable pattern and also has some uniform low traffic load; but within the bursty zone, the communication pattern used changes in each step. Applications in this category perform computations over a data set for a while before start communication process to other nodes in the network. Afterward, a new set of computations will be executed again, until communication phase is needed and the cycle will start over again. Applications under this scheme can be those with task migration between a set of nodes, or where communications depends on data results obtained from previous computation.

We can see from Figure 17 that bursty traffic is composed of many portions of uniform and some other communication pattern load. The bursty nature of HPC traffic is not entirely random [36], and it is governed by application criteria such as process mapping. Using the application patterns extraction strategy within the cyclic behavior of traffic in HPC could be a powerful way to obtain better result for routing and congestion control policies based on historical data about application behavior and application traffic.

3.8 Prior Work: DRB

PR-DRB is based on the algorithm presented at [29] and it is called *Distributed Routing Balancing (DRB).* Below a short description of the algorithm.

The DRB algorithm dynamically configures *MetaPaths* and distributes messages in a communication between all *MultiStepPaths* available in a particular *MetaPath*, in order to maintain latency controlled under minimum values and make proper use of all the resources available in the network. DRB routing fundamentals are:

- Monitoring of messages latency in the network, to detect current traffic conditions.
- Based on the latency registered, dynamically configuration of *MetaPaths*.
- Distribution of messages between the *MultiStepPaths* of the *MetaPath* found.

Based on this fundamentals, the DRB algorithm is broken up in three phases:

- Phase 1: Application traffic load ***Monitoring.***
- Phase 2: *MetaPath* C***onfiguration.***
- Phase 3: *MultiStepPath* **S*election.***

This phases are independent from each other, and are executed concurrently at every source/destination pair involved with running application.

To achieve a global uniform distribution of traffic, authors mention that a global work is needed, hence each source/destination node in the network is capable of expanding their paths according to particular traffic loads which they are subject to.

Monitoring is performed at every channel, and as all other nodes in the network can perform the same task, every channel is able to detect a situation where latency goes beyond tolerated values and then decide to pull apart from the original path.

By means of alternative paths an uniform traffic load distribution is intended, to maintain low latency values and try to eliminate traffic unbalance in the network. If total resources demanded by applications do not exceed network and nodes maximum capacity, DRB will keep latency values low. According to traffic conditions and its location over the network, DRB distributes traffic from more loaded paths to less loaded paths in the network.

To summarize, DRB evaluates all source/destination behavior, and when there are high latency values then is activated the mechanism in charge of adding more paths to the original path, thus improving its performance. Its procedure is based on the phases mentioned in section 3.8 ; *Monitoring, Configuration and Selection;* and its basic flow diagram is shown in Figure 18.

Monitoring is accomplished at every intermediate router, by measuring buffer latency values while a message is traversing the network. Each latency registered is saved, added up and moved into a header packet. When a message arrives at destination node, it further generates an acknowledge message *(Ack)* towards the source node with the entire path latency, to perform the actual *notification* of path congestion.

Once the ACK message reach the source node, it then evaluates the latency value and make proper decisions about how to treat the congestion situation detected along the way to the destination. If the latency value surpass a maximum threshold established for the latency, then the event to start contention is triggered. The decision to perform contention is responsibility of the *Configuration Module*, where decisions about to open new alternative path or close existing ones are carried out.

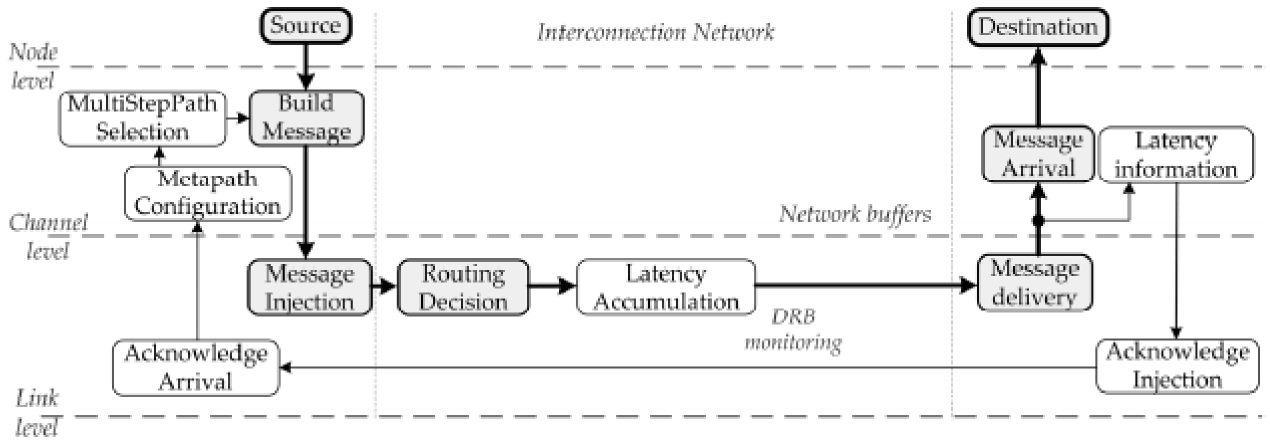


Figure 18: DRB flow diagram.

Finally, *selection* of alternative paths encountered at the configuration phase for each message to be injected into the network is performed. This *selection* tries to avoid congested portions of the network. Proper distribution of messages into corresponding paths is guided by latency values received from ACK packets, sending more messages through paths with lower latency values notified.

3.8.1 Analysis of DRB features

This section analyzes limitations of the DRB algorithm and situations where the performance obtained with it is poor.

In Figure 18 we saw that latency is monitored and saved at every intermediate router in the network that is in the path toward the destination. This procedure adds and overhead to the total amount of time invested in transmission, due to actual operation involved in these procedures. This overhead depends on the amount of intermediate routers to traverse and the time involved in the monitoring/saving operations.

DRB also generates an acknowledge message (Ack) for every message that reaches the destination, to inform the source node about the state of latencies in the path involved in the source/destination path. This ACK messages can also cause a situation of congestion, because if the network is under a heavy traffic load, inserting more messages would worse the situation the algorithm is trying to avoid. Another limitation is the total notification time since it reaches the destination, until it gets to the source node to notify about the situation encountered. This can be considered as a limitation due to the distance between the source/destination nodes in the network, as if it is large enough it could take considerable time to reach the source under congestion situation, and the average congestion present at intermediate routers. Under bursty traffic patterns these limitations can go even worse, because notification can reach source node when the congestion condition has already finished or now it is completely different respect to the original information notified.

A proper response time is then a must to these kind of algorithms, lesser the time invested in notification greater the benefits of using the algorithm and the good effect over the network.

3.8.2 DRB behavior under bursty traffic conditions

In 3.7 was introduced the bursty traffic, that represent the traffic pattern of applications that perform cyclically between computation and communication. This section is going to study the DRB algorithm and its behavior under this specific traffic pattern.

In previous sections we have seen that DRB open new alternative paths to control congestion based on latency values of the entire path towards the destination. The aperture of new alternative paths is performed gradually, meaning that after a ACK notification is sent the actuation phase on the source node is invoked, and DRB proceeds to open one (or more) alternative path to try to control the congestion situation. This process is repeated until a combination of proper alternative paths is encountered, where proper mean that the latency values received at the source node is under a pre-defined low threshold. In Figure 19 we can see DRB behavior under a bursty traffic situation, and the raising portion of the blue (latency) curve represent the process of opening new alternative paths progressively. It can be seen from this figure that DRB invest considerable time to find the combination of alternative paths to get a stable latency value, and all these steps happen just for the first stage of each of the repetitive bursty traffic condition. Once all the alternative paths to keep latency values controlled are found, there is a period where the traffic adapts to the new given condition and the global latency value tend to become stable, as it is shown in Figure 19 where the curve is decreasing.

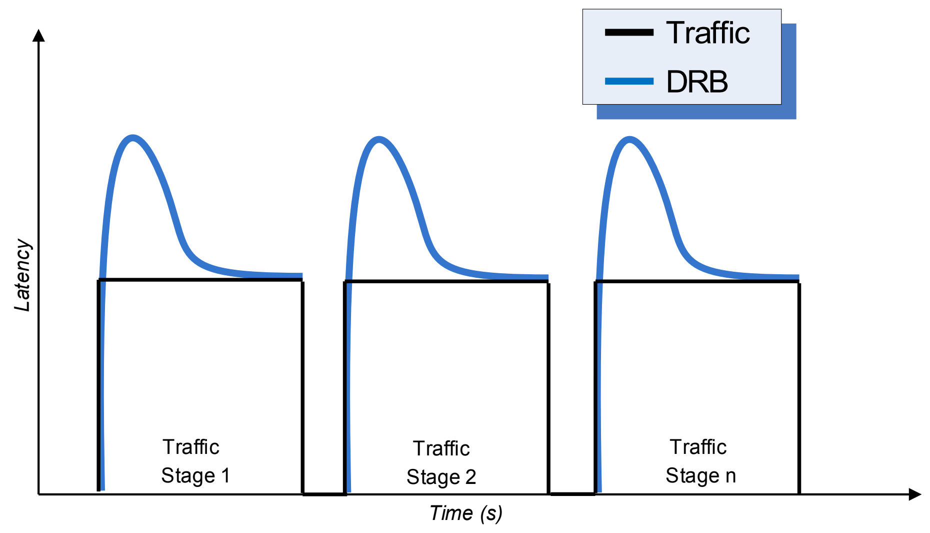


Figure 19: DRB behavior under bursty traffic.

All these steps happens during the first stage of the bursty/repetitive traffic condition. After the application has ceased computation and starts communication again, represented by the peaks in bursty traffic, DRB will carry out again the same series of task mentioned above to stabilize latency values.

The behavior of DRB shown in the Figure 19 illustrates that the algorithm does not perform well under repetitive traffic patterns, thus allowing high latency values during the period of time dedicated to find all the alternative path and the adjusting process taking place in the network after the paths have been found. If the traffic pattern represent a hotspot situation, then this hotspot would remain causing congestion during the whole process of finding the alternative paths until it reaches an end.

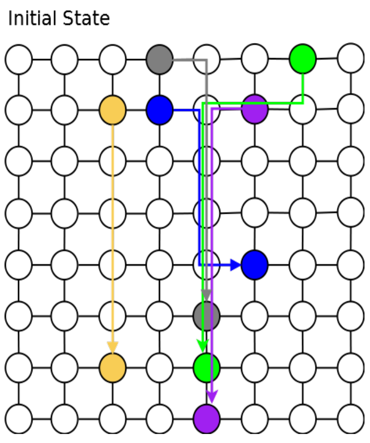
3.8.3 Process of finding alternative paths

We have seen the main features of DRB under bursty traffic conditions; such as the time it takes to find the global good combination of paths to keep latency values stable, the situation of hotspot is extended during this period and the poor reaction under repetitive traffic. In the following figures the cause of this behavior will be shown, step by step, until a stable solution is found.

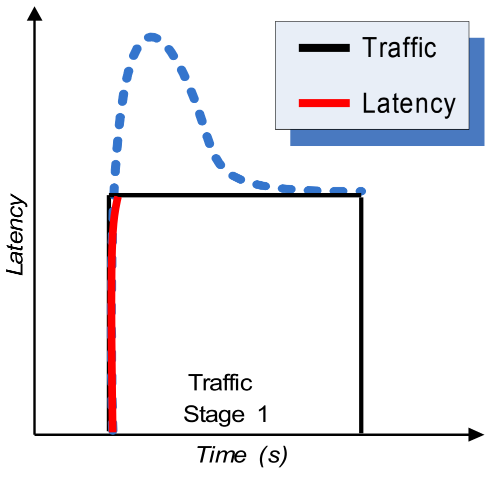
Figure 20 *(a)* shows an example of a direct network configuration. Each pair of colored circles represents a source/destination node in a communication. By simple inspection can be determined that many flows are converging at some nodes in the network (gray, blue, pink and green) in their communication to their respective destinations nodes, and there is a communication external to this situation not colliding with the others (yellow). Figure 20 *(b)* show the latency values, corresponding to network situation in that particular time instant. Latency value is rising as traffic is rising also.

Figure 21 (a) shows the situation where messages of all flows are traversing the network through intermediate routers. There, latency values surpass a high threshold value due to traffic collision in routers. Because the routers are handling more traffic than they are capable of, messages are waiting at the router buffers thus increasing their latency values. This congestion situation is registered by the DRB algorithm on every intermediate router each message traverses. The red colored circles represents congested routers. Global latency values in the network are still rising, because the network congestion situation is not controlled yet, as it can be seen on Figure 21 (b). It can also be seen that even though traffic has already stabilized (i.e. traffic is not rising and remain constant for that period of time), latency does not follow this similar behavior because the congestion is still in the network. The traffic through the yellow source/destination pair is still not compromised due to low load at intermediate routers in its path toward its destination.

Figure 22 (a) shows how the aperture of new alternative path is performed. In this case the blue node starts opening an alternative path due to congestion situation detected by its original path, marked in red in the network. Once the destination node receive the total latency value registered through out the path, then it sends back an ACK message to the source with this value contained in the packet. The ACK message acts as a trigger to start the process of balancing the traffic load and find the proper combination that leads to a lower latency value in the network. Figure 22 (b) still shows that latency value are on rise, and traffic remains constant.

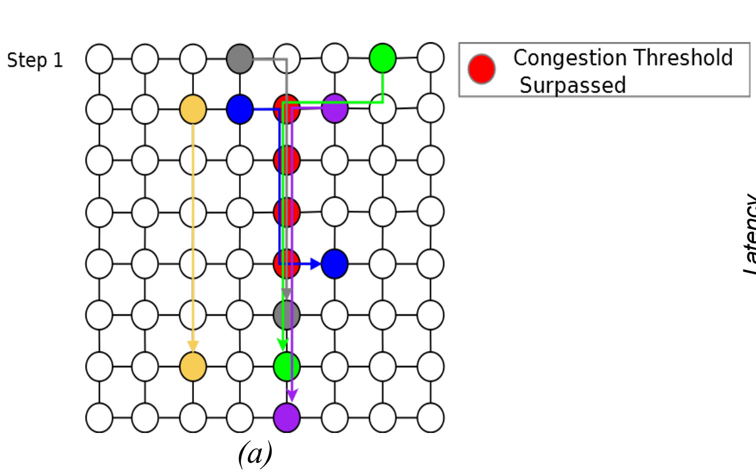


(a)



(b)

Figure 20: Initial state - DRB behavior.



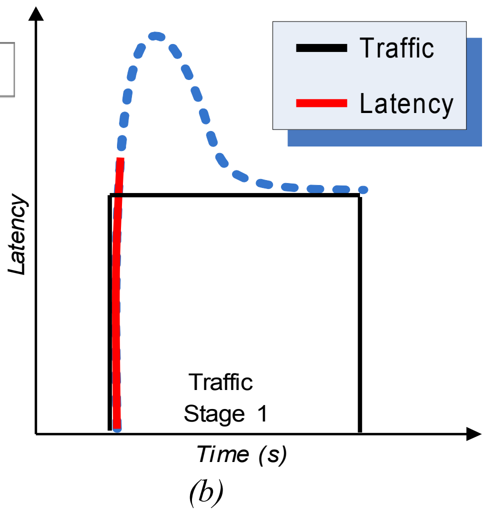


Figure 21: Step 1: DRB behavior.

The process of opening new alternative paths is performed by all nodes in the network involved in the congestion, so a global effect of uniform load distribution and latency control will be obtained. To visualize the effect of the alternative paths under hotspot situations only some nodes in the network will be analyzed with these examples. The original path is named P1, and the alternative paths are named P2 for the first alternative path open, P3 for the second alternative path and P4 for the third correspondingly.

Figure 23 *(a)* shows the situation where two different source nodes are performing path apertures, reacting to the congestion detected by all the intermediate routers. The important point to remark here is that the effect of aperture of the blue and gray nodes involves alternative paths that collide in their new trajectories towards their respective destinations. We can see with this example that congestion is partially moved from the original location where it took place to this spot in the network, reason why the latency is still rising as shown in Figure 23 *(b)*.

This "transfer" of congestion would be monitored by intermediate routers now dealing with this situation, and DRB will start to measure again that latency are surpassing the high threshold defined as correct value under it can work seamlessly. When DRB positively identifies a new situation of congestion will continue opening new alternative path to minimize the congestion situation.

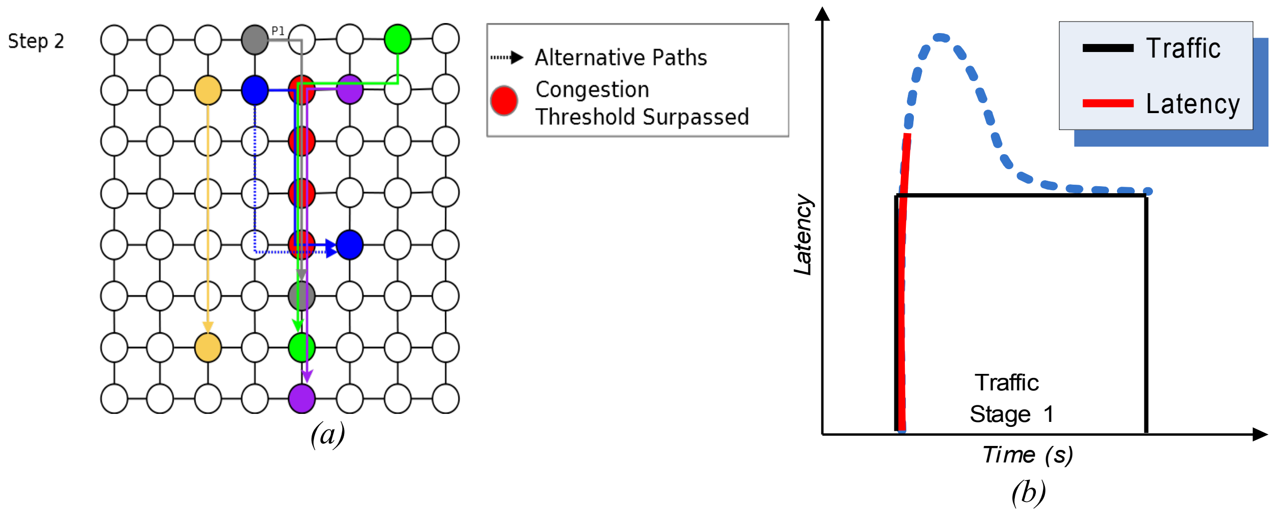


Figure 22: Step 2: DRB Behavior

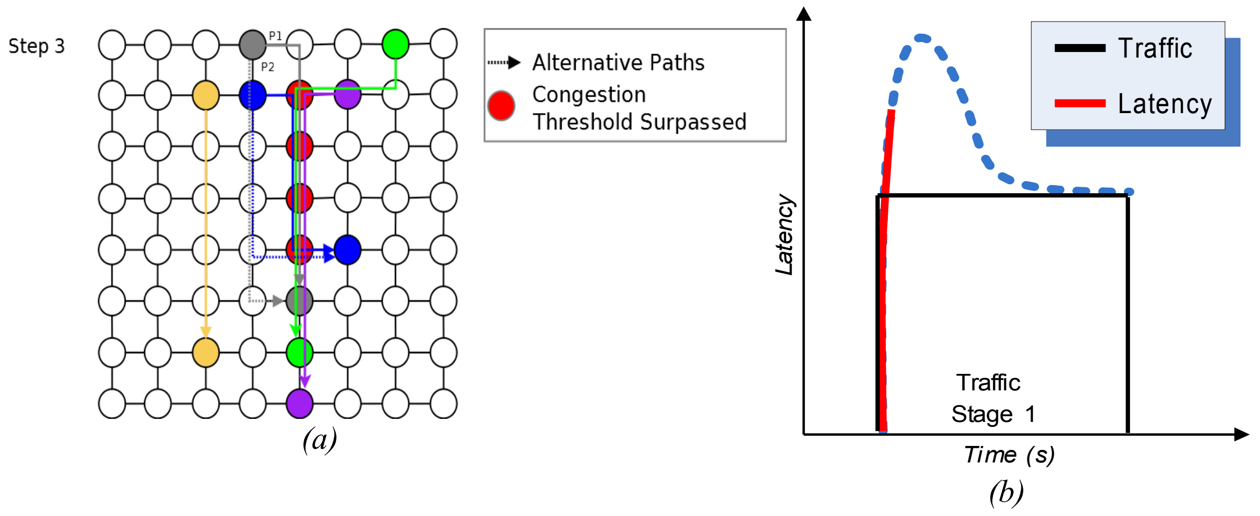


Figure 23: Step 3: DRB behavior.

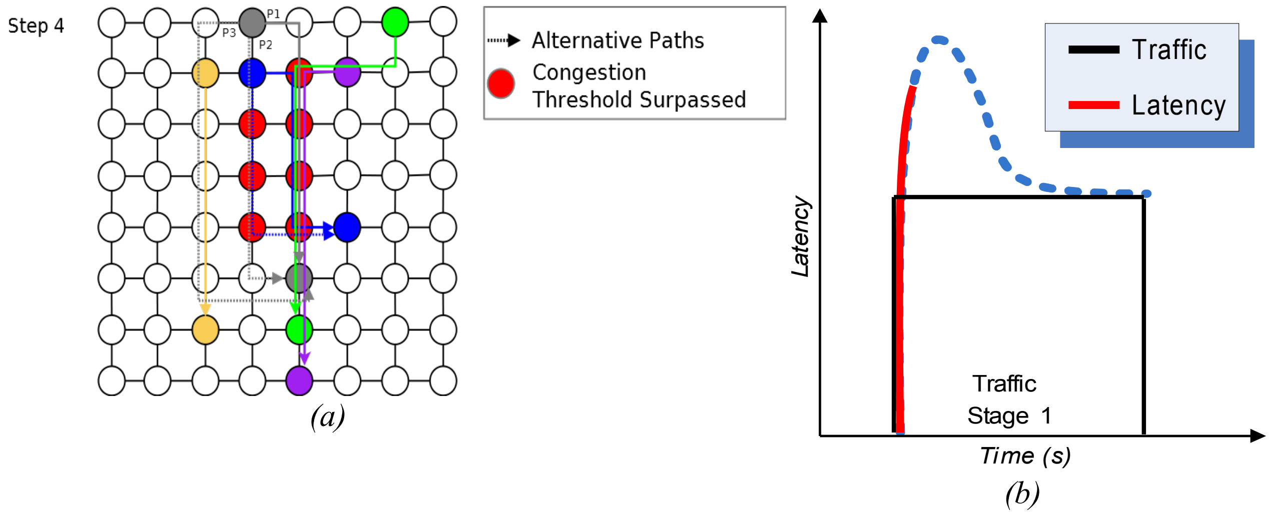


Figure 24: Step 4: DRB behavior.

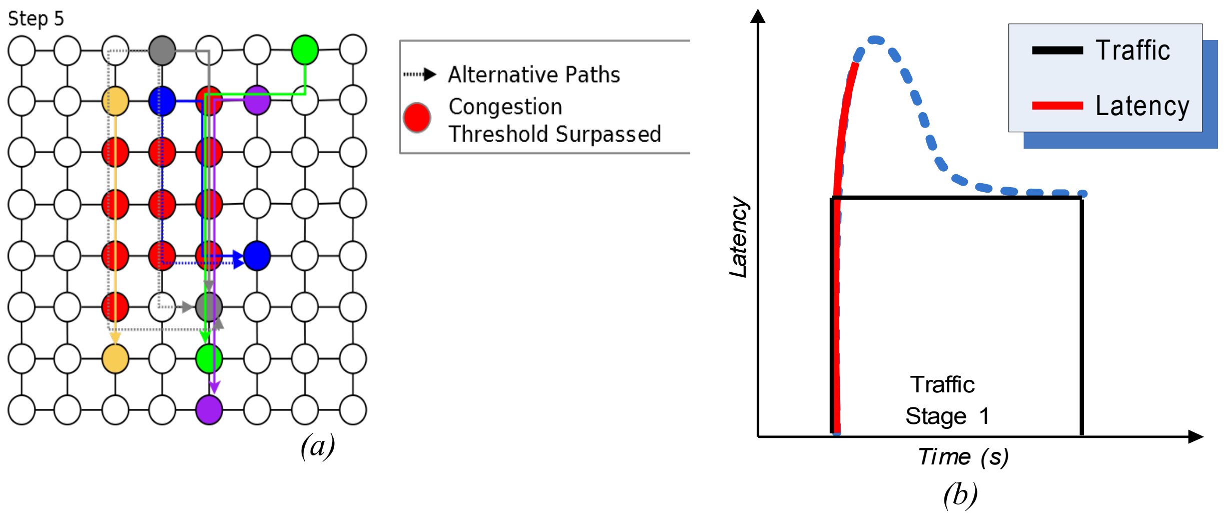


Figure 25: Step 5: DRB behavior.

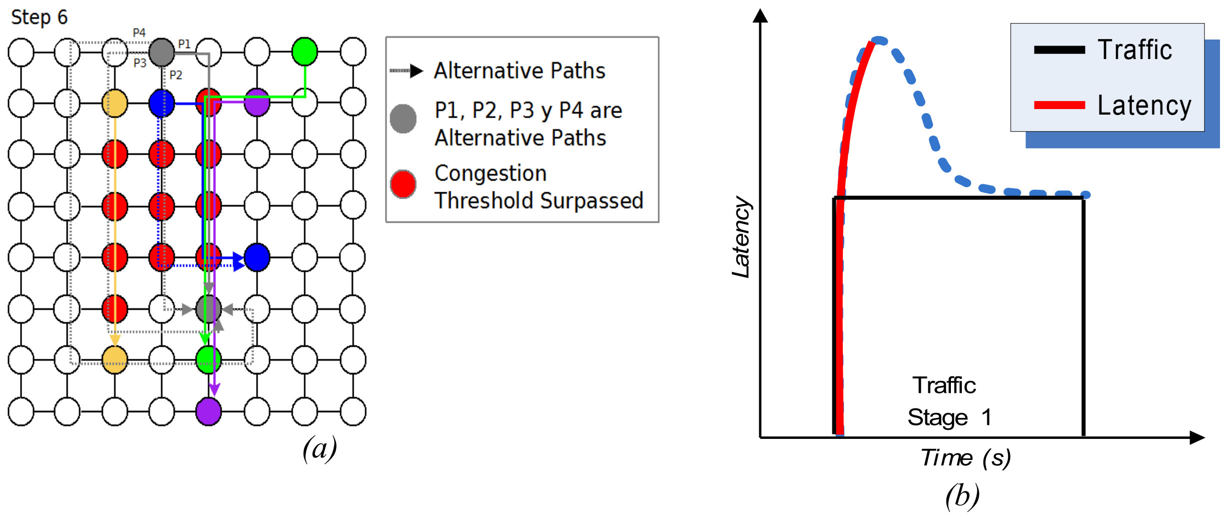


Figure 26: Step 6: DRB behavior.

As a reaction to the notification of congestion mentioned above, Figure 24 *(a)* and *(b)* shows a new alternative path from the gray source node, as a response to the congestion detected while colliding with the alternative path of the blue trajectory.

Now there is a particular situation worth of comments. The yellow flow, so far not involved in congestion, is now affected by the decisions of DRB in his search of global traffic distribution and controlled latency values. The P3 alternative path after collision with the original yellow flow will unchain similar detection/notification mechanism. A similar latency condition as previous steps is given in this case as shown in Figure 25 *(a)* and *(b)*.

Again, DRB will detect the congestion in Figure 24 and after a notification (via an ACK message), will decide to open the alternative path P4 to restrain the congestion situation so far still in the network. This last alternative path, P4 in Figure 25 *(a)*, now completely avoid collision with any other trajectories in the network, so DRB *MetaPath* configuration mechanisms will stop opening new paths. This decision of stop opening alternative path is because global latency values notified under this combination of alternative paths is currently acceptable in the network.

Latency, shown in Figure 26 *(b)*, at this point reaches its highest value and from now on it will start to decrease until a stabilized value is reached. This is the effect of all the alternative paths opened by the DRB algorithm. To keep latencies stable during the life of the traffic condition, all paths opened, the original P1 included, are used but with different weights. With this approach those path less loaded are going to receive more messages and those whose capacity is nearly surpassed will receive less traffic load.

Because all nodes in the network are capable of opening alternative path to improve their particular congestion situation, a global state of controlled latency and traffic balance can be achieved in the interconnection network.

This procedure of finding a global stable latency value, through new multiple alternative paths in the network, is shown to be a controlled and ordered mechanism. This mechanism have many steps as we have seen, as just one example, to obtain the proper combination of alternative paths to control the congestion situation spread over the network. When traffic shows a repetitive pattern or behavior, DRB algorithm will perform the same amount of tasks to reach to a global stable latency value every time the pattern appears, and now we can see that lengthening the whole process of finding the right path combination could be a extremely penalized process.

All the limitations included in the original DRB mentioned earlier show that it is candidate to modifications, to work well under different environmental others than the original designed to work with, and extend its functionality to adapt to other traffic conditions, requirements, etc. not included in the original version.

3.9 Comments about the DRB algorithm.

So far inappropriate traffic balance problems in the interconnection network has been analyzed. Based on the problems exposed, design algorithms based on path distribution have been introduced. This work considers a good approach the effect of introducing multiple alternative paths in the interconnection network under traffic congestion, and then presents how can be effectively created the alternative paths to fulfill the goals of the proper traffic distribution.

The DRB algorithm was later introduced with the idea of multi path routing validation. In this work we are confident that traffic load balancing is one of the best alternatives to control congestion and hotspot situations, also we analyze the actual solution given by DRB to handle its weak points, as the time it invest to find a good combination of solutions or to stabilize latency values, and present a solid proposal about how to improve DRB behavior, analyzing specific transient points that leads to those congested or hotspot situations.

Chapter 4 Predictive Module Proposal

Previous chapters have shown the interconnection network benefits and importance in the area of high performance computing systems. Other topic analyzed have been the problematic found under specific traffic patterns such as the bursty traffic condition, together with the behavior of the DRB algorithm under bursty traffic to practically show the problematic exposed. One possible angle to attack the situations explored so far is the proposal of a predictive module capable of distribute the traffic over the interconnection network and behave efficiently under repetitive traffic pattern conditions.

Therefore, here it is introduced the routing mechanism called *Predictive and Distributed Routing Balancing (PR-DRB)* that aim to solve the inconvenient found in the original algorithm, in an efficient and scalable manner.

This chapter goal aims to expose the ideas that shape the main objective of this work, following the methodology carried out to present a solid proposal of the predictive module **PR-DRB**. Accordingly, a detailed technique description of all the components configuring the module and oriented to avoid hotspot situations and get a proper traffic load distribution under controlled values of latency are included.

The proposed method, **PR-DRB**, follows the three standard phases proposed in section 3.5 . The first phase is in charge of the *monitoring, analyzing and notification* of high latency value that can lead to a congestion situation in the interconnection network. The second phase proceeds with the *decision* about the *configurations* of new alternative paths according to the congestion situation notified in the first phase. The third phase *activates* properly the alternative paths to effectively control the congestion situation.

Sections composing this chapter are focused on the details of the proposal and the internal mechanisms of each composing part, and the interactions between all these modules to fulfill the main goals established.

4.1 Predictive Approach

PR-DRB is based on the DRB algorithm presented in [29] but improves the predictive performance with the integrated monitoring and learning module, in charge of analyzing congestion situations in order to decrease that congestion and keep latency values stable in the shortest possible time. The proposed model consists basically of three basic phases: network traffic monitoring, congestion detection with the corresponding traffic pattern that influenced the situation, and the subsequent effective control of congestion.

PR-DRB basically tries to learn from the repetitive situations of bursty traffic conditions and overcome the limitation in behavior of the DRB algorithm mentioned in the previous chapter.

This work is based on the concept of repeatability of important stages in parallel applications. At the phase of monitoring, latency value of a message is logged throughout its journey, passing through intermediate routers, in addition to storing information related to traffic pattern who caused congestion. When the message reaches destination node, then it notifies the source node the situation through a notification message (ACK), including latency values and contending flows. With this information, the source node is able to carry out the controlled alternative path opening based on latency value, and verify if the congestion situation has been analyzed before in order to reuse the best solution already applied. Besides, PR-DRB updates its database of *best encountered solutions* in case the registered values differs from the ones found in this procedure.

To sum up, once congestion have been detected, DRB algorithm proceeds with the controlled aperture of alternative path to control the situation. To find the solution that keeps latency values stable and traffic balanced can be a time consuming process. PR-DRB proposes to save this optimal situation encountered so far to tackle the congestion situation, and re apply it directly when the same pattern that caused congestion is detected again.

4.1.1 Functional Aspects

In Figure 27 we can see, graphically, the proposed method behavior. During the first stage of the bursty traffic, the curve for both algorithms is practically the same, due to the fact that PR-DRB is learning from the hotspot situation that is causing the congestion in the interconnection network.

Ah the end of traffic stage 1, latency values are stable and the best solutions encountered to obtain that particular latency value are saved at source node. Best solutions are identified by the fact that latency curve has reached its highest point in the curve and from that moment on it starts decreasing, meaning a good balance of traffic have been found and all messages injected in the interconnection network are appropriately distributed.

When the bursty traffic condition stops, meaning that communication phase of the parallel application is over and now is probably doing some computation with the data transferred, threshold latency values goes under minimum values and in such situations the network can process all the messages currently in the network.

When the application starts communicating again, assuming some degree of repetitiveness in the application pattern, latency values starts rising accordingly to traffic conditions as we can see during the second stage of the traffic in Figure 27.

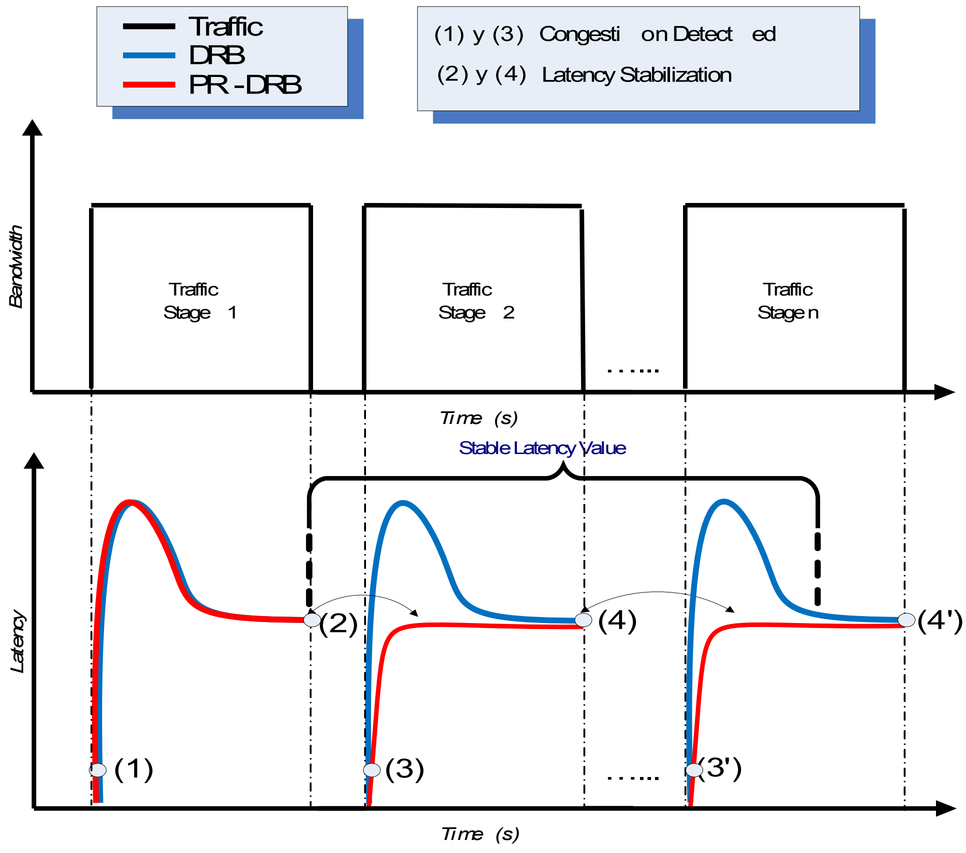


Figure 27: PR-DRB behavior proposal.

This repetitiveness may lead to a situation where the same set of communication patterns appears again in the network, hence giving the ideal situation to reapply the good already saved solutions found in previous stages of traffic. The proposed behavior is shown in stage 2 of Figure 27 where both curves, red and blue representing PR-DRB and DRB respectively, rise similarly until some specific point and then starts deviating from each other. Both curves go similarly practically until some points after the detection have been identified and the control mechanisms have been activated. This point is marked in the figure by (1) and (3), and it represents the precise moment where both algorithms perform the detection and start controlling the situation. We can mention that the proposed method, *PR-DRB,* get such a deviation from the original curve partially due the fact that good solutions have been applied earlier and even though the global latency is still rising, PR-DRB controls the sharpness of rising procedures.

Greater improvement can be seen when the bursty traffic stabilizes and remain constant. *PR-DRB* practically avoid the hotspot situation caused mainly by the process of finding the alternative paths. The main goal of *PR-DRB* is to stabilize the latency value in the shortest possible time, thus leaving resources available to new communications. The points marked by (2) and (4) shows the final effect intended by *PR-DRB.* The latency value encountered in (2) is similar than the one used in (4), practically without any hotspot prolongation due to adaptivity process of the algorithm.

The marks (3') and (4') shows that, when the behavior of both algorithms under subsequent bursty traffic stages is repeated, *PR-DRB* will always apply the good solutions it has and the results will be determined mainly by the capacity of *PR-DRB* to re apply the solutions.

4.1.2 PR-DRB Phases

As stated before, one of the main goals of *PR-DRB* is to improve the response time of the DRB algorithm to react to congestion situation detected in the interconnection network. The improvement is based on historical information of communications between any pair of source/destination nodes. The volume of historical information recorded depends on monitoring information saved by each message in their journey to destinations. While more information is gathered, the more precise the solutions will be in the future. This volume also can negatively impact on performance, because of the time invested by intermediate routers to the process of collecting and saving the information. Therefore a balance must be found between all the inputs and the expected outputs.

PR-DRB uses the *MetaPath* concept as a set of alternative paths to communicate messages from a source towards a destination. *MetaPath* configuration defines how the new alternative paths are created and the final step of *Selecting* the appropriate number of paths created to confront a particular situation is then executed.

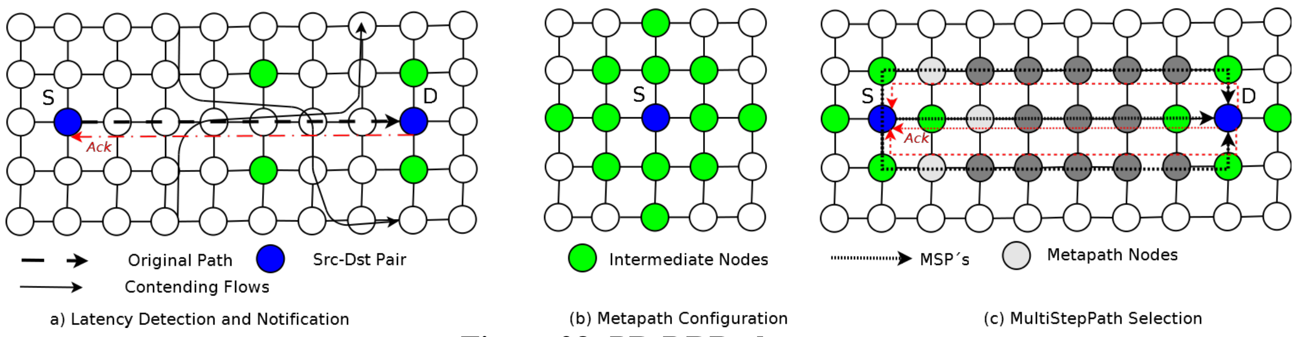

S
D
Ack
Original Path
Src-Dst Pair
Contending Flows
a) Latency Detection and Notification
S
Intermediate Nodes
(b) Metapath Configuration
S
D
Ack
MSP´s
Metapath Nodes
(c) MultiStepPath Selection

Figure 28: PR-DRB phases.

PR-DRB phases are depicted in Figure 28. Detection of congestion, registry of latency values and the communication pattern involved are shown in Figure 28 (a). Also in the figure is the process of notification via an ACK message to the source node, about the congestion detected in the interconnection network, more specifically in the path traversed from the source to the destination. In Figure 28 (b) is depicted part of the process of *MetaPath* configuration once the module has considered pertinent the creation of new alternative paths. The figure also shows the candidate nodes to be used as intermediate nodes for the *MultiStepPath* to create. A complete set of *MultiStepPath* is shown in Figure 28 (c), where besides the original trajectory from source to destination, now there also are two more alternative paths.

With the predictive module active, the steps shown in Figure 28 (b) will be reduced to the only task of selecting the best solution available in its database.

Monitoring is the phase where the latency of the whole path where a message traversed is recorded. Intermediate routers are in charge of detecting the congestion and then save the information that later on will be used again to reopen new alternative paths. Another task of the monitoring phase is the source node notification about the global state of the path where its messages are traversing.

Once the notification arrives at the source node, alternative paths are created based on latency values recorded in the received ACKs messages. All messages are then distributed among this new created alternative paths. Every alternative path is created using intermediate nodes (INs) close to source and destination respectively. These INs are created this way to define a controlled area where new messages would traverse in their path from source to destination. *PR-DRB* uses a three step scheme to create alternative paths, going from the source node to the first intermediate node *IN1*, then from here to the second intermediate node *IN2*, and from IN2 to the final destination node. For every single step in the sequence, the basic routing strategy selected for the network is used. This phase is known as the *MultiPath Configuration.*

In the MSP *Selection phase,* each source node proceeds with the injection of new messages into the network having into consideration the alternative paths available for a particular destination node in the interconnection network.

4.1.2.1 Monitoring and notification phase

Every intermediate router executes the monitoring task when messages go across its internal ports. Once inside a router, the latency of the message is monitored, where the latency is the time the message occupies internal buffers waiting to be accepted for delivery to the next intermediate router or the destination node. Besides latency value, routers analyze what others flows of data are at the same time in the buffers and look for information about them. If latency values registered overpass a high threshold defined, then before a message is accepted for delivery, the header of the packet is filled with information about the contending flows found in the router. Table 2 resumes the monitoring phase.

Contention latency value is determined by the time a message must wait at internal router buffers before it is re injected into the network. Contention is given because other flows are also trying to use as well router resources. The latency value is registered and added up in every intermediate router, so that when reaches destination node the whole path latency would be available to make the proper decisions afterward.

The procedure of saving the contending flows identified in a congestion is a key factor to determinate in other phases that the same pattern has occurred again, and proceed with the proper solution. The pseudo code of the monitoring phase also controls that the congestion situation is only recorded in the first router where it encounters that situation. This is explained by the reason that when the global ACK message is delivered to the source node, then it would perform the aperture of new alternative paths and with this the congested portion of the network will lighten resources hence diminishing latency.

```
Traffic Load Monitoring (Message M, Threshold, MSP)
/* Performed at each intermediate router in the interconnection network */
Begin
For each message M at every hop,
    1. if (first router in path) {
         - predictive = FALSE;
    }
    2. Accumulate latency (queue latency) to calculate path latency
       if (latency > threshold) and not predictive {
       - Identify traffic patter involved in congestion situation
       - Record the traffic pattern
       - predictive = TRUE
       }
    3. Record in message current latency
    4. Continue to next intermediate router or destination
    5. At destination, the latency and patter  information received is sent back to source via an ACK
       message, to the MetaPath Configuration function
End Traffic Load Monitoring.
```

Table 2: Monitoring and Notification phase.

Once the message reaches destination node, then a notification message (ACK) is sent back to the source to allow new messages to be injected into the network but through different paths than the original one. ACK messages have higher priority than others, and due to its small size it can be considered negligible.

Each intermediate router perform the task of monitoring and notification independently with local information, including contention information about other flows contending in its respective buffers, thus improving the global information or knowledge about the whole network.

4.1.2.2 MetaPath Dynamic Configuration Phase

Based on the information received by the monitoring phase, the dynamic configuration of MetaPath is performed. The main goal of this phase is to determine the proper number of alternative path to expand in order to control the congestion based on latency value registered for a particular path in the network, identified by a source/destination trajectory. The process of creation of new alternative paths is carried out using two intermediate nodes (INs) disjoint from the original one. Also, latency values are monitored continually and taken into account to perform the procedure of new path apertures. Congestion if effectively controlled extending current bandwidth by opening as many alternative paths as necessary between a source/destination pair. Table 3 Depicts the configuration phase pseudo code.

Figure 29 shows the flow diagram of the configuration phase in PR-DRB. Those actions are performed when a new ACK packet arrives at source node. If the ACK packet has a recorded latency value showing the network is stabilized, I.e. latency values are between an acceptable high and low threshold value, then no action is needed by the configuration module and new packets are injected into the network without preliminary path information. High threshold value identifies the zone where latency switches from the working zone to a more congested zone in the network. The low threshold identifies the point where more alternative paths creation are not necessary and the set of those alternative paths are reduced. The interval between both threshold then determines the range where a MetaPath is valid and there is no need of modifications.

```
MetaPath configuration
/* Executed at source node every time a new ACK message arrives */
Begin
        - Receive ACK message with latency info and flows involved in congestion
        - Calculate MetaPath                                           latency values
                        $Latencia(MP)=(\sum Latencia(MSPs)^{(-1)})^{(-1)}$
        - If  $(Latencia(MP)>Umbral)$                                  (
                If (already exists saved solution for source/destination pair)
                – Look for the best solution registered in the DB.
                Else
                – Increment INs number to create more alternative paths.
                End If-
        Else If  $(Latencia(Mp)<Umbral)$
                – Decrement INs number to diminish MetaPath.
                – Save latency information of the (MP) and the source/destination nodes of the flows involved in
                congestion.
        End If
End MetaPath Configuration.
```

Table 3: Configuration Phase.

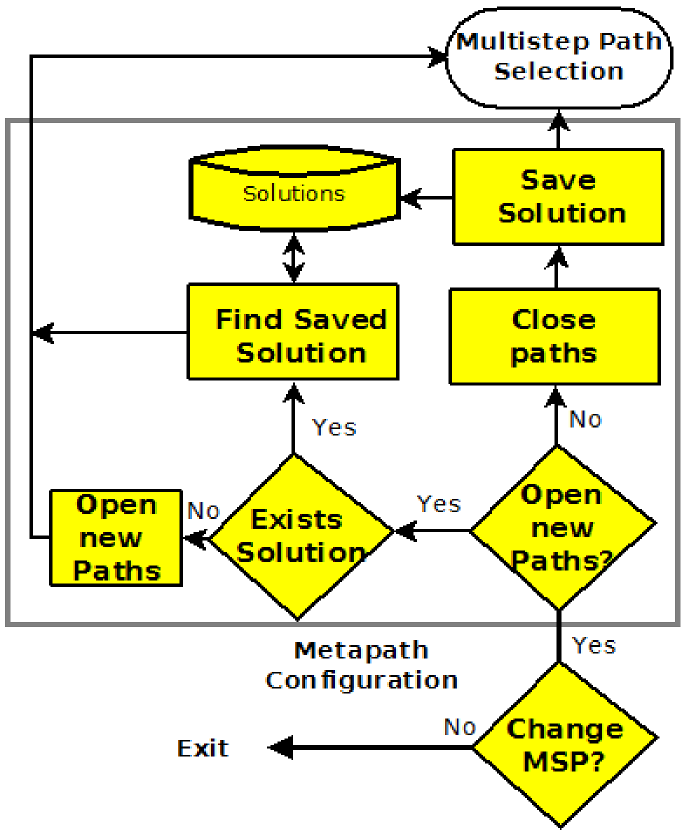


Figure 29: Configuration phase flow diagram.

Recall that each ACK packet has information about one MultiStepPath latency value, as well as the information about the contending flows patterns found in the congested areas, then MetaPath latency is calculated. This latency calculated is compared with a high threshold value, and if it is greater than threshold, then the configuration phase must proceed with path expansion procedure. The first step in path expansion is the task of searching for a previously recorded solution in the database. If a solution exists, then it is retrieved and delivered to the path selection phase, so new messages will use this retrieved path when they are injected into the network for a particular source/destination pair.

If solutions are not found, new alternative path opening procedure must be executed. Once created the path with proper intermediate nodes and other requirements, the path selection phase is notified as the previous case.

This configuration phase main goal is to keep trajectories working under a traffic load that produces low latency values and within threshold limits. If latency values rises, new alternative paths should be opened to satisfy bandwidth demand. If latency value decreases and goes below minimum threshold, then this channel is using more resources than necessary, thus MSP must close some alternative paths to free systems resources. The process of closing paths can be considered as a trigger or indicator to save solutions, because when the configuration phase detects this situation, is a sign that global latency is stabilizing for this source/destination pair and good combination of alternative paths have been found.

MetaPath configuration opening is done gradually, based on the of intermediate nodes selection function presented in previous chapters. Metapath latency is calculated with the following equation:

$$L(MP) = \sum_{i=1}^{s} \left(\frac{1}{(L(MSPi))} \right)^{-1}$$

Equation 7: MetaPath latency.

According to this latency value, the module makes the decision of increment or decrement the alternative paths available to one MetaPath. This reduction/increment of paths is governed by the latency value and the threshold defined for the whole network. When the calculated value overpass the maximum latency threshold, then the module proceeds with the opening of new paths. When the latency is below the minimum threshold then the number of paths decrement procedure is executed. We must remember that when latency is under the minimum threshold value then the global latency is near or soon to be in the stabilized portion of the curve, shown in Figure 27 (2).

The main function of this phase is to permanently control the balance between latency and traffic load and also that the whole network could accept more messages. If latency values rises, then this phase must dynamically open new alternative path to try to balance the traffic and decrement latency registered during the traversing of messages through intermediate routers. When latency values decrement, meaning that the path is using more resources than necessary, this phase then properly starts closing unnecessary paths.

4.1.2.3 MultiStepPath selection phase

When a new message is ready to be injected into the network, one of the paths found at the configuration phase must be selected to carry that message. Selection is done with the idea of fair distribution of traffic in mind. Aggregate latency values are the key point in decisions about the path to choose for a particular message, when latency of a MSP notified by an ACK is low, then that path is prone to be used more frequently than others. This strategy ensures that distribution if done properly, hence more loaded paths are going to be used less, and less loaded paths will receive as much traffic as possible to keep latency values under controlled values.

Given a source node with *N* alternative paths, we can define *Lci (I:1..N)* as latency registered for path *Ci.* The alternative path *Cx* will be selected for the next injection of messages accordingly to Equation 8:

$$\rho(Cx)=\frac{1/L_{Cx}}{(\sum_{i=1}^{N} 1/L_{ci})}$$

Equation 8: Selection probability function.

From this equation we can see that finding the good distribution of paths for a message is a time consuming task, due the fact that several paths may be tried before finding the right one, and send a message over a loaded path under a congestion situation can impact negatively on performance. *PR-DRB* is constantly monitoring this value and saves the best combination of found probabilities that perform better for each source/destination pair. These distribution are saved in addition to the source/destination and the communication pattern to help identify later the same situation and apply directly the best combination encountered to solve this particular congestion situation. This whole process help avoiding the hotspot situation caused by the process of sequentially trying multiple paths to find the lesser loaded paths. Table 4 depicts the *selection of MultiStepPath* phase.

```
MultiStepPath Selection
/* Executed at the source node before inserting a new message into the network */
Begin
        -Build the cumulative distribution function and MSP bandwidth normalization.
        -Select a MSP based on the cumulative distribution function created.
        -Inject the new message into the network
                -Build multiple headers, with intermediate nodes information and the final destination
                -Link together all the headers
                -Inject the new message with the PR-DRB format
End MultiStepPath Selection
```

Table 4: Selection of MultiStepPath phase.

4.2 All Phases Integrated

The three phases integrated are shown in Figure 30. To this point, each phase was described separately, but the *PR-DRB* actually performs all of them integrated and in most cases concurrently. When a new message is ready for injection, a new *MultiStepPath* is chosen from all the are current available in the *MetaPath.* Latency values are considered during this selection phase, paths with lesser latency values are going to be selected more often.

Once a message has been injected into the network, then the time it takes to pass each of the intermediate routers in its paths is recorded. The time of a message inside a router is defined as the time it must wait in the router's buffers with other messages, thus sharing resources. Each router also monitor the congestion level by comparing the recorded latency value with a maximum pre-established threshold; if latency value goes beyond that value, then besides the latency already recorded, intermediate routers identify contending flows in its tables and save information about them. Information about contending flow may be defined in many ways, but for *PR-DRB* are the source/destinations pairs that collide during communications in the router where detected. This process of monitoring latency is performed in each intermediate router, to register and add up latency values until reaching destination node. The process of saving information about contending nodes is not done in the same way as the latency procedure, here the information is saved only in the first router where was detected a congestion situation. This is done this way because when new messages will be injected into the network using the *PR-DRB* strategy, new alternative paths will avoid completely the congested situation and hence saving information besides the one found in the first router is useless.

When a message reaches its destination, it would have all the information about the network situation in its header, and then the destination node proceeds with the notification back to the source node about what the message have learned during its journey so far. The notification message is named and ACK message. This ACK message reaches the source node and the process of packet information analysis is performed. New alternative paths will be created if the congestion situation is detected, and *MetaPath* configuration phase will start. If latency values are controlled, then the normal injection of new messages if guaranteed. If congestion is detected, then, before opening new alternative path, the configuration module will look for a suitable saved solution in its “best solutions database”, and if one is found for that particular communication pattern being currently analyzed, then it retrieves and re applies directly that solution, thus avoiding time dedication to the task of assembling new alternative paths. If no paths or solutions are available at the database, then the normal process of creating new ones is performed. When the congestion is controlled by the alternative paths created in this phase, ACK messages will confirm that with latency values under a normal low value and the configuration phase will proceed to close those paths no more necessary to the communication and then proceeds with the updating process, where those encountered good solutions will be inserted or upgraded into the database.

All the tasks are performed concurrently, as extracted from the figure. Messages delivery is performed while the tasks of configuring new paths are executing. At the intermediate routers this condition is also true, while messages are waiting in the internal router's buffers, the task of accumulating latency and identifying contending flows it is being carrying out, so no extra overhead is included. Metapath Configuration and Multistep Path Selection are performed at source nodes and not at intermediate routers, so intermediate routers are not more complex than normal ones found under other implementations.

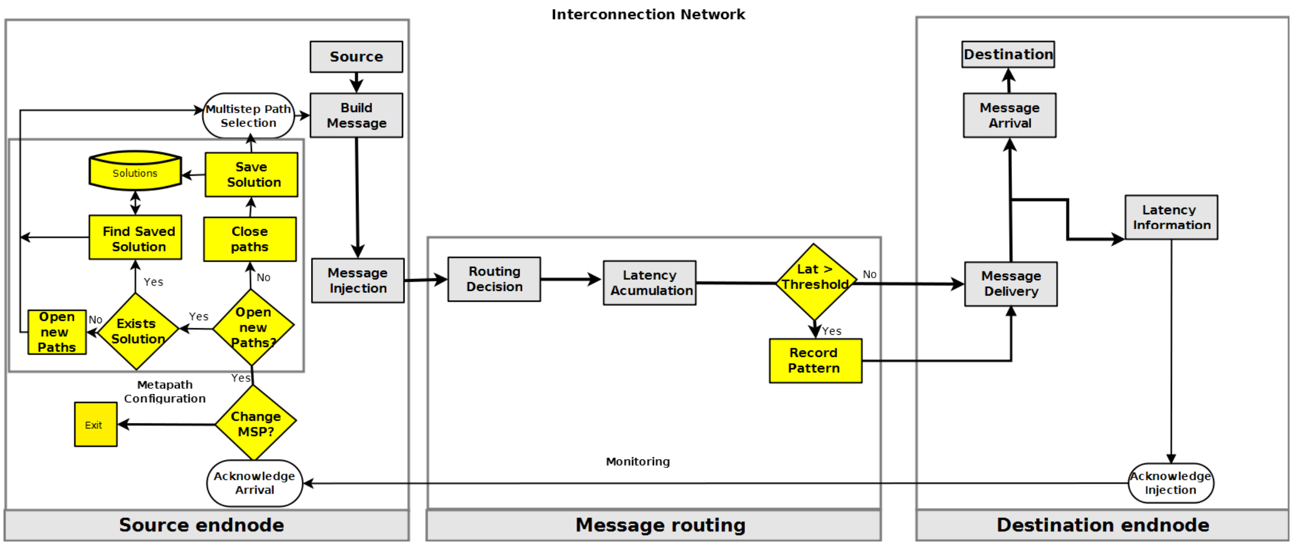


Figure 30: All PR-DRB phases integrated.

4.3 Predictive example

We have seen so far all phases of the PR-DRB algorithm. First, a detailed overview of each of the phases was given separately, and latter was introduced a section where all this concepts were integrated and a global vision about how all phases worked together to purse a single goal was shown. When monitoring phase detects a situation that can be considered “dangerous”, in terms of congestion and high latency values, then it proceeds with conflicting pattern identification that is later notified to source nodes, specifically to configuration phase, to perform new alternative paths apertures accordingly. During the first stage of bursty traffic pattern, PR-DRB is immersed in a learning process about how the best combination of alternative paths is accomplished. This learning process concludes when traffic pattern in bursty situations also ends. During next bursty traffic stages, PR-DRB will proceed to look for the best solutions encountered and re-apply those solutions recorded in order to avoid unnecessary alternative path creation. In Figure 31 there is an example of how this whole procedure works, and helps understand the integration of all phases described in previous sections. Figure 31 *(a)* shows an initial hotspot nodes configuration, where circles of the same color represents a pair of source/destination nodes, during communications of packets. Red circles represents a congestion situation detected by an intermediate router, as explained in section 3.8.3 . In Figure 31 *(b)* we can see that alternative paths have been opened by source nodes, in response to latency values notification by intermediate routers. When PR-DRB has effectively opened the correct number of alternative paths, and when notification procedures indicates that global latency value is stabilized, then good solution encountered for this particular traffic situation must be recorded, as historical data. In Figure 32 *(a)* we can see the values recorded for this communication pattern. Original source and destination nodes, represented as S1 and D1, as well as contending flows information, represented by S2 and D2, are saved as historical data. Also, the two alternative paths opened to solve this congestion situation, P1 and P2, are recorded as well.

Now, in Figure 31 *(c)* we can see another congestion situation depicted. In this case, the same pair of blue nodes (S1 and S2) are communicating in the interconnection network. Also, red intermediate routers detects a congestion situation and start collecting information about contending flows encountered at internal routers buffers waiting for resources, information that will be sent back to source nodes via an ACK messages once it reaches its destination. The difference here are the contending flows source and destination, in this case S3 and D3, which are different from the previous example. Figure 31 *(d)* then shows the alternative paths opened for this particular case. We can see also in Figure 32 *(b)* the best solution encountered to deal with this congestion problem, that is recorded into the database. We can see here that both situations save the original source/destination pair, as well as the contending flows identified during notification phase.

This contending flows saved with the original source/destination pair is going to be used to identify the same troubled pattern later.

Now, recall that the interconnection network is working under bursty traffic conditions, and scientific parallel applications posses repetitive behavior. This assumption leads us to the situation where troubled traffic patterns is going to appear again in the network. When that situation happens again, PR-DRB will identify at intermediate routers contending flows causing congestion in the network, and when a source node is notified about congestion, it will proceed with search procedures trying to find good solutions already recorded previously. In this case, if congestion presented in Figure 31 *(a) appears* again, then PR-DRB will recognize this situation, and alternative path P1 and P2 will be directly opened, besides the original path, to start contention procedures.

Similar actions will be performed when hotspot situation depicted in Figure 31 *(c)* appears again. In this case, the index to find proper solution would be the combination of S1-D1 + S3-D3. This will retrieve the best combination of alternative paths encountered for this situation, P3 and P4 in Figure 32 *(b).*

Because the original DRB will invest considerable time to find proper combination of alternative paths to control the congestion situation, and during that period of time more intermediate routers and nodes in the network will suffer high latency in their communications, it is an effective way to face congestion situation based on historical data of best generated alternative paths.

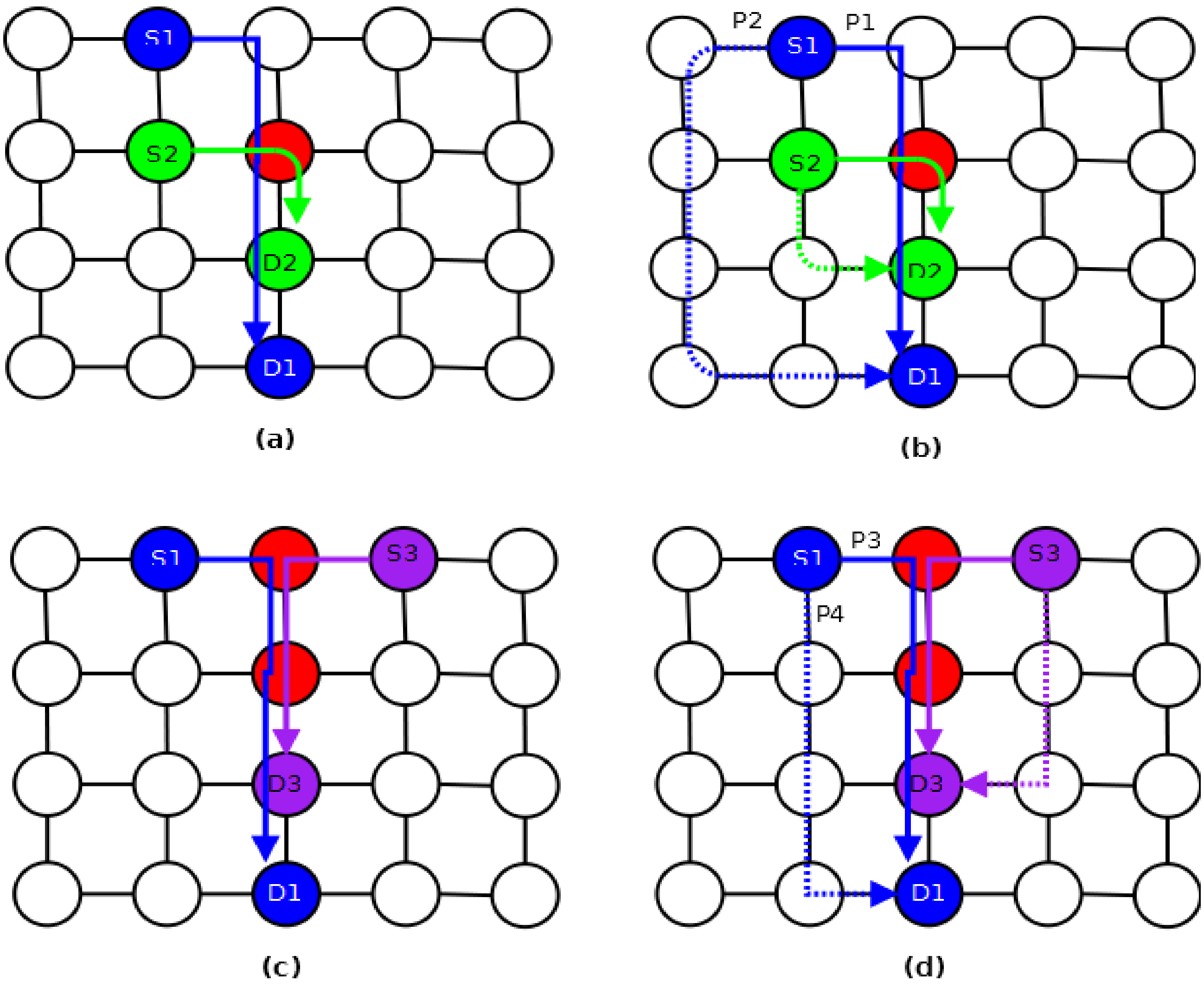


Figure 31: Predictive traffic pattern example.

S1 -D1 [S2 -D2]	• P1 • P2

(a)

S1 -D1 [S3 -D3]	• P3 • P4

(b)

Figure 32: Solutions recorded by PR-DRB for the example.

Chapter 5 Experiments And Initial Results

5.1 Introduction

This section describes the experiments carried out to probe the idea presented in this work proposal, and the results obtained. Also, the environment of the tests is introduced and described.

To evaluate the proposed idea, simulation techniques and tools are used. Comparison between the original routing algorithm DRB [30] against the proposed PR-DRB are analyzed. The intention so far with this work is to present the idea of a predictive module capable of efficiently adapt to repetitive traffic patterns and conditions, and show the possible benefits and limitations of the predictive module.

To accomplish this goals comparison of performance are carried out, where the main metric used is latency of the entire network for each algorithm.

5.1.1 OPNET Modeler

A simulation model provides mechanisms to analyze performance of a real system or behavior under different configurations when the system is not really implemented yet. Even if the system is already implemented, a simulation tool can be very useful because allows many configurations to analyze specific components or perform different execution under distinct environmental variables, to study or stress some particular feature of the system. Simulation and modeling are very useful because can provide insight knowledge about a system [37].

To implement our proposal, the commercial simulation tool OPNET *modeler* was used [38]. This tool allows the simulation of distinct communication network protocols, distributed systems and other technologies under different configurations, also providing the tools to analyze those results.

Some features of the modeler tools are mentioned below:

- Discrete event driven engine, very fast and efficient
- Modeling through object oriented techniques
- Hierarchical modeling environment
- Debugging tools

- 32 and 64 bits simulations environment, and with parallelism execution support.
- Distributed executions support
- Modeling and design framework, with finite state machines as the main modeling technique, and C/C++ language support

Opnet has three level hierarchy for modeling purposes: *network, node, and process levels*.

Network level: Includes nodes, links, and subnets interconnected between them, and composed topologies. At this level, models attributes are set and parametric simulations are configured.

Node level: Network components are represented by using *modules* with such features as: Messages processing (creation, transmission, reception, and storage); and internal routing,content analysis, queuing, multiplexing, etc. Modules typically represent applications, protocol layers, and physical resources, such as buffers, ports, and buses.

Process models: Behavior of modules is programmable via their process models. They consist of finite state machines (FSM) containing blocks of C/C++ user code and kernel procedures (KP). The state machines respond to interrupts generated by the simulation kernel, and are intended to represent the logic of real world processes (i.e. communications protocols and algorithms, queuing disciplines, etc.).

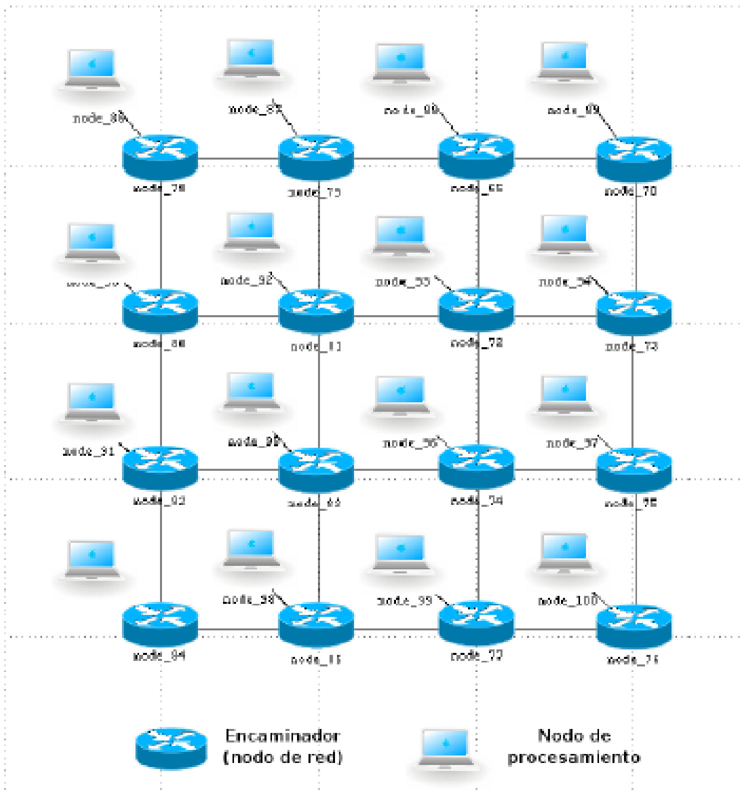
Link parameters such as bandwidth, bit error rate, propagation delay, supported packet formats, as well as other attributes can be specified by users. The simulation kernel handles a single global event list and a shared simulation time clock. Events are attended from the list in the appropriate time order.

Error: Reference source not found Shows an example of a network model. This network is composed by routers and processing nodes. Routers are interconnected by links between them.

Figure 34 Shows a processing node model. It has a component that is in charge of packet generation and injection. This generator simulates the communication patters of an application or a synthetic workload. Also contain the components of transmission and reception found in any network interface card, using to inject into the simulated network, at the lowest possible level, the generate packet. A component for the destination node is also modeled, where packets are received and simulation of packet delivery to higher layers is performed. At the destination node, all the statistics or metrics analyzed are also managed, such as the latency of paths, throughput, among others. In Figure 35 is shown the finite state machine of the SRC (source) component of the processing node. The I*nit* state reads all the configuration parameters of the model for one execution of the simulated system. *Begin* state performs the actions related to identification of nodes (source and destination) and if a trace file is going to be used. Also the statistic collected by this module are initialized here. The *Generate* state effectively creates a new packet and proceeds with the injection into the network, based on the packet format and other parameters defined before. *Bursty* state represent the traffic condition used in this work, where a combination of uniform traffic load and a specific communication pattern is combined to get a chainsaw like pattern. Source procedures are shown in Figure 35. Processing of

statistics of packets arriving at the destination node are shown in the finite state machine of Figure 36.

Figure 37 shows the model of a router. There are 8 ports in this router, and a crossbar that act as a bridge between input and output ports. The switch info module manages the information about initial configuration of the router in the network, such as port id, among other .



33: Network model (mesh 4x4).

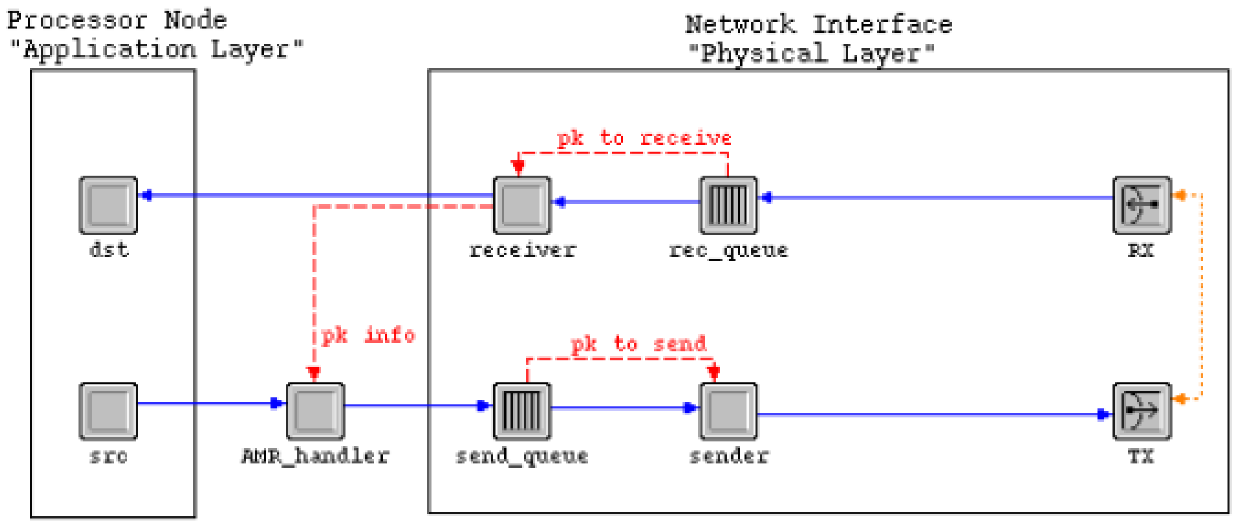


Figure 34: Processing nodes model.

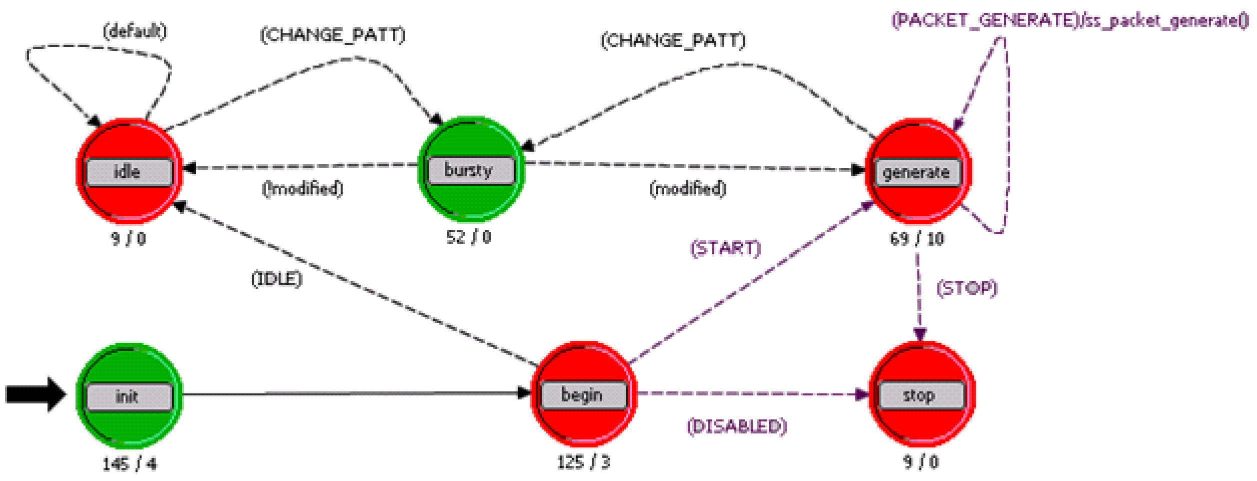


Figure 35: Finite state machine of SRC processing nodes.

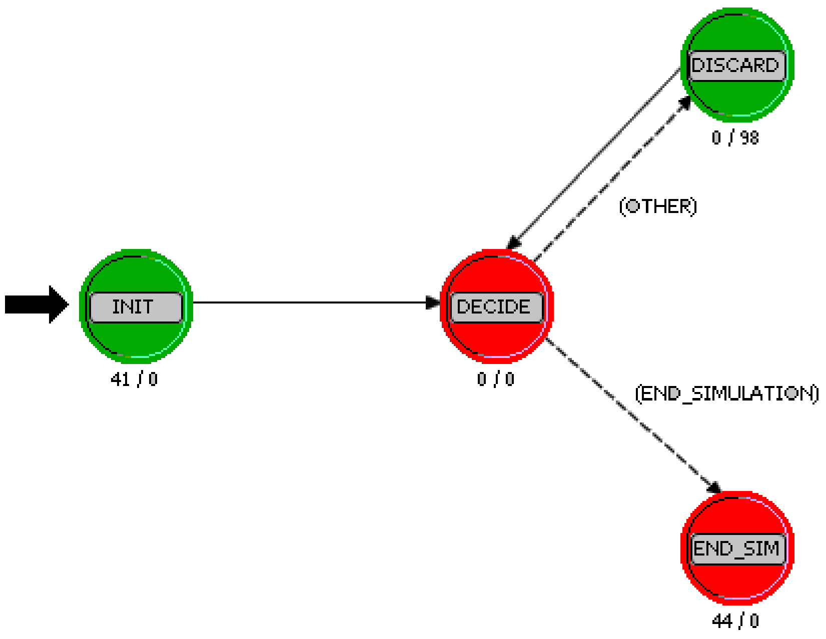


Figure 36: Finite state machine of DST processing nodes.

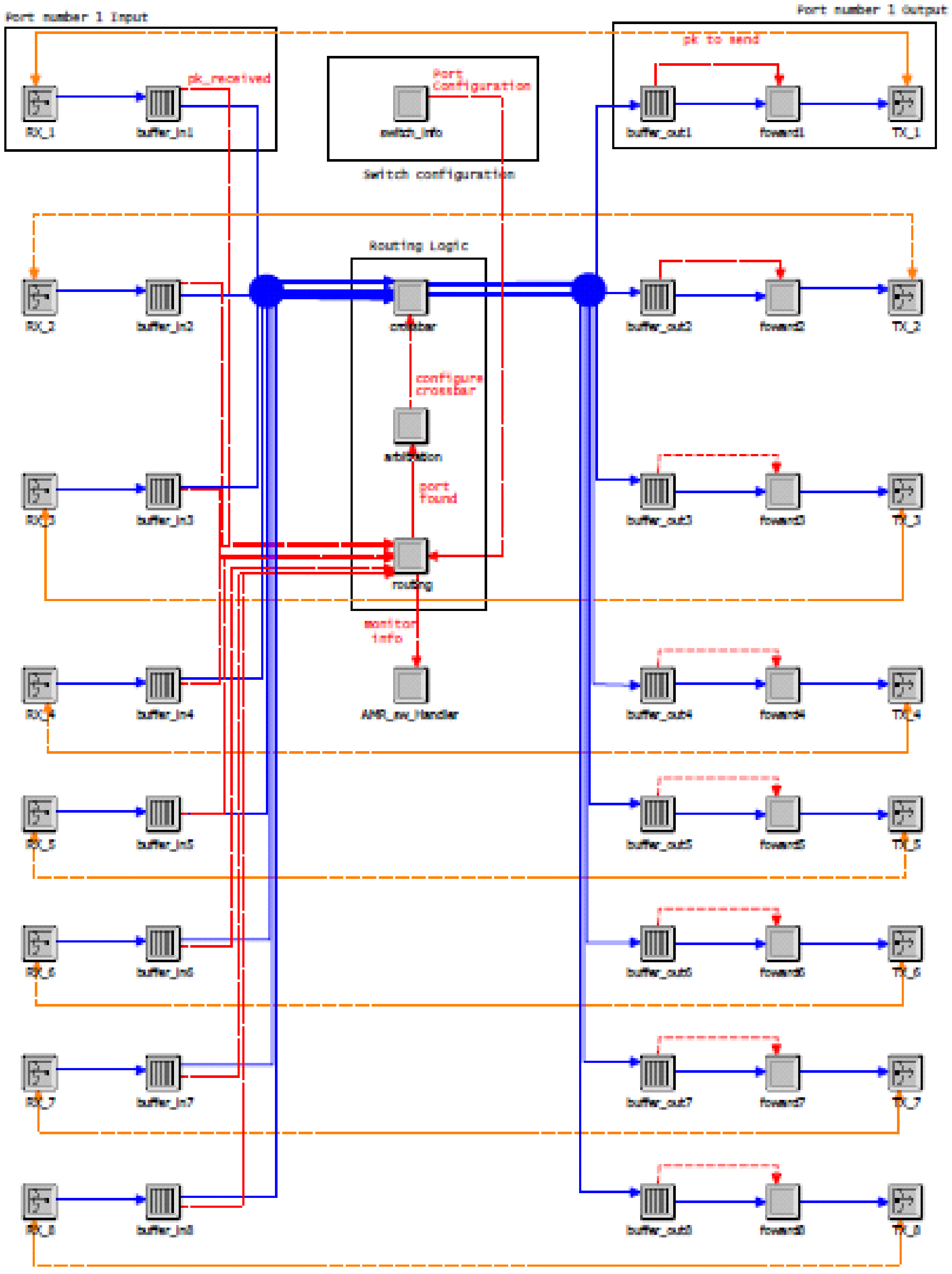


Figure 37: Router model implementation.

5.2 Experiments Designs

Methodology of the experiments executed is based on the response of the algorithm PR-DRB under situations of hotspot traffic. The main idea is to identify PR-DRB behavior under transitory traffic condition, based on repetitiveness of communication and applications patterns, as well as to verify traffic load distribution of the algorithm under all of this traffic load.

Hotspot situations are created making some nodes in the network receive a lot of flows from communications between different source/destination pair, converging at some specific point during its traversal, thus forcing saturation of some areas in the network.

Experiments were done over a 8x8 2D mesh network topology, distributed across 64 intermediate routing nodes, and one processing node connected to each of the routers. Simulations under different test scenario were performed, changing hotspot situations to determine and observe the response of the algorithm, watching latency values, and traffic load distribution performed during each step of the bursty traffic, as shon in Table 5.

Parameters	Value	Description
Time	1.02 seconds	Execution time
Traffic generation	Exponential	Packet injection rate
Load	1000-1800 (Mbps)	Traffic load offered by a node
Routing Algorithms	DRB ; PR-DRB	Routing algorithms used
Network size	8x8 (64 nodes) mesh	Network topology
Packet size	1024 bytes	Packet size used
Flow control	Virtual cut-through	Flow control technique used
Link bandwidth	2 Gbps	Maximum bandwidth per link node

Table 5: Common experiments configurations.

The mesh network topology is selected for this proposal, because it offers one essential feature for our work, which is the great number of alternative paths that can be found to reach a destination node, besides it works well in the validation of the idea of the predictive module because it is used in many experiments in the area of interconnections networks. Future experiments, after the ideas has been correctly validated, would be based on other topologies such as torus, among others.

5.3 Path distribution analysis

To study the behavior of DRB algorithm under congestion situation, several examples were run to determine how the procedure of new alternative path creation is performed. This is a key situation to understand traffic behavior under different situations of traffic in the network, thus giving a wide idea of how the algorithm can be improved. To produce the suitable congestion situation to force aperture of new alternative path, hotspot is introduced into the network. This is accomplished concentrating traffic in some areas of the network, thus overloading this subset of intermediate routers forcing processing nodes to start procedures of contention to avoid congested paths.

In Figure 38 a situation of hotspot is depicted and a complete scheme of the solutions given by DRB can be visualized. Figure 38 (a) show the initial state of the network, where all nodes start transmitting and enter a common area in their path toward their destination, where congestion will be produced as a result of this situation. There is one flow that is not part of the congestion zone, and its communications advances without major inconvenient. In figure (b), DRB has been notified about the congestion situation detected by intermediate routers, and the figure shows one alternative path opened, complementing the original trajectory. This new alternative path opened now collides with the trajectory that until this point was not involved in the common congestion situation. A snapshot of the complete picture of the network can be shown in (c). There, all the processing nodes involved in the congestion detect the situation and advance with the procedure of new paths aperture. In this case, the red processing node now is affected by the decision taken previously by DRB, and to alleviate the latency occasioned by having to share resources, a new alternative path is also created here.

Figure 39 depicts another situation of hotspot. This has the peculiarity of opening several alternative paths until reach a stable latency value in the entire network. This behavior of the algorithm is obtained in a controlled manner, opening one path a the time and evaluating the effect of that path into latency values. If latency values are still beyond an upper threshold, then another alternative path must be introduced into the network, otherwise the combination of paths opened so far is good and latency values are under a controlled threshold and consider good solutions.

Figure 40 Shows a different congestion situation in the network, where hotspot appears at two different points. One packet traversing the network in this situation must pass though all congested areas before reaching its destination. Every intermediate router adds up the latency value registered when this packet uses its resources, until the packet get to destination and proceed with the notification of that latency.

Again, the source node first open one path (P2) to try to solve the congestion situation, but after receiving another notification of latency registered in the original path and in the new alternative path (P2), source node decides to open other path (P3) in order to control the situation. In this scenario, opening the first path and then waiting for the whole notification process can be very costly, because source and destination node are far away from each other, and they may travel through still congested areas to deliver the message.

We can conclude after this hotspot situations examination, that a valid method to improve the DRB algorithm is to collect information about past behavior of the communications, and then use that information to make intelligent prediction about future traffic conditions, and avoid most unfavorable situations.

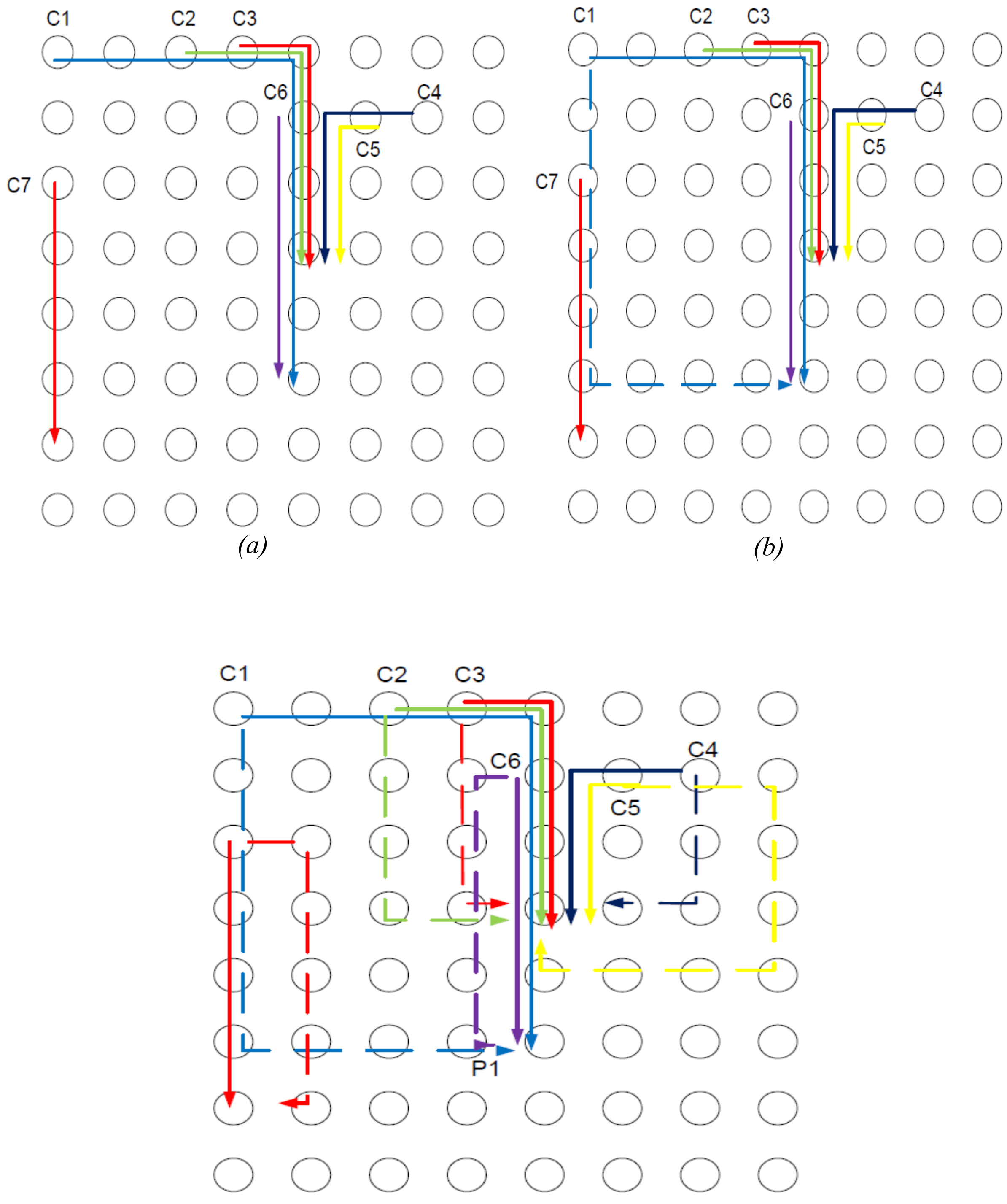
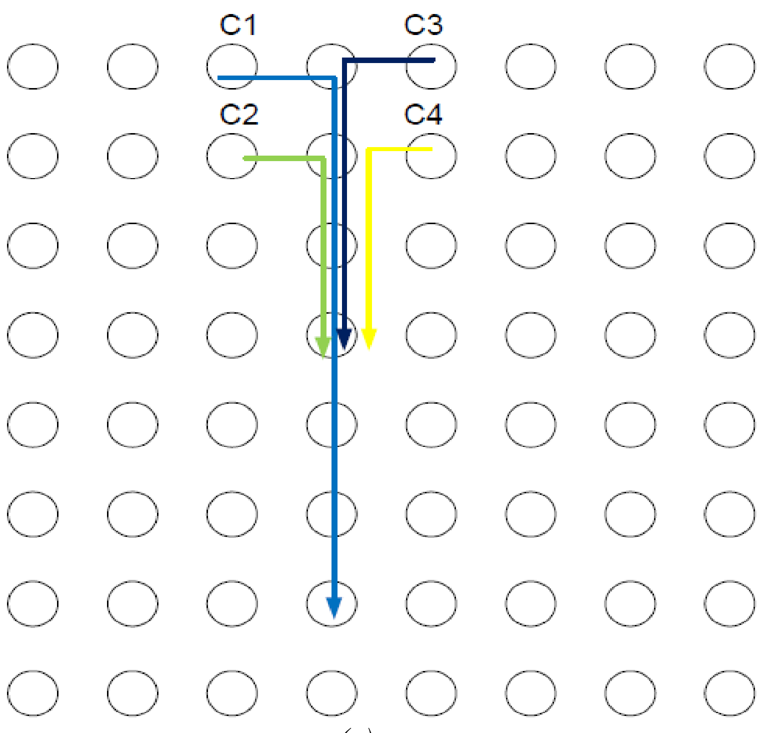
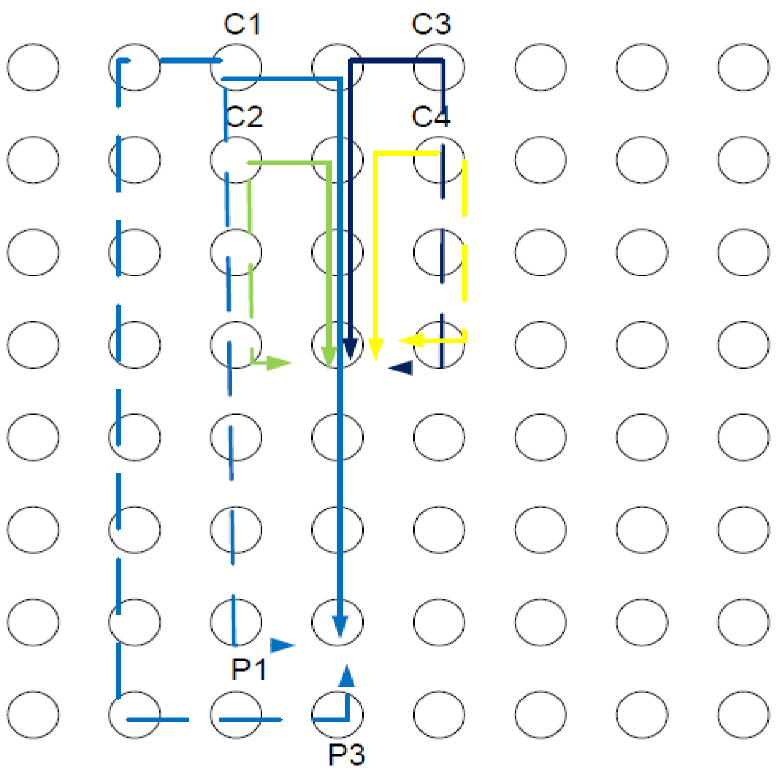


Figure 38: Hotspot situation 1.

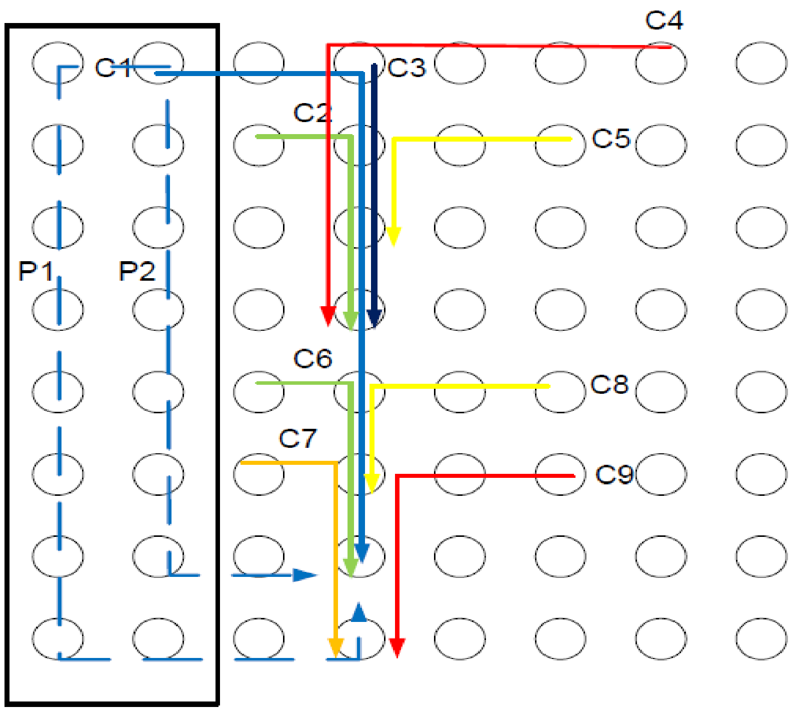


(a)



(b)

Figure 39: Hotspot situation 2



(b)

Figure 40: Hotspot situation 3.

5.4 PR-DRB evaluation.

This section analyzes PR-DRB behavior against the original DRB algorithm. Latency value is the metric used to perform those comparison. The main goal of this evaluation is to visualize the effect of the original DRB algorithm under bursty traffic conditions, where repetitiveness in traffic communications patters is perceptible, and then compare the PR-DRB performance to make some final conclusions.

We must have in mind bursty traffic characteristic to fully understand the test scenario. As seen in section 3.8.2 , there are several stages during traffic generation, one of them is well represented by applications communicating and the rest where applications are performing some computation. During the communication stage, traffic load is considerable high because depending on the communication pattern decided by the application, many nodes may be involved in such transmissions thus augmenting network resources demand during that period of time. The other phase is represented by computations, where all nodes stop transmitting application data, and start doing some real work with the data obtained, and communication are represented only for synchronization and other management related. This two phases may be repeated several times during the whole life of the application.

Based on bursty traffic, Figure 41 shows the latency behavior of both approaches, DRB and PRD-DRB. We can see in this figure that both algorithms practically behaves the same way. This situation is because during the first stage of the bursty traffic, both algorithms do not have any knowledge about the network status so far, so they are in a "learning stage". PR-DRB is continuously opening and closing paths to try to stabilize global latency value, and is also learning from the communication pattern actually been transmitted over the network. Figure 42 Depicts other concrete situation, corresponding to further stages of bursty traffic. Here, we can see that PR-DRB start behaving slightly different than its predecessor DRB. Because during this phase PR-DRB already have some information about the network and communications patterns, saved in the first stage of bursty traffic, it is able to identify an already analyzed situation and then re apply the best solution encountered to stabilize latency values. We can see that latency values in the raising area of the figure is diverging from each other at approximately between 1 and 1.5 us. This is because PR-DRB has re applied the best known solutions, and no extra time to find them is necessary. Also, high latency peaks caused by this search of good solutions is avoided with PR-DRB.

On average latency values, as the one represented in the figure, PR-DRB maintain better results during the total execution time of the experiments, meaning that all solutions learned, were intelligently re applied after communication patterns identification process where carried out. On the other hand, DRB still behaves as predicted and latency reach high values during all the consequent stages, before encounter a proper combination of paths to stabilize the system.

In Figure 43 we can see raw instant data, during the first portions of a second, where the algorithm is still adapting to traffic injection before adapting to a stabilized latency. We can see that between 0.0002 and 0.0004 values in the x axis, PR-DRB practically is stabilized when DRB is still looking for good solutions to avoid high latency peaks. We can also see in the figure that PR-DRB also has some peaks, but compared to DRB are always shorter. Another interesting point is that latency peaks encountered in DRB appears in PR-DRB curve afterward, or displaced, because PR-DRB has a sharper control of paths. This situation is due to detection of congestion is done at the same time in both algorithms, but applying of good solutions directly in PR-DRB force it to remain longer in the working zone of the network. We can see that this particular situation is repeated several times during the exposed period of time in the figure.

Under some points in Figure 43, PR-DRB is over the curve of DRB, resulting in higher latency values during that period of time. This is caused by the dynamic behavior of DRB, that is capable of adapting to changing traffic conditions according to different parameters. To carry out this experiments, PR-DRB module was not fully implemented and of course, some situations are better handled by DRB because it is a fully modeled implementation.

Figure 44 Depicts a similar situation, but during the last period of time of execution, and in terms of global average latency values. In this figure, we can see that PR-DRB have better response in latency values even during the final stage of simulations, meaning that it can find good solutions during the first stage of traffic and the idea of re applying of them, once similar communication pattern appears again, is feasible. Here, both curves have similar behavior, marked only by the difference between latency values among each other. Traffic injection is affecting both of the algorithms, but reaction of one is better comparing to the other, in this case, PR-DRB outperforms DRB.

In respect to traffic injection, no penalization was introduced into the network. The predictive module with all its phases executed, at intermediate routers and source nodes basically, did not make the network to drop any packet, thus maintaining the global throughput achieved with DRB. This throughput was maintained by PR-DRB with the advantage of keeping low latency values, or out of hotspots situations during the stabilization phase, as much as possible.

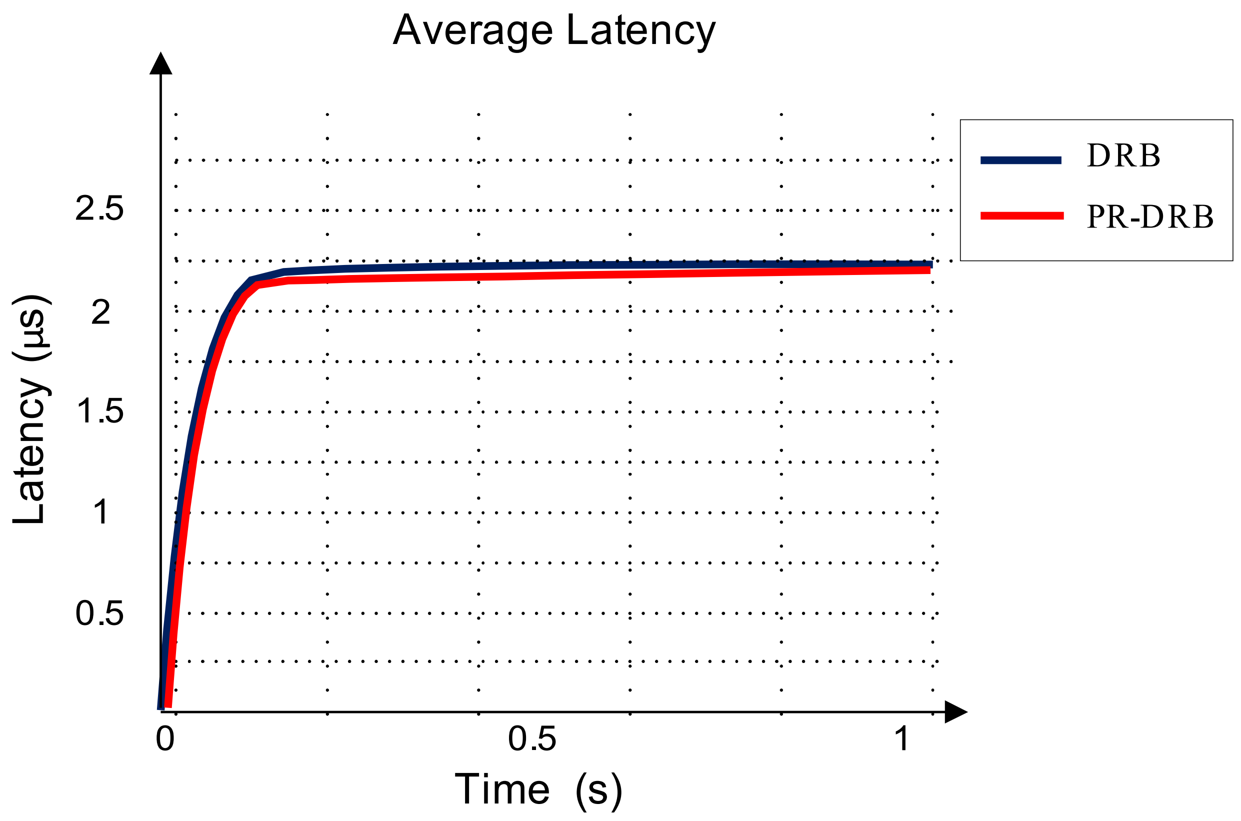


Figure 41: First stage latency behavior.

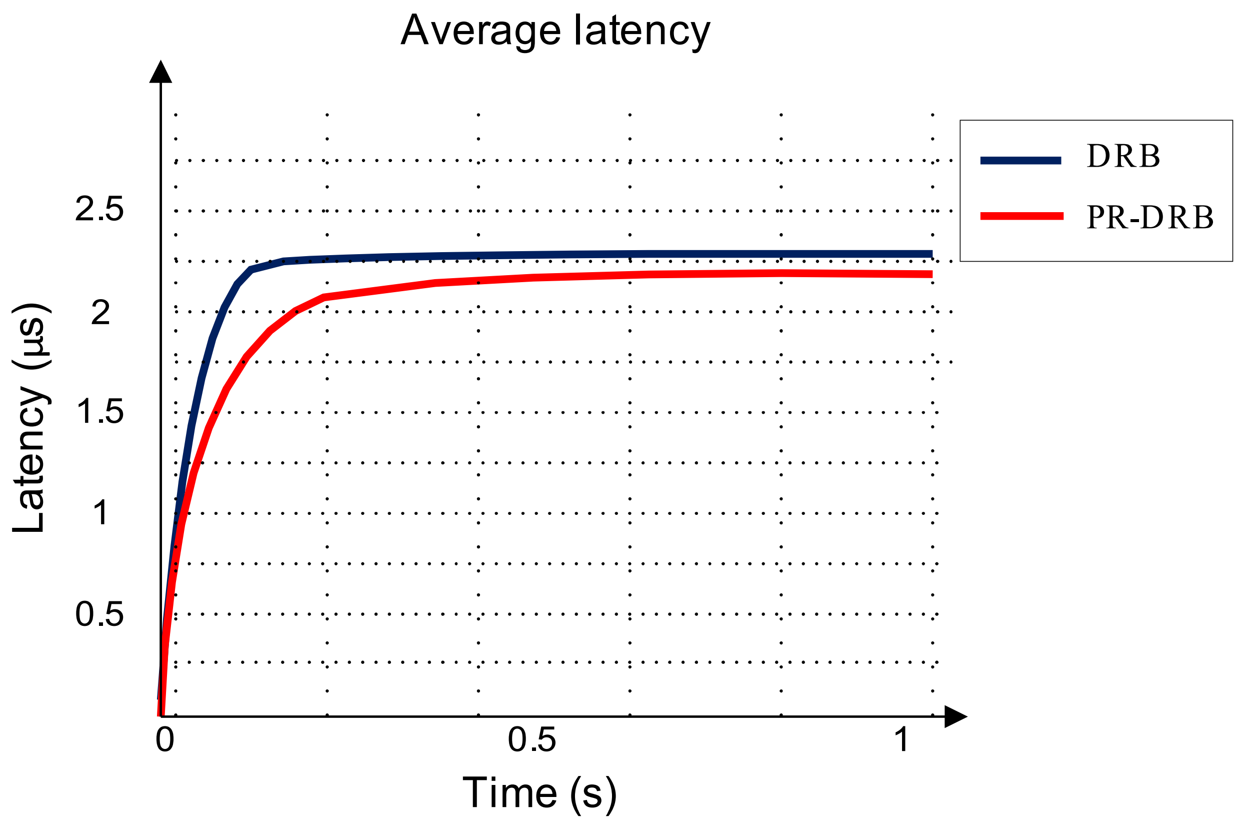


Figure 42: Next stages under bursty traffic.

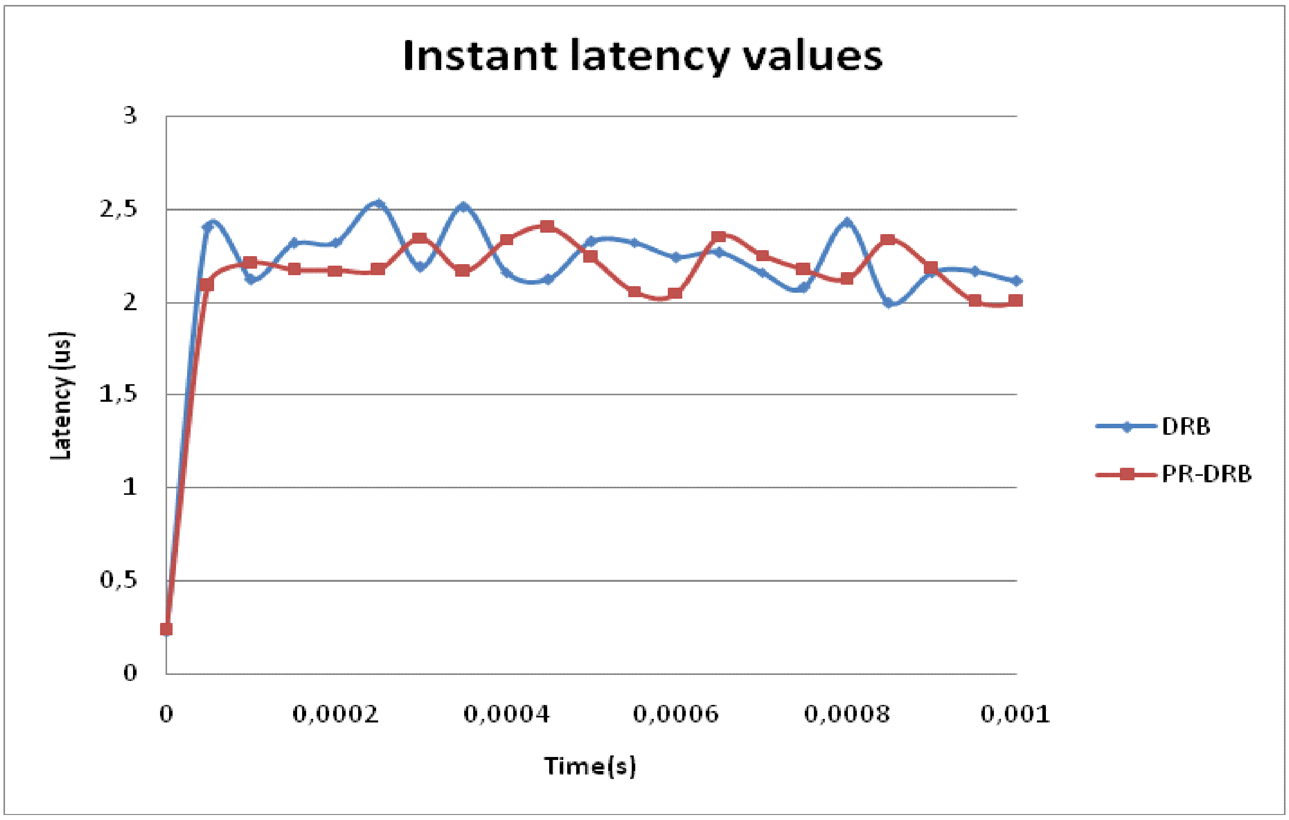


Figure 43: Latency values during initial communications.

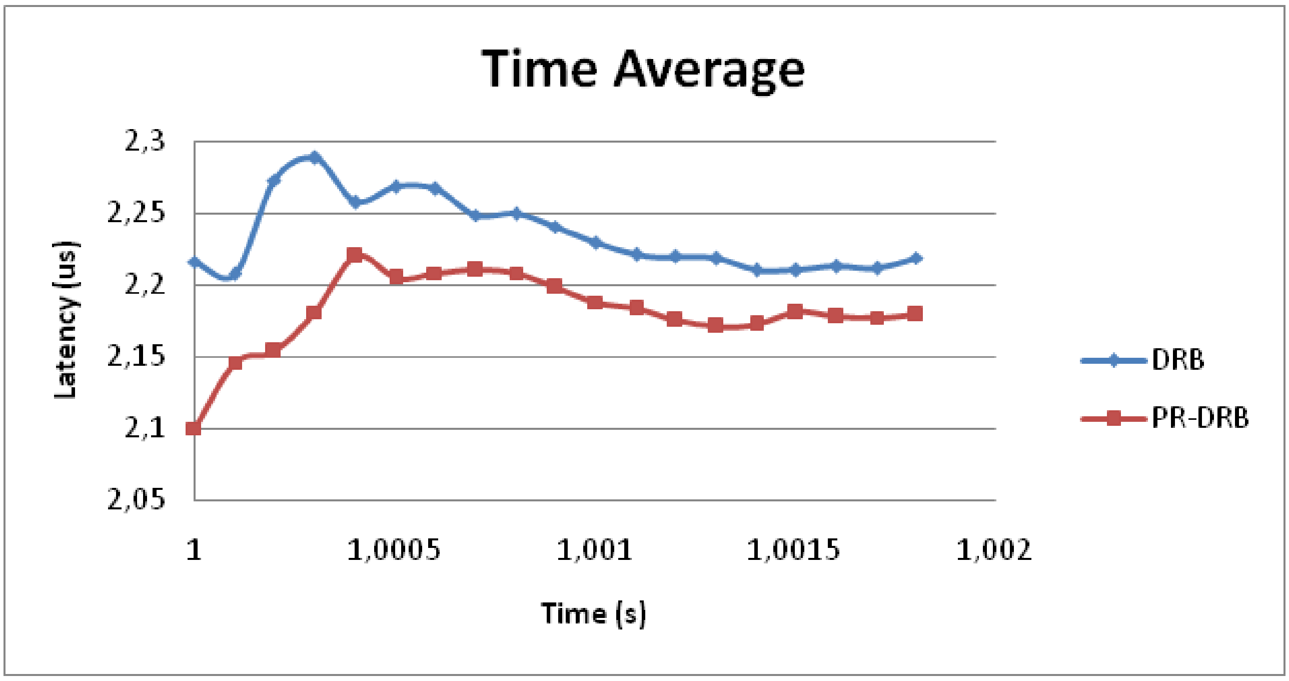


Figure 44: Latency values (average) during final communications.

5.5 Latency map

Latency distribution across the entire set of routers is presented in Figure 45 and Figure 46. Latency map shows average traffic managed by a router throughout the entire execution of the simulation. The idea of these graphs is to show how traffic is distributed across the network and how the predictive approach influence the gain in latency values expressed so far.

Figure 46 depicts PR-DRB behavior, where we can see a high peak of latency concentrated mostly in one spot of the network. This peak is the reaction of the algorithm to traffic injection produced during bursty stages, and can be considered as a “normal” value, because it is accordingly with traffic load. Recall that this is a controlled situation and traffic is not injected at the whole network, but to controlled areas where the hotspot is wanted. Compared to DRB behavior, shown in Figure 45, it is clearly visualized that PR-DRB has less congested areas in the whole network. For example, routers indexed by numbers 0 to 2 are heavily loaded under DRB traffic. Similar situation occurs at routers corresponding to greater values in the x axis. This routers corresponds to most external nodes in the network, and are heavily used by DRB because the hotspot situation in focused in central part of the network, thus using most external nodes as escape path.

The reason of this behavior, is related to the fact that DRB opens many alternative paths during its search for the optimum combination of alternatives, as a result it may potentially overload some portion of the network that already have high traffic in its links. This unnecessary overload causes that the now-congested sources start to open new alternative paths, to control what is notified to them as a congestion situation. Because DRB will take some important time to find all the alternative paths to control a particular situation, at future time maybe this now-congested path will not be under the same traffic load, thus having to close those alternative paths already opened and re adapt to traffic conditions. We can also see that most external routers do handle some traffic, as the slightly raised spots shown in Figure 46 at border routers, because PR-DRB main goal is to distribute traffic in the best possible way, and in this case it involves the use of these resources to alleviate global latency value and traffic load in the entire network.

Another interesting aspect shown in these figures, is that PR-DRB has its maximum peak of latency below the maximum registered for DRB, as a result of less congested paths introduced by using the optimum combination previously encountered.

All this situation leads to a better distribution of traffic load across a correct number of communication nodes, avoiding situations where intermediate routers are overloaded handling traffic, while other routers may be idle or underutilized.

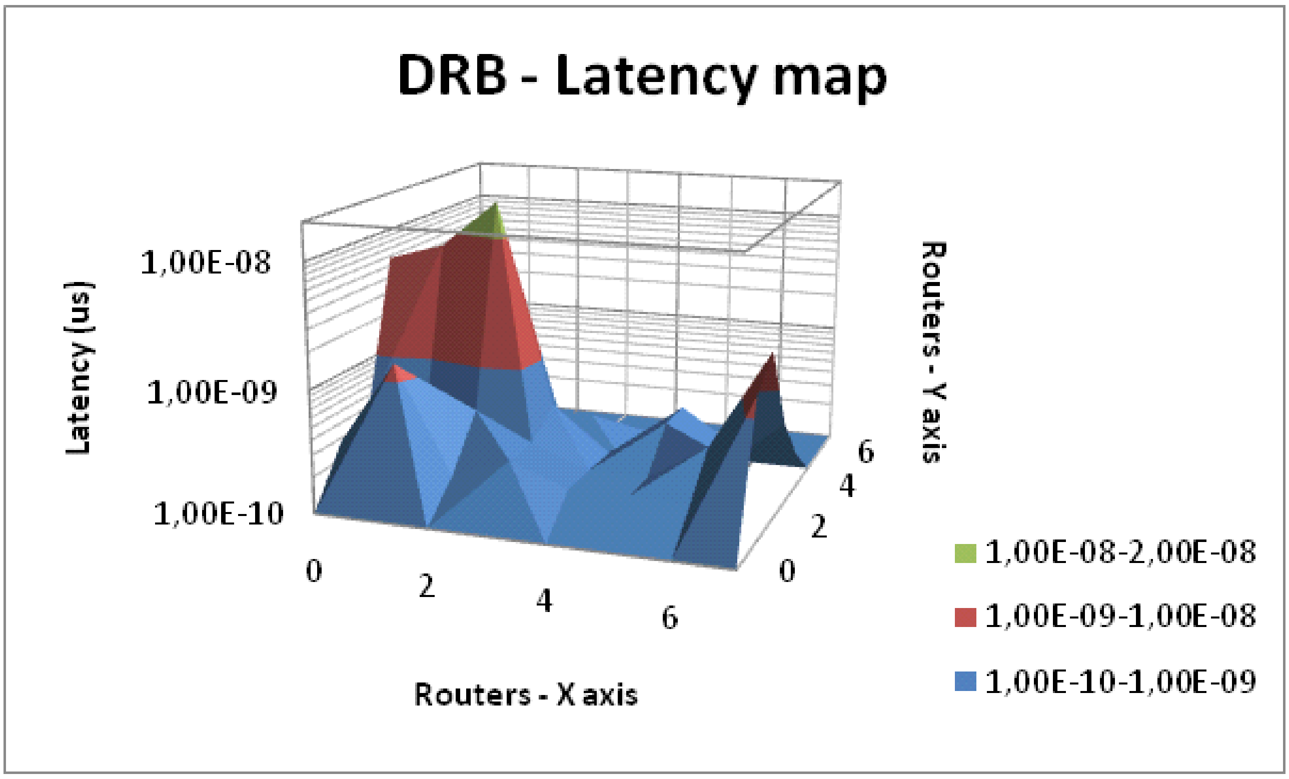


Figure 45: DRB latency map.

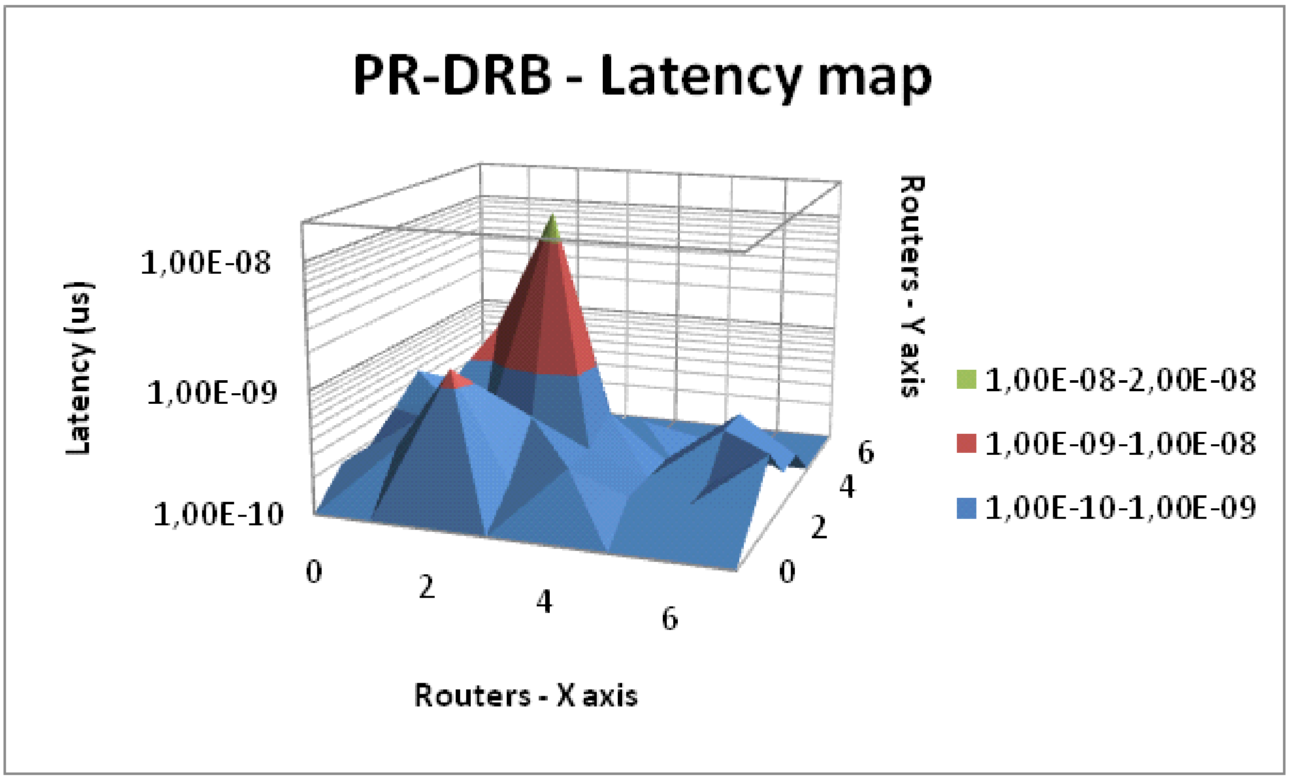


Figure 46: PR-DRB latency map.

Chapter 6 Conclusions And Future Work

6.1 General conclusions

This is the last chapter of this dissertation, but in any way it means that there is not more work to do about the ideas presented here. On the contrary, now there is a wide spectrum of alternatives to start thinking about how to continue what have started with this work.

This work basically proposes a new method of routing in an interconnection network, called ***Distributed and Predictive Routing Balancing***. This new strategy is based on path distribution under congestion situations across many alternative paths, to keep latency values controlled and enough bandwidth available to applications running in the interconnection network. All of this in the shortest possible time and dynamically.

Because parallel application present well defined phases during its life time, and repetitive behavior, PR-DRB is able to learn about the situations that cause congestion in the network, tagging the communication pattern involved in congestion as troubled, and then identify this tagged pattern and re-apply the best solutions encountered for each particular case.

Throughout all chapters of this document, many concepts and ideas relevant to the field of interconnection network were presented, as well as the problematic details of modern strategies used to improve performance. All these led to the main motivation to start this work, and propose new alternatives to improve the performance of interconnection networks.

Description of the fields were interconnections network are applicable and the problems it help to solve were commented in chapter one. More theoretical concepts were presented in chapter two, where all definitions relevant to this work were presented and explained. In depth analysis of traffic load distribution is studied in chapter three, giving also enough background to understand and delimit this work scope, improve the overall performance of interconnection network, through communication balancing and low response time. Having analyzed the internals of path distribution, details of the proposal to accomplish the defined goals were described in chapter four. Then, details of what have been modified in existing models to basically implement our proposal were described in the fifth chapter. In this chapter also is presented the environment aspects of the experimentation carried out and the results obtained.

The main goals initially defined were accomplished along the development of this work. The extension of the original DRB algorithm to work accordingly under controlled bursty traffic conditions were probed with successfully results.

6.2 Results Conclusions

In this work we have proposed a predictive routing approach, called ***PR-DRB***, which is targeted to improve interconnection network latency values under congestion or hotspot situations. Results obtained from the experiments carried out show that our proposal is valid as a mechanism of predictive routing in interconnection networks. We also observed that by applying our mechanism, we can obtain relevant improvements in latency values under specific traffic loads. Based on repetitive behavior of parallel applications, and bursty traffic load conditions, the proposal algorithm can extract relevant information about the agents involved in the congestion situation at any particular intermediate node. The information gathered during the process of troubled pattern recognition, is notified to involved source node, so that it can take proper actions to control congestion in the network. PR-DRB use all these collected information, in a particular source node, to start the procedure of contention. Contention is done opening new alternative paths to alleviate high latency values registered during hotspot situations. New alternative paths opening can be a costly process, in terms of time and latency, and PR-DRB apply directly the best historical information recorded previously, when the troubled communication pattern re-appears.

After our experiments we saw that PR-DRB performance, under bursty traffic conditions, was better than the original DRB, through a series of latency improvements. We also observed that latency values in PR-DRB raises smoothly and stabilizes sooner, thus avoiding critical increasing in latency values during the process of finding (or opening) alternative paths to balance communications in the network. In addition, and as a result of the predictive scheme applied, the global load distribution in the network is also optimized. This implies that intermediate routers are less loaded because traffic is flowing through the best available paths right away, avoiding unnecessary communications and keeping these resources available to other communications requests.

This work basically presents the methodology of a predictive routing approach, and after observing initial experiments results, in depth analysis of proposed approach is necessary, in order to extend the idea and make solid proposals about performance gain. During the development of this work, many other related ideas emerged as well, and can be considered as promising.

6.3 Open Lines And Future Work

To continue this work, short term goals and activities can be described. First, more experiments must be executed, in order to finely tune predictive module behavior and scope. This fine tuning can lead to full routing predictive modules implementation, based on already existing models developed with OPNET simulator tool. Having the predictive module partially or completely developed, then new experiments may be carried out in order to analyze different conditions or features to enforce predictive scheme methodology, such as:

- Use different interconnection network topologies, such as k-ary n-cube, 3D meshes, among others

- Execute experiments under different traffic loads, to study PR-DRB behavior under different load scenarios and its response in time and latency values
- Study the effect of scientific applications in the interconnection network
    - Analyze parallel applications and its communications patterns

As open lines to continue this work we can mention the following:

- Predict future traffic conditions and be able to detect congestion situation before it effectively appears in the interconnection network
    - By speculating about future traffic congestion, based on historical latency values, analyzing latency trend distribution in time.
- Propose *"application-aware"* predictive modules
    - Obtaining communication patterns from parallel applications, in order to maximize performance and resources availability

6.4 Paper Published

As a result of this work, an article have been elaborated and submitted. The conference where the paper have been submitted is CLEI (Latin American Conference of Informatics) in its 36th edition. As the content of the article the whole methodology of the predictive module was included and described, as well as initial results validating the proposal. By the time this document was finished, notification phase of acceptance or rejection was not yet concluded.

Bibliography

[1] Ortega,Julio. and Mancia, Aguita. and Alberto, Prieto., Arquitectura De Computadores, Thomson,Thomson,1,2005, pp.

[2] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers,,1,2004, pp. pp. 1-20

[3] Top500 Supercomputers site, Interconnection family share for 06/2010, 2010 [online]. Available: http://www.top500.org.

[4] Flynn, Michael J., Very high-speed computing systems, , 2000, pp. 519--527

[5] Wang, Mu-Cheng and Siegel, Howard Jay and Nichols, Mark A. and Abraham, Seth, Using a Multipath Network for Reducing the Effects of Hot Spots, *IEEE Trans. Parallel Distrib. Syst.*, 1995, pp. 252--268

[6] Dandamudi, Sivarama P., Reducing Hot-Spot Contention in Shared-Memory Multiprocessor Systems, *IEEE Concurrency*, 1999, pp. 48--59

[7] Baydal, Elvira and Lopez, Pedro and Duato, Jose, A Family of Mechanisms for Congestion Control in Wormhole Networks, *IEEE Trans. Parallel Distrib. Syst.*, 2005, pp. 772--784

[8] Wong, A. and Rexachs, D. and Luque, E., Parallel application signature, *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 2009, pp. 1-4

[9] D. Franco., Balanceo distribuido del encaminamiento de redes de interconexión de computadoresparalelos, 2000, pp.

[10] J. Duato, S. Yalamanchili, and L. M. Ni, Interconnection networks. An Engineering Approach, Morgan Kaufmann,,,2003, pp.

[11] AS. Tanenbaum, Computer Networks, Prentice-hallH,Englewood Cliffs,3rd,1997, pp.

[12] Dally, William J., Performance Analysis of k-ary n-cube Interconnection Networks, *IEEE Trans. Comput.*, 1990, pp. 775--785

[13] Song, Yong Ho and Pinkston, Timothy Mark, A New Mechanism for Congestion and Deadlock Resolution, *ICPP '02: Proceedings of the 2002 International Conference on Parallel Processing* , 2002, pp. 81

[14] Shihang Yan and Geyong Min and Irfan Awan, An Enhanced Congestion Control Mechanism in InfiniBand Networks for High Performance Computing Systems, *Advanced Information Networking and Applications, International Conference on*, 2006, pp. 845-850

[15] Tamir, Yuval and Frazier, Gregory L., Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches, *IEEE Trans. Comput.*, 1992, pp. 725--737

[16] M. E. Gómez and J. Flich and A. Robles and P. López and J. Duato, VOQ_SW: A Methodology to Reduce HOL Blocking in InfiniBand Networks , [online]. Available:

[17] Kyung Min Su and Ki Hwan Yum, Simple and Effective Adaptive Routing Algorithms inMulti-Layer Wormhole Networks, *IEEE International Performance, Computing and Communications Conference*, 2008, pp. 176-184

[18] Paul, Gratz. and Boris, Grot. and Stephen, Keckler, Regional Congestion Awareness for Load Balance in Networks-on-Chip, *Proceedings of the 14th International Symposium on High Performance Computer Architecture* , 2008, pp. 203-214

[19] Hu, Jingcao and Marculescu, Radu, DyAD: smart routing for networks-on-chip, *DAC '04: Proceedings of the 41st annual Design Automation Conference* , 2004, pp. 260--263

[20] Vishnu, A. and Koop, M. and Moody, A. and Mamidala, A. R. and Narravula, S. and Panda, D. K., Hot-Spot Avoidance With Multi-Pathing Over InfiniBand: An MPI Perspective, *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid* , 2007, pp. 479--486

[21] Lu, Zhonghai and Zhong, Mingchen and Jantsch, Axel, Evaluation of on-chip networks using deflection routing, *GLSVLSI '06: Proceedings of the 16th ACM Great Lakes symposium on VLSI* , 2006, pp. 296--301

[22] Arjun Singh and William J Dally and Amit K Gupta and Brian Towles, GOAL: A load-balanced adaptive routing algorithm for torus networks, *International Symposium on Computer Architecture (ISCA* , 2003, pp. 194--205

[23] Duato, J. and Johnson, I. and Flich, J. and Naven, F. and Garcia, P. and Nachiondo, T., A New Scalable and Cost-Effective Congestion Management Strategy for Lossless Multistage Interconnection Networks, *HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture* , 2005, pp. 108--119

[24] Singh, Arjun and Dally, William J. and Towles, Brian and Gupta, Amit K., Globally Adaptive Load-Balanced Routing on Tori, *IEEE Comput. Archit. Lett.*, 2004, pp. 2

[25] Singh, Arjun and Dally, William J. and Gupta, Amit K. and Towles, Brian, Adaptive channel queue routing on k-ary n-cubes, *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures* , 2004, pp. 11--19

[26] Thottethodi, Mithuna and Lebeck, Alvin R. and Mukherjee, Shubhendu S., BLAM: A High-Performance Routing Algorithm for Virtual Cut-Through Networks, *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing* , 2003, pp. 45.2

[27] Thottethodi, Mithuna and Lebeck, Alvin R. and Mukherjee, Shubhendu S., Self-Tuned Congestion Control for Multiprocessor Networks, *HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture* , 2001, pp. 107

[28] Puente, V. and Izu, C. and Beivide, R. and Gregorio, J. A. and Vallejo, F. and Prellezo, J. M., The adaptive bubble router, *J. Parallel Distrib. Comput.*, 2001, pp. 1180--1208

[29] Franco, D. and Garces, I. and Luque, E., Distributed Routing Balancing for Interconnection Network Communication, *HIPC '98: Proceedings of the Fifth International Conference on High Performance Computing* , 1998, pp. 253

[30] Franco, D. and Garces, I. and Luque, E., A new method to make communication latency uniform: distributed routing balancing, *ICS '99: Proceedings of the 13th international conference on Supercomputing* , 1999, pp. 210--219

[31] Glass, Christopher J. and Ni, Lionel M., The turn model for adaptive routing, *SIGARCH Comput. Archit. News*, 1992, pp. 278--287

[32] Valiant, L. G. and Brebner, G. J., Universal schemes for parallel communication, *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing* , 1981, pp. 263--277

[33] Bolding, Kevin and Fulgham, Melanie and Snyder, Lawrence, The Case for Chaotic Adaptive Routing, *IEEE Trans. Comput.*, 1997, pp. 1281--1292

[34] Dally, William and Towles, Brian, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc.,,,2003, pp.

[35] Sherwood, Timothy and Perelman, Erez and Calder, Brad, Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications, *PACT '01: Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques* , 2001, pp. 3--14

[36] Rodriguez, German and Beivide, Ramon and Minkenberg, Cyriel and Labarta, Jesus and Valero, Mateo, Exploring pattern-aware routing in generalized fat tree networks, *ICS '09: Proceedings of the 23rd international conference on Supercomputing* , 2009, pp. 276--285

[37] Lugones, D. and Franco, D. and Luque, E., Modeling adaptive routing protocols in high speed interconnectionnetworks, *OPNETWORK 2008 Conference*, 2008, pp.

[38] OPNET Technologies, Opnet Modeler Accelerating Network R\&D, 2008 [online]. Available: http://opnet.com

Alphabetical Index