Universitat Autònoma
de Barcelona

Escola d'Enginyeria

Departament d'Arquitectura de Computadors i Sistemes Operatius

Predictive and Distributed Routing Balancing for High Speed Interconnection Networks

Thesis submitted by **Carlos Heriberto Núñez Castillo** for the degree of Philosophiae Doctor by the Universitat Autònoma de Barcelona, under the supervision of Dr. Daniel Franco Puntes, done at the Computer Architecture and Operating Systems Department, PhD. in High performance Computing

Barcelona, July 2013

Predictive and Distributed Routing Balancing for High Speed Interconnection Networks

Thesis submitted by Carlos Heriberto Núnez Castillo for the degree of Philosophiae Doctor by the Universitat Autònoma de Barcelona, under the supervision of Dr. Daniel Franco Puntes, at the Computer Architecture and Operating Systems Department, Ph.D. in High-performance Computing.

Barcelona, July 2013

Supervisor

Dr. Daniel Franco Puntes

Abstract

English

In high performance clusters, current parallel application communication needs, such as traffic pattern, communication volume, etc., change along time and are difficult to know in advance. Such needs often exceed or do not match available resources causing resource use imbalance, network congestion, throughput reduction and message latency increase, thus degrading the overall system performance. Studies on parallel applications show repetitive behavior that can be characterized by a set of representative phases. This work presents a Predictive and Distributed Routing Balancing (PR-DRB) technique, a new method developed to gradually control network congestion, based on paths expansion, traffic distribution, applications pattern repetitiveness and speculative adaptive routing, in order to maintain low latency values. PR-DRB monitors messages latencies on routers and saves the found solutions to congestion, to quickly respond in future similar situations. Traffic congestion experiments were conducted in order to evaluate the performance of the method, and improvements were observed.

keywords: Interconnection Networks, High Performance Computing, Predictive Routing, Application-Aware Routing

Castellano

En los clusters de altas prestaciones, los requerimientos actuales de las comunicaciones de las aplicaciones, como el patrón de tráfico, el volúmen de comunicaciones entre otras, pueden cambiar a lo largo del tiempo y son difíciles de predecir. Estas necesidades generalmente exceden o no se corresponden con los recursos disponibles realmente, lo cual conlleva a una situación de desbalanceo de los recursos, congestión en la red, reducción del throughput y un incremento considerable en los valores de latencia de los mensajes. Todo esto conlleva una degradación general del rendimiento de todo el sistema computacional. Los estudios de las aplicaciones paralelas demuestran que estas tienen un comportamiento repetitivo. Además, esta repetitividad puede detectarse y caracterizarse a través de unas fases representativas. Este trabajo propone un Algoritmo de Encaminamiento Predictivo y Distribuido (PR-DRB). Este nuevo método propone controlar la congestión de la red de manera gradual basándose en la expansión controlada de caminos, la distribución del tráfico, la repetitividad en las aplicaciones paralelas y el encaminamiento adaptativo especulativo; de manera a mantener los valores de latencia controlados. PR-DRB monitorea la latencia de los mensajes en los encaminadores y guarda las mejores soluciones adaptativas encontradas a una situación de congestión. Esto se realiza de manera a re aplicar estas mejores soluciones de manera rápida ante situaciones similares futuras. Fueron desarrollados varios experimentos que generen congestión de tráfico a fin de evaluar el rendimiento de la propuesta, y se han logrado mejoras importantes en el rendimiento global del sistema.

Palabras clave: Redes de interconexión, Computación de Altas Prestaciones, Encaminamiento Predictivo, Encaminamiento Basado en las Aplicaciones.

Català

En els clústers d altes prestacions, els requeriments actuals de les comunicacions de les aplicacions, com el patró de trànsit, el volum de comunicacions entre altres, poden canviar al llarg del temps i són difícils de predir. Aquestes necessitats generalment excedeixen o no es corresponen amb els recursos disponibles realment, la qual cosa comporta a una situació de desbalanceig dels recursos, congestió a la xarxa, reducció del throughput i un increment considerable en els valors de latència dels missatges. Tot això comporta una degradació general del rendiment de tot el sistema computacional. Els estudis de les aplicacions paral · leles demostren que aquestes tenen un comportament repetitiu. A més, aquesta repetitivitat pot detectar i caracteritzar a través d unes fases representatives. Aquest treball proposa un Algorisme d Encaminament Predictiu i Distribuit (PR-DRB). Aquest nou mètode proposa controlar la congestió de la xarxa de manera gradual basant-se en l expansió controlada de camins, la distribució del trànsit, la repetitivitat en les aplicacions paral · leles i l encaminament adaptatiu especulatiu, de manera a mantenir els valors de latència controlats. PR-DRB controla la latència dels missatges en els encaminadors i guarda les millors solucions adaptatives trobades a una situació de congestió. Això es realitza de manera a re aplicar aquestes millors solucions de manera ràpida davant de situacions similars futures. Van ser desenvolupats diversos experiments que generin congestió de trànsit per tal d avaluar el rendiment de la proposta, i s han aconseguit millores importants en el rendiment global del sistema.

Paraules clau: Xarxes d interconnexió, Computació d Altes Prestacions, Encaminament Predictiu, Encaminament Basat en les Aplicacions.

Agradecimientos

Todo este trabajo no puede acabar sin la dedicación del mismo a la que fue el motor, el engranaje principal y la llama que me mantuvo motivado a lo largo de todo estos anos. Para vos,**Nati**. Sin tu apoyo incondicional y sincero, esto no hubiera podido acabar del todo, y más que nada, no hubiera tenido valor ni significado alguno.

A la personita que hoy llevás dentro, que mucho antes de ver la luz de este maravilloso mundo y a la maravillosa mujer que la trae, ya gobierna mis pasos y mi vida. Increíble lo que una vida puede ejercer en otra.

"De todo corazón, GRACIAS."

Agradecimientos

A todo el departamento y en especial a todos los integrantes del grupo de trabajo, quienes estuvieron a mi lado en cada momento, cada reunión y cada minuto que se tomaron para conversar conmigo y dar forma a esta tesis. Gracias a Emilio y Lola por la oportunidad, los consejos, la experiencia y más que nada por el tiempo que me han dedicado. A Sandra, Hai, Gonzalo y Diego quienes estuvieron a mi lado, y con sus críticas y comentarios fueron matizando el trabajo.

A Dani Franco, mi director, de quien siempre he admirado su capacidad para resolver problemas y sin importar el grado, la complejidad o las horas que invirtamos, nos hace culminar con una sonrisa.

No puedo olvidar a mi familia, mis padres y tía Ana, que desde que tengo razón me han ensenado que el trabajo duro, honesto y constante forman la mejor receta para salir adelante. Que no existen barreras que no podamos superar con ella. Hoy, una vez más, veo que han tenido razón.

Siempre me habían dicho que a lo largo de una travesía de esta envergadura, uno logra conocer y hacerse de muy buenos amigos. Pues tenían razón. Gracias al doctorado he podido conocer a personas a quienes los llevaré presentes durante toda la vida. Ronal, Hugo, Mimí, Caro, Juan Diego, Javier Martinez y Tami sin duda han sido invaluables. Mis amigos brasilenos; Tharso, con quien hemos podido sonar un mundo mejor cada vez que hablamos; y Aprigio, extranare las sesiones de sobremesa con nuestros cafés. De ahí salieron muchas ideas que fueron como el combustible para seguir o sonar que todo se puede. Es difícil enumerar a las personas, pues muchas a lo largo de todo este tiempo han sido más que companeras. Mis más cercanos al escritorio, Javier Panadero, Andrea y Joao. Conversar con ustedes 5 minutos siempre ha servido para cambiar de aire, reírnos y aprender algo nuevo. No puedo dejar de agradecer a Sandra, por los mates, chicles, dulces y demás provisiones que siempre ha tenido disponible para todos, además por el aguante y la compañía en las interminables reuniones. No puedo olvidar a mis otros companeros que también siempre estuvieron a mi lado, Eduardo, César, Julio, Álvaro y Marcela.

A la gente del RINCE en Irlanda, especialmente a Martin Collier, Olga Ormond y

Diego Lugones; por recibirme y darme la oportunidad de formar parte de tan prestigioso centro en Dublín. También a los amigos que conocí allí, Mithun Reddy, Khalid Javed y Jin Jie.

A los peques de la familia que sirven de inspiración para seguir construyendo un país, un mundo, un lugar mejor para todos. Gracias a la Nune por su ternura infinita, al Thi por su inocencia, a Ma del Carmen por su gracia, al Joaco por sus energías y Dieguito por su carino. Ustedes son el futuro, y los que hacen que cada día tenga razón de ser. Tampoco la familia puede estar fuera, para quienes constantemente pese a la distancia nos envían sus palabras de apoyo, de amor y de esperanza. Gracias Ceci, Diego, Ricardo, Ma Cecilia, Ada, Silvia, Sebas, Vivi, na Nena, Don Julio, Nacha, Mari. Mención especial para Sonia que supo (y tuvo que) aguantarme durante estos cuatro anos.

Una mención que no puede faltar, es a mi Alma Mater, La Facultad Politécnica de la Universidad Nacional de Asunción Paraguay. En especial a María Elena y Abel. Gracias por confiar en mí y apoyarme como lo han hecho. Esto no tiene precio. La Facultad de Ingeniería de la Universidad Nacional de Itapúa también merece mi reconocimiento. Por haberme dado la oportunidad de llevar adelante el primer grupo de investigación, y llegado el momento, de apoyarme en mi decisión de conseguir el doctorado.

A mi querido país, Paraguay, por darme la oportunidad de formarme en él y salir a competir y demostrar lo que producimos en carácter intelectual y social. Gracias a las oportunidades que he encontrado en mi tierra, me enorgullece llevar siempre mi bandera en alto y espero, en breve, poder devolverle con creces todo lo que ha hecho por mí. Creo rotundamente en un país mejor, y estoy decidido a aportar mi grano de arena para conseguirlo.

"A todos, sinceramente GRACIAS."

Contents

List of Figures

List of Tables

List of Algorithms

List of Equations

Chapter 1

Introduction

"Man is still the most extraordinary computer of all."

John F. Kennedy

1.1 High Performance Computing

In the last years we have been witnesses of how the technology not only arrived at our lives, but became an essential tool in our every day activities. This technology acted as a valuable source of knowledge for the entire society. Among all these technology, the computer systems have played a fundamental role in the whole ecosystem, by linking people s demand against applications and services (globally) available. While societies evolve, the relation between them and technology evolves accordingly, leading to situations where the society demands a whole new set of computing services, but more integrated and personalized.

The massive use of personal computers and devices have steadily demanded more computer power. These increase in general use also meant an increase in massive computer systems and large datacenter in order make it feasible to accomplish communications and processing required. From this outcome, new technology emerged and lead to specialized field such as the High Performance Computing (HPC). The world of the 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). In order to limit the increases in power and cooling in HPC, the architecture are expected to change dramatically in the future. The algorithms and the applications should also change in order to remain in line with future constraints. 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.

In order to successfully achieve useful Exascale computing; the applications, the algorithms and the technology should evolve accordingly. All of these within any reasonable budget.

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 [62].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.

1.1.1 Interconnection Networks

The supercomputer concept, as reviewed above, 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 is 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. A key factor 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.

1.1.2 Interconnection Networks Analysis

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

Traffic imbalance 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. On the other hand, other portions of the network have enough resources to handle the traffic but are being locked due to poor decisions [63].

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 [13] . 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 [3].

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, is considered as an affordable technique.

1.2 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 that runs in the network, but a good algorithm must also consider the effect of the traffic dynamics that can lead to unexpected situations due to traffic imbalance 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 [64]. 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, by investing less time to adapt to imbalanced communications, avoid congestion situations among others.

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 to 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.

1.3 Objectives

The ultimate goal of this thesis is to design, implement and evaluate a predictive routing policy capable of serving interconnection networks, specifically to deal with and learn from real parallel scientific traffic and applications. 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 communication traffic pattern, known as bursty traffic, and the repetitiveness in program execution flows. Therefore, the main goals of this work can be summarized as follows:

- Improve overall system performance.
- Perform proper traffic load distribution among all communication resources.
- Avoid hotspot situation in the network.
- keep global latency values low.
- 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 the impact of parallel applications patterns 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 *Computer Architecture and Operating System* group.

1.4 Research Method

The research in this thesis is oriented to the design, implementation and evaluation of predictive routing policies; and is framed in the academic program of applied research of the *Universitat Autònoma de Barcelona*.

1. **Existing theories and observations.** Pose the question in the context of existing knowledge, theory and observations.
2. **Hypothesis.** Formulate a hypothesis as a tentative answer.

3. **Predictions.** Deduce consequences and make predictions.

4. **Test and new observations.** Test the hypothesis in a specific experiment/theory field.

5. **Old theory confirmed within a new context or new theory proposed.** When consistency is obtained the hypothesis becomes a theory and provides a coherent set of propositions that define a new class of phenomena or a new theoretical concept.

As a rule, the loop 2-3-4 is repeated with modifications of the hypothesis until the agreement is obtained, which leads to 5. If major discrepancies are found the process must start from the beginning. The results of stage 5 have to be published. Theory at that stage is subject of process of *natural selection* among competing theories. The process can start from the beginning, but the state 1 has changed to include the new theory/improvements of old theory [16].

This thesis share theoretical basis with the method developed and discussed in depth by Franco et al. [19] [21], [20], Distributed Routing Balancing (DRB); and subsequently improved by Lugones et al. [40]. These methods, together with the theory of applications repetitiveness and interconnection networks, constitute the first stage of the scientific research method of this thesis. Throughout this stage, we have conducted the fist study on interconnection networks. Books on interconnection networks by Duato et al. [17], Dally and Towles [62], and Hsu and Lin [24] have been particularly useful. Since DRB and its related methods have been discussed in depth in two previous theses, we focus on the design, implementation, and evaluation of a novel and complete predictive and application-aware mechanism. In fact, Stages 2 and 3 comprise the proposal of the predictive routing policies based on DRB. Then, in Stages 4 and 5, we have evaluated and analyzed the effectiveness of the proposed predictive routing method through simulation, using a standard discrete event simulator. To this end, we have enhanced existing models of network components [39] by including the proposed predictive and application-aware mechanisms and policies. This has allowed us to develop the simulation models used in the experimentation of our proposals.

1.5 Contributions

The work developed for the thesis has been published in the following papers.

1. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing (PR-DRB)*, In X Distributed and Parallel Processing Workshop (WPDP), pages 112-121, XVI Argentine Conference on Computer Science (CACIC),. Moron, Buenos Aires, ARGENTINA, 2010. ISBN: 978-950-9474-49-9.**
   URI http://sedici.unlp.edu.ar/handle/10915/18913 [5]
   This work focuses on the problem caused by the imbalance of network communications. The proposed method control network congestion by means of alternative paths, traffic and load distribution, in order to keep latency values low.

2. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing for High Speed Interconnection Networks*, In Proceedings of the XXII Spanish Conference on Parallelism, pages 397-402, Tenerife, SPAIN, 2011.** [9] This work introduces the congestion problem at interconnection networks and the idea of repetitive behavior in parallel scientific applications. This repetitiveness could help to improve the existing congestion control procedures. In order to accomplish this the policy uses communication path redundancy.

3. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing (PR-DRB)*, Journal of Computer Science & Technology (JCS&T) Selected Papers, pp. 59-70, ISBN: 978-950-34-0757-8, October 2011. Pearson.** [7]
   Idem [5]

4. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing for High Speed Interconnection Networks*, In Proceddings of the 2011 IEEE International Conference on Cluster Computing, CLUSTER, pages 552 -556, Austin, Texas, USA, 2011. IEEE Computer Society.** [6]
   This paper discusses the routing algorithm and the relation to the patterns from applications. The repetitiveness of parallel scientific applications is introduced. An overview of the internal mechanism proposed is explained.

5. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing for HPC Clusters*, In Proceddings of the 2011 IEEE International Conference on Parallel and Distributed Systems, PDPTA, pages 875 -878, Las Vegas, Nevada, USA, 2011.**

WorldComp Congress. [8]
This work presents a Predictive and Distributed Routing Balancing technique (PR-DRB) to control network congestion based on adaptive traffic distribution. PR-DRB uses speculative routing based on application repetitiveness. PR-DRB monitors messages latencies on routers and logs solutions to congestion, to quickly respond in future similar situations. Experimental results show that the predictive approach could be used to improve performance.

6. **C. N. Castillo, D. Lugones, D. Franco, and E. Luque. *Predictive and Distributed Routing Balancing on High-speed Cluster Networks*, In Proceedings of the 2011 23rd International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD '11, pages 72–79, Washington, DC, USA, 2011. IEEE Computer Society.** [10]
This work focuses on the predictive approach for distributed routing, based on synthetic traffic. The predictive mechanism is based on ACK notification from destination nodes. The information carried out by ACK are collected by intermediate routers traversed by each packet. The study of thresholds used by the method is discussed and explained.

7. **C. N. Castillo, G. Zarza, D. Lugones, J. Navarro, D. Franco, and E. Luque. *Cluster GUI, an Application to Launch OPNET Simulations within Resource Management Environments*, OPNETWORK Conference, pages 1–7, USA, 2011. Online: www.opnet.com** [45]
This work resumes the procedures to launch OPNET simulations within a distributed cluster such as *Sun Grid Engine* (SGE).

8. **C. N. Castillo, D. Lugones, D. Franco, E. Luque and Martin Collier. *Predictive and distributed routing balancing, an application-aware approach*, In Proceedings of the 2013 International Conference on Computational Science, ICCS '13, "To be published", Barcelona, SPAIN, 2013. Elsevier.** [11]
This paper proposes an application-aware predictive routing. The predictive routing approach is adapted to work with traces from real applications, along with synthetic traffic. The routing unit is able to dynamic identify communication patterns that caused congestion in the network. The policy learns from past situations and then re applies the set of multipath solution in order to tackle congestion.

1.6 Thesis Outline

According to the objectives and the research method described above, the outline of the remaining chapters of the thesis is as follows.

Chapter 2: Thesis Background.

This chapter introduces some basic concepts about interconnection networks. Then, exposes concepts about parallel applications and its characteristics and presents the impact of interconnection networks over parallel applications

Chapter 3: Predictive and Distributed Routing Balancing.

In this chapter, we present the proposed predictive and distributed routing mechanism *PR-DRB*. Here, we discuss in detail the rationale behind the predictive approach.

Chapter 4: Evaluation.

This chapter describes the test scenarios and provides the explanation of experimental results for our proposal.

Chapter 5: Conclusions.

Concludes the thesis and presents the further work and open lines for the predictive and application-aware technique.

The list of references and one appendix complete the document of this thesis. The Appendix A includes the algorithms used in the methodology discussed in Chapter 3 and complementary results of the evaluation presented in Chapter 4.

Chapter 2

Thesis Background

In this chapter, we discuss some basic concepts about interconnection networks and parallel scientific applications in order to frame the thesis.

First, we introduce a general description of interconnection networks in section 2.1. Then, in section 2.2, we present some concepts extracted from scientific parallel applications and focus on specific aspects such as the impact of the network into the applications (subsection 2.2.1), the communications patterns and the requirements of real scientific parallel applications (subsection 2.2.2), the bursty traffic (subsection 2.2.3) and the repetitiveness that exists in communication patterns (subsection 2.2.5). Some final remarks are given at section 2.3.

2.1 Interconnection Networks

The interconnection network can be thought as programmable physical system. This system comprises links and switches that interact with each other to perform the communications needed by numerous components of the computing system. [21]. Interconnection networks can be found in computer systems and in communication systems. In the former, the interconnection network is in charge of the task of connecting processors to memories and input/output (I/O) devices to controllers; in communication switches, they connect input ports to output ports [62]. Many different systems are currently based on communicating data through interconnections networks, ranging from very specific hardware equipment 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 for communications

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.

Nowadays, interconnection networks are fundamental for systems where efficient communication technologies have a direct impact in the overall performance of the system. These system can be categorized as *computer clusters*, and *massively parallel processing (MPP)* systems [58] [59]. Computing cluster with the 82.2% in the TOP500 list of November 2012 [58] is the largest group. This represents platforms for implementing parallel applications, even though they have been subsequently used in other application fields such as computation-intensive purposes, computational simulations, storage networks and internet services. The other group comprises the MPP systems, such as the *IBM BlueGene/P* supercomputer [25], and represents 17.8% of the TOP500 systems. These systems were the first using high-speed interconnection networks.

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.

We will now give an introduction to important features of interconnection networks design related to the thesis, including topologies (subsection 2.1.1), switching techniques (subsection 2.1.2), flow control (subsection 2.1.3) and routing (subsection 2.1.4).

2.1.1 Topology

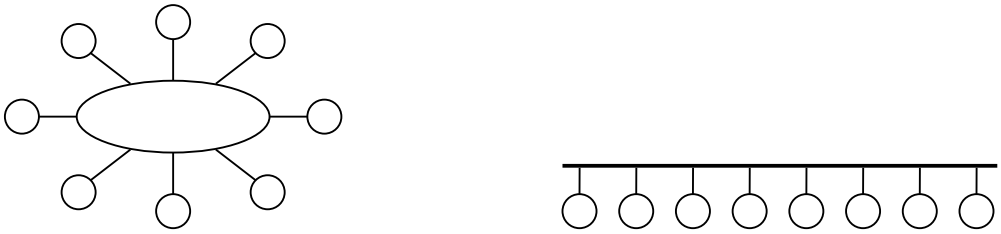
Topological disposition of nodes in the interconnection network are known as the topology of the network. This represent the path where messages must traverse during communication. Topology is modeled as a graph, where vertex represent nodes and edges represent links between a pair of vertex. Network topologies can be classified in four main groups, as defined by Duato et al. [17, Ch. 1] and W. Dally [62, Ch. 3]: *shared-medium networks*,

direct networks, *indirect networks* and *hybrid networks*.

In shared medium, the physical medium is shared by all communicating devices, as seen in Fig. 2.1 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. Fig. 2.2 depicts some direct networks. It 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 supercomputers. 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.

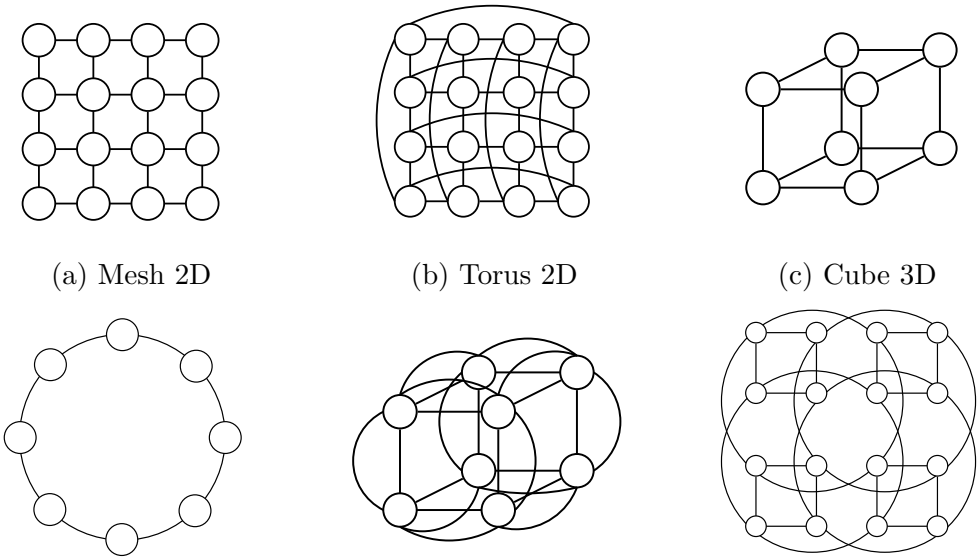
Instead of directly connecting communication devices to each other interconnection can be done by means of one or more switches. Several switches are connected to each other through point to point links. This concept is known as indirect networks, as shown in Fig. 2.3. Crossbar topology of size NxM is shown in Fig. 2.3a. Crossbar network allow communications between any pair of processors or memory unit simultaneously. Fig. 2.3b 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. The 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. Fig. 2.3c shows another indirect network topology, known as fat-tree. In [49] they explain the fat-tree concept. According to them, the fat-tree is represented in a tree structure where links weights gets thicker as they go toward the root node. Fat-trees have been adopted by many research as well as commercial machines such as Infiniband [26] and Myricom [43]. The connectivity degree (the arity) of internal switches of the fat-tree increases as it goes closer to the root. Because of this, its physical implementation is unfeasible. Some alternatives have been proposed in order to avoid this problem. One approach, mentioned in the paper, is to trade connectivity with simplicity. This could be accomplished by building blocks with fixed-arity. One of the possible variation of fat-tree is the *k-ary n-tree*. A *k-ary n-tree* has k^n leaf nodes and n levels of k^{n-1} switches. Each switch has 2^k links. An example of a *4-ary 2-tree* is shown

in 2.3d.



(a) Local Area Network (b) Bus-based Network

Figure 2.1: Examples of shared-medium network topologies.



(a) Mesh 2D (b) Torus 2D (c) Cube 3D

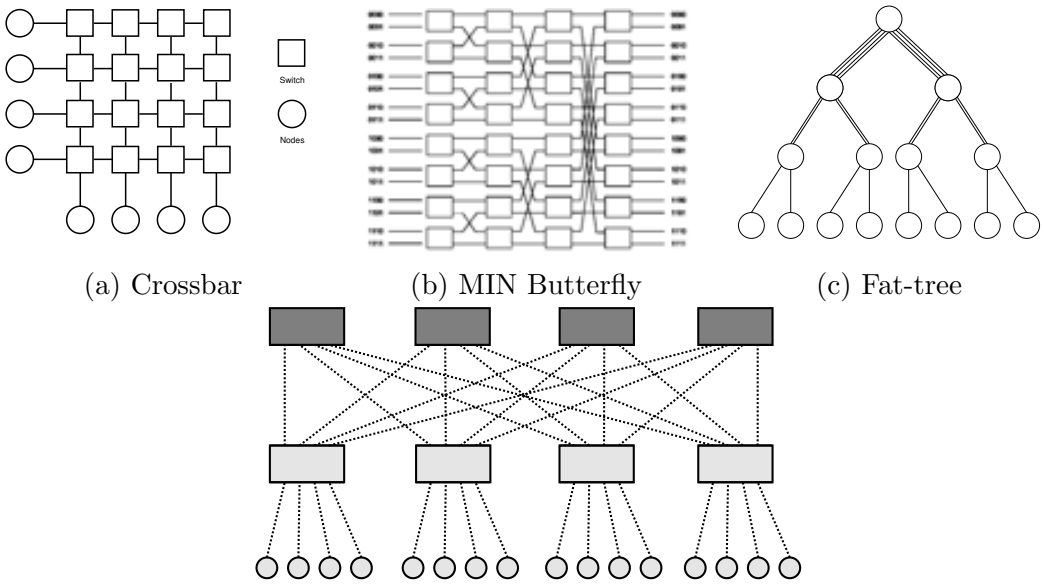
(d) Ring (e) Torus 3D (f) Cube 4D

Figure 2.2: Examples of direct network topologies (orthogonal).

2.1.2 Switching Layer

The switching technique is in charge of moving packets within each switch along source-destination paths. In fact, switching is the mechanism that transfers data from an input channel into an output channel [44]. The most widely accepted techniques in the literature are *Store-And-Forward*, *Virtual Cut-Through* [34], and *Wormhole* [12].

Store-And-Forward (SAF). 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



(a) Crossbar (b) MIN Butterfly (c) Fat-tree

(d) A 4-ary 2-tree

Figure 2.3: Examples of indirect network topologies.

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 (VCT). 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. In *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. Consequently, the small buffer requirement and packet pipelining enable the construction of routers that are small, compact and fast [17, Ch. 2].

2.1.3 Flow Control

Currently communication devices have input and output buffers to store sent and received 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 [17, Ch. 2], [62, Ch. 13]. The most commonly used schemes are *Credit-based* and *On/Off*.

Credit-based Flow Control. This mechanism assigns credits to each router in the network, in order to allow message transfer to other neighbors routers. Each transmission decrements the credit count and once the router runs out of credit, then it stops transmitting.

On/Off Flow Control. A 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.

2.1.4 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.

Although the number of existing options is quite large, routing algorithms can be classified into four main groups using the taxonomy proposed by Duato et al. [17, Ch. 4]. The resulting classification is based on *number of destinations*, *routing decisions*, *implementation* and *adaptivity*. The classification is summarized in Figs. 2.4 and 2.5.

Number of destinations. Routing algorithms where packets have a single destination are known as *unicast* routing algorithms, while those having multiple destinations are called *multicast* routing algorithms.

Routing decisions. This criterion is based on determining *who* and *where* are taken the routing decisions. Decisions can be either taken by a centralized controller (*centralized* routing), or in a *non-centralized* manner. In the latter case, decisions can be taken at the source node prior to packet injection (*source-based* routing) or in a distributed manner while packets traverse the network (*distributed* routing). *Multiphase* routing is an hybrid scheme where the source node selects some destination nodes and the path between them is established based on distributed approaches.

Adaptability. This is probably the most important classification criterion in the context of this thesis. Adaptability refers to how routing algorithms select a path between the set of possible paths for each source-destination pair. *Deterministic* routing algorithms always chose the same path between a source-destination pair, even if there are multiple possible paths. *Oblivious* routing algorithms choose a route without considering any information about the network s present state (note that oblivious routing includes deterministic routing). Finally, *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.

Implementation. Basically, routing algorithms can be based on *routing tables* storing paths information; or also on *routing functions* (logic or arithmetic) determining the path for each source-destination pair. The routing algorithm can be either deterministic or adaptive in both cases.

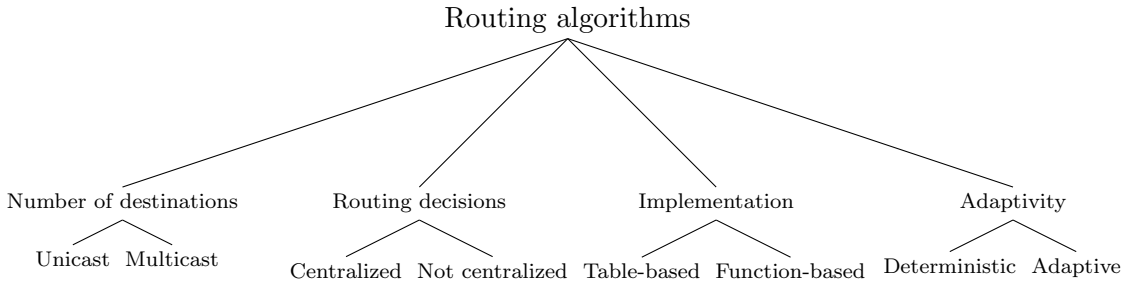


Figure 2.4: Classification of routing algorithms.

2.1.5 Routing in *k-ary n-tree* networks

Because the *k-ary n-tree* (Fig. 2.3d) is one of the topology we use in this work, we will explain some basics about its routing. Minimal adapting routing between a pair of nodes can be accomplished by sending the packet to one of the nearest common roots or *nearest*

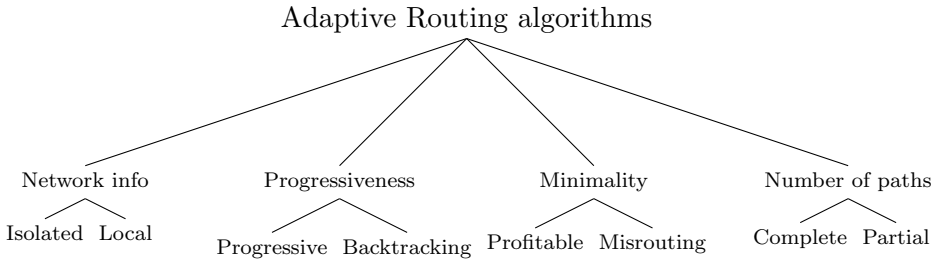


Figure 2.5: Classification of adaptive routing algorithms.

common ancestors (NCA) of source and destination, and from there to the destination. Each packet traverses through two phases, and *ascending adaptive* phase to get to one of the NCA, followed by a *descending* deterministic phase [22] [49].

2.2 Parallel Applications

2.2.1 Interconnection network impact over applications

Applications that are executed in an interconnection network may suffer from hotspot or congestion situations that 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 strategy to avoid long waiting times during communication time, could be the overlapping of computation and communications. In order to efficiently overlap communications and computations, 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 impact of latency 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.

The distribution of the Application s task 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.

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 towards an unpredictable latency s behavior .

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 the communication time, which is a delicate issue due to latency and congestion situations as mentioned earlier.

2.2.2 Parallel Applications Characteristics

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 s data distribution, computation, synchronization, and many other depending on particular application s 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 program s basic 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, present some features such as strong periodic behavior and alternation between different portions of code [55]. Based on the repetitiveness exposed above, relevant parts of applications can be extracted and then create a signature to identify them properly during execution [64].

Communication Requirements of Real Applications

Many of the routing algorithms in the literature base its results on synthetic traffic analysis, some examples could be [18] [15] [57] [19] just to mention some of them. In their work they try to probe some specific new functionality or feature of the routing algorithm or the architecture. On the other hand, and despite of the large number of works studying new features of routing algorithms, only a few works (mostly in the last decade) started to use real applications patterns for HPC traffic studies.

The work presented in Kamil et al. [30] performs and in-depth study of the communication requirements across a broad spectrum of important scientific applications. The work tries to characterize the parallelism and communication requirements of such a diverse set of applications. The main goal encompasses a guide for architectural choices for the design and implementation of interconnects for future HPC systems. In Kamil et al. [31], they extend their last work and propose a hybrid architecture that uses circuit switches to dynamically reconfigure lower-degree interconnect to suit the topological requirements of a given scientific application. Barker et al. [2] present a two-network interconnect: An optical circuit switching (OCS) network handling for bulk data transfers using optical switches; and a secondary lower-bandwidth electronic packet switching (EPS) network. Here, Kamil et al. [31] and Barker et al. [2] both propose changes to the core architectural design of network components. Their major contribution, besides its new design proposal, is the extensive evaluation of their work with real scientific applications and its communication requirements.

Riesen [52] studies message-passing applications patterns. In his work he proposes a novel framework in order to extract the information from running applications, instead of getting from the source code. From the framework expects to get significant knowledge (such as message density distribution, data density, number and type of collectives and message size distribution). Basically he collect significant data from the application in order to be used later in the design of networking hardware and software.

Oliker et al. [46], Vetter and Yoo [60], Bhatelé et al. [4] test a set of representative applications on candidate petascale platforms. These works are based on the premises of studying the behavior of these applications in high-end computing environments. Although they do not propose a specific method to improve performance, they do present a thorough result set of application performance on these architectures. These result could be used to analyze the application characteristic and could help the design of novel hardware and software for HPC environments. Later, Kamil et al. [32] extend their work by conducting a broader and updated study of high-end application communication requirements. Using these characteristics, the propose a *fit-tree* approach for designing network infrastructure

that is tailored to application requirements. They suggest a cost-effective solution by introducing a new network architecture. While they prove that their solution is effective, they basically propose a completely new architecture and infrastructure. This could be good for new HPC systems, but impractical for existing installations.

Gómez et al. [23] propose a deterministic routing algorithm for fat-trees, comparing it with adaptive routing in several workloads. Besides using synthetic traffic, they propose to validate its methodology against some I/O specific traffic patterns. In Vishnu et al. [61] they provide an *MPI functionality* which provides hot-spot avoidance for different communications, without a priori knowledge of the pattern. Even though they do present one of the first *dynamic* approach, they require a new layer between the MPI protocol layer and the Infiniband Layer in order to operate. Desai et al. [14] study the behavior of real and synthetic supercomputer workloads to understand the impact of the network s nonblocking capability on overall performance. Kandalla et al. [33] present a methodology to detect the topology of large *Infiniband* clusters and then leverage this information to create topology-aware algorithm for collective operations. They also propose a model to analyze the communication of collective communications on large scale supercomputing systems. They focus only on collective communication primitives, while many parallel scientific applications rely mostly on point-to-point primitives (send-receive). A framework for application-aware routing that guarantee deadlock freedom under one or more virtual channel is proposed in Kinsy et al. [35]. The framework uses an offline approach in order to minimize a set of constraints such as latency, number of flows per link, bandwidth or any combination thereof.

In Rodriguez et al. [53] a family of oblivious routing scheme for *Fat Trees* and their slimmed version is presented. They propose a new generalized family of algorithms that provides a better oblivious solution for the routing in this kind of networks. They extract some information from the application, such as the connectivity matrix (source-destination pairs) for each communication phase. Then they feed their routing algorithm with the information extracted previously (the connectivity matrix, the topology and the mapping). All this process is performed offline, and then fed into their simulator. In Rodriguez et al. [54] a new static source routing algorithms for High Performance Computing is proposed. They have evaluated their proposals against realistic traffic generated by HPC applications such as WRF (Weather Report Forecast), FEM (Finite Element Method) and the NAS Parallel Benchmarks. This work also proposes an offline contention metric for the network and the adapter, as well as a new pattern-aware routing heuristic to get optimized routes according to the offline contention metrics. Johnson et al. [28] show that for a particular application communication pattern, the performance of an Infiniband

network is greatly affected by contention within the network. They propose a methodology to optimize the network performance from an application centric point of view. They analyze the application and extract its requirements. Then the fabric of the network is customized accordingly. They propose this application approach instead of optimizing particular network features such as bandwidth, latency or collectives communications.

2.2.3 Bursty Traffic

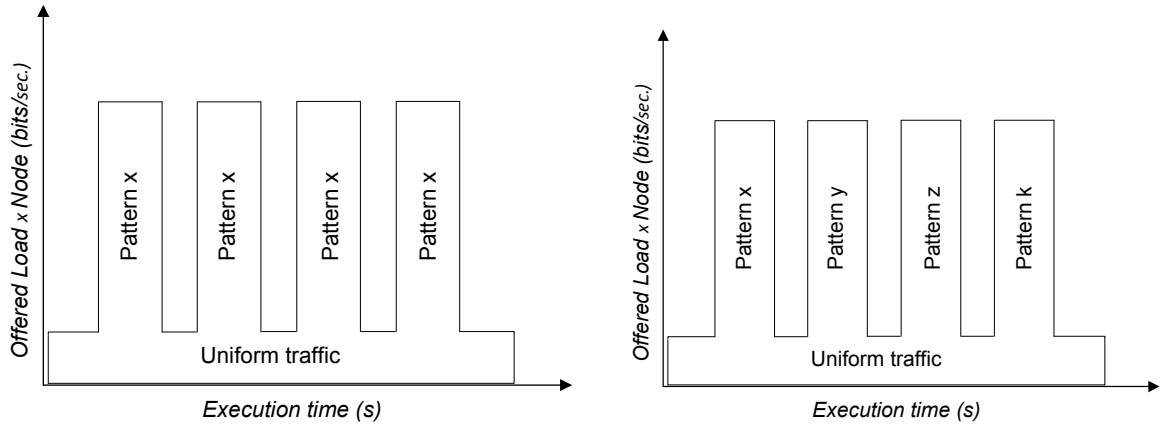
Repetitive communication patterns alternated by computation phases can be extracted by identifying relevant stages of applications. In high performance computing (HPC), we could find many traffic load patterns according to the paradigm or model of the applications. DRB [19] for example works well under uniform traffic load distributions, which represent applications where heavy communications are done at the beginning and remains mostly the same until execution finishes.

There are, though, other types of load distribution found in HPC, as the bursty traffic. Bursty traffic is 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 Fig. 2.6. This kind of traffic load represents applications where computation stage is well differentiated from communications stage. Fig. 2.6 depicts two traffic load patterns examples. The first one, shown in 2.6a, represents an application with an uniform distribution for low traffic load, and one distribution for high traffic load, such as the bitreversal for example. 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 computation again. This process repeats until execution ends, and communications are always performed among the same set of nodes.

The second figure, depicted in Fig. 2.6b, 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 Fig. 2.6 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 [54], and it is governed by application criteria such as process mapping.

Using the strategy of application patterns extraction within the cyclic behavior of traffic in HPC could lead to a powerful way to obtain better policies of routing and congestion s control, based on historical data about application behavior and application traffic.



(a) Simple pattern. (b) Variable pattern.

Figure 2.6: Bursty traffic.

2.2.4 Communication Primitives

The breakdown of MPI calls is shown in table 2.1. We take into consideration only the calls used by the applications used in this thesis. Also, we only consider communications and synchronization calls. We can see that the applications only use a subset of the entire communication library available. The use of the primitives depends on each particular application and how it is coded. The applications selected mostly use non collective communication primitives. The only applications analyzed that make heavier use of one collective communication primitive (MPI_Allreduce) are the POP with nearly 30%, followed by the Lammps Molecular Dynamic application with 10% of the total. The rest of applications rely on point-to-point communications. From this non collective subset, the POP is the one which uses non-blocking primitives. The rest of applications use mostly blocking MPI communication models for their codes.

2.2.5 Parallel Application Phases - Repetitiveness

Studies of parallel applications in HPC reveal they have repetitive behavior, based on computing and communication phases [64]. Programs in general tend to be written in a modular fashion, and have a very strong periodic behavior [55]. An example is given on Fig. 2.7, where the repetitive behavior of the NAMD [50] application is shown.

Function	POP	Lammps	NAS LU	NAS MG S	NAS MG A	NAS MG B	Sweep3D
MPI_ISend	34.9%	0%	0%	0%	0%	0%	0%
MPI_Waitall	34.9%	0%	0%	0%	0%	0%	0%
MPI_RecvS	0%	0.04%	0%	0%	0%	49.09%	0%
MPI_Send	0%	43.6%	49.8%	44.42%	46.49%	0%	49.99%
MPI_SendR	0%	0.04%	0%	0%	0%	0%	0%
MPI_Wait	0%	43.6%	0.31%	44.42%	46.49%	49.09%	0%
MPI_Irecv	0%	0%	0.314%	0%	0%	0%	0%
MPI_Recv	0%	0%	49.52%	0%	0%	0%	49.99%
MPI_Reduce	0%	0%	0%	0.10%	0.06%	0.17%	0%
MPI_Allreduce	29.3%	10.75%	0.003%	9.6%	6.04%	1.56%	0.007%
MPI_Barrier	0.3%	0.008%	0.0007%	0.65%	0.41%	0.10%	0.0007%
MPI_Bcast	0.3%	1.88%	0.0035%	0.76%	0.48%	0.12%	0.0009%

Table 2.1: Breakdown of MPI Communication Calls.

Here, some communications take longer than others (green bars). This imbalance in the communications causes that some processes become idle (red bars). These inefficiencies are repeated in each application phase. Fig. 2.8 shows an extract from one communication pattern of MG. Here if communications between nodes 0-1 and 0-3 could be reduced (green and orange arrows), then also the time for the ***MPI_Wait*** state will decrease (magenta bars). This will allow the next primitives for nodes 1 ***MPI_Send 4*** and node 3 ***MPI_Send 12*** to execute earlier. As this procedure is part of a significant phase of the MG application, a reduction in this communication will be applied every time this patterns re-appears. This repetitive behavior represented by fundamental phases of the entire application can lead to situations where specific traffic patterns reappear. A phase of an application includes one or more communicating patterns, represented by a set of source/destination pairs as shown in Fig. 2.9. Representative phases from parallel applications can be extracted by using the PAS2P performance prediction tool [64]. In table 2.2 we can see a summary of parallel applications, their representative phases and weights (how many times it is repeated during execution).

2.2.6 Matrix of Communications

We have also analyzed the the topological connectivity of applications. The idea here is to show the communication pattern used by the application as well as the volume of communication. This information would be valuable to identify the applications which would be optimal to use with our proposal. We have also identified the pattern of the (repetitive) phases that were extracted previously (shown in the table 2.2). In the fig. 2.10a

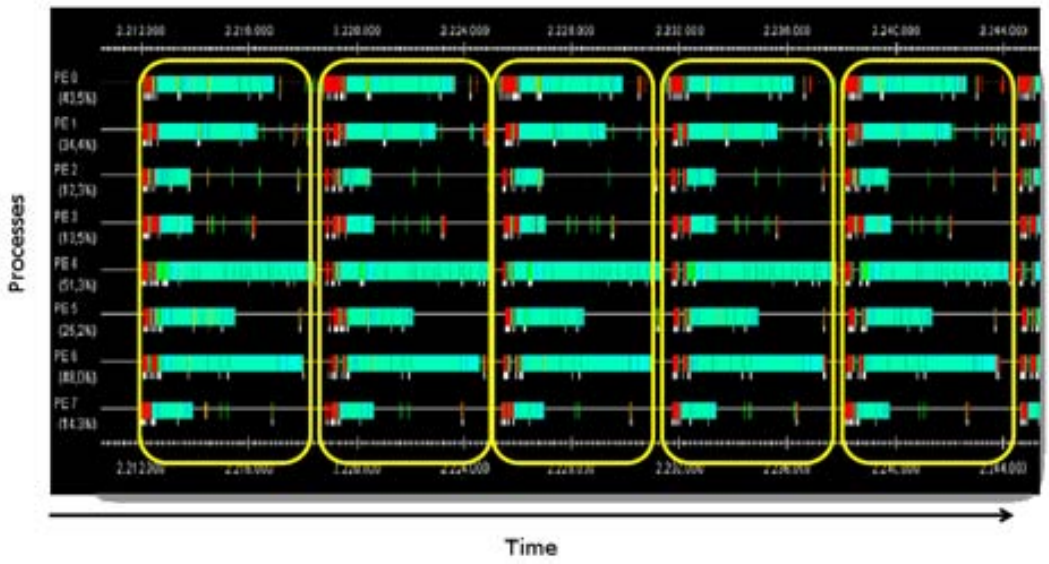


Figure 2.7: Repetitiveness in parallel applications.

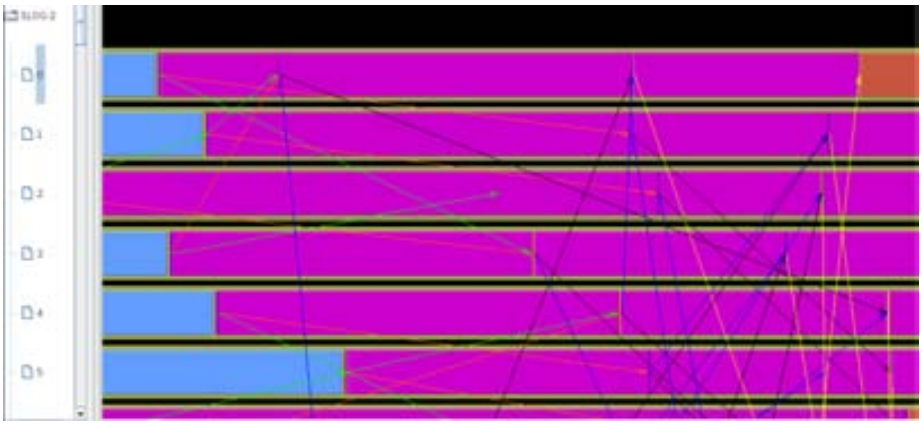


Figure 2.8: NAS MG pattern (6 of 64 tasks shown)

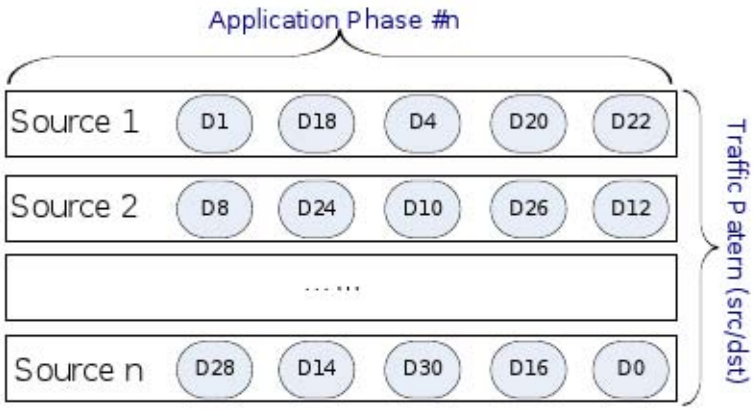


Figure 2.9: Traffic pattern in one application phase

Parallel Application	Total Phases	Relevant Phases	Weight (# of phases' repetitions)
Lammps Comb (64 nodes)[51] [37]	49	2	1698
Lammps Chain (256 nodes) [51] [37]	52	19	1802
NAS FT Class A [1]	6	5	9
NAS FT Class B	6	5	22
NAS MG Class S	21	12	164
NAS MG Class A	25	11	185
NAS MG Class B	27	3	424
SMG2000 Semicoarsening Multigrid Solver [56]	10	4	1200
SWEEP3D: 3D Discrete Ordinates Neutron Transport [36]	80	5	46000
Parallel Ocean Program (POP) [29]	140	120	38158

Table 2.2: Parallel applications phases

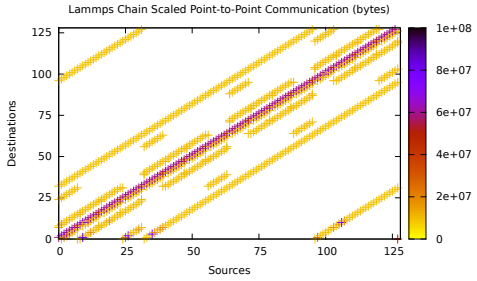
we can see the communication matrix for the Lammps molecular dynamics application. It is represented for the Chain problem. In the color box at the right side we can see the volume of communication reference (in bytes). Here, the diagonal represents the largest volume of communication. Chain communicates with its neighbors nodes and also has communication with other nodes located further away. The average TDC (connectivity) is 7 per node for this program. This means that each node communicate with 7 other nodes. This TDC is independent of number of nodes. We can see this because it remains the same for 25 or 100 execution nodes, on average. If we watch the fig. 2.10b, we can also determine that for 256 nodes the behavior is similar. The difference here lays in the volume of communication, which for the 256 example is a bit higher. Despite the TDC is only 7 for lammps, we can see in fig. 2.10c that the matrix of communication from one of the representative phases of the application involves communications between several nodes. Recall from table 2.2 that Lammps Chain has 19 relevant phases. In this case, this phase is the number 43 and the pattern in the figure is repeated 69 times throughout execution. We can observe that the phase has similar behavior compared to the whole application. The routing algorithms must deal with this communication pattern every time it appears in the network. If the routing algorithm does not have any information about the communication pattern, or the cause and the solution given, it must perform the same operations over and over in order to control latency values under this repetitive patterns.

In the case of Lammps Comb, in fig. 2.11, we can see similar results with the diagram for the entire application. Regarding the application phase shown in fig. 2.11b, we can see that the communication is mostly performed around the diagonal band. Under this scenario no gain can be expected in network communication, due that communication is performed almost locally within source routers. But because this application has two relevant phases (see table 2.2), we have to analyze both of them in order to decide if this application would be suitable to network optimization. The phase #2 is not shown here, because it is composed solely by collective communications (MPI_Allreduce). This collective communication would produce heavy traffic into the network. Also, this collective phase has a weight of more than 800 (repetitions). This implies that this communication pattern would inject the same amount of load into the network every time. Because of this, it should be considered to be used with our proposal.

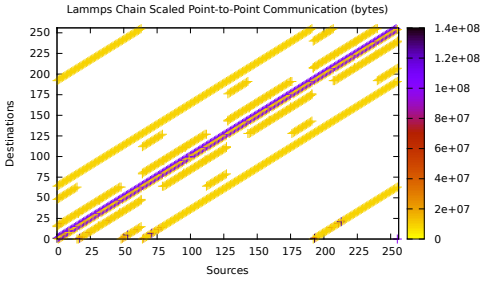
The connectivity matrix for Sweep3D is shown in fig. 2.12. We can see that the communications are performed around the diagonal, mostly between neighbors. The TDC for Sweep3D is 4 on average. Although this applications has a high rate of repetitive phases(5 phases which are repeated 46000 times; see table 2.2), none of this communications

are relevant to the routing. This happens because most of the communications are performed among neighbor nodes and the network can handle all the communications without congestion. Therefore, this application is not suitable to be optimized based on its communications characteristics.

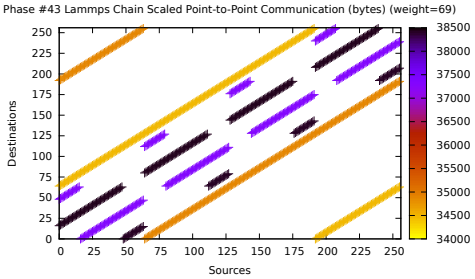
In fig. 2.13 we see the communication matrix extracted from the POP application for 64 nodes. The communication topology of the entire application execution is shown in fig. 2.13a. We can see the communication among close nodes represented by the diagonal bands. Also, some scattered communications exists. They represent the communications between remote nodes in the network. The maximum TDC here is of 11. We can see in fig. 2.13b one relevant phase of this applications. Here, the volume of communication is not as large as the other examples. On the other hand, we can mention that this particular phase even though having low volume of communications, has a high degree of repetitiveness over time, 5050 out of 38158 according to the information extracted in table 2.2. Other phases for this application behaves similarly. For this application the analysis and study of its communications characteristics would result in benefits at communication level, and in some extent to application level as well. This benefits could be achieved because by reducing the communication times of repetitive communications patterns, we could make that blocked packets (like those of MPI_wait, Barrier, All_reduce, etc) waiting for notifications move faster.



(a) Lammps Chain Scaled (128x4x4x4)

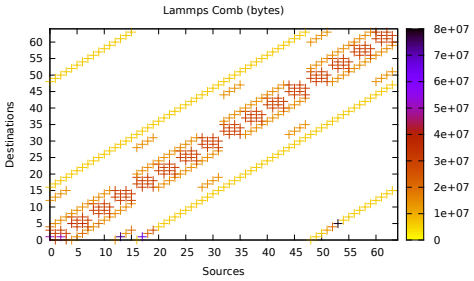


(b) Lammps Chain Scaled (256x4x4x16)

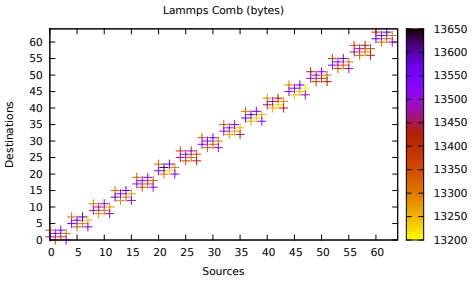


(c) Phase 43 with Weight 69 - Lammps Chain Scaled (256x4x4x16)

Figure 2.10: Lammps Chain Benchmarks Matrix of Communications.



(a) Lammps Comb fixed size



(b) Phase 30 with Weight 854

Figure 2.11: Lammps Comb Benchmarks Matrix of Communications.

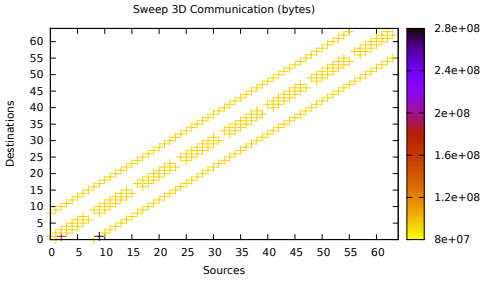
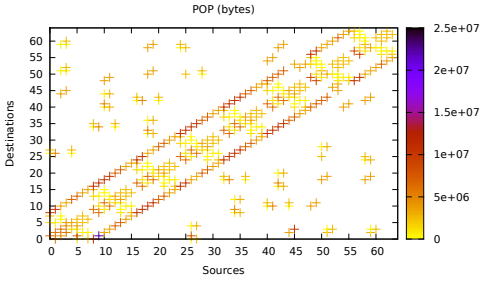
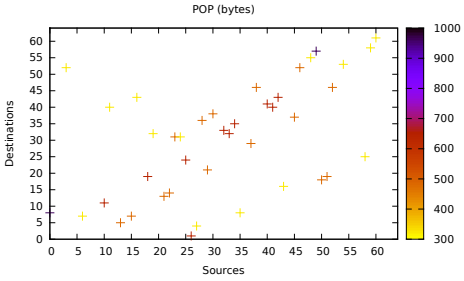


Figure 2.12: Topological Connectivity of Sweep3D



(a) Topological Connectivity of POP with 50 steps (Application Load)



(b) Phase 113 with Weight 5050

Figure 2.13: POP 64 nodes.

2.3 Final Considerations

After analyzing these works we can arrive at some common conclusions. The authors from the analyzed papers conclude that the communication patterns exhibited by some real scientific applications generally under-utilize the bisection bandwidth of fully-connected networks. They also identify that applications patterns are a crucial factor when analyzing performance and the benefits in overall performance cannot not be realized by hardware alone. All the previous work basically rely on the extraction of information from applications offline. After getting all the information such as: the communication pattern, the connectivity, the topology, the perfect mapping among other; then they feed that meta-information to their simulator (in most of the cases) or real implementations (in others). We also saw that besides the patterns, their behavior could also be extracted and analyzed. We can say then that all the application information extracted could be used to guide or help in the routing and/or congestion control mechanisms by means of static or dynamic approaches. The evaluation of the applications explained here are described in the chapter 4. Specifically the NAS Parallel Benchmark in section 4.8.2, Lammps Comb in section 4.8.3 and POP in section 4.8.4.

Chapter 3

Predictive and Distributed Routing Balancing

In this chapter we propose a routing algorithm based on the study of communication latency and repetitive communication patterns in HPC applications. The proposed method is called *Predictive and Distributed Routing Balancing*(PR-DRB). This method is also based on the concept of path redundancy available in many network topologies, by using a multipath routing approach.

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

This method shares theoretical basis with two previous theses; it is based on the method developed by Franco et al. [19] [20], [21], *Distributed Routing Balancing* (DRB); subsequently improved by Lugones et al. [41], [42].

Conceptually, the proposed predictive routing method is based on state information obtained from source and destination paths. Among the state information we can mention the latency value of the whole path, the latency at intermediate routers and the flows contending for a resource at every intermediate router. The proposed method follows the three standard phases proposed for reactive congestion control techniques. The first phase is in charge of the monitoring, detection and notification of high latency values that can lead to a congestion situation. Also in the first phase, the information about the communication pattern involved in congestion is analyzed. The second phase decides about the configuration of new alternative paths according to the congestion situation previously notified. The third phase uses the configured alternative paths in order to

effectively alleviate the congestion situation.

The rest of the chapter is organized as follows: The justification of the predictive idea is given in section 3.1. The functional aspects are explained in section 3.1.1. The predictive approach is analyzed in section 3.2 along with the monitoring, detection and notification phases (subsection 3.2.2), the configuration of alternative paths (subsection 3.2.3), the thresholds and the related zones used (subsection 3.2.4), the path selection procedures (subsection 3.2.6) and the contending flows notification and management (subsections 3.2.7 & 3.2.8). The architecture of network components is exposed in section 3.3. Additionally, some design alternatives are disclosed in section 3.4.

3.1 Justification

Based on previous examples of communication patterns repetitiveness, we can say that High Speed Interconnection Networks (HSIN) routing performance depends mostly on the communication pattern used and the mapping of nodes to processors. PR-DRB tries to improve communication imbalance situations by distributing communications in a better way. To improve communication performance, hence applications currently running in the network, a technique capable to fully dynamically combine adaptive algorithms and communication patterns is needed, so that routing and congestion control can perform as fast as possible and minimize overhead.

Initial Assumptions

For this thesis, we use the term *router* to indicate network nodes (devices) which can receive packets, determine their destination based on a defined routing algorithm, and then forward the packet through the corresponding output port [62, Ch. 2]. Consequently, the term *node* will be used to reference terminal and processing nodes.

3.1.1 Functional Aspects

In Fig. 3.1 we can see the proposed method behavior. During the first stage of the traffic, the curve for *DRB* and *PR-DRB* algorithms are practically the same. This is because *PR-DRB* is learning from the hotspot situation that is causing the congestion in the network. At the end of traffic stage 1, latency value is stable and the best solution found is saved at source node. Best solution is identified because the latency curve has reached its highest value and from now on it starts decreasing. This means a good balance of traffic have been found and all messages injected into the network are properly distributed.

When the bursty traffic stops, because the communication phase of the parallel application is over and now is probably doing some computation with the data transferred, the latency value decreases to a minimum. Under such situation the network can accept all the traffic currently ongoing. When the application starts its communication phase again, the latency value starts rising accordingly to traffic conditions, as we can see during the second stage of the traffic in Fig. 3.1. This scenario would be repeated as many times as the weight (# of repetitions of one phase) assuming some degree of repetitiveness in the application pattern.

The repetitions give the ideal situation to apply the solutions already found in previous stages. The proposed behavior is shown in stage 2 of 3.1 where both curves rise similarly until some specific point and then start 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 because good solutions have been applied earlier and even though the global latency is still rising, *PR-DRB* controls the sharpness of the rising procedures. Greater improvement can be seen when the bursty traffic stabilizes and remain constant. *PR-DRB* practically avoid the hot-spot situation that happens while the method is still in 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.

3.2 Predictive Approach

The predictive approach is based on the concept of repetitiveness of relevant stages in parallel applications. Each stage of one application may incur in the congestion control procedures explained here. At the monitoring phase, latency value of a message is logged throughout its journey through intermediate routers. In addition, the information related to the communication traffic pattern which caused congestion is also saved.

Based on the design alternatives, notification can be performed at intermediate router (*router based notification*) or destination nodes (*destination based notification*). In either case, the source node is notified about the network status via an *acknowledge* or *notification*

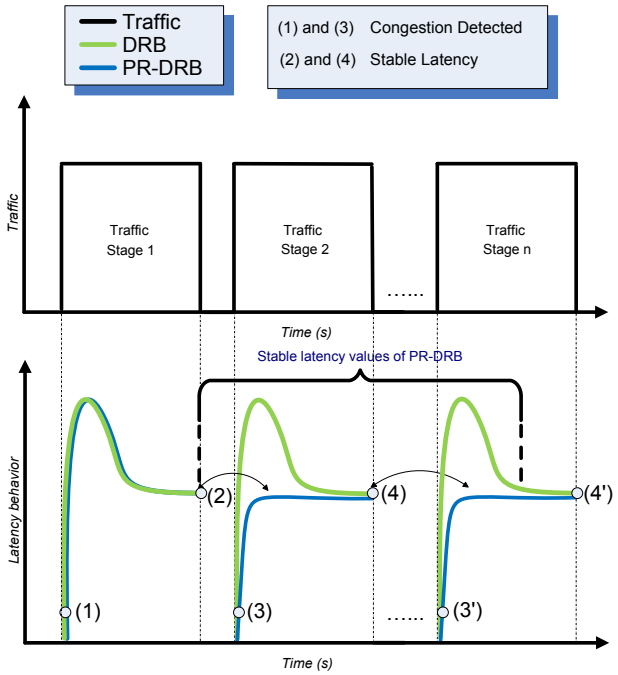


Figure 3.1: PR-DRB overview

message (ACK). This acknowledge includes the latency value registered throughout the path and the set of contending flows which caused congestion.

The set of contending flows are extracted from the router where a latency threshold has been surpassed. For *destination based notification*, only the information from one router can be managed at one time. On the other hand, for *router based notification*, every intermediate router could send a notification ACK to the sources involved in congestion. With the ACK message, the source node is then able to start the procedures to control the congestion, if any, in the network.

At the end, the congestion situation along with the solution applied is saved for future similar situations. The best solution saved may be further updated, if the method finds a better combination of paths which improves performance. Fig. 3.2 shows the general overview described earlier. Here, the *destination based* scheme is shown. The *router based* variation is shown in section 3.4.1.

PR-DRB basically tries to learn from the repetitive situations encountered during bursty traffic conditions, and overcome the limitation found in adaptive algorithms in general under this situation. Specifically, we focus on the improvement of the original DRB adaptive algorithm.

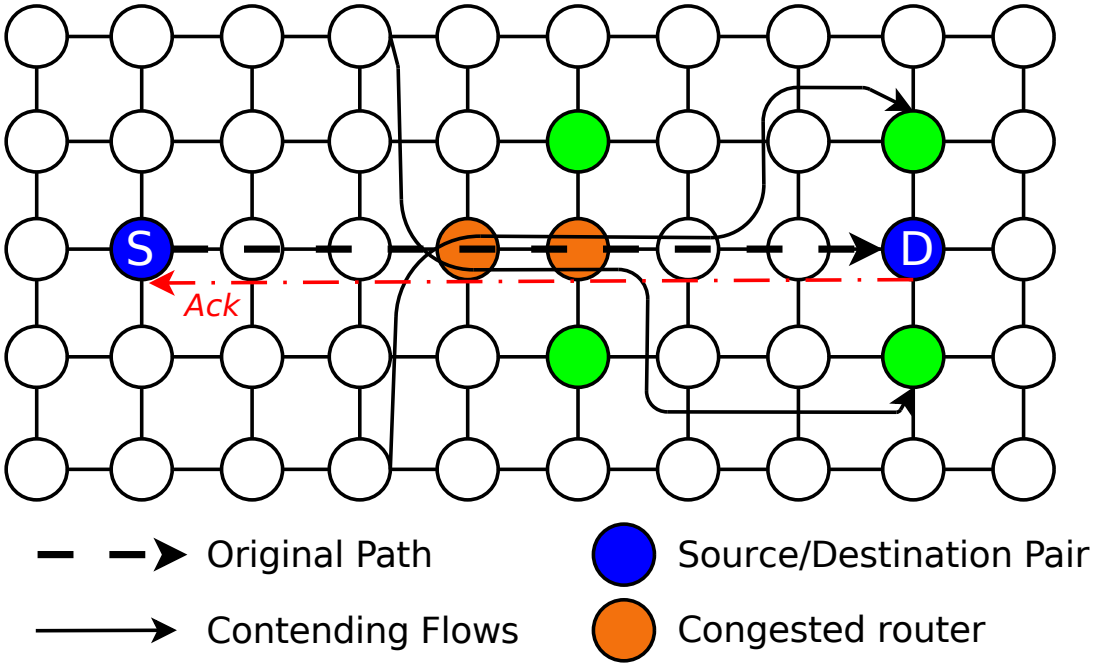


Figure 3.2: Latency detection and notification

3.2.1 PR-DRB Working Scheme

PR-DRB seeks better response time by using cached communication and alternative paths. In Fig. 3.3 we can see the standard Packet delivery Process. The diagram is composed of three main blocks: *Source node*, *Package Routing* and *Destination node*. The actions of PR-DRB are performed in one or more of these main blocks. The source and destination nodes performs the local operations.

The packet routing block represents the activities performed at intermediate routers along the path from source to destination. Each block is composed by several elements (stage boxes and decision elements) representing the flow of actions performed by routers, source and destination nodes. When a source node wants to send some data, depicted in the *Source node* block, a message is built and injected into the network.

Then, as seen in *Package Routing* block, a multi-header message is forwarded through intermediate routers. Once the message reaches destination endpoint, as seen in *Destination node* block, the packets are received and delivered to the end node for processing. Before the delivery process takes place the packets are re assembled, and eventually ordered, into a valid message.

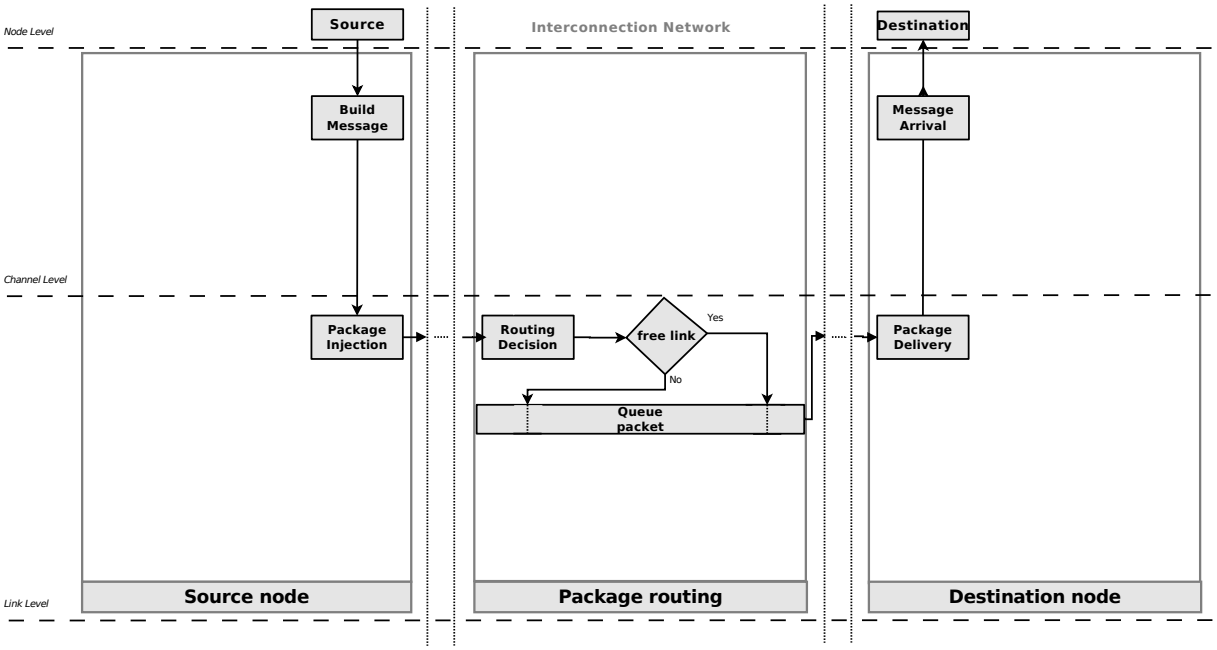


Figure 3.3: Standard packet delivery process

3.2.2 Monitoring, Detection and Notification

Monitoring includes the tasks of latency accumulation and contending flows identification, performed at intermediate routers. Every intermediate router executes the monitoring task when packets traverse its internal ports.

Once inside a router, the latency of the packet is monitored. The latency is the time the packet occupies internal buffers waiting to be accepted for delivery to the next intermediate router or the destination node. Besides latency value, routers analyze which 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.

Contention latency is determined by the time a packet 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 router resources at the same time. 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 then re apply the best solution already used. The pseudo code of the monitoring phase also controls that

the congestion situation is only recorded at routers which experiment a certain level of congestion. 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. The flow diagram for the *destination based* monitoring and notification approach, is shown in Fig. 3.4. The general algorithm is also included in the appendix, in section A.1.

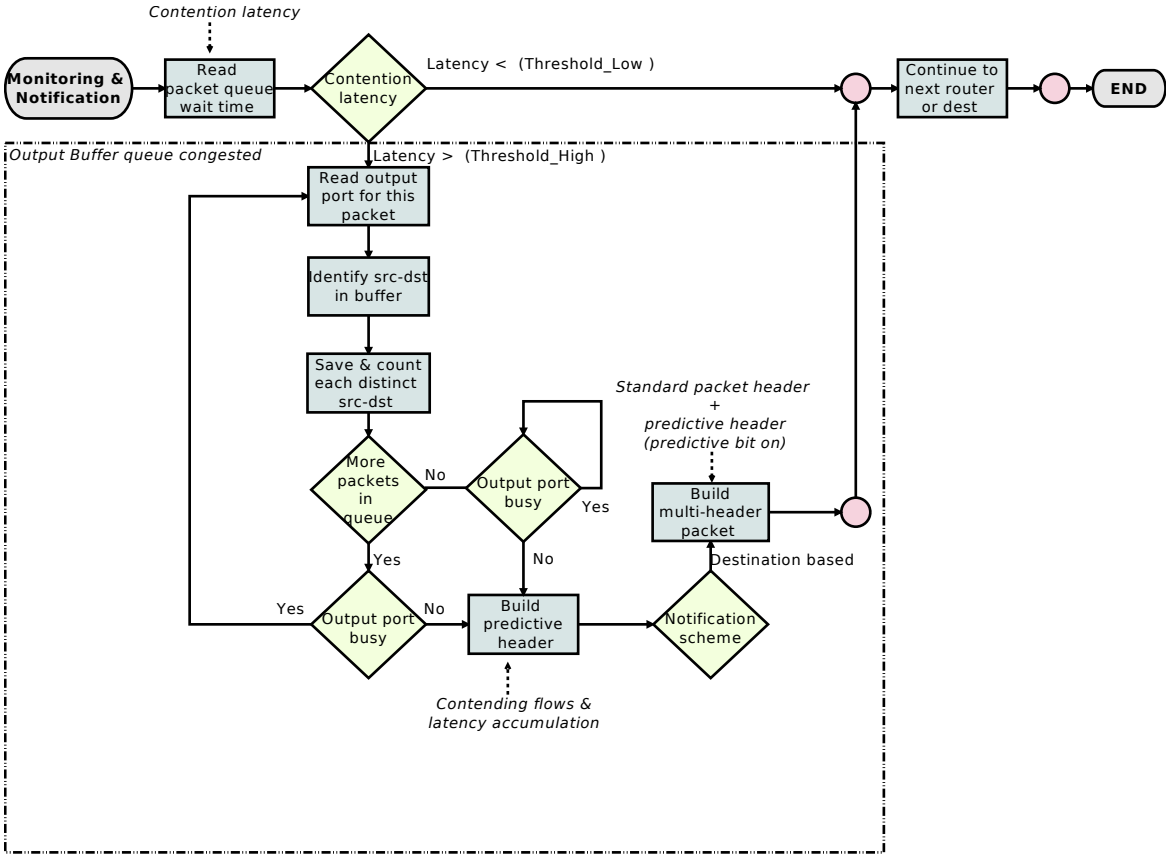


Figure 3.4: Monitoring and detection flow diagram (destination based).

As expressed before, the notification process could be *router based notification* as well, as detailed in section 3.4.1. Here we explain the *destination based notification*. Notification is initiated at destination nodes. Here, and Acknowledge (ACK) message with path information is created and sent back to the source. The diagram of the monitoring and notification procedures explained here are shown in Fig. 3.5. As shown in the figure, the delay suffered in switch buffers (queuing latency) is logged into the message. If queuing latency values exceeds a threshold while still at intermediate routers, contending flows patterns are also logged by PR-DRB. Monitoring tasks are performed while the packet is queued. Once the message reaches destination, as seen in *Destination node* block, Notification takes place. The Notification box depicts the task involved in this procedure.

Here, latency as well as contending communication patterns found are sent back to the source in an ACK. Not all contending flows are notified, but only those which contributes most to congestion. In order to identify the contribution of each flow to the congestion, we find the average of occupation of every unique source currently in the ACK packet. The notification functionality is shown in Fig. 3.5

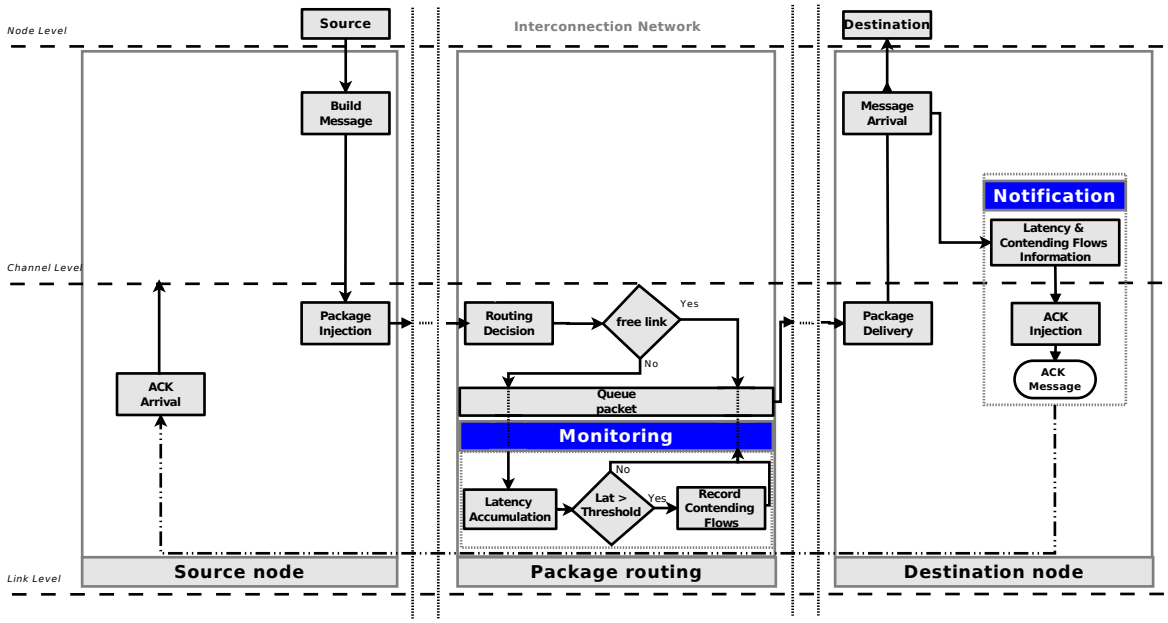


Figure 3.5: Monitoring, Detection and Notification procedures

3.2.3 Configuration of Alternative Paths

PR-DRB defines a *metapath* (MP) as a set of possible alternative paths between each source-destination pair. *Metapath Configuration* defines how to create alternative paths in order to expand single paths, and when to use them according to congestion level in the network. Using the information gathered during monitoring tasks, PR-DRB executes the dynamic metapath configuration. The objective of configuration is to determine, for each source-destination pair, the size of the metapath according to message latency between both nodes. This is achieved by the selection of intermediate nodes (IN) belonging to a path different from the original. An example of a set of intermediate nodes that can be used in alternative path is shown in Fig. 3.6. Each alternative path is created, schematically, using a three step path (*Multi-Step Path, MSP)* by selecting two intermediate nodes, neighbors from the source (IN1) and destination (IN2) node, respectively. PR-DRB builds the alternative paths around the original path as shown in 3.7.

The mathematical definition of a multistep path (MSP) is given in Eq. 3.1, where S

corresponds to the source nodes; D represents the final destination node; and intermediate routers are denoted as In_i. Furthermore, each P_j is a simple segment between any two nodes (terminal or intermediate); and the symbol $\bullet$ represents concatenation.

$$\begin{aligned} MSP &= \prod(S, In_1, In_2, \ldots, In_{i-1}, In_i, D) \\ &= P_1(S, In_1) \bullet P_2(In_1, In_2) \bullet \ \ldots \ \bullet P_{j-1}(In_{i-1}, In_i) \bullet P_j(In_i, D) \end{aligned} \quad (3.1)$$

We can see from the definition of Eq. 3.1 that a MSP may not be minimal, although each of their composing segments are based on minimal routing. The length of a MSP is defined in Eq. 3.2 as the sum of the length of each individual segment, $Length(P)$. In case of minimal static routing, $Length(P)$ equals the minimum number of links that must be crossed to go from the source node to the destination node. The latency of the MSP is determined according to the expression given in Eq. 3.3.

$$\begin{aligned} Length(MSP) &= Length(P_1(S, In_1)) + Length(P_2(In_1, In_2)) + \ldots + \\ &\quad Length(P_{j-1}(In_{i-1}, In_i)) + Length(P_j(In_i, D)) \end{aligned} \quad (3.2)$$

Each single path is traversed using original routing defined for the topology. The intermediate nodes register the latency values at every moment, together with topological characteristics of interconnection network. Thus, the Traffic load is fairly distributed over the network resources.

$$Latency(MSP) = Transmission\ time + \sum QueuingDelay(router) \quad (3.3)$$

As defined above, a **metapath** is composed by a set of alternative paths between a given source and destination pair. Congestion is controlled by increasing the available effective bandwidth between source and destination pair. This metapath expansion is performed adding more surrounding intermediate nodes neighbors. Intermediate nodes are selected according to their distance to the source and destination nodes (The intermediate nodes of 1-hop distance are considered first, then intermediate nodes of 2-hop distance, etc). This metapath configuration also depends on the predictive procedures proposed in this work (section 3.2.6). If we still do not know anything about the pattern, it means there is no solution saved for it. So, the metapath configuration process is executed. After

a while, the predictive module would have saved enough information about the pattern. If this is the case, then the set of alternative paths is extracted from the saved paths database , thus avoiding the metapath configuration process.

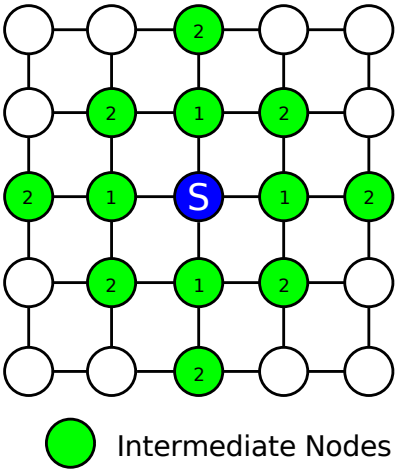


Figure 3.6: Intermediate nodes

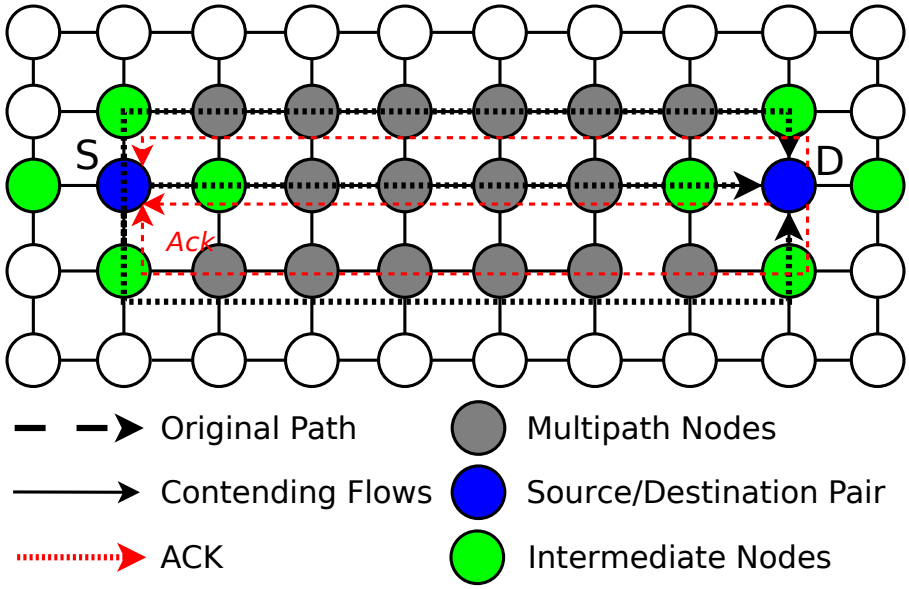


Figure 3.7: Example of a *metapath* (MP) composed by 3 *multistep paths* (MSP)

Recall from section 2.1.5, on *k-ary n-trees* networks minimal routing between two nodes is defined in two steps. On the first step, a packet is sent from the source node to the closest ancestor between the source and destination nodes. From this point, the packet is delivered towards the destination. This is the second step. As the packet moves through

the tree levels, more paths are available to perform adaptive routing. Each router now has the ability to select an output port in the upward direction. Once the packet arrives the common ancestor, then it is sent back in the backward direction towards its destination.

3.2.4 PR-DRB Thresholds

Congestion level in the network is used by the routing unit in order to perform the load distribution. To evaluate this congestion level then the *metapath latency* concept is introduced. This latency is defined as the inverse of the inverse aggregate latency sum of every path conforming the metapath. These latencies are produced while a message is traversing along one path from the metapath. The inverse of the latency is in fact one path s capacity. Thus, the inverse of the aggregate sum is the metapath s capacity. The metapath latency equation is shown in 3.4. The threshold is a set of predefined latency values. *Threshold_High* is the maximum limit for the latency before it enters into a *saturation zone*. *Threshold_Low* represents the point where the alternative paths starts to decrease. The interval between *Threshold_High* and *Threshold_Low* defines the latency value which is acceptable. This is the working zone of the network. While in the working zone, the metapah remains unchanged.

$$L(MP) \;=\; (\textstyle\sum_{i=1}^{S} \frac{1}{L(MSPi)})^{-1} \tag{3.4}$$

According to the latency value $L(MP)$, the source node would:

- Increase the number of MSPs if $(L(MP) > Threshold\_High)$.
- Maintain the same number of MSPs if $((Threshold\_High) > L(MP) > (Threshold\_Low))$.
- Decrease the number of MSPs if $(L(MP) < Threshold\_Low)$.

The metapath configuration process is outlined in Fig. 3.8. The pseudocode is also shown in the Appendix in Alg. A.2. The thresholds used, also define zones which are used to trigger the predictive behavior. This zones are explained in the next section: 3.2.5.

3.2.5 PR-DRB Zones

In order to identify the proper time when an action must be performed by PR-DRB, thresholds are used, as shown in Fig. 3.9. The thresholds corresponds to those used to

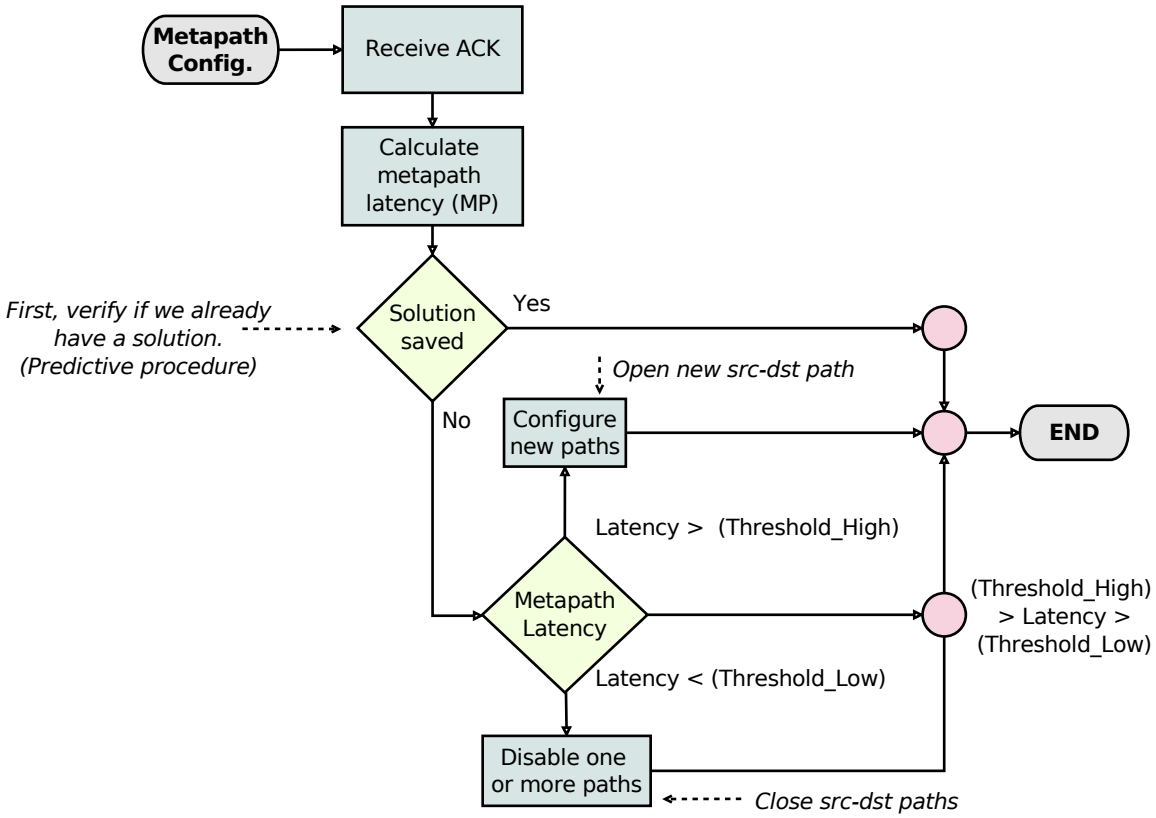


Figure 3.8: Metapath configuration flow diagram.

perform the metapath configuration (see section 3.2.4). By using this thresholds values, three well defined zones are created:

$$low(L), medium(M), high(H) \tag{3.5}$$

The low latency area represents low congestion situations. The medium latency area represents situations when congestion is raising, but the network can handle all communications properly. This is the normal working zone. High latency area represents high congestion in the network. While in low and medium latency zones, PR-DRB does not open new paths. When latency values identifies a transition between medium to higher zones, PR-DRB starts opening procedures. These includes the searching of saved solutions or new paths opening procedures if no solutions are saved so far. When congestion is controlled, latency returns to the middle zone indicating that good paths have been found. Here, path saving procedures are initiated. Finally, when latency values drops back to the lower zone, path closing procedures are started.

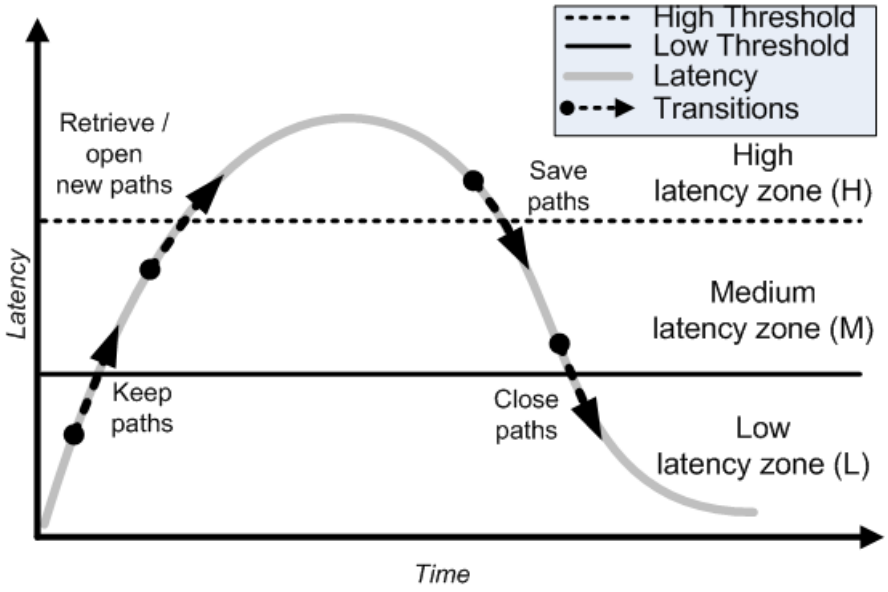


Figure 3.9: Thresholds and the zones defined

3.2.6 Predictive Procedures

Metapah procedures configure alternative paths to be used accordingly to latency value and thresholds, as shown in Fig. 3.12. If transition is from *M (medium) to H (high)* latency denotes congestion, then new alternative paths are needed. PR-DRB then looks for an already analyzed congestion situation. If this is the case, the set of alternative paths is retrieved from the database of saved solutions. If no solutions are found, then alternative paths opening procedures are initiated. If transition is from *H to M*, information about contending flows during congestion situations is updated into the saved paths database. If transition is from *M to L* latency values denotes low congestion, then alternative paths closing procedures are invoked. All these procedures feed the current paths database.

Later, when a message is ready to be injected into the network, PR-DRB performs the Path Selection. Here, PR-DRB selects which paths are going to be used from those available in the current paths database. Path selection and Configuration diagram is shown in Fig. 3.10. Paths having lower latency values are more frequently used, and they receive proportionally a greater number of messages. Given a source node with N alternative paths, let s be $L_c i(i : 1..N)$ the latency recorded by path Ci. The alternative path Cx will be selected in the following injection according to the probability (PDF):

$$p(Cx) \;=\; \frac{(1/L_C x)}{\sum_{i=1}^{N} 1/L_C i} \tag{3.6}$$

Furthermore, paths are selected according to their length. If paths are long in hops,

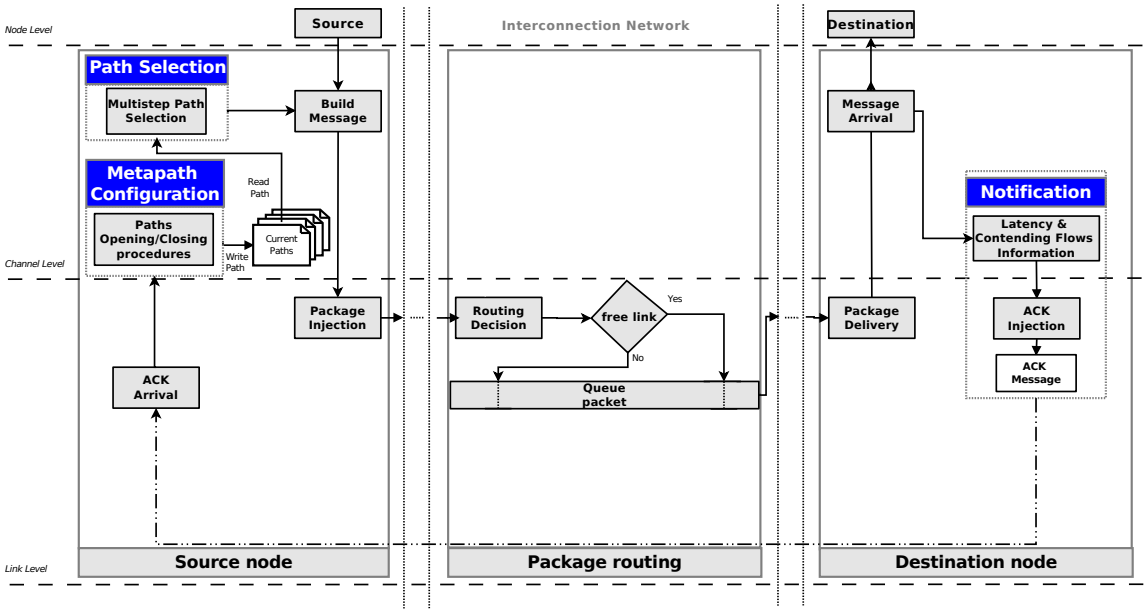


Figure 3.10: Path selection and metapath configuration diagram

message transmission time could be high enough and lead to performance degradation, so shortest paths are selected. Bandwidths are calculated as the inverse of latencies received in source node. This information is used to build a probability density function, and a path is selected according to the equation shown in 3.6. Finally, a multiple header packet (containing the intermediate nodes) is injected into the network. Path selection algorithm is resumed in Alg. A.3 in the Appendix. The path selection procedure is shown in Fig. 3.11.

As shown previously in Fig. 3.5, a packet is forwarded without any overhead when the output port is free. Otherwise, packet is queued and latency is simultaneously accumulated until the packet is ready to be forwarded again. PR-DRB is based on the DRB algorithm. DRB has been already proposed as a congestion control for Infiniband [40]. As IBA already has functionalities required by PR-DRB (e.g. monitoring functions at IBA switches, the CCA has procedures for congestion notification and path opening), PR-DRB integration into the IBA standard is feasible.

3.2.7 Contending Flows Notification

In section 3.2.2 we have mentioned that not all flows were notified about a congestion situation. Only those source-destination pair which contribute most to congestion are to be notified. Then this nodes would start the metapath configuration process properly. The *monitoring, detection and notification* phase could be *destination based*, as seen in

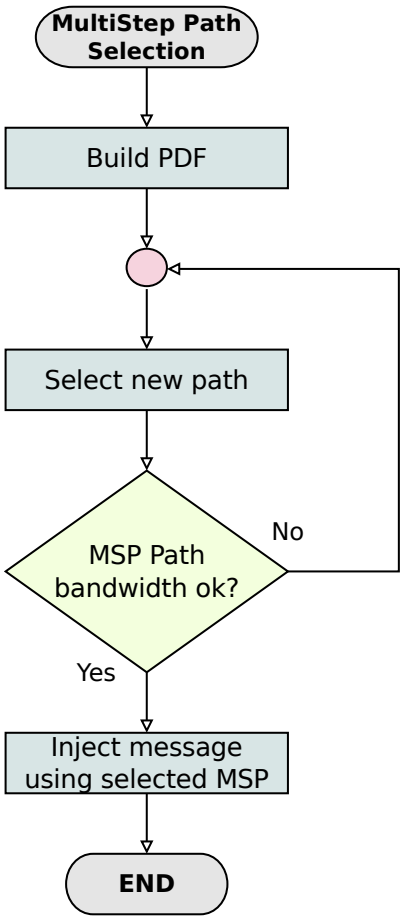


Figure 3.11: Multistep path selection.

3.2.2 or *routed based* as seen in section 3.4.1. Either case, the base information about what to send to the source is the same. Both strategies use the set of contending flows (and the latency) information. The only difference is that for the destination based the information about all the contending flows should travel into the PR-DRB header packet, to be processed at destination. On the other hand, the routed based approach reads the local buffers queues and then determine what to send via and ACK to the source. For this case, no additional payload should arrive to the destination. In order to identify which sources should be notified, among all racing for router resources, we select those which are causing congestion. We estimate this value by reading the average latency value (contention latency) of every packet in one output queue. We then select those packets (src-dest) which are the oldest and their average contention latency is constantly raising. In Fig. 3.13 we can see an example of which source/destination pairs are going to be notified. In the example, the source/destination pair (1-5) represents 50% of the packets causing congestion. The packets (2-7) represents 30%. Assuming the other packets do not

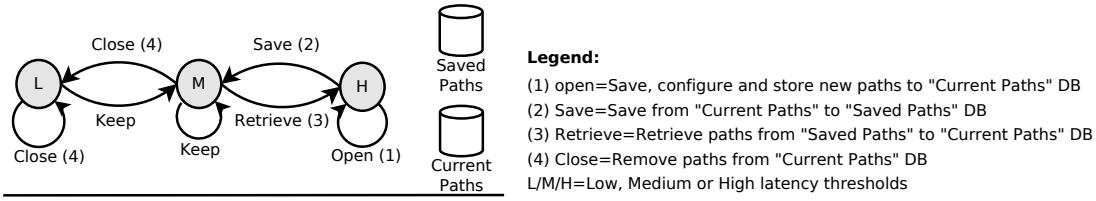


Figure 3.12: Metapath Configuration FSM

contribute to congestion, then only these two sources (1 and 2) will be notified. Our policy also checks that even if a source/destination pair appears more than once, the notification is performed only once per buffer s access.

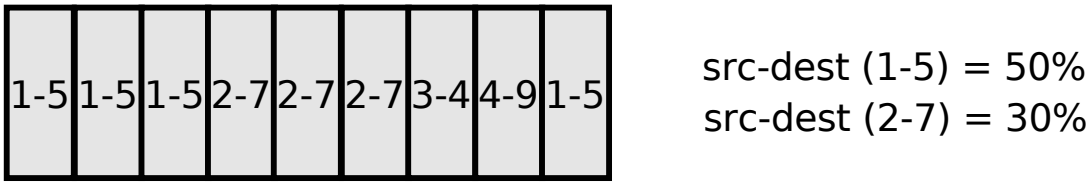


Figure 3.13: Example of a pair of source/destination to be notified

3.2.8 Contending Flows and Solution Management

A complete path expansion example is given in Fig. 3.14. This set of alternative paths conform the best solution found so far. In order to reuse the same known solution afterwards, PR-DRB saves contending flows and best solutions information. Contending flows pairs (S1-D1, S2-D2) are identified, as well as the paths opened for this solution (P1, P2, P3). The info registered is given in Fig. 3.14 Node S1 - Saved Solution . This diagram corresponds to what the node S1 knows about the congestion situation, and the paths it should open once it contends again against node S2. Each source involved fills its own table with particular paths opened for this situation. PR-DRB is based on contending flows comparisons during congestion. As parallel scientific application do have repetitive communication patterns in time, when PR-DRB identify a similar already analyzed situation, it looks for a set of optimal paths into its database of saved solutions. The process of detecting already analyzed situations is based on contending flows similarity, which is based on approximation matching. The percentage used for similarity is of 80%.

PR-DRB node level operations have not a high overhead because these operations are performed locally, they are simple (comparisons and accumulations for latency evaluation, logging small traffic info), and they do not delay send/receive primitives. During multi step path creation, deadlock freedom is ensured by having a separate escape channel for

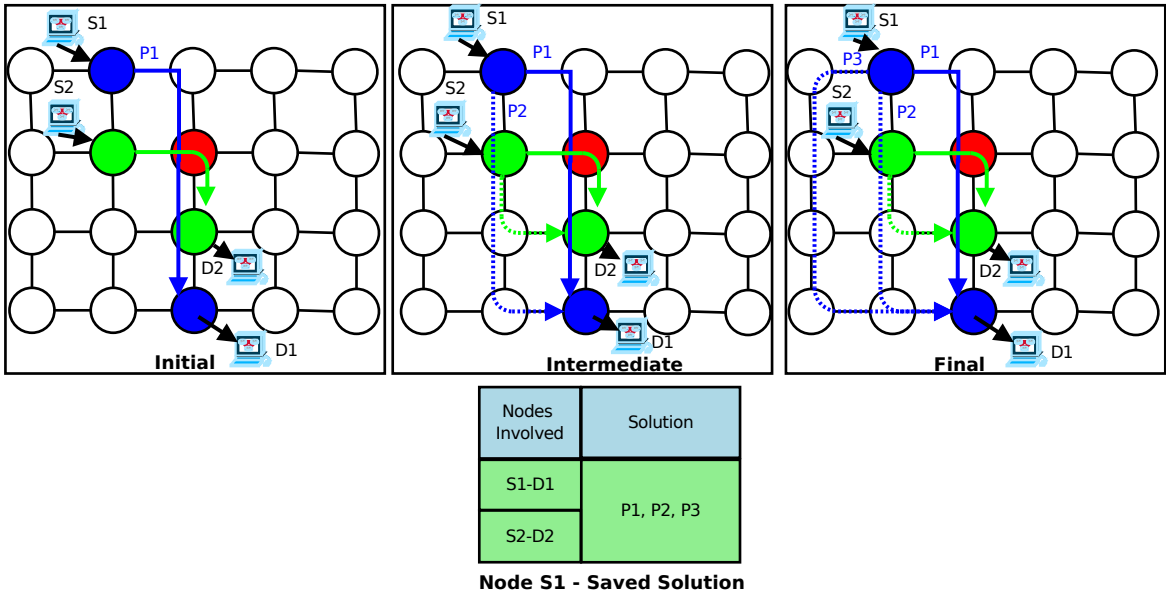


Figure 3.14: PR-DRB final solution saved

each segment. With two intermediate nodes (INx), one escape channel is used from S to IN1, another from IN1 to IN2, and a third one from IN2 to D. This way, each segment defines a virtual network, and the packets change virtual network at each intermediate node. Although each virtual network relies on a different escape channel, they all share the same adaptive channel(s).

3.2.9 PR-DRB Integrated

All the phases conforming the PR-DRB policy is shown in Fig. 3.15. Up to this point, each phase was described separately, but the PR-DRB actually performs in most cases concurrently. When a new message is ready for injection, a new *MultiStepPath* is chosen from those currently available in the *metapath*. Latency values are considered during this selection phase, paths with less latency values are going to be selected more often.

Once a packet 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 packet inside a router is defined as the time it must wait in the router s buffers with other packets, 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. 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 flows is recorded only at congested routers, as seen in section 3.2.2. This is done this way because when new messages are injected into the network using the PR-DRB strategy, new alternative paths will avoid the congested situation and hence saving information of other routers would be useless.

When a packet reaches its destination, it would have all the information about the network situation in it. Then, the destination node proceeds with the notification process. This notification informs the source node about what the packet have measured during its network trip. The notification message is the ACK packet. This ACK packet reaches the source node and the analysis process is performed. New alternative paths will be created if a congestion situation is detected, and metapath configuration phase will start. If latency values are controlled, then the normal injection of new messages is guaranteed. If congestion is detected, the PR-DRB starts looking for a suitable saved solution in its best solutions database . If one solution is found for the particular communication pattern being currently analyzed, then that solution is retrieved. The new packet is then injected into the network. With this solution, PR-DRB tries to save time by avoiding the procedure to create new alternative paths. If no paths or solutions are available at the database, then the procedure for new alternative paths is executed. If the congestion is controlled by the alternative paths created in this phase, ACK packets will confirm it by carrying latency information that is below a threshold. The configuration phase will then proceed to close those paths. Here the updating process will start, where those encountered good solutions will be inserted or upgraded into the database.

3.3 Architecture of Network Components

In this section we will describe the physical design and the implementation features required for the proposed policy, the PR-DRB. The use of adaptive routing may lead to situations where some abnormal situations arises, such as:

- **Deadlock.** Our method is based on the strategy presented in . This strategy makes use of virtual channels in order to get rid of cycles that produce deadlock.

- **Livelock.** Like our predecessor [21] and their descendants [41] [38] and [66], PR-DRB does not use paths of infinite length. Also, messages always reach their destination within a finite number of steps. Considering these concepts, we can affirm that Livelock would not be a problem.

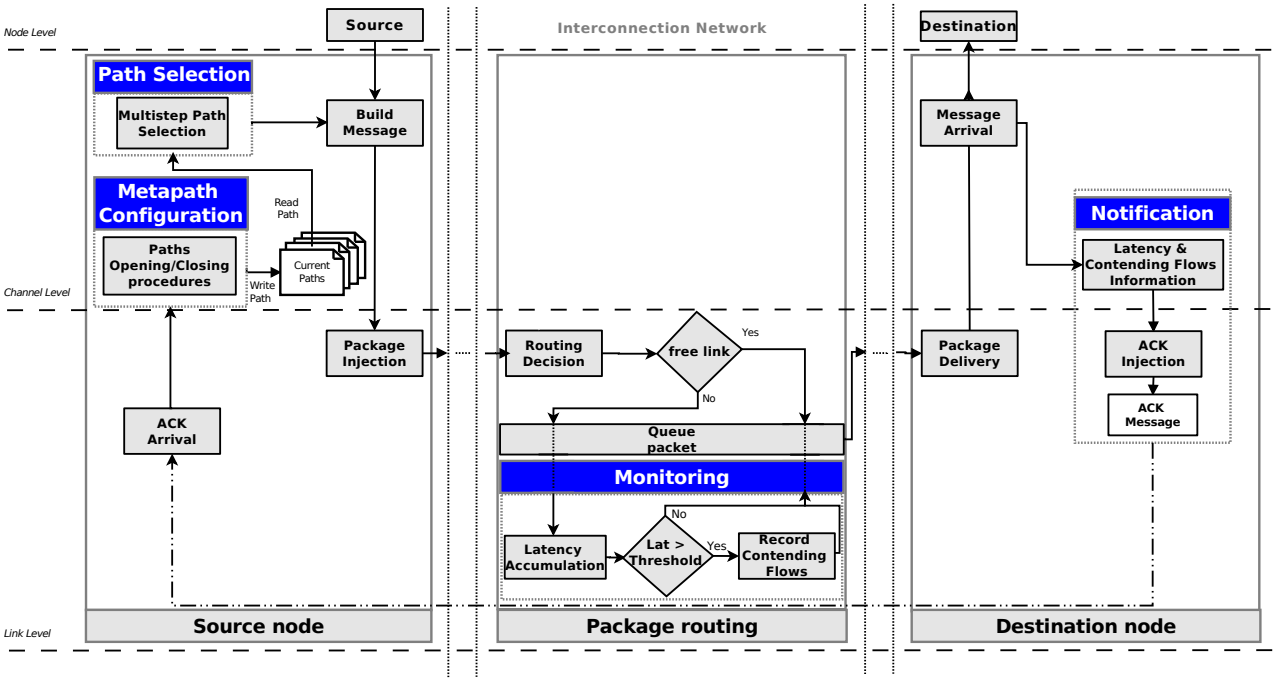


Figure 3.15: PR-DRB with all its phases

- **Starvation.** PR-DRB would not suffer from starvation, because it does not limit the injection of packets for indefinite time periods. All packets in intermediate routers have equal chances to access output links, therefore they can not be indefinitely blocked.

As we previously explained, PR-DRB is based on three main phases: Monitoring, detection and notification ; metapath configuration (section 3.2.3) and the selection of alternative paths to react to congestion situations (section 3.2.6). The following subsections address the implementation issues of these phases, together with the packets formats used for the PR-DRB policy.

3.3.1 Packets Formats

PR-DRB uses the information in the packets in order to carry information about the alternative paths, as well as latency and contending flows information. PR-DRB uses two types of packets, *Data Packets* and *ACK packets*. Data packets include a multiple header to store information about the intermediate nodes to traverse to get to the destination. The method uses only two intermediate nodes, to identify the *Multistep Path* (MSP). Every router should read the header properly in order to determine if it should process it.

The field two bits *Header_id* identifies which of the multiple options available for header should be used. This is performed at each step of the MSP. Packet forwarding is performed using the minimal static routing defined for the topology, at each segment of the MSP. When a packet reaches the router identified with one of the intermediate nodes in the header, then it changes the value in *Header_id* to point to the new intermediate node. The forwarding process then continues, and the other routers perform similar tasks until the packet arrives to destination. For this strategy to work, the router module needs a *Header Detection and Processing* (HDP) module. This module should be capable of handling the intermediate header of every packet. The *Data Packet* format is shown in Fig. 3.16. The $<$ *Reserved* $>$ MUST be sent as 0 and ignored on reception and at intermediate routers.

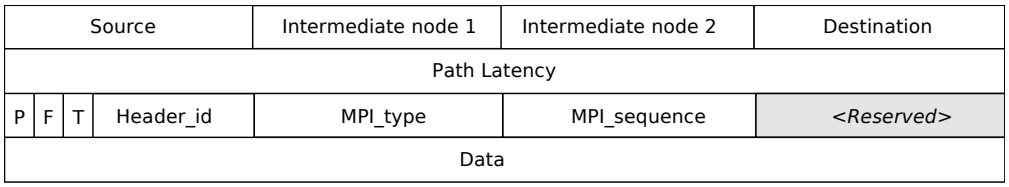


Figure 3.16: PR-DRB data packet

Data packet fields:

- **Source:** MUST be set to the id of the node originating this packet. Intermediate nodes that retransmit the packet to MUST NOT change this field. Of integer-size type.
- **Intermediate node 1:** The id of the first intermediate node (close to the source) used to construct the MSP. Of integer-size type.
- **Intermediate node 1:** The id of the first intermediate node (close to the destination) used to construct the MSP. Of integer-size type.
- **Destination:** MUST be set to the id of the node intended to receive this packet. Intermediate nodes that retransmit the packet to MUST NOT change this field. Of integer-size type.
- **Path Latency:** An integer-size field used for recording the path latency value.
- **Predictive bit (P):** Set to indicate that the PR-DRB router has injected a *Predictive ACK packet*. Therefore, the destination node should not inject a simple ACK.

- **MPI_final packet (F):** Set to indicate that this is the last packet from a fragmented set. This is used to indicate that no more segments are needed in order to construct the message to pass to the upper layer.
- **Type (T):** Bit-sized field to set the type of the packet, *Data* or *Acknowledge* (ACK).
- **Header_id:** MUST be set to the id of the intermediate node intended to process the header.
- **MPI_type:** Set to the id of the type of MPI_call executed. Used by the procedures that guarantee the logical trace execution of packets. Of integer-size type. Examples of these calls are: MPI_Send, MPI_Receive, MPI_Wait, MPI_Bcast, MPI_Allreduce, etc.
- **MPI_sequence:** A unique value generated by every MPI_call executed. Used by the procedures that guarantee the logical trace execution of packets. Of integer-size type. Also used to guarantee the arrival of messages (the order of packets) in an orderly fashion.
- **Data:** The payload of the packet. To be used by the application.

PR-DRB also uses the ACK packet, or the notification packet. This packet is used to notify the source about the network status. Recall that to PR-DRB the network status comprises the latency of the path and the information about the set source/destination pair contending for an output port on a router. The ACK packet is similar to the Data packet, but it is simpler. It basically contains the routing info and the network status info. The former contains the the source node, the two intermediate nodes, the destination and the Header_id. The latter includes the information about the status: the latency and the fields used to properly identify the related logical call. The ACK packet is shown in Fig. 3.17.

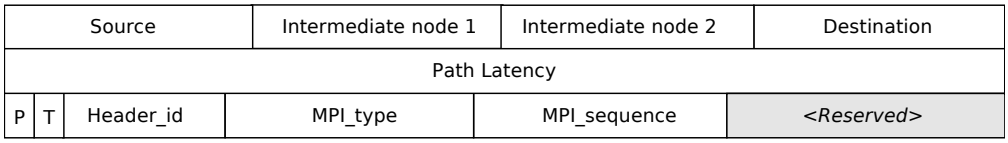


Figure 3.17: PR-DRB ACK packet

We have mentioned in section 3.2 that the PR-DRB strategy could be *router based* or *destination based*. Based on the approach implemented the information about the

contending flow should be communicated to the source. For this purpose, the *predictive optional header* is used. For the *router based* approach, this predictive header is added to the ACK packet. For the *destination based* approach, the predictive header is added to the *data packet* until it reaches the destination. Once there, the predictive header is copied to the *ACK packet* to inform the source node about congestion. Fig. 3.18 shows the predictive header.

Type	Opt Data Len	Router id	*<Reserved>*
Contending flow[1]		Contending flow[2]	
Contending flow[...]		Contending flow[n]	

Figure 3.18: PR-DRB predictive packet

Predictive packet fields:

- **Type:** Set to indicate a full predictive search or a tendency latency search at source node.
- **Opt Data Len:** Length of the option, integer-size, excluding the Option Type and Opt Data Len fields. MUST be set equal to $(integer_size \cdot n) + 1$ where n is the maximum contending_flows number to send. This n is a system parameter.
- **Router id:** The id of the router that detects the congestion and starts the notification procedure. If notification is *destination based*, this field must be set to 0.
- **Contending flow[1..n]:** The set of source + destination contending for resources (output port) at this router. The procedure explained in section 3.2.7 is used to compress the information to inject via the *predictive ACK* in the *router based* and as an additional header in the *destination based* approach.

3.3.2 PR-DRB Router

The architecture of the PR-DRB router is depicted in Fig. 3.19. Our router share some design architectures with the work presented in *Multipath Fault-tolerant Routing Policies to deal with Dynamic Link Failures in High Speed Interconnection Networks* (FT-DRB) [65]. The main difference with our method is that FT-DRB uses its own mechanism in the router in order to provide fault tolerance in the network. On the other hand, the capabilities for ACK packet injection is shared between this two policies. Our method

diverges from the previous work in that it has mechanisms to analyze contending flows. The router includes the following modules:

- **Latency Update (LU).** In charge of the latency accumulation of the packets. No synchronization is needed to measure the time a packet is waiting in a queue. A router *clock* governs this task.

- **Header Detection and Processing (HDP).** In charge of modifying the *Header_id* bits of both data and ACK packets, at intermediate routers. At destination nodes it is in charge of the intermediate headers removal process. This mechanism is part of the *Routing and Arbitration* (R+A) unit.

- **Contending Flows Detection (CFD).** This module has been designed to analyze the flows currently at any particular output port. When a congestion situation is detected, it starts the procedures to extract the source and destination of every packet actually in the congested queue. This module has a register where a threshold value is maintained. Once a packet reaches the output queue, its latency is added to the global latency of the output port. This sum is then compared against the threshold value stored in the internal register. If it surpasses the threshold, then the procedure of identifying contendingflows is started. Once the contendingflows process is finished, and the PR-DRB approach is *router based*, then the GPA module is invoked.

- **Generation of Predictive ACK packets (GPA).** In the situation of a congested output port, this module generates the notification packet and sends it back to all the corresponding source nodes.

3.4 Design Alternatives

In section 3.2.2 we explained the standard Monitoring, detection and notification procedures. This process could be used when the router doesn`t have the capabilities to inject packets into the network. We saw that the detection was performed at intermediate routers, but all the information about contending flows was not processed there. Instead, all this information was introduced into the packet and sent to the destination node. Once at the destination, the information about contending flows was pre-processed and then sent back to the source for the final metapath configuration process. In order to speed up the process, the intermediate routers can start the notification immediately after analyzing

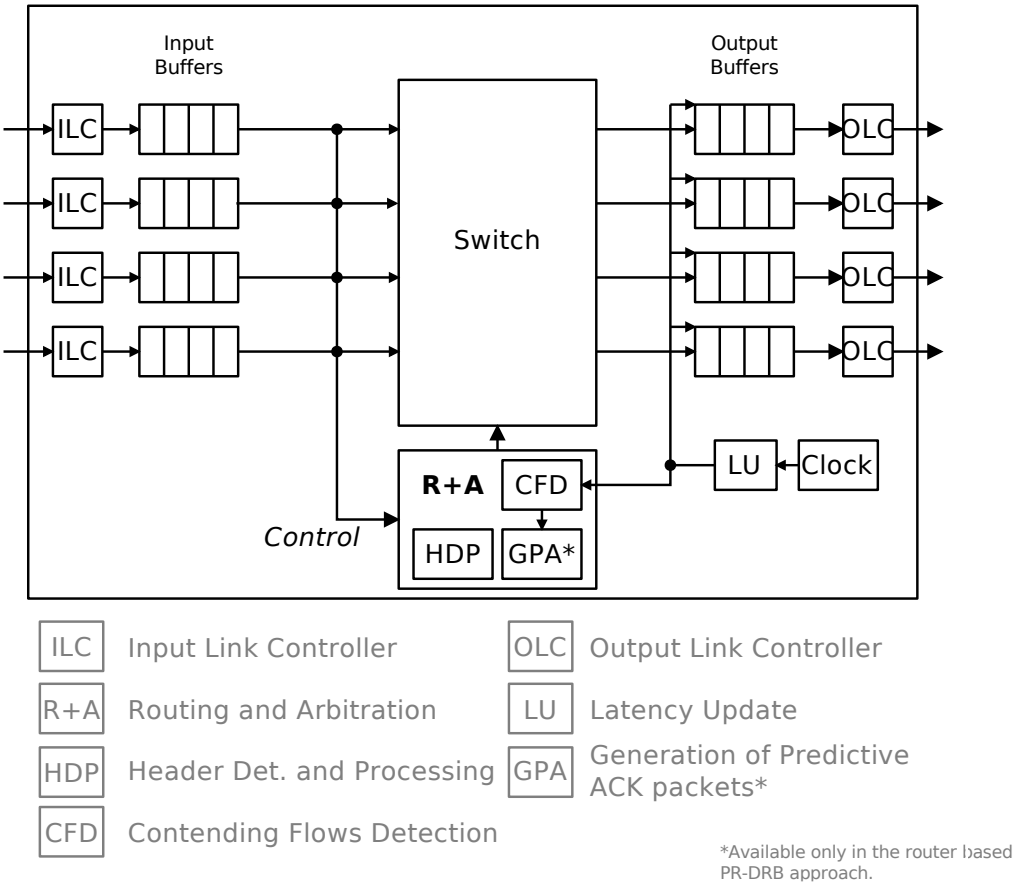


Figure 3.19: PR-DRB router

the latency and contending flows information. This would benefit the predictive procedure, specially when detecting and reacting to already analyzed patterns.

3.4.1 Early Detection & Notification - Router Based

Basically, most of the tasks described in 3.2.2 remains unchanged. Only some specific functions are now performed at the intermediate router, instead of the destination node. Here, the router also monitors the latency and the set of contending flows. When latency surpasses a high threshold value, instead of forwarding all the information towards the destination, the notification process is started. The new ACK message injected into the network is shown in the *Predictive Acknowledge Injection* box in Fig. 3.20. This approach requires more resources at the router, but makes it more robust under network congestion. This approach requires that the router should be capable of packet injection in order to send the predictive ACK packet. The flow diagram for the *router based* approach is shown in Fig. 3.21. The monitoring algorithm for the *router based* version is also shown in the

Appendix in A.4. Recall that the same analysis for the *destination based* approach was made at section 3.2.2.

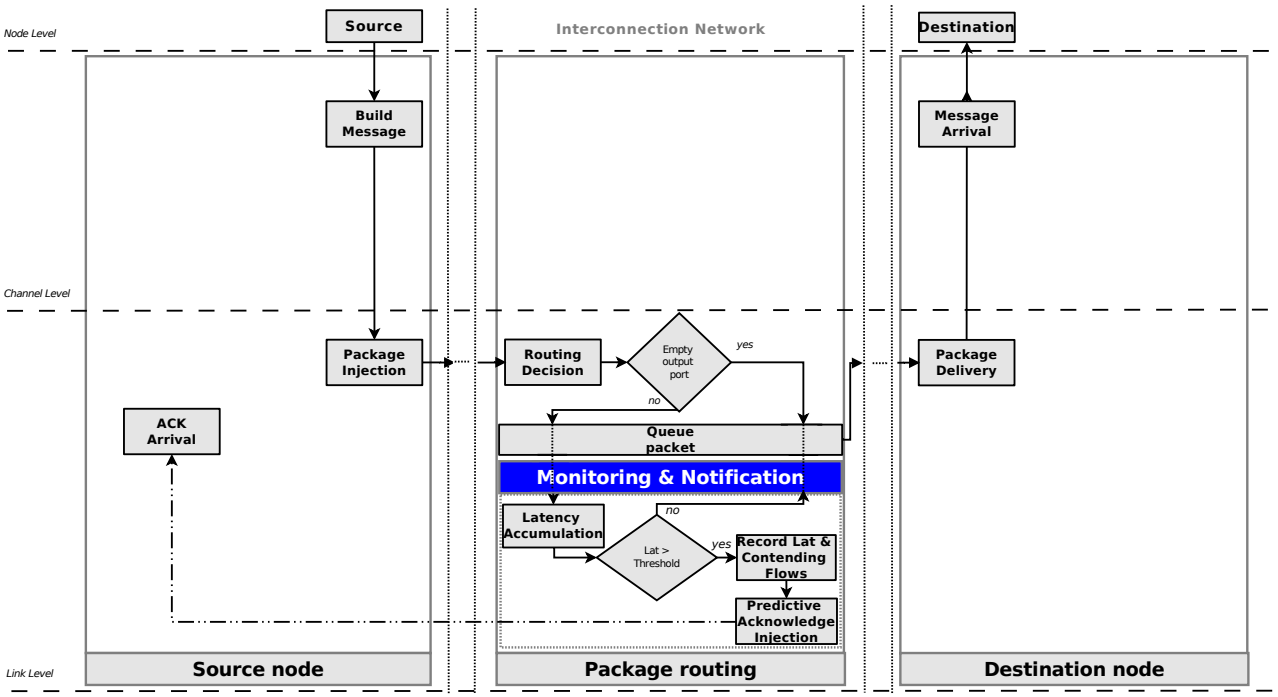


Figure 3.20: Early monitoring procedures

3.4.2 Latency Notification - Router Based

If the detection procedure is executed earlier, then the notification process must be also modified. Having the ACK notification been executed in the intermediate router, then destination notification is much simpler. Here, only the latency information will be sent to the source node. This is because the contending flows information was already been sent. If the router has injected a new ACK, then the predictive bit from header is modified in order to include this fact. The destination now has to read the predictive bit in order to inject the ACK message. This ACK now only has the latency information, which will be used by the metapath configuration process. The latency notification process and general overview of this new design alternative are shown in Fig. 3.22.

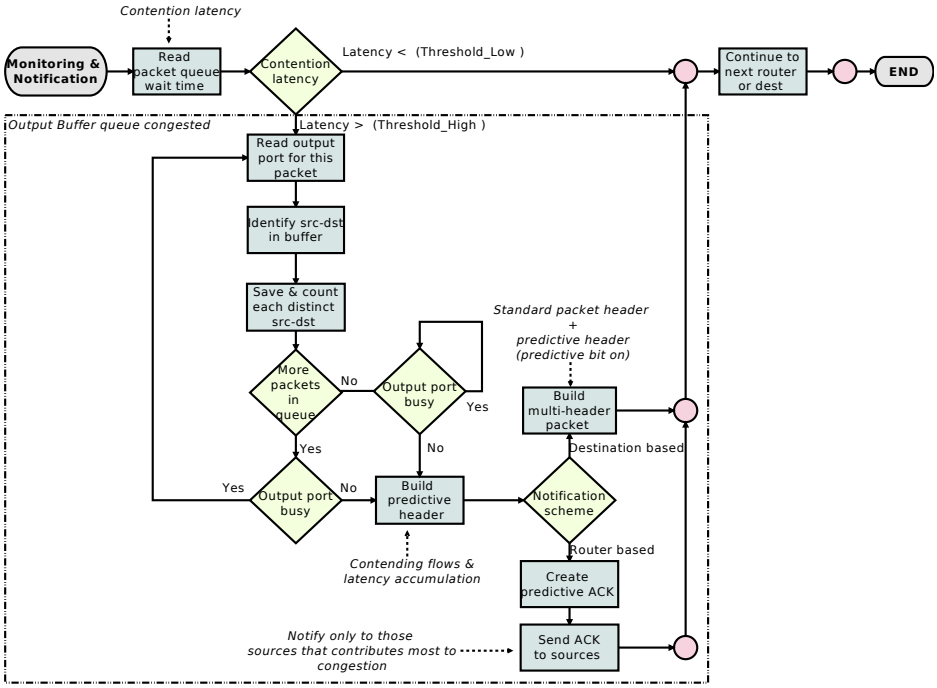


Figure 3.21: Monitoring and detection flow diagram (router & destination based).

3.5 Discussion

In this chapter we have presented the complete predictive and distributed routing approach. The method s main goal is to dynamically learn from parallel applications communication patterns and then try to relate the pattern information with the adaptive solution given by the routing algorithm.

The proposed method does not need any meta information injection prior to network usage. All the information *PR-DRB* needs is extracted dynamically from the communications patterns actually traversing the internal buffers of routers. Some adaptive routing algorithms, such as *DRB*, adapts by performing a controlled path expansion. These paths are opened and closed based on the interpretation of the network load by the routing algorithm. The complete adaptive procedure produces a good combination of paths to handle all the communication requirements of the parallel application. But in order to achieve this, the routing algorithm requires some time. One feasible approach would be to *statically* analyze the application and extract the pattern information out of it. We would also need to obtain the best routes available in the network for this particular application s traffic pattern. The process should be repeated for every application intended to use. Instead of this approach, we ve leverage a fully dynamic approach, capable of learning from the communication patterns currently in the network. One of the items of our future work proposal (Section 5.2) includes a *static* variation of our method.

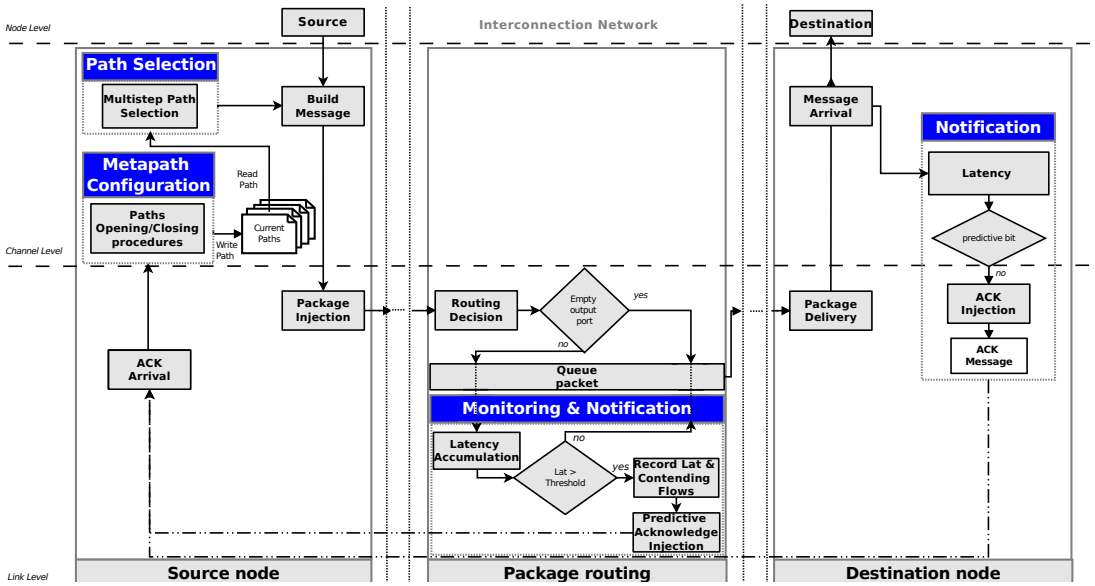


Figure 3.22: PR-DRB with all its phases - Early detection and notification

The decision of a fully dynamic predictive approach encompasses some situations where some overhead is introduced. For example, when the method decides to re-apply a saved solution (set of alternative paths) based on the interpretation of a repeated communication pattern that actually is a new one. Under this scenario, the method does not introduce more congestion because the adaptability of *PR-DRB* will detect that our solution is not good and will start the standard opening path procedures. Also, the dynamic approach involves the need for special containers for the real-time information obtained. The method should be capable of treat/save this information in order to properly make routing decisions. Because all the resources needed are not far beyond today s standards (such as Infiniband [26]), all the proposed solutions are feasible.

As conclusion remarks, we could mention that our method *PR-DRB* allows the design of a predictive and application-aware approach in order to solve congestion situation in the network, for parallel applications that exhibit basic repetitive behavior.

Chapter 4

Evaluation

In this chapter, we present the evaluation of the *Predictive and Distributed Routing Balancing* methodology. The evaluation aims to corroborate that our proposal works properly, using a set of different test scenarios within a simulation environment. We use three main components to carry out the evaluation: the simulation models to represent the system under study, e.g. the interconnection network; the workloads to be injected as input of the simulations models; and the metrics to asses the benefits of our proposal.

The purpose of the evaluation process is to confirm the operation of PR-DRB as it was described in previous chapters. To this extent, we have defined a set of metrics to observe both functional and performance features of our proposal. These metrics have been chosen with the aim of measuring several aspects related to the capability of interconnection networks and their impact on the applications. The most representative metrics are the average network latency, the contention latency at intermediate routers and application execution time. The workloads represent a broad set of input, ranging from synthetic traffic to real applications traces. Our proposal have been implemented using accurate InfiniBand-based simulations models.

4.1 Simulation Models

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.

PR-DRB operations together with network components were modeled [39] using the

standard simulation and modeling tool OPNET [47]. OPNET provides a Discrete Event Simulator engine and offers a hierarchical modeling environment with an enhanced C++ language. This allows defining network components behavior by a Finite State Machine approach (FSM), and it supports detailed specification of protocols, resources, applications, algorithms, and queuing policies.

OPNET offers a hierarchical structure for modeling purposes. It defines the *network, node and process* levels. At network level OPNET includes the nodes, links and subnets among them, which form the topology. At node level the network components are represented as *modules*

Simulation Management Tool

In order to launch a large set of simulations for the evaluation of our proposal, we have designed and implemented a simulation management tool [45] to perform this task automatically. The tool acts as an interface between OPNET Modeler and non-dedicated computer clusters. With this tool we have been able to increase the simulation capacity by using more of the available computing systems, including an efficient use of multicore processing nodes.

4.1.1 Processing Nodes

In order to implement the source or processing node, several OPNET specific objects, such as the *processor* are used, as shown in Fig. 4.1. The source has a processor node that mimics the behavior of the application by injecting data according to a traffic pattern. The model also has a network interface, which receives the traffic generated and then insert them into the network. In Fig. 4.2 we can see the finite state machine used for the processing node. We can see that our models can deal with synthetic traffic, as well as to read the instructions to generate the logic from real applications. Several attributes related to the injection procedures can be modified, such as the injection rate, start and stop time, packet format and length, workload characterization, etc. The destination node FSM is shown in Fig. 4.3. Here, the packets are analyzed and then the statistics are updated, before they are *consumed* by the processor. In the FSM of the Fig. 4.4 we can see the flow that packets traverse when they arrive to the network interface. Here, the packets are received and buffered until the processing unit is ready to process them.

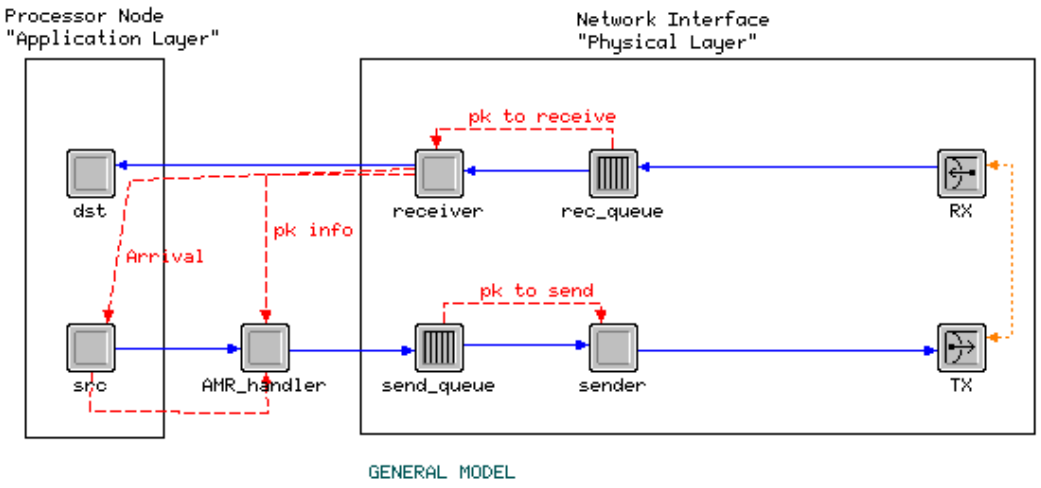


Figure 4.1: Processing node model implementation.

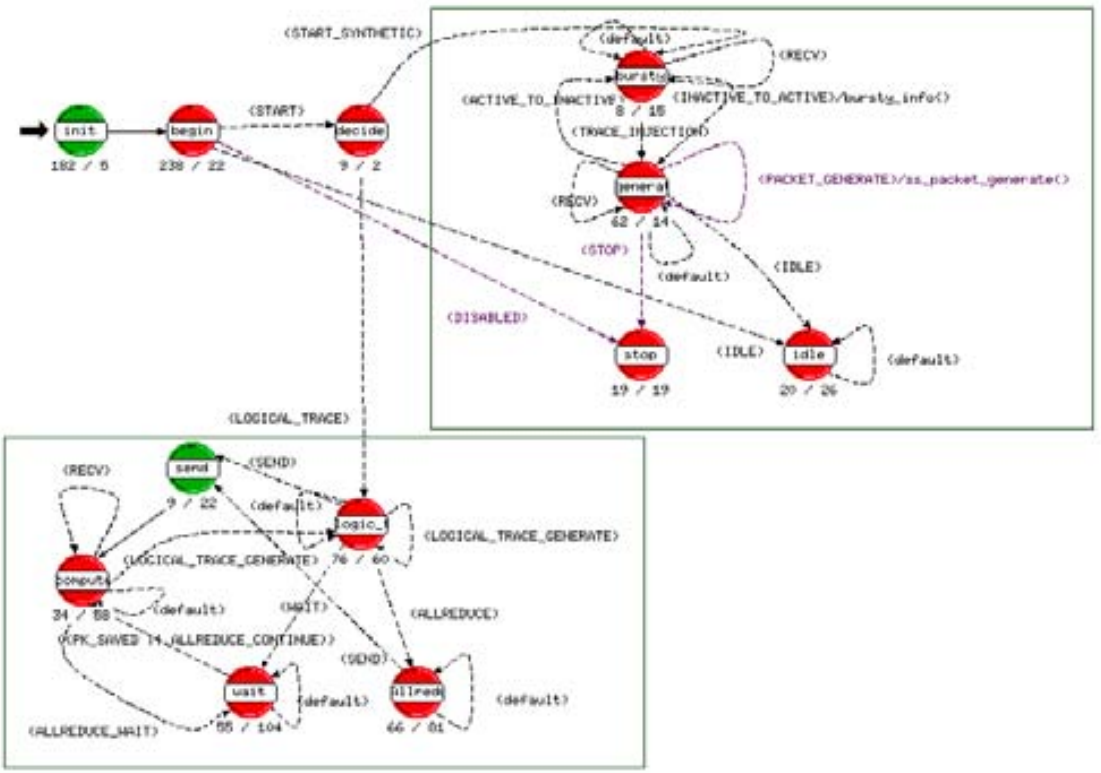


Figure 4.2: Source node FSM.

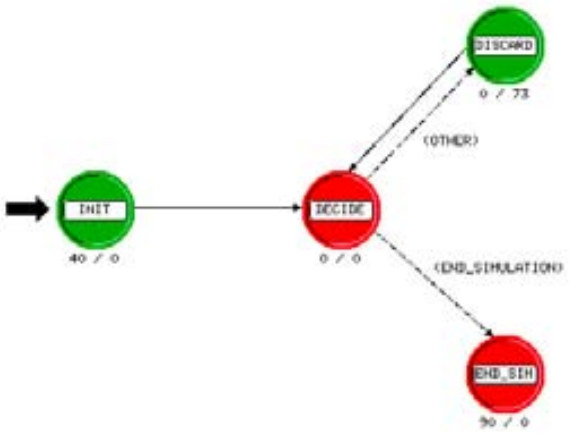


Figure 4.3: Destination (sink) FSM.

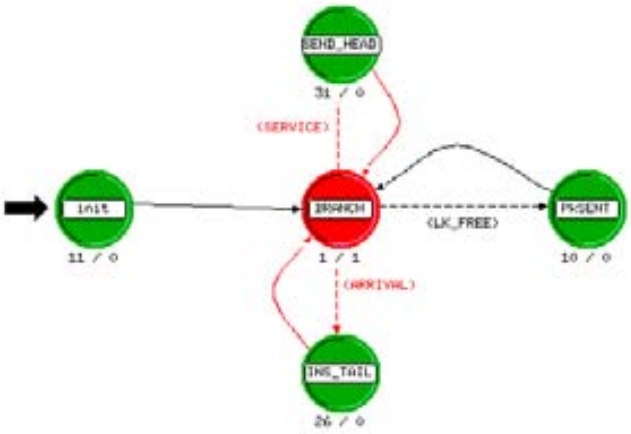


Figure 4.4: Processing node FIFO FSM.

4.1.2 Network Nodes

The internal structure of the implementation of the 12-ports network node is shown in Fig. 4.5. This model provides a set of modules that allow to experiment with several routing policies. The logical behavior of the router is given by four main modules: the switch manager, the routing unit, the arbitration unit, and the crossbar. It also has the *switch_info* unit, which is basically the subnet manager, in charge of the whole network initialization and the discovery and configuration of the network topology, and also defines the algorithmic routing or the routing tables if applicable. The routing unit is shown in Fig. 4.6 we can see the routing unit FSM. This unit decides the output port for each incoming packets, besides the task of handling error situations. The routing unit handles simultaneous requests by applying a round-robin technique. If more than one packet tries to access the output port, then they are stores in specific data structures and are served sequentially. Basically, PR-DRB is implemented within this module. Once the routing unit has found an output port for a packet, then one signal to match input and output ports are emitted by the crossbar unit.

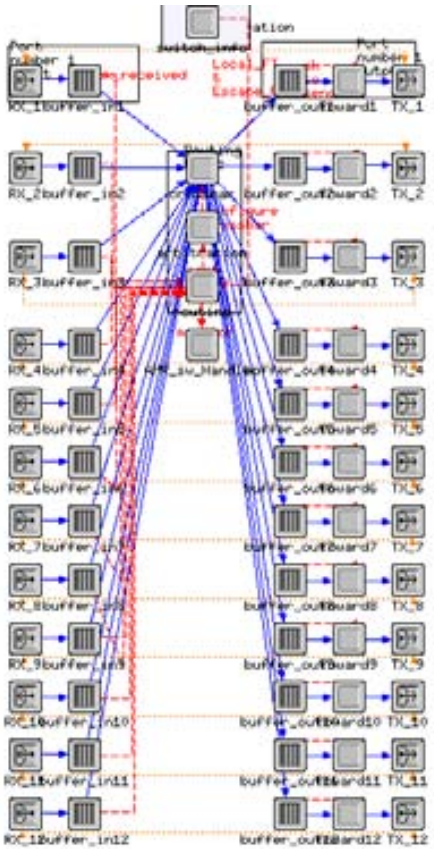


Figure 4.5: Router node model implementation.

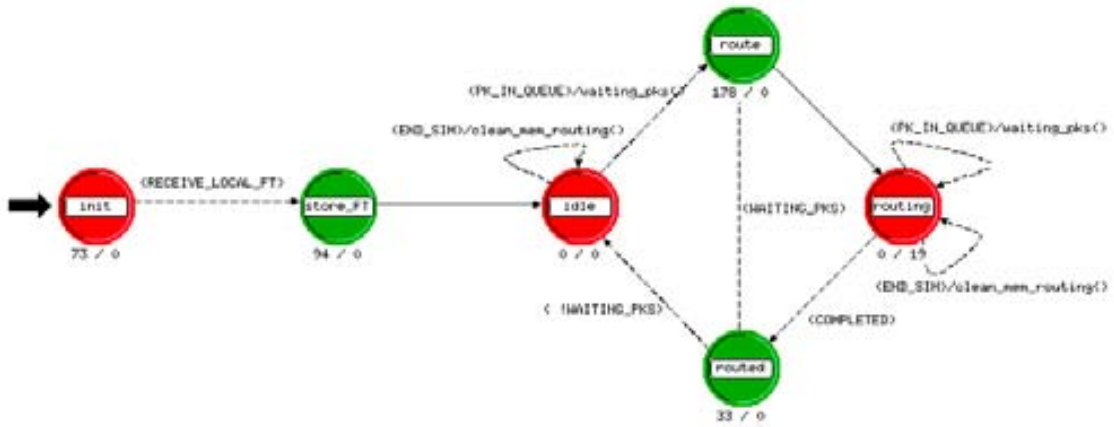


Figure 4.6: Routing FSM.

4.2 Evaluation Metrics

In order to assess the performance of the network we consider the latency of messages. The latency is the total time elapsed since a packet is created until it reaches the destination. Basically it is measured since the head of the first packet enters the network until the tail of the last packet leaves the output port [62, Ch. 3]. The average latency for each packet x to a destination i is then obtained, as given in Eq. 4.1, where $l_i[x]$ is the latency value of the packet x at the node i.

$$\overline{L}_i[x] = \frac{1}{x}(l_i[x] + (x-1) * \overline{L}_i[x-1]), \quad \forall x \neq 0 \tag{4.1}$$

The global average latency is calculated by averaging the latencies of every packet, and is measured in seconds as defined in Eq. 4.2, where n is the number of destination nodes.

$$\overline{L} = \frac{1}{n} \sum_{i=1}^{n} \overline{L}_i \tag{4.2}$$

The *throughput* is the data rate in bits per seconds that the network accepts per unit of time (seconds) per input port. To this end, we have taken into account the ratio between the number of packets received at destination nodes (accepted load) and the number of packets injected at source nodes (offered load). In the evaluation of our proposal, we guarantee that the ratio between the offered load and the accepted load is always maintained, ensuring that there are no traffic lost during our simulations.

In order to show the load distribution of messages in the network, a metric to measure

the traffic load in every router has been defined. This metric is the *latency surface map* or just *latency map*. It is defined by a three-dimensional graph where each point xy represent a router in the network and the z represent the average latency of internal buffers for that router. With this metric, is possible to analyze the latency reduction in the network. In the fig. 4.7 we can see an example of a latency map.

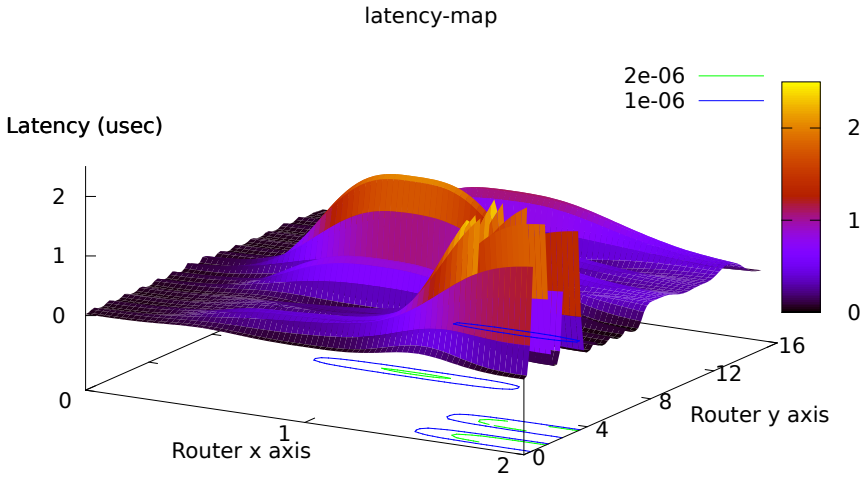


Figure 4.7: Latency surface map.

4.3 Evaluation Method

The statistical validity of the results from our discrete event simulations are assured by executing multiple instances of the simulation with a different set of random seeds. In order to provide a confidence interval, we have run each simulation between two to thirty times according to [27].

Therefore, in order to avoid statistical anomalies, the results obtained from each individual simulation were averaged to estimate the typical behavior of the system. By following this procedure, we expect to avoid erroneous and inaccurate results and obtain greater degree of confidence.

The experiments presented along this chapter have been conducted taking into account the above mentioned aspects for statistical validity. More precisely, we have used the methodology proposed in [48]. Evaluation results and test scenarios of our approach are explained in detail in further sections.

4.4 Workloads

A workload must be defined in order to evaluate our proposal. A workload can be defined as the communication traffic traversing the network. We have used the following approaches in order to cover a wide spectrum of communication characteristics, as listed below:

1. Specific pattern scheme
2. Synthetic performance benchmarks
3. Real applications logical traces

The first approach is obtained by defining a specific set of source and destination in the network. The second correspond to a set of application benchmarks , while the third is obtained from the real application itself. The *Specific pattern scheme*, is used to generate known imbalance situations in order to analyze the correct behavior of our policy under these special situations. The *hot-spot* specific pattern is shown in section 4.5. The second approach comprises a set of application-inspired *performance benchmarks*, that describes the behavior frequently found in many scientific applications. The set of *traffic patterns* used in these benchmarks is explained in subsection 4.6. The last approach covers the behavior of *real applications patterns* by executing the application and extracting the logical trace used for each particular application, as shown in section 4.7.

4.5 Hot-Spot Specific Traffic Patterns

Under this traffic pattern a set of paths are strategically defined in the network so that they collide and produce high network congestion load. The paths that collide do not share the source and destination nodes, but they do share some portion of their trajectories. The shared trajectories are the ones where the congestion is produced.

4.5.1 Path Distribution Analysis

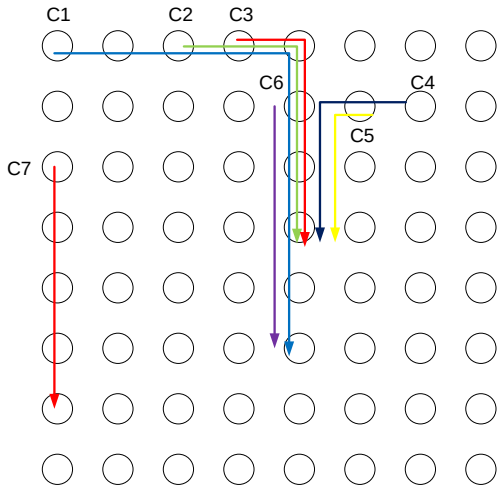
As part of the specific traffic pattern, we define a set of experiments to analyze and understand the alternative path opening procedures of adaptive algorithms, specifically the behavior of the DRB algorithm under congestion situation. This is a key situation to understand traffic behavior under different 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.

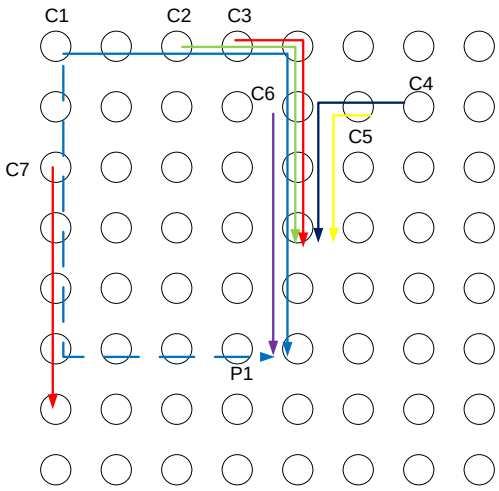
In Fig. 4.8 a situation of hotspot is depicted and a complete scheme of the solutions given by DRB can be visualized. Fig. 4.8a shows the initial state of the network, where all nodes start transmitting and enter a common area in their path toward their destination, where congestion will be produced 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 Fig. 4.8b, 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 4.8c. 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. Fig. 4.9 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.

In Figs. 4.9c and 4.9d we can see 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,

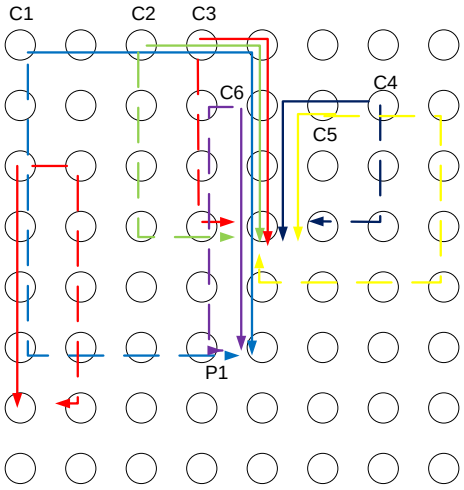
and avoid most unfavorable situations.



(a) Hotspot 1. Original state.



(b) C1 after the first alternative path.

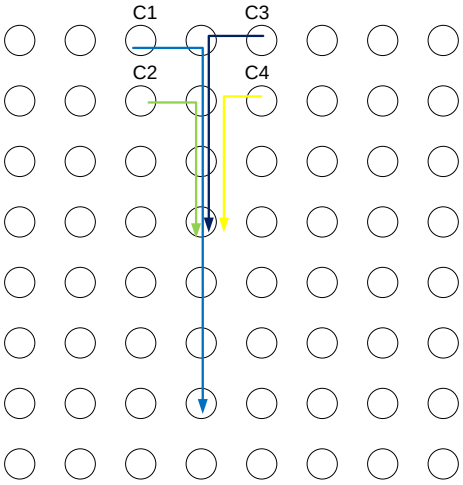


(c) All nodes after the first alternative path.

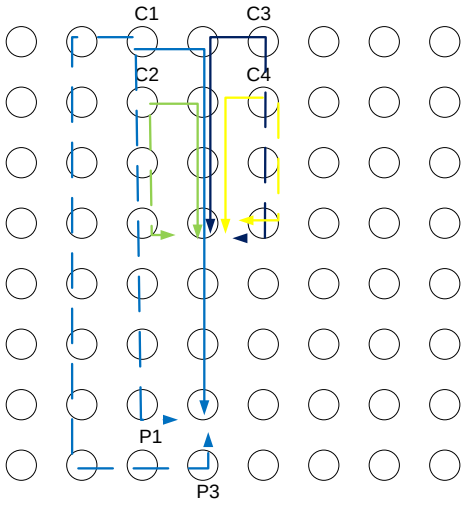
Figure 4.8: Path opening procedures & hot-spot situation 1

4.6 Synthetic Traffic Patterns

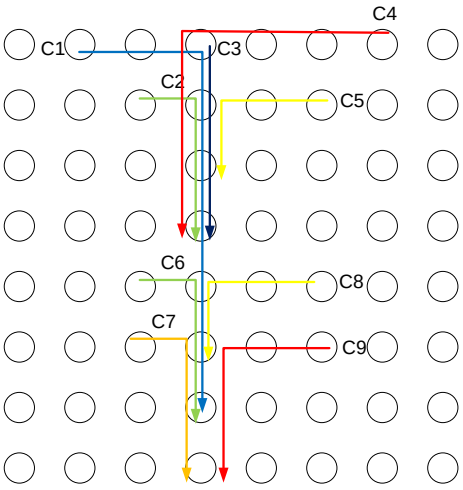
In order to simulate the communication patterns commonly used in computational intensive scientific applications, a set of performance benchmarks are used. This benchmarks describes the permutation performed in mathematical or numerical programs[Ch. 9]Duato2002, [62, Ch. 3]. The destination nodes remain invariable throughout the pattern for a particular source node. The traffic patterns used for this work are *Bit reversal*, *Perfect shuffle* and *Matrix transpose*. Their mathematical descriptions are shown in Table 4.1,



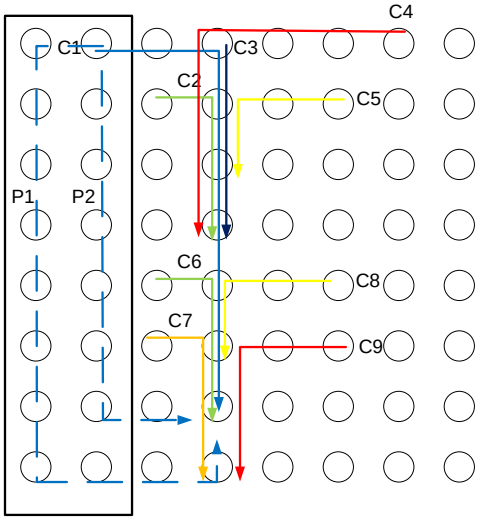
(a) Hotspot 2. Initial state.



(b) After path opening procedures.



(c) Hotspot 3. Initial state.



(d) After path opening procedures.

Figure 4.9: Path opening procedures, hot-spot situation 2 & 3

where source and destination nodes are denoted as s and d, respectively; and n is the number of bits used to represent the nodes. For some experiments, the *Uniform* has also been used. This pattern randomly selects its destination node.

Pattern	Destination	
Bit reversal	$d_i = s_{n-i-1}$	$\forall i : 0 \leq i \leq n-1$
Perfect shuffle	$d_i = s_{(i-1)\ mod\ n}$	$\forall i : 0 \leq i \leq n-1$
Matrix Transpose	$d_i = s_{(i+\frac{1}{2})\ mod\ b}$	$\forall i : 0 \leq i \leq n-1$

Table 4.1: Mathematical description of synthetic traffic patterns.

4.6.1 Synthetic Traffic Evaluation

Evaluation methodology for synthetic traffic is divided in two parts. The first is designed to perform a network response analysis under *hot-spot* traffic using a mesh topology and to evaluate PR-DRB dynamic behavior and traffic load distribution. We designed the *hot-spot* experiment to analyze PR-DRB under extreme conditions. This *hot-spot* experiment establishes some fixed destinations to increase traffic in a particular network area, and produces network congestion. Remaining network nodes inject uniform load to create noise traffic. The second part uses the *Systematic performance benchmarks*, in order to evaluates PR-DRB using various well known communication patterns such as: Perfect Shuffle , Bit Reversal and Matrix Transpose , under a fat tree topology with 32 or 64 nodes.

4.6.2 Hot-spot Analysis

For these experiments, bursty traffic were injected in a 64-node network arranged in an 8x8 mesh topology. The detail of the simulation parameters used to evaluate PR-DRB with *synthetic specific traffic* is presented in Table 4.2. Communication traffic patterns from bursty injection appear many times throughout the simulation. DRB response to the repetitive bursty traffic is always the same, because it is not capable to learn from past communications. On the other hand, PR-DRB algorithm identifies communication patterns already analyzed and use past solutions to control the congestion. Figs. 4.10 and 4.11 show average latency map of the mesh network after the execution of the whole bursty simulation. The map only shows the coordinates for the nodes that experiment some level of contention, in order to make the graph clearer. Latency surface represents the average contention latency at buffers. Fig. 4.10 shows the behavior of the original DRB algorithm,

Network Parameters	Value
Network topologies	Mesh 8x8,
Flow Control	Virtual Cut-through
Link bandwidth	2 Gbps
Buffer size	2 MBytes
Packet Size	1024 Bytes
Packet generation rate	400 Mbps 600 Mbps
Traffic patterns	Perfect Shuffle, Uniform

Table 4.2: Simulation parameters used in the evaluation of PR-DRB under hot-spot systematic traffic.

where high values of latency can be seen under congested areas. Also, load distribution at routers in coordinates (x,y) (3,1) and (3,2) are considerable high, because DRB uses these routers in its alternative paths. We can also see from the map surface`s contour , the use of DRB to handle congestion. The amber contour shows a greater use of links and bandwidth compared to the PR-DRB policy. Fig. 4.11 shows the latency map for PR-DRB; where its highest value is lower than the original DRB. Better load distribution is accomplished by PR-DRB compared to DRB, because PR-DRB has directly applied best solutions already saved, and unnecessary load at routers are avoided. For this case, global latency reduction of about 20% is accomplished. Fig. 4.12 shows average latency values for the entire mesh network, also under a series of repetitive bursty traffic. Here, on average, PR-DRB outperforms DRB because it reaches better global latency values in less time. The figure represents a second phase of the application. PR-DRB improves latency because it has already learned from previous communications. Throughput is not penalized whatsoever with latency gains of PR-DRB. Fig. 4.12 also shows that DRB has an initial latency raise due the fact that it is opening alternative paths in order to control a particular congestion situation. Recall that PR-DRB will behave similarly to DRB only under the execution of the first phase of the parallel application, because in this stage PR-DRB is learning from the alternative paths opening procedures. In the next phases of parallel applications, like the one shown here, PR-DRB will apply directly the best solutions encountered previously. From time 1.5 second, latency values of both algorithms tend to become stable and converge.

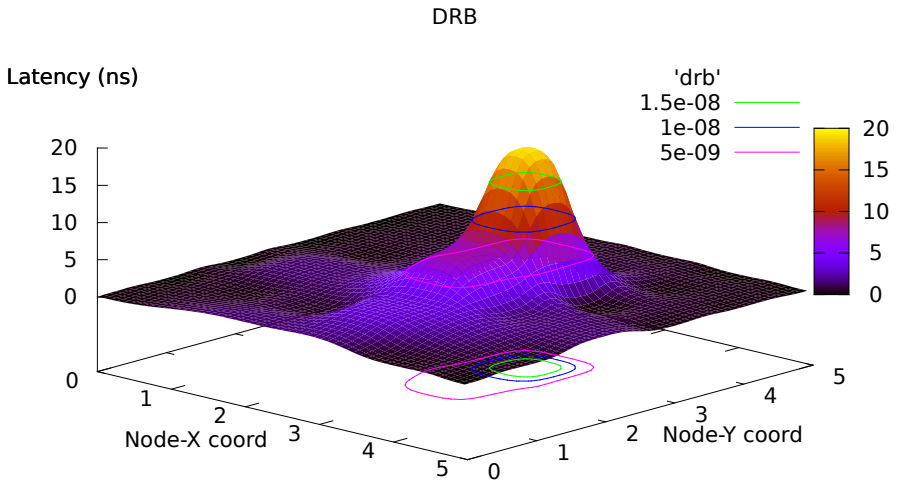


Figure 4.10: Latency map - drb

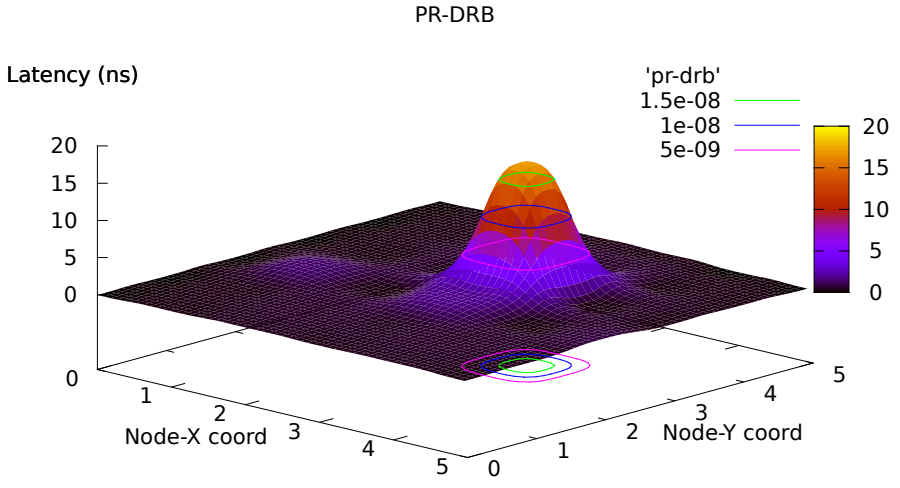


Figure 4.11: Latency map - pr-drb

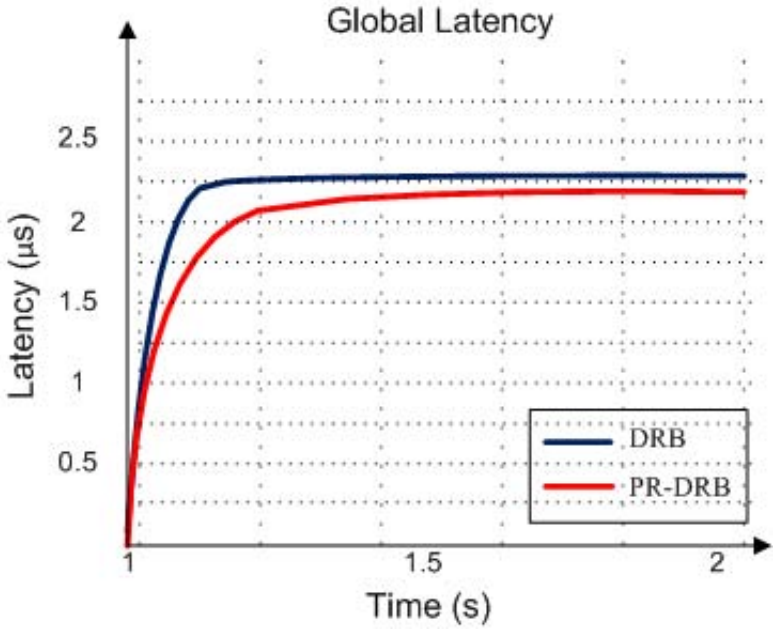


Figure 4.12: Average latency in mesh topology

4.6.3 Analysis with Permutation Traffic

The detail of the simulation parameters used to evaluate PR-DRB with systematic permutation traffic is presented in Table 4.3. Fig. 4.13 and 4.14 show the network

Network Parameters	Value
Network topologies	Fat-tree 4-ary 3-tree
Flow Control	Virtual Cut-through
Link bandwidth	2 Gbps
Buffer size	2 MBytes
Packet Size	1024 Bytes
Packet generation rate	400 Mbps 600 Mbps
Traffic patterns	Bit Reversal, Perfect Shuffle, Matrix Transpose

Table 4.3: Simulation parameters used in the evaluation of PR-DRB under systematic traffic.

performance under Shuffle traffic pattern with traffic load from 400 to 600 Mbps/node, for 32 communicating nodes, respectively. It can be observed that PR-DRB achieves lower latencies than DRB. Latency gain achieved is 29% for low load situations and 22% for higher loads. Here we can see that the increase in traffic injection, from 400 to 600 Mbps, is handled properly by PR-DRB routing mechanisms. Proper communication balancing

procedures and packets sent to destination through optimal alternative paths from the beginning, keep congestion under minimum values. Under 600 Mbps/node injection, PR-DRB uses progressively the maximum number of alternative paths to deliver messages. For repetitive traffic pattern situations, maximum path expansion is directly done. By avoiding intermediate path expansion, unnecessary ACK messages are not generated and processed by source nodes and intermediate routers. With a maximum number of 4 alternative paths for these experiments, our proposal performs a remarkable lower latency than DRB.

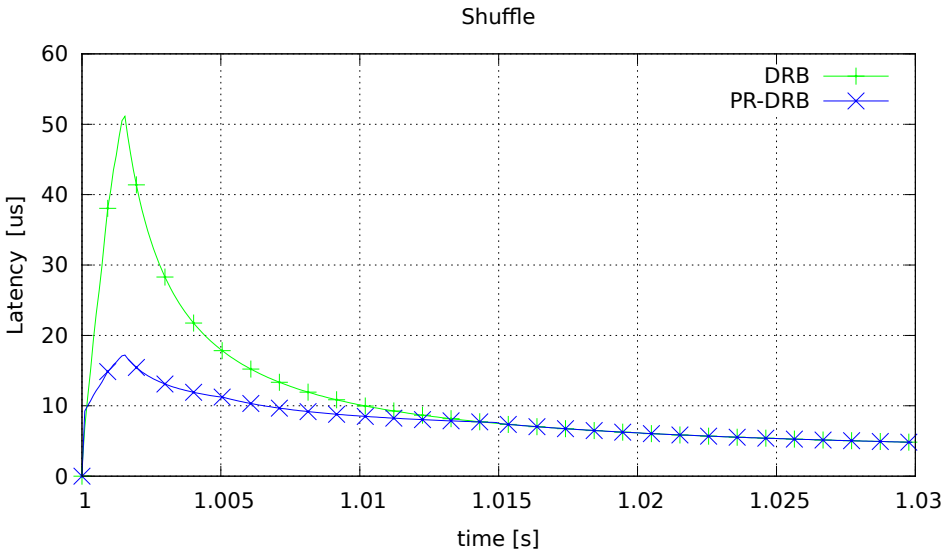


Figure 4.13: Fat tree - Shuffle 32 nodes. 400 Mbps/node

Fig. 4.15 and 4.16 show the network performance under Bit Reversal traffic pattern with traffic load from 400 to 600 Mbps/node, also for 32 communicating nodes respectively. It can be observed that PR-DRB achieves lower latencies than DRB. Under this traffic pattern, in the absence of a mechanism to save and use the information about the path opening procedures, the highest latency peak reaches more than 60 ns of latency. By using PR-DRB, hence by learning from the controlled path expansion procedures of DRB, we can achieve a latency reduction of around 23% for 400 Mbps and 18% for 600 Mbps. From time 1.01 onward, both algorithms tend to stabilize and remain stable until the end of the simulation. This behavior is expected, because one of the main goal of PR-DRB is to reduce the latency during the transitory state. This state is defined by the fact that the DRB policy is constantly adapting itself (dynamically) to the network status. While in

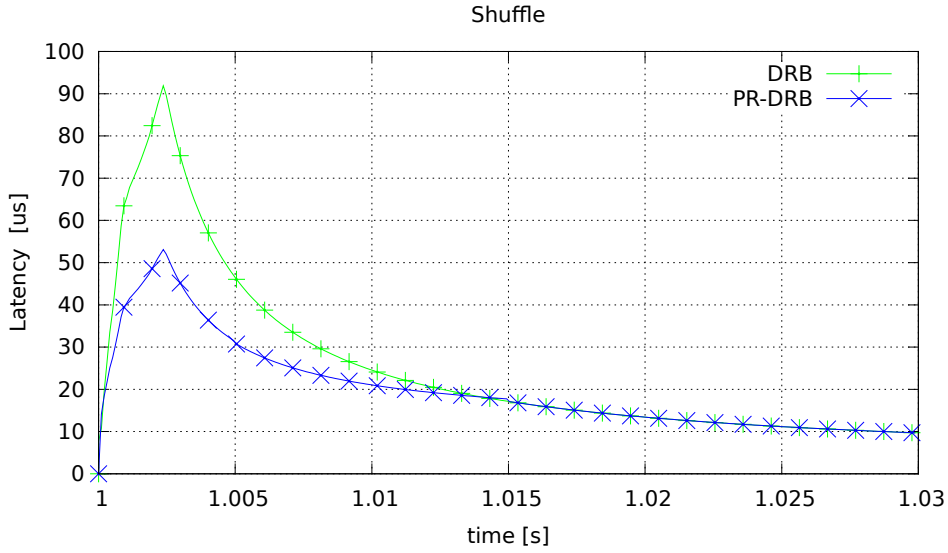


Figure 4.14: Fat tree - Shuffle 32 nodes. 600 Mbps/node

this transitory state, DRB needs to open and close a number of paths in order to find the right combination that stabilizes the network latency.

Figs. 4.17 and 4.18 show latency metric for a fat tree topology with 64 communicating nodes. Figs. 4.17 shows a latency reduction of 31% for Matrix traffic pattern. For the 600 Mbps configuration tested in Fig. 4.18, higher traffic load is injected into the network and latency remains bounded. Latency is considerable reduced here, around 40%, compared to DRB. PR-DRB uses less network resources for a given load, because those resources are efficiently handled. Additional complementary set of results; for example Matrix Transpose with 32 nodes, Bit Reversal and Shuffle for 64 nodes; for this section are shown in the Appendix A, subsection A.2.

4.7 Parallel Applications' Analysis Technique

The analysis of applications communication performance suggests a well defined procedure for estimating the suitability of a given network architecture/topology for a parallel application. This procedure is based on [4] but adapted to communication performance in mind. This method can be useful in many scenarios: (1)The end user can can choose

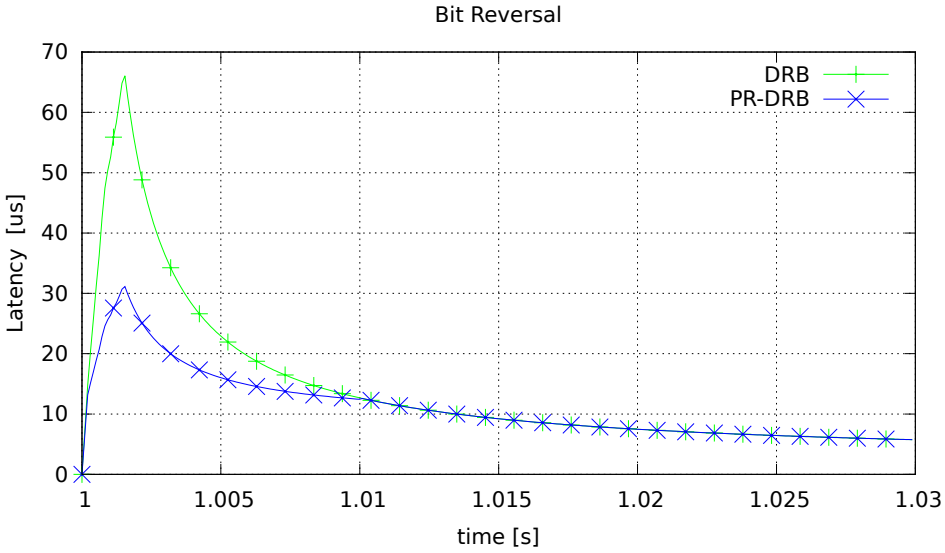


Figure 4.15: Fat tree - Bit Reversal 32 nodes. 400 Mbps/node

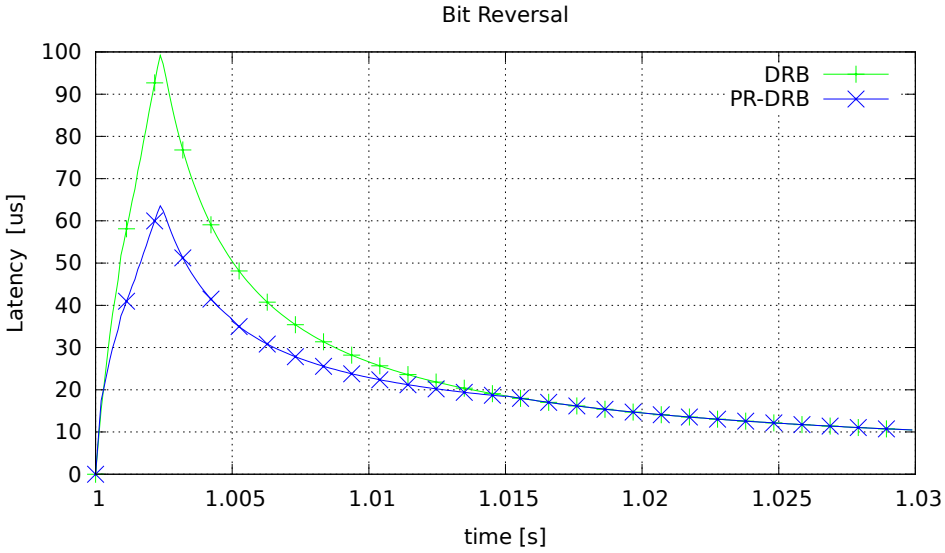


Figure 4.16: Fat tree - Bit Reversal 32 nodes. 600 Mbps/node

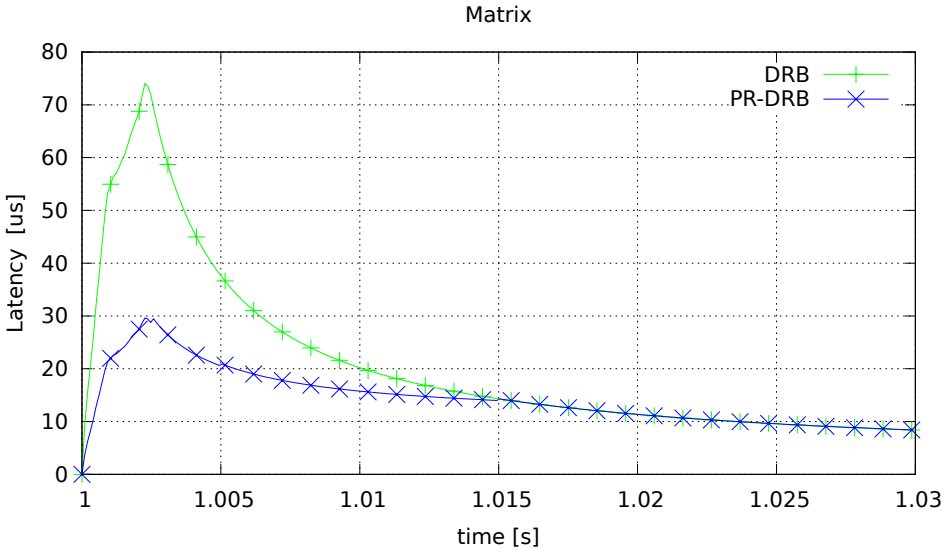


Figure 4.17: Fat tree - Matrix Transpose 64 nodes. 400 Mbps/node

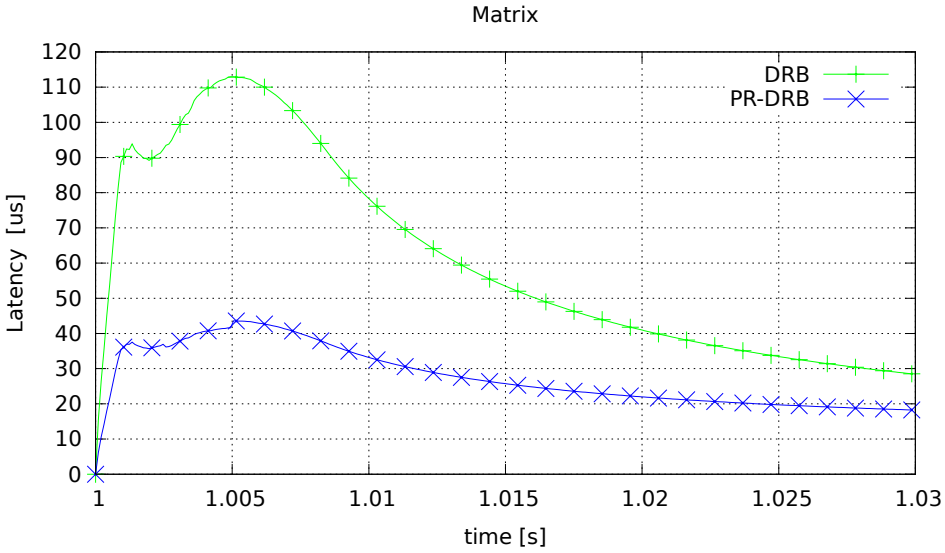


Figure 4.18: Fat tree - Matrix Transpose 64 nodes. 600 Mbps/node

specific runtime features based on this analysis; (2) it can help managers make the correct budget decisions based on the type of applications and resources really needed, and (3) application developers can analyze the performance of applications on different platforms in a systematic manner. We now outline the steps involved in this analysis technique.

4.7.1 Obtain Information from the Application and the Architecture

We first need to identify various application characteristics which affect performance. With network performance in mind it is useful to measure the communication requirements of the application. The network topology and routers/switches in question also needs to be carefully analyzed. It is important to take into account factors such as internal bandwidth available, network speed and latency. In fig. 4.19 we can see the procedure to get all these information about one application. A trace file is then obtained from an application execution. Later, each node in the network will read an input trace file and will simulate the events (for example ***MPI_Wait***, ***MPI_Send***, ***MPI_Receive***,***MPI_Broadcast***). Every event has a ***Compute (t)*** event, which emulates a serial computation of duration *t*.

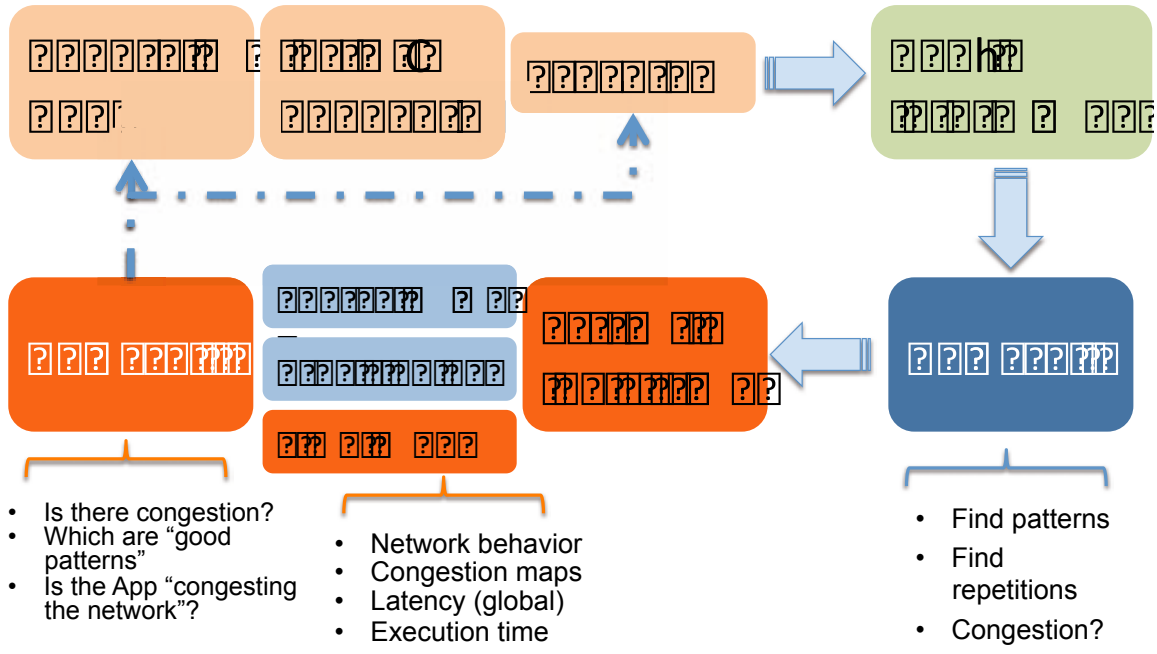


Figure 4.19: Application characterization framework

4.7.2 Determine Communication Characteristics

Based on the application specifications or execution of a parallel application signature on an available machine, one should be able to get a sense of whether the application is communication-bound. One way to do this is by executing the application with the PAS2P tool[64] to identify relevant phases of the application.Later, only those relevant phases could be executed and analyzed. Other way would be to build a histogram of message sizes. A large number of messages over time, or frequent collective communication operations, may also indicate that the application will be hindered by network contention on the target machine. Based on the message size histogram and a rough estimate of whether contention will be a problem, we can use bandwidth and latency graphs on the target machine.

4.7.3 Benchmark the Application on the Real Machine

If access to the machine is available, we can obtain additional and more reliable information about the application and communications performance and do a more concrete analysis. Again, PAS2P tool gives a set of useful information. The capability of an application to overlap communication with computation and the tendency to create network contention are important factors affecting communication performance. Although this can be deduced through actual runs on a certain number of processors on the real machine, the tool has the option of executing the application on a smaller cluster footprint and yet get the desired information. The grain size of the application is another important consideration; it can be used to estimate the effect of noise on application performance.

4.8 Application's Performance with PR-DRB

In order to evaluate the Application-Aware PR-DRB (just PR-DRB from now on), a simulation approach was chosen. This section shows the results with real parallel scientific applications. In [10], we presented the evaluations with synthetic traffic for mesh and fat tree topologies. We use NAS Parallel Benchmarks [1], Lammps [51] and Parallel Ocean Program (POP) [29] for our evaluation. Latency and global application execution time are evaluated in order to assess PR-DRB. Latency is the time elapsed since a packet is created until it reaches its destination, in seconds. Execution time is considered for each one of the policy under evaluation. Tests were conducted for 64 nodes under fat tree topology.

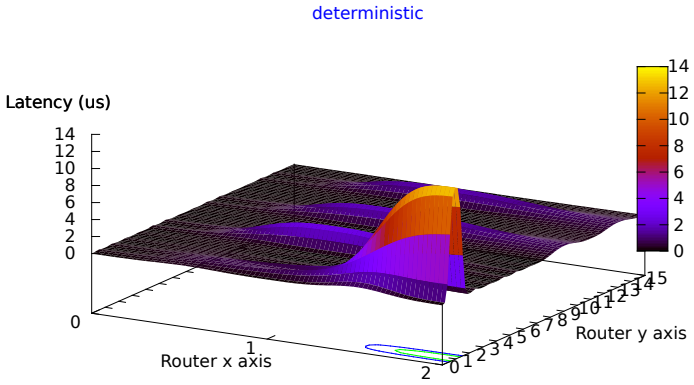
4.8.1 Modeling Environment

We have assumed virtual Cut-through flow control [17]. Link Bandwidth was set to 2Gbps, packet size was set to 1024 bits and the size of routers buffers was 2MB.

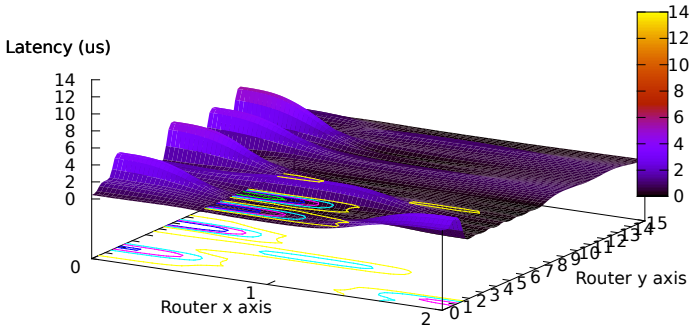
4.8.2 NAS Parallel Benchmark

For NAS Parallel Benchmarks, we focus on the LU pseudo application and MG kernel, which uses long- and short-distance communication. We use classes S, A and B problem size for evaluation. Although not included here, we have not seen performance degradation for rest of the NAS Parallel Benchmarks using DRB and PR-DRB. Fig. 4.20 shows average latency map for the fat tree network after the execution of the whole LU Class A application. Figs. 4.20a, 4.20b and 4.20c show the results for Deterministic, DRB and PR-DRB respectively. Latency surface represents the average contention latency at buffers. We see an improvement of 57% on average at the highest peak in the map, between the Deterministic and DRB algorithms. We can also see that DRB concentrates all the traffic into the routers closest to the sources (index 0 of Routers x Axis). This is due to alternative paths opening procedures used by DRB. PR-DRB on the other hand achieves 41% latency reduction compared to DRB and 75% compared to the Deterministic algorithm. Here we can see that PR-DRB has re applied the best solutions directly and avoided the contention introduced by DRB previously.

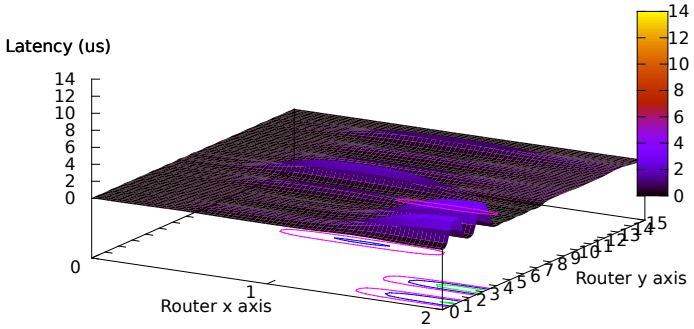
In fig. 4.21a we can see the global network latency results for the MG benchmark classes S, A and B. For class S we do not see any improvement, since the contention in the network is negligible. However, for classes A and B we can see a 65% and 60% in latency reduction respectively, comparing the Deterministic and DRB algorithms. Even though DRB and PR-DRB shows similar final latency values, we can see in figs. 4.22 and 4.23 that contention latency is reduced by PR-DRB. In 4.22a, between time 1.005 and 1.0125 both DRB and PR-DRB have the same latency. This represent the phases where PR-DRB is learning or monitoring the patterns currently in the network. From time 1.0125 until the end of the simulation, PR-DRB has lower latency values. This is caused by the fact that PR-DRB has applied its best known solution for one or more communication pattern. Now the traffic is redirected to others paths, thus the traffic is reduced in this router. In fig. 4.22b, between time 1.0225 and 1.03 we can see that PR-DRB has higher latency value than DRB, but from 1.03 it keeps latency bounded below DRB s value. In this case PR-DRB has applied a solution it considers good, but the traffic pattern has changed and now its solution is not longer needed. Here, PR-DRB detaches from work and lets DRB find a good solution for this pattern. Figs. 4.23a and 4.23b show similar results for other



(a) NAS LU Deterministic

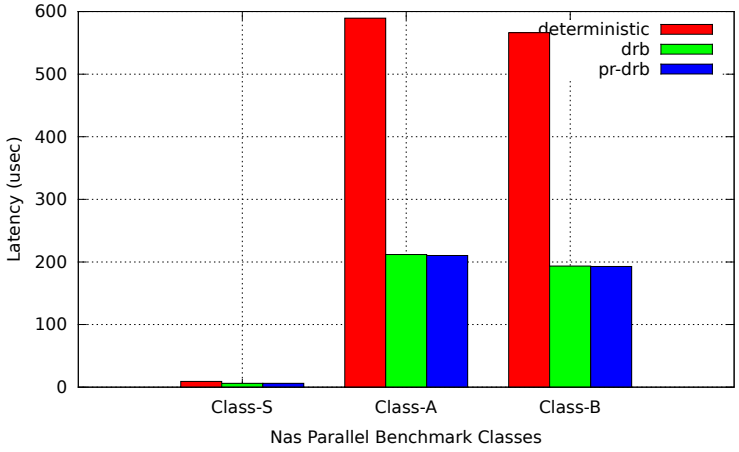


(b) NAS LU DRB

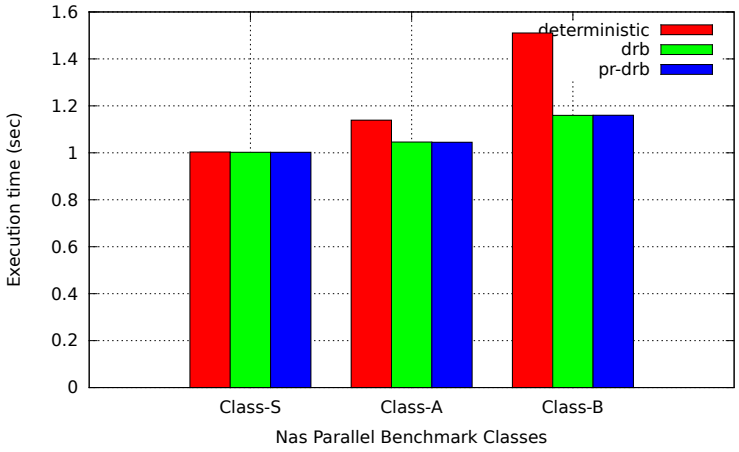


(c) NAS LU PR-DRB

Figure 4.20: NAS LU latency map.



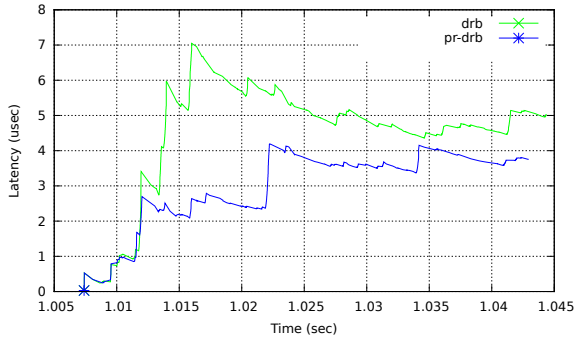
(a) NAS MG latency



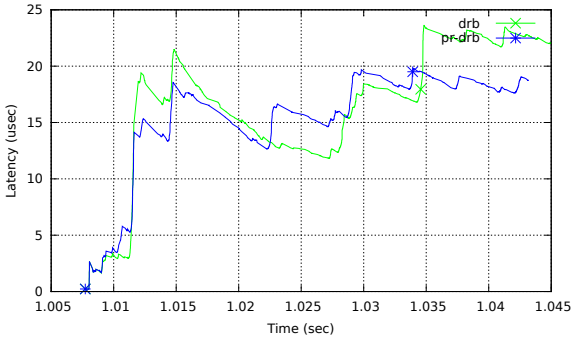
(b) NAS MG execution time

Figure 4.21: NAS MG Global latency & execution time.

congested routers in the network. Regarding the execution time, as shown in fig. 4.21b, DRB and PR-DRB outperform the Deterministic in 8% for class A and 23% for class B.

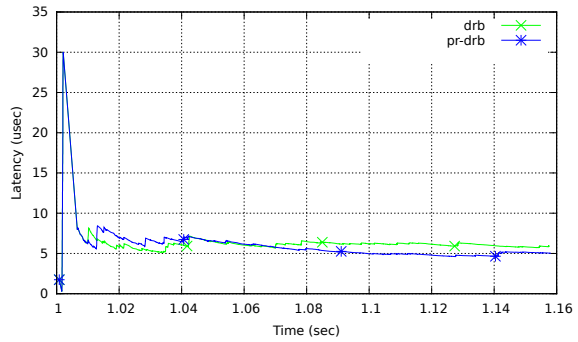


(a) NAS MG class-A router A;

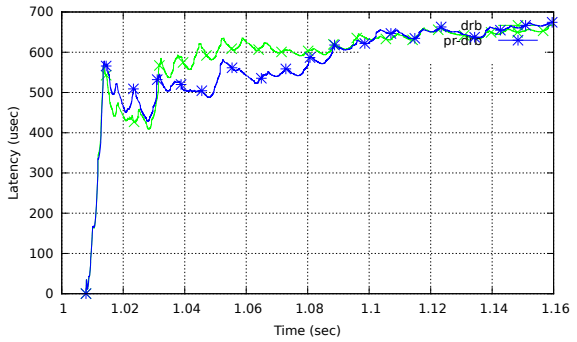


(b) NAS MG class-A router B

Figure 4.22: Contention latency of NAS MG routers.



(a) NAS MG class-B router A;



(b) NAS MG class-B router B

Figure 4.23: Contention latency of NAS MG routers.

4.8.3 Lammp Molecular Dynamics Application - Latency Map

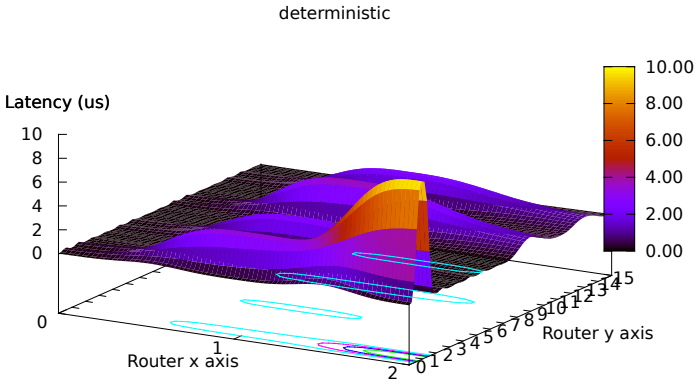
Here we show the results for the application LAMMPS Molecular Dynamics Simulator . In the map shown in fig. 4.24 the average latency with DRB algorithm (Fig. 4.24b) is reduced by 65% compared to the Deterministic algorithm (Fig. 4.24a). PR-DRB (Fig. 4.24c) has a similar reduction than DRB in latency compared to the Deterministic algorithm. However, PR-DRB has better global latency values compared to both Deterministic and DRB, as shown in fig. 4.25a. Here, latency is reduced by PR-DRB on 5% compared to DRB and 36% compared to the Deterministic algorithm. PR-DRB achieves this improvement by using the best known solution every time the same pattern re-appears in the network. Fig. 4.25b shows the execution time reduction obtained by PR-DRB compared to DRB and Deterministic algorithms. In this case, 6% and 37% respectively. The improvements

achieved, in latency and execution time is caused by using the best solutions for each representative pattern it finds. The best solution comprises a combination of shorter and longer paths towards a destination. Thus, global latency is slightly reduced by contention latency reduction in buffers, as shown in fig. 4.26a. In fig. 4.26b PR-DRB found 80 different contending flows patterns during the first stage of the application. Later, 7 patterns were identified or repeated again. One of those patterns was repeated 279 times, and the best solution encountered and saved previously was applied every time. Here we can see that with this pattern recognition procedure, PR-DRB reaches better latency and execution time than DRB and the Deterministic algorithms.

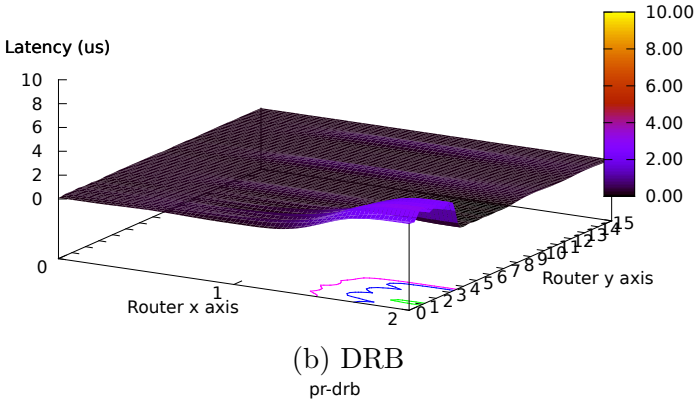
4.8.4 Parallel Ocean Program (POP) Application

Here we show the results for the application Parallel Ocean Program (POP) . For this evaluation, we have included the *random routing* as well as the *cyclic periodic routing* to compare against these general algorithms. Also, we have included another variant of the original DRB, called the *Fast Response DRB* [38] in order to assess our method. Our proposal is designed as a modular implementation, so we intend to show that we can apply our predictive policy to other algorithms also derived from the original DRB. We can see in fig.4.27 the global average latency and the execution time of the POP application. For the latency values, we can see that the Deterministic, and the Cyclic reaches 16 μseconds of average latency while the Random algorithm reaches 14 μseconds. Compared to these three values, PR-DRB outperforms them by 38%. If we have into consideration the *Fast Response pr-drb*, then the reduction goes up to a 57% of the latency for the worst case algorithm, which for this experiment is the Deterministic algorithm. Once again, the capability of PR-DRB to distribute the traffic is a key factor in order to reduce the latency value. Latency is considerably reduced between the standard PR-DRB and its predictive Fast Response variation. In this case, about 31%.

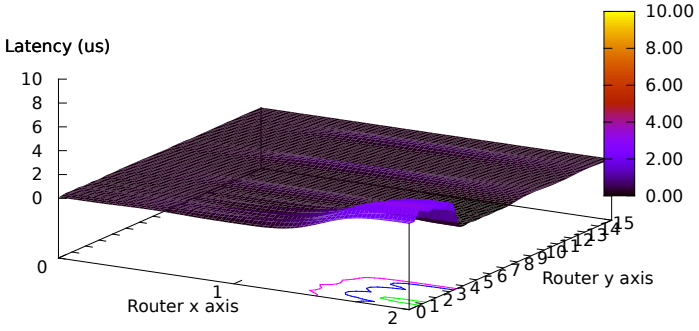
If we analyze the global average latency between the DRB and PR-DRB, and also between the FR-DRB and the predictive FR-DRB, the reduction obtained in both cases for the predictive approaches are of about 2%. In order to fully understand this last behavior, we should look at the contention latency suffered by router s internal buffers, as shown in fig. 4.28. We can see in fig. 4.28a that the DRB algorithm has higher contention latency values from time 1.03 onward, until the end of the simulation. The PR-DRB in this case outperforms the DRB because the it is capable of permanently re apply its best known solutions. For this router PR-DRB has found 143 different contending flows patterns. Later 40 patterns were repeated, including all of the 64 nodes. The communication patterns re appeared 50 times throughout the execution of the application. Regarding the graph seen



(a) Deterministic

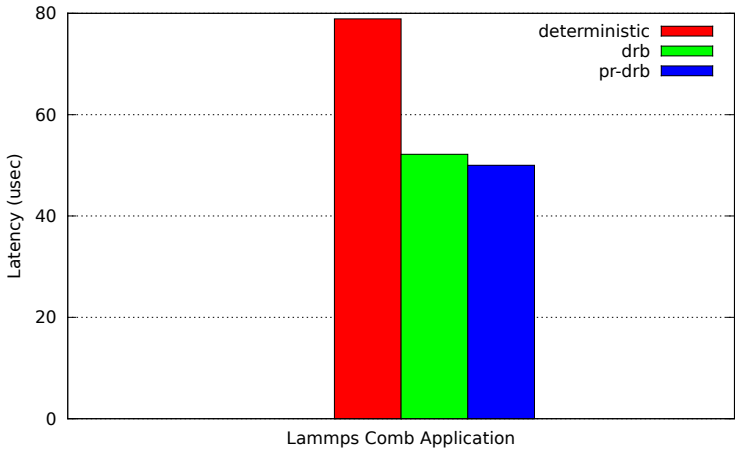


(b) DRB

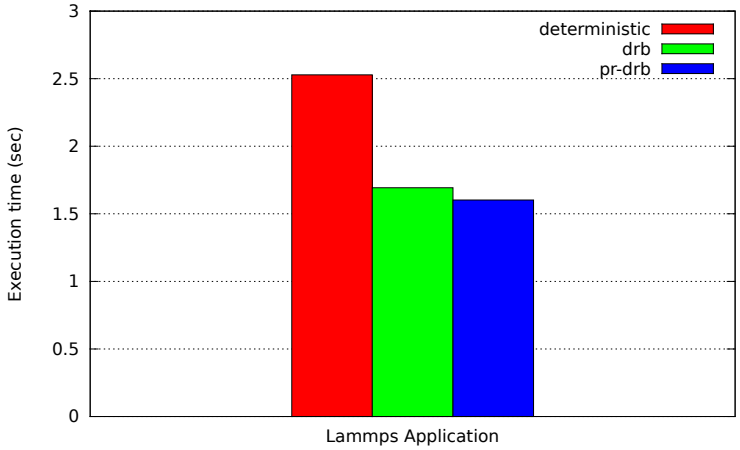


(c) PR-DRB

Figure 4.24: Lammps latency map.

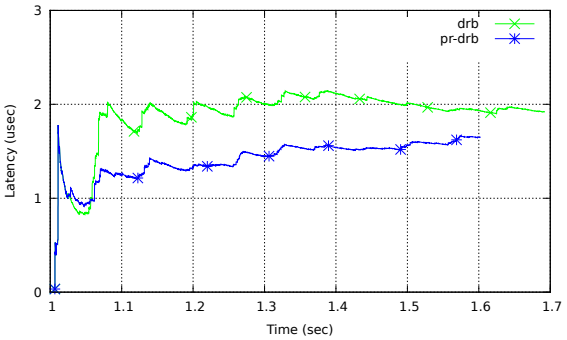


(a) Lammps latency

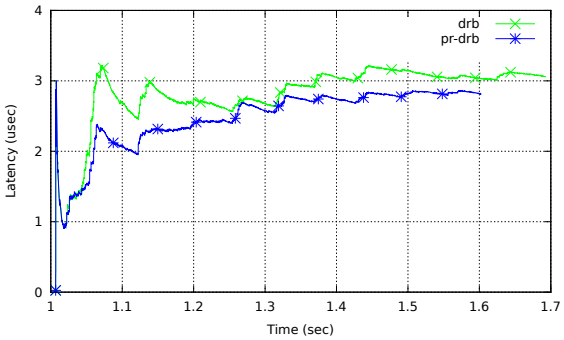


(b) Lammps execution time

Figure 4.25: Lammps global latency & execution time.



(a) Lammps Router A contention latency;



(b) Lammps Router B contention latency

Figure 4.26: Contention latency of lammps routers.

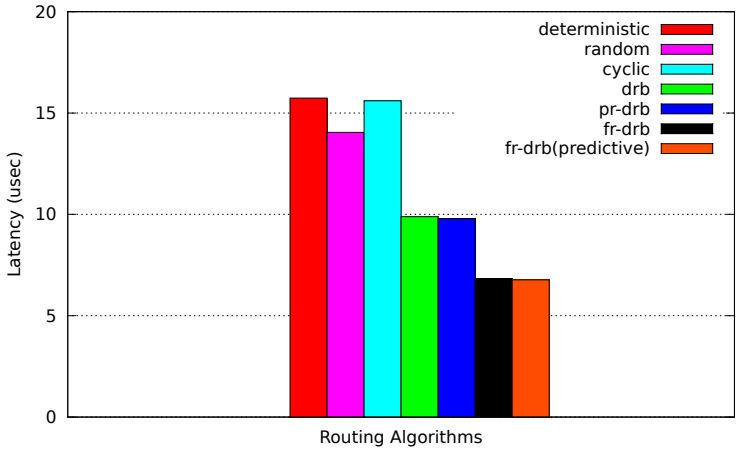
in fig. 4.28c, we can mention that PR-DRB found 160 different phases at this intermediate router. Then, with this information propagated or notified to different sources, in total PR-DRB found again 69 communication patterns. These 69 patterns, were repeated 87 times and the best solution found when they first appeared were re applied by PR-DRB. The contention latency shown in figs. 4.28b and 4.28d corresponds to the routers in the first level of the tree (nearest to the sources nodes), so they do not have any contending flows registered throughout the simulation. The gain obtained is because the common ancestors (nearest to the roots) have found the contending flows and after notifying this information to the sources, they managed the paths available using the right combination of paths that PR-DRB found in order to keep latency bounded. Result from other routers for this applications are shown in the appendix in section A.3.1.

Regarding the execution time as shown in fig 4.27b, the DRB family on average outperforms the Deterministic, Cyclic and Random algorithms on 27%. Each of the predictive approach, *PR-DRB* and *FR-DRB predictive*, outperform their non predictive approach by 2%. The real gain here is shown in message latency, contention latency and overall network performance explained above.

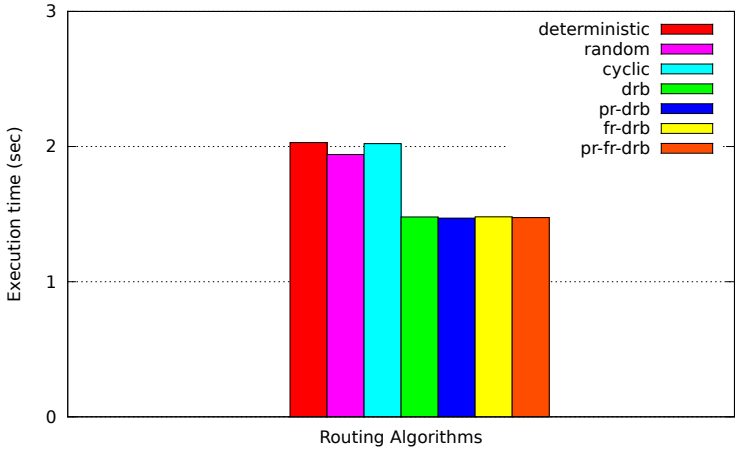
In the map shown in fig. 4.29 we can see the latency map for the Deterministic, Cyclic Priority and Random routing algorithm. Here, the occupation latency of Deterministic reaches the highest value of the three. Although, the Cyclic Priority algorithm uses more resources at the top of the tree (near the roots), as we can see in figs. fig:eval-pop-64-50-steps-map-cyclic.

By comparing these three algorithms against the PR-DRB, our results show that contention latency is reduced by 87% compared to the Cyclic algorithm and Deterministic, and by 50% compared to the Random algorithm. Here we have included the *Fast Response DRB - FR-DRB* in order to test the capability of our proposal against other DRB based algorithm. Recall that the PR-DRB requires an ACK message in order to start the path opening procedures. PR-DRB uses the *router based* or *destination based* to accomplish this task. On the other hand, the *FR-DRB* has a watchdog timer which expires after a time t with no ACK received. This is a sign that congestion is present in the network. This expiration does not require the use of an ACK, at least to start the opening procedures. PR-DRB is built in a modular fashion on top of DRB, and because FR-DRB is also a DRB s descendant, then by applying or policy to *FR-DRB* we could improve it. As we can see in fig. 4.30b latency map is greatly reduced by *FR-DRB*, compared to the original PR-DRB, to nearly 10% less contention latency in the router. Our proposal after applied to *FR-DRB* also shows an improvement of 5%. This means that better usage of the network is obtained, and that our policy could be positively adapted to work with any

current of future DRB implementation.



(a) POP latency

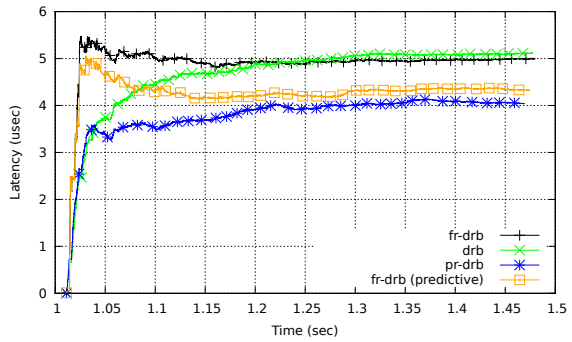


(b) POP execution time

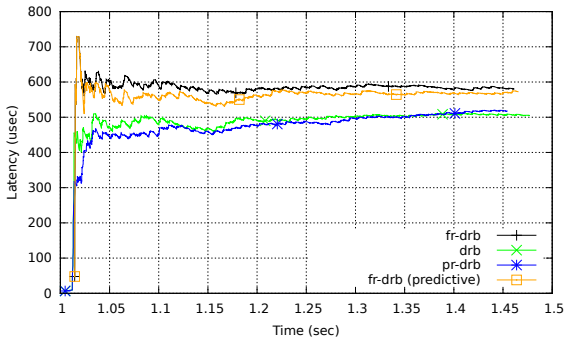
Figure 4.27: POP global latency & execution time.

4.8.5 Discussion

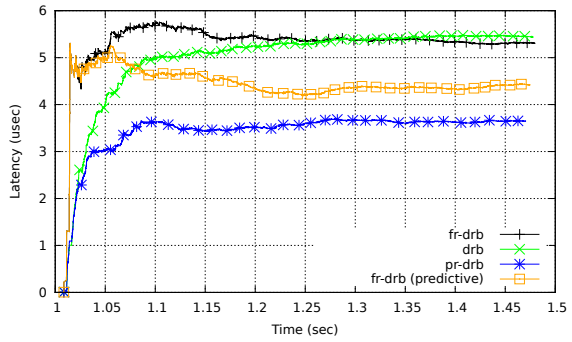
We can mention that our proposal, PR-DRB, is strong because it can *dynamically* learn from the traffic pattern currently in the network and react based on that knowledge. The contribution of this work ranges from the study of parallel applications and their communication patterns, to the effect they have into the interconnection network. PR-DRB detaches from the overhead of extra information about one specific communication pattern into the routers and processing nodes. We can also infer from our evaluation results that our proposal does not degrade performance even under extreme situations, and the performance of the whole system is maintained even with a smaller network footprint.



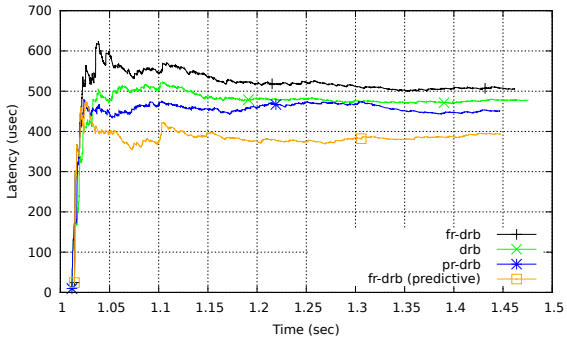
(a) POP Router A contention latency;



(b) POP Router B contention latency



(c) POP Router C contention latency

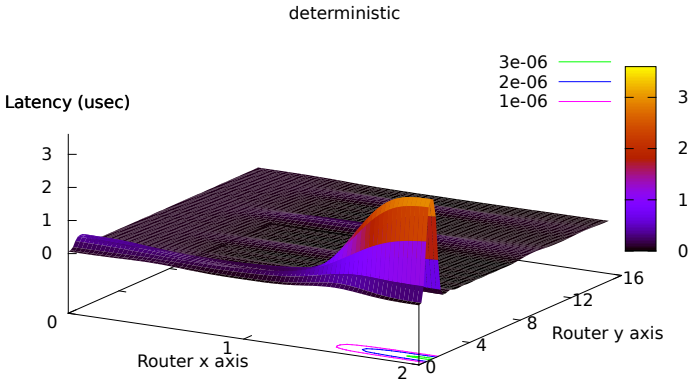


(d) POP Router D contention latency

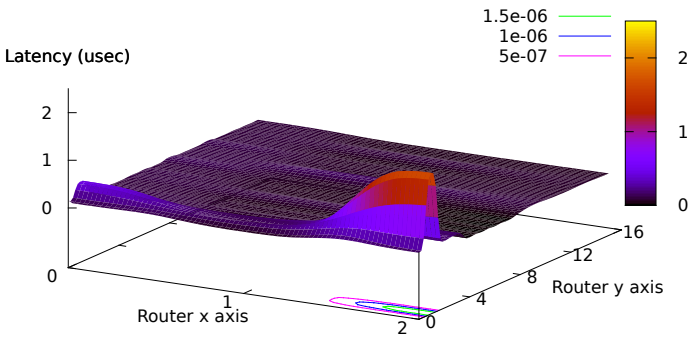
Figure 4.28: Contention latency of POP routers.

Acknowledgments

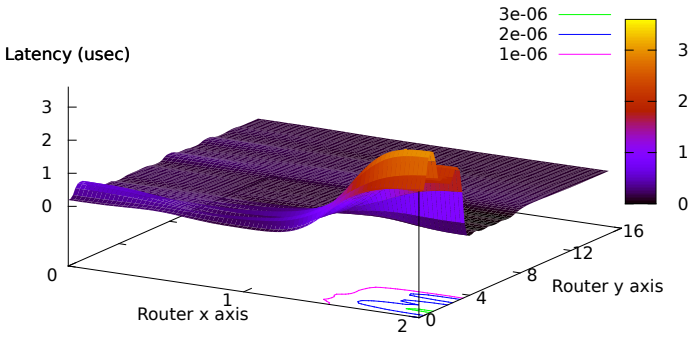
We would like to thank *OPNET Technologies, Inc.* for providing us the *OPNET Modeler* licenses to perform the experimental evaluation of this thesis and the previous work [5], [9], [6], [7], [45], [8], [10], [11].



(a) Deterministic

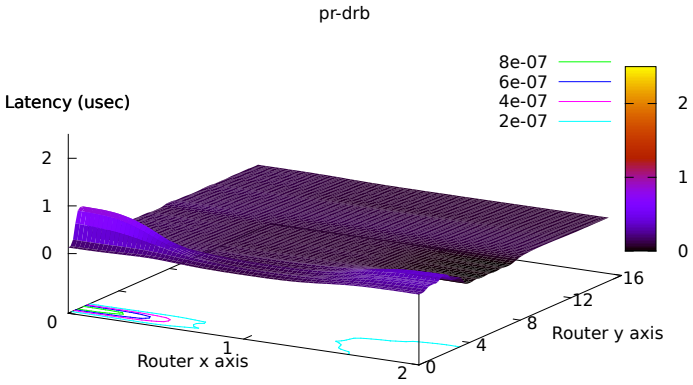


(b) Random

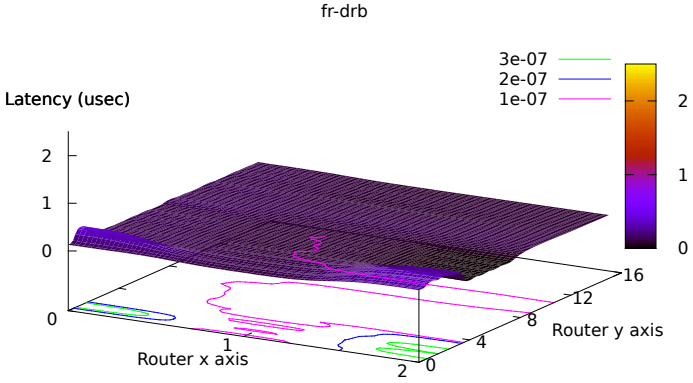


(c) Cyclic Priority

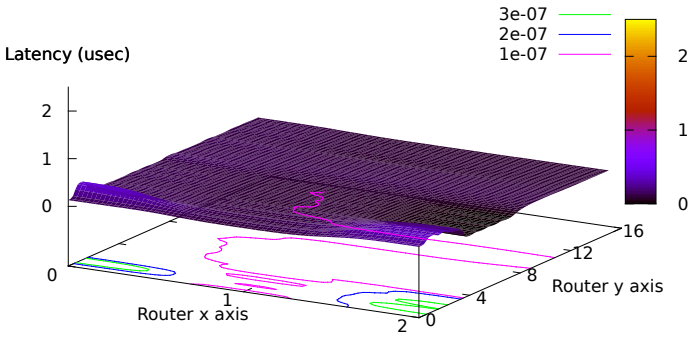
Figure 4.29: POP latency map for non DRBs.



(a) PR-DRB



(b) FR-DRB



(c) FR-DRB Predictive

Figure 4.30: POP latency map for DRBs.

Chapter 5

Conclusions

5.1 Final Conclusions

This thesis, as any work covering several aspects within a certain field of science, is intended to be complete and entirely closed. However, this research also gives rise to a wide range of affordable open lines and further work. These open lines are described below, grouped according to the contribution from which they are originated.

Predictive and Distributed Routing Balancing - (PR-DRB)

We have proposed the Predictive and Distributed Routing Balancing PR-DRB. This strategy uses alternative paths under congestion situation to reduce latency and to increase bandwidth availability, by considering time as well as traffic dynamic behavior constraints. Routing algorithms try to adapt parallel applications traffic load to network topology. These applications, that run on an HSIN, change along time and possess repetitive behavior, and PR-DRB is capable to learn from them and save information for later use.

PR-DRB has been developed to fulfill HSIN design objectives such as all-to-all connection, and low and uniform latency between any pair of nodes under any message traffic load. The proposed method is also in line with current approaches used in commercial interconnects (as InfiniBand). Our policy allows heavier communication load in the network, or in cost-bounded data centers it allows using less network components, because they are more efficiently handled.

The evaluation performed to validate PR-DRB has revealed very good improvements in latency. Saturation is reduced allowing the use of the network at higher loads. We have

shown that PR-DRB is a fast and robust method with a very low overhead. Additionally, PR-DRB is useful for permutation and bursty communication patterns, which are commonly created by parallel applications and can produce the worst hot-spot situations.

Methodology to Extract Parallel Applications Information

We have proposed a framework to study and analyze the behavior of parallel scientific applications. The methodology has allowed us to extract the logical trace of parallel applications. This information was the source for many subsequent analysis. With this information, important features of applications could be analyzed, such as their communication patterns, their degree of repetitiveness, the topological information of communications and communications requirements of parallel applications in general. The information extraction was non-intrusive, therefore allowing us precise data from the parallel applications.

Original Distributed Routing Balancing (DRB) Extended

In our work we have also worked on improvements of the original DRB routing algorithm. We have modified and extended the DRB in order to include logical applications traces into its injections capabilities. The DRB algorithm was widely tested against synthetic traffic, and the only application traces used previously were physical traces. By using physical traces we could only simulate one scenario, because of the limitations of the information extracted itself. It lacks the vital information about the logical dependencies of communications. To tackle this situation, a combination of the information extracted from our proposed framework and the modifications to the original DRB were used. By proceeding this way, we could successfully tested the DRB with real applications traces and analyzed its behavior.

5.2 Further Work and Open Lines

This work was intended to be as complete as possible and also entirely closed. After being working with this for the last four years, a set of related work appears and could be treated

as open lines and future work. The description of the open lines originated from this work are mentioned below.

Predictive and Distributed Routing Balancing - (PR-DRB)

One extension to PR-DRB has to do with congestion detection time. Actually PR-DRB waits until congestion reappears, in order to start the predictive module. To speed up this phase, latency trend could be used. With enough historic latency values and traffic information, PR-DRB could predict future congestion before it actually arises. This trend analysis could greatly improve system performance.

An additional point could be the information obtained from parallel applications. PR-DRB routers could have *offline* meta-information about the communication patterns and communication requirements. This information could help leverage the predictive phases, specifically the monitoring & detection phase. Recall that the methodology proposed in this work, the *PR-DRB* is completely dynamic, and has the ability to learn from communication patterns actually running in network buffers.

This decision could have a positive impact in communication performance as well as for the application execution time. The rationale behind this performance gain is based on the idea of the availability of extra information. This information would help the routing module to decide faster, notify sooner and apply best solutions smarter.

PR-DRB Models

Up until now we were focused on network behavior simulations. Our models were specifically designed and implemented with this idea in mind. By studying the network performance and the relation with parallel applications, we found that other topics could also be analyzed. Among the topics that could get some benefits we could mention:

- Provisioning: The models could be useful to start thinking about dedicating some specific portions of the network to one application, based specifically on its communication requirements. This could be useful to predict and accommodate several applications into the network without disturbing each other.
- Energy-Aware routing: With the idea of the predictive module in the routing unit, we could use the knowledge of future communications patterns to start applying energy-aware policies.

Appendices

Appendix A

Appendix

In this appendix, we include additional information from specific chapters. For example, we can find the algorithms that describes our methodology in chapter 3. Also, the results of the evaluation presented in chapter 4 are included here.

A.1 Algorithms

```
1  MonitorPathLatency(Packet P, Threshold)
2
3  /* At each PR-DBR Router */
4    for each step of P do
5  /* contention latency = wait time of the packet
6     in buffer's output port */
7    read contention latency
8    if (contention latency > threshold) then
9      Identify traffic patter involved in congestion situation
10     Record the traffic pattern
11   end if
12
13 /* Accumulate contention latency,
14    to get the path latency afterwards */
15     Accumulate latency (queue time)
16     Continue to next router or to final destination
17   end for
18
19 /* At the destination node */
20    Send the path latency and contending flows information ACK
21    to the source
22
23 end MonitorPathLatency
```

Algorithm A.1: Monitoring and Notification of path latency and contending flows (destination based)

```
1  MetapathConfig(metapath Mp, Thresholds )
2  /*  Executed at source node every time a new ACK message arrives
3      The metapath info is maintained for every
4      source destination pair
5   */
6
7    Receive ACK message
8  /*  ACK message = [latency info + contending flows] */
9    Calculate Mp latency value according to Eq. 3.4
10
11 /*  First, verify if we already have  have a solution */
12   Execute_predictive_procedure() according to 3.2.6
13
14 /*  Second, if we don't have a a saved solution,
15     then start the metapath configuration
16     procedures in order to find a good set
17     of alternative paths.
18  */
19   if (NOT saved solution for source-destination pair) then
20     if (Mp latency > Threshold_High) then
21       Increment the number of alternative MSPs
22     else if (Threshold_Low < Mp latency > Threshold_High) then
23       Maintain the same number of alternative paths
24     else if (Mp latency < Threshold_Low) then
25         Decrease the number of alternative MSPs
26     end if
27   end if
28
29 end MetapathConfig
```

Algorithm A.2: Metapath configuration.

```
1  MultiStepPathSelection()
2  /* At the source node (before injecting an app. message) */
3    Build the probability density function (PDF) of MSPs bandwidths
4    Select the correct MSP using the PDF
5    Inject the application message into the network
6  end selection
```

Algorithm A.3: Multistep path selection.

```
1  MonitorPathLatency(Packet P, Threshold)
2
3  /* At each PR-DBR Router */
4    for each step of P do
5    if (latency > threshold) then
6      Identify traffic patter involved in congestion situation
7      Record the traffic pattern
8      Generate a "Predictive ACK packet"
9    end if
10
11     Accumulate latency (queue time)
12     Inject the "Predictive ACK packet" toward the source node
13   end for
14
15 end MonitorPathLatency
```

Algorithm A.4: Monitoring and Notification of path latency and contending flows (router based)

A.2 Synthetic Traffic Evaluation

A.2.1 Analysis with Permutation Traffic

Average latency

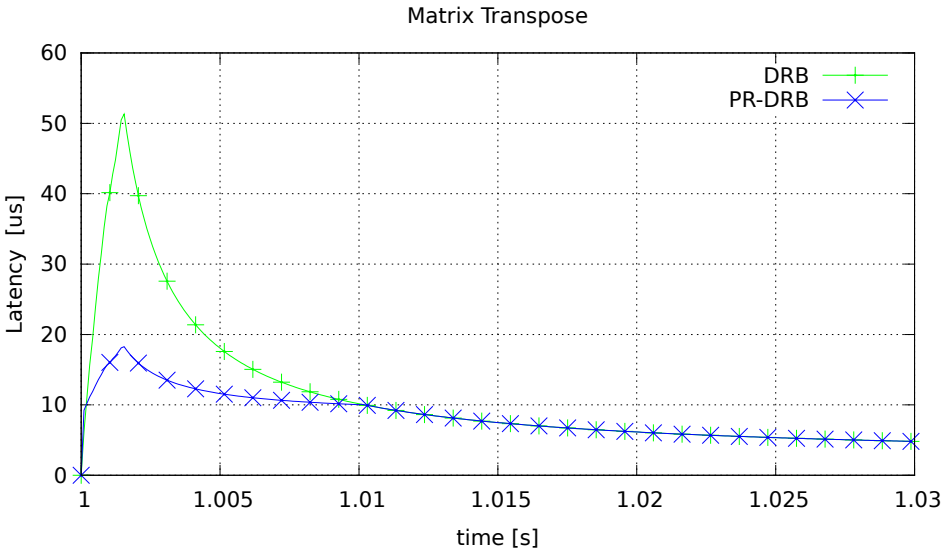


Figure A.1: Fat tree - Matrix Transpose 32 nodes. 400 Mbps/node

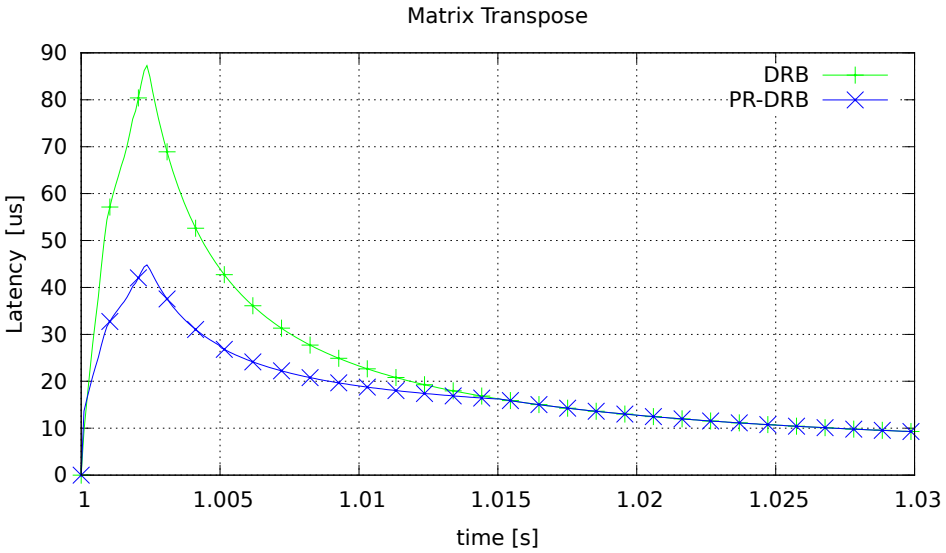


Figure A.2: Fat tree - Matrix Transpose 32 nodes. 600 Mbps/node

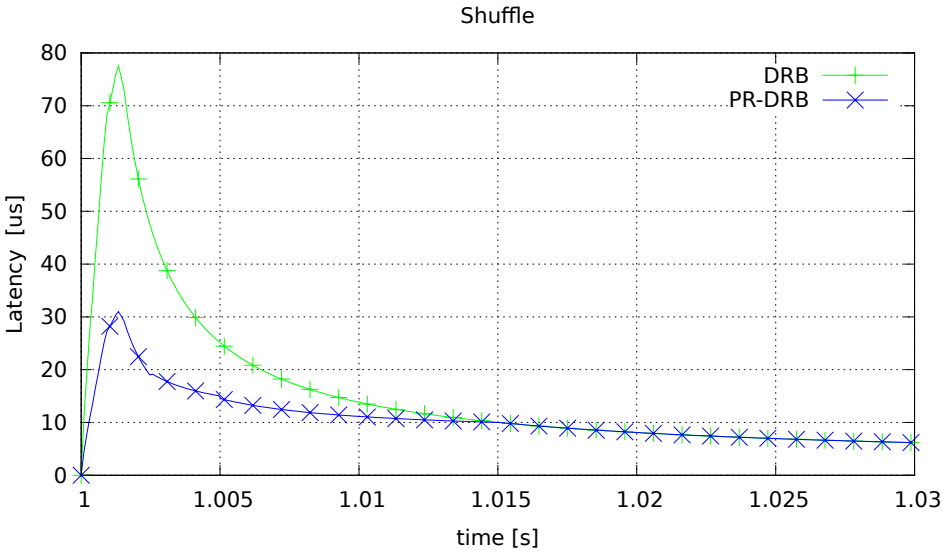


Figure A.3: Fat tree - Shuffle 64 nodes. 400 Mbps/node

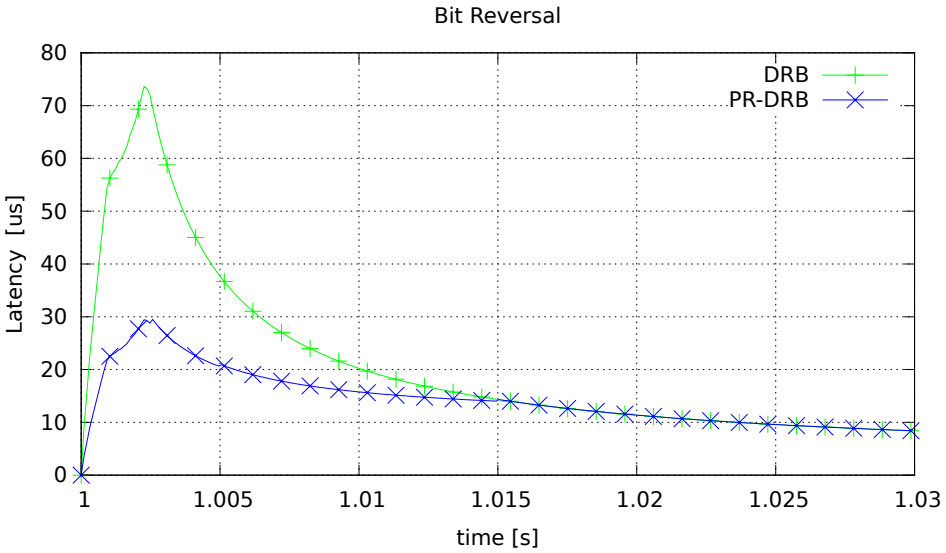
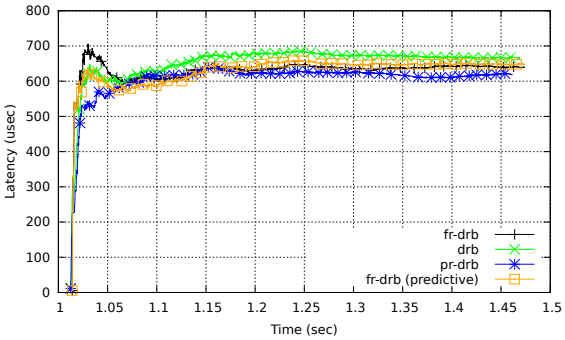


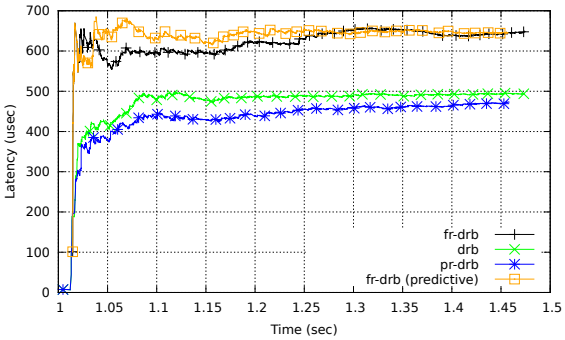
Figure A.4: Fat tree - Bit Reversal 64 nodes. 400 Mbps/node

A.3 Application's Performance with PR-DRB

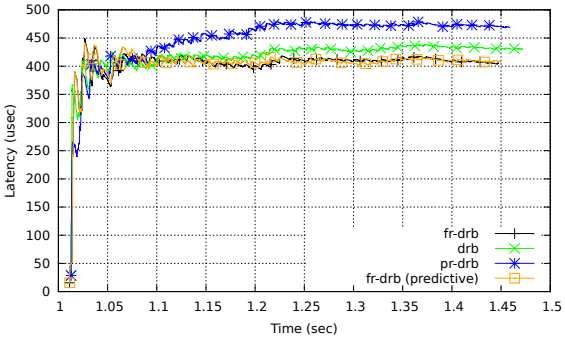
A.3.1 Parallel Ocean Program (POP) Application



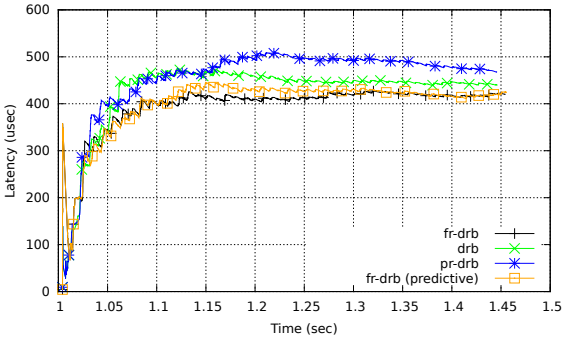
(a) POP Router 48 contention latency;



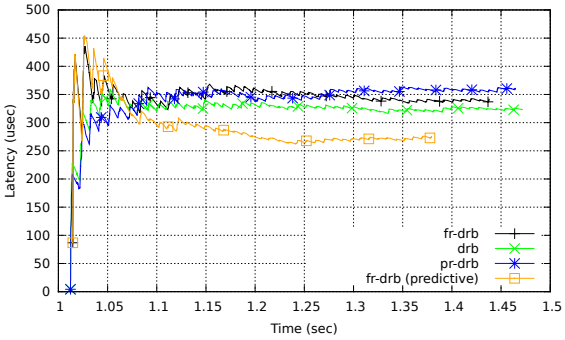
(b) POP Router 50 contention latency



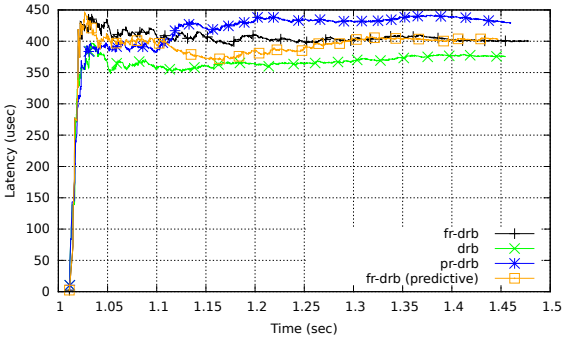
(c) POP Router 51 contention latency



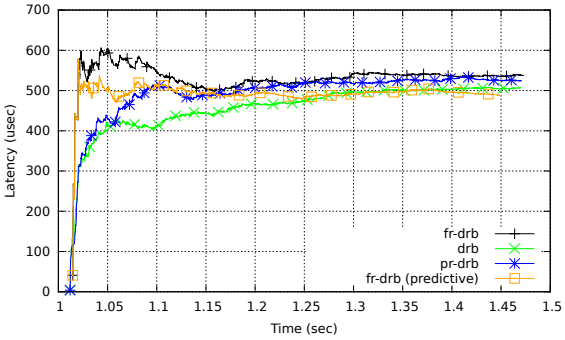
(d) POP Router 53 contention latency



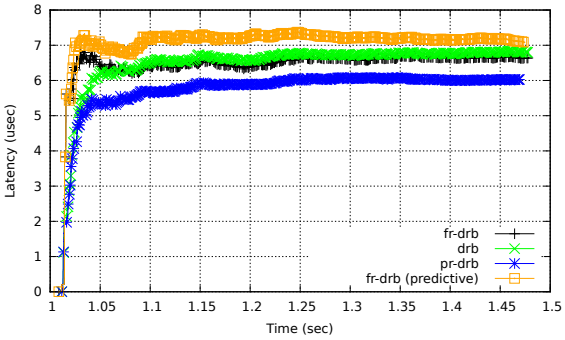
(e) POP Router 59 contention latency



(f) POP Router 60 contention latency

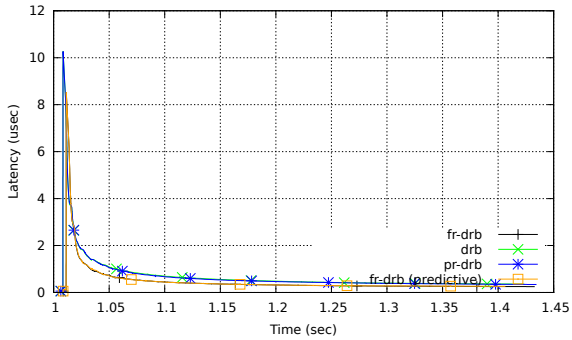


(g) POP Router 64 contention latency

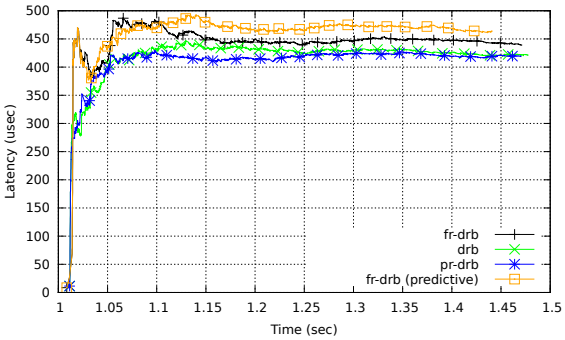


(h) POP Router 65 contention latency

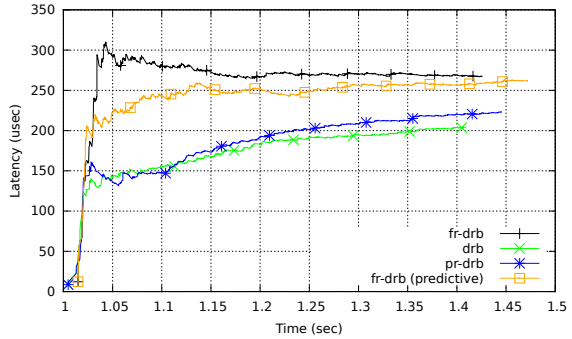
Figure A.5: Contention latency of POP routers (1/3).



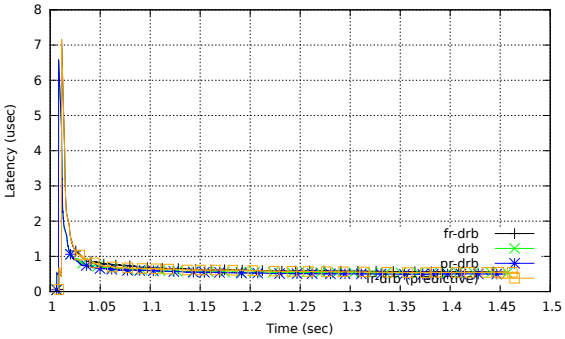
(a) POP Router 149 contention latency;



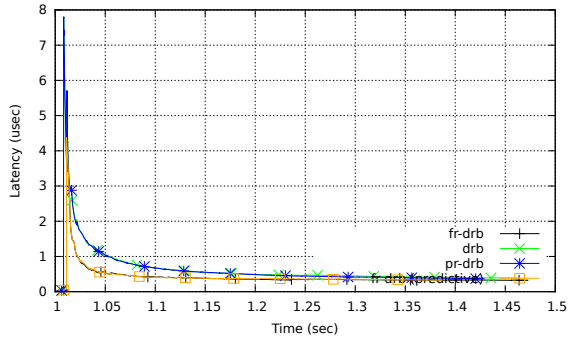
(b) POP Router 68 contention latency



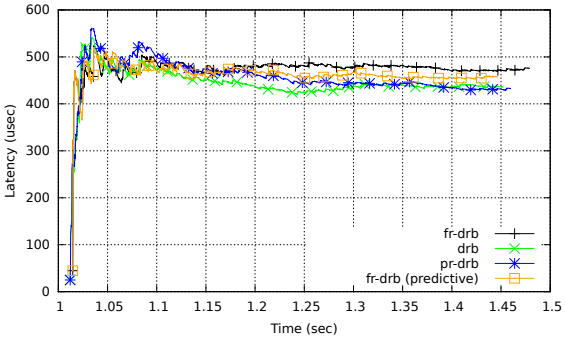
(c) POP Router 70 contention latency



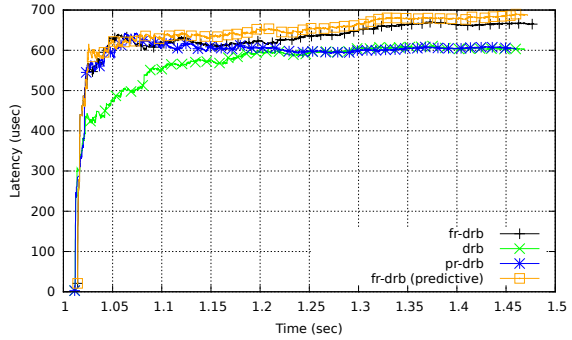
(d) POP Router 80 contention latency



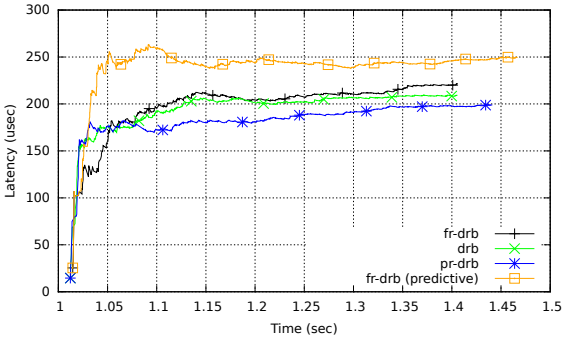
(e) POP Router 93 contention latency



(f) POP Router 108 contention latency

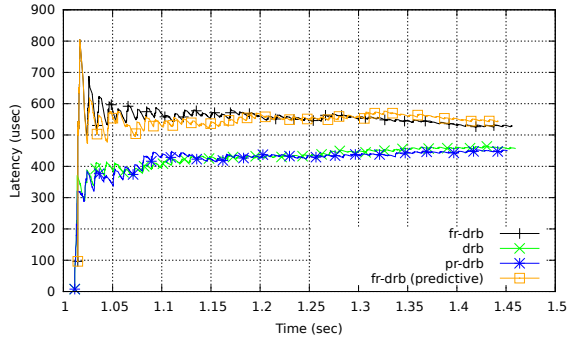


(g) POP Router 111 contention latency

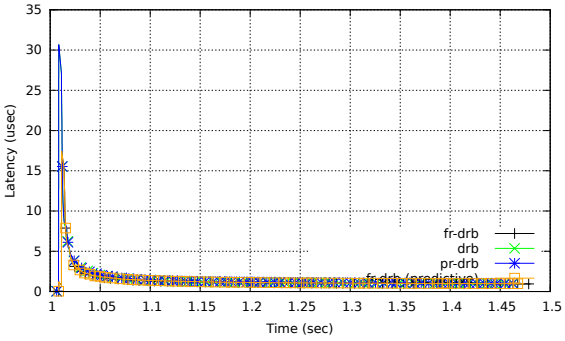


(h) POP Router 116 contention latency

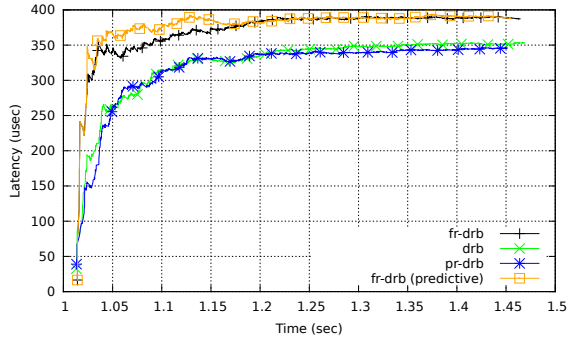
Figure A.6: Contention latency of POP routers (2/3).



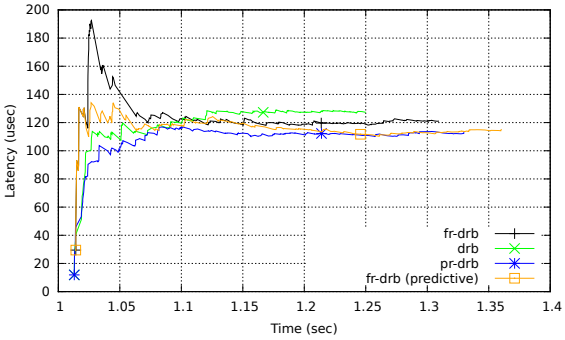
(a) POP Router 118 contention latency;



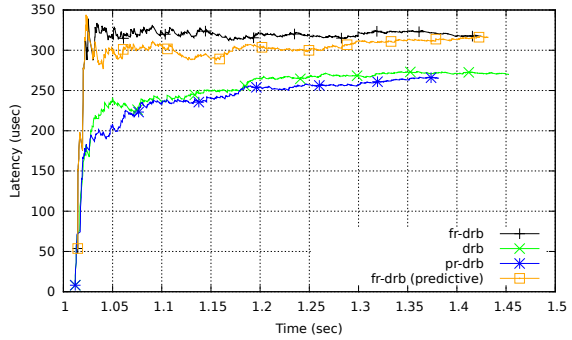
(b) POP Router 121 contention latency



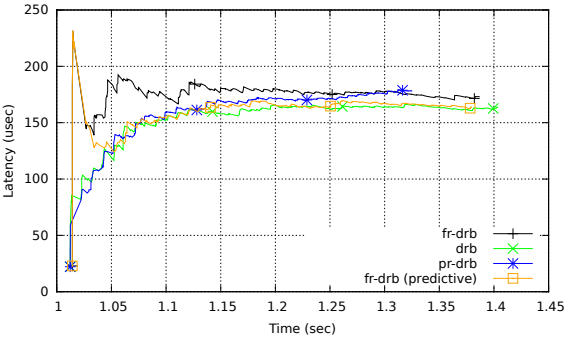
(c) POP Router 132 contention latency



(d) POP Router 134 contention latency



(e) POP Router 136 contention latency



(f) POP Router 142 contention latency

Figure A.7: Contention latency of POP routers (3/3).

Bibliography

[1] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, Supercomputing 91, pages 158 165, New York, NY, USA, 1991. ACM. ISBN 0-89791-459-7.

[2] K. Barker, A. Benner, R. Hoare, A. Hoisie, A. Jones, D. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker. On the feasibility of optical circuit switching for high performance computing systems. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, page 16, nov. 2005. doi:10.1109/SC.2005.48.

[3] E. Baydal et al. A family of mechanisms for congestion control in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 16(9):772 784, 2005. ISSN 1045-9219. doi:http://dx.doi.org/10.1109/TPDS.2005.102.

[4] A. Bhatelé, L. Wesolowski, E. Bohm, E. Solomonik, and L. V. Kalé. Understanding application performance via micro-benchmarks on three large supercomputers: Intrepid, ranger and jaguar. *Int. J. High Perform. Comput. Appl.*, 24 (4):411 427, Nov. 2010. ISSN 1094-3420. doi:10.1177/1094342010370603. URL http://dx.doi.org/10.1177/1094342010370603.

[5] C. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing (pr-drb). In *CACIC, Argentine Conference on Computer Science*, pages 112 121, July. 2010.

[6] C. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing for high speed interconnection networks. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference*, pages 552 556, sept. 2011. doi:10.1109/CLUSTER.2011.66.

[7] C. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing (pr-drb). In *Journal of Computer Science & Technology (JCS&T)*, pages 59 70, October. 2011. ISBN: 978-950-34-0757-8.

[8] C. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing for hpc clusters. In *WorldComp, 2011 International Conference*, pages 875 878, July. 2011.

[9] C. N. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and Distributed Routing Balancing for High Speed Interconnection Networks. In F. Almeida, V. Blanco, C. León, C. Rodríguez, and F. de Sande, editors, *Actas XXII Jornadas de Paralelismo (JP2011)*, pages 397 402. Universidad de La Laguna, Sept. 2011. ISBN 978-84-694-1791-1.

[10] C. N. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing on high-speed cluster networks. In *Proceedings of the 2011 23rd International Symposium on Computer Architecture and High Performance Computing*, SBAC-PAD 11, pages 72 79, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4573-8. doi:10.1109/SBAC-PAD.2011.27. URL http://dx.doi.org/10.1109/SBAC-PAD.2011.27.

[11] C. N. Castillo, D. Lugones, D. Franco, and E. Luque. Predictive and distributed routing balancing, an application-aware approach. In *Proceedings of the 2013 International Conference on Computational Science*, ICCS-2013, Barcelona, SPAIN, 2013. Elsevier. ISBN 978-0-7695-4573-8.

[12] W. J. Dally and C. Seitz. The torus routing chip. *Distributed Computing*, 1:187 196, 1986. ISSN 0178-2770. doi:10.1007/BF01660031.

[13] S. P. Dandamudi. Reducing hot-spot contention in shared-memory multiprocessor systems. *IEEE Concurrency*, 7(1):48 59, 1999. ISSN 1092-3063. doi:http://dx.doi.org/10.1109/4434.749135.

[14] N. Desai, P. Balaji, P. Sadayappan, and M. Islam. Are nonblocking networks really needed for high-end-computing workloads? In *Cluster Computing, 2008 IEEE International Conference on*, pages 152 159, 2008. doi:10.1109/CLUSTR.2008.4663766.

[15] Z. Ding, R. Hoare, A. Jones, and R. Melhem. Level-wise scheduling algorithm for fat tree interconnection networks. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 9 9, 2006. doi:10.1109/SC.2006.40.

[16] G. Dodig-Crnkovic. Scientific methods in computer science. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, April 2002. URL http://www.mrtc.mdh.se/index.php?choice=publications&id=0446.

[17] J. Duato et al. *Interconnection Networks: An Engineering Approach*. M. Kaufmann Pub. Inc., USA, 2002. ISBN 1558608524.

[18] J. Flich, M. Malumbres, P. Lopez, and J. Duato. Improving routing performance in myrinet networks. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 27 32, 2000. doi:10.1109/IPDPS.2000.845961.

[19] D. Franco, I. Garces, and E. Luque. Distributed routing balancing for interconnection network communication. In *HIPC '98: Procs of the Fifth International Conference on High Performance Computing*, page 253, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-9194-8.

[20] D. Franco, I. Garcés, I.s, and E. Luque. Dynamic routing balancing in parallel computer interconnection networks. In V. Hernández, J. Palma, and J. Dongarra, editors, *Vector and Parallel Processing-VECPAR98*, volume 1573 of *Lecture Notes in Computer Science*, pages 494 507. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-66228-0. doi:10.1007/10703040_37. URL http://dx.doi.org/10.1007/10703040_37.

[21] D. Franco et al. A new method to make communication latency uniform: distributed routing balancing. In *ICS '99: Procs of the 13th int. conf. on Superc.*, pages 210 219, USA, 1999. ACM. ISBN 1-58113-164-X. doi:http://doi.acm.org/10.1145/305138.305195.

[22] F. Gilabert, M. Gómez, P. López, and J. Duato. On the influence of the selection function on the performance of fat-trees. In W. Nagel, W. Walter, and W. Lehner, editors, *Euro-Par 2006 Parallel Processing*, volume 4128 of *Lecture Notes in Computer Science*, pages 864 873. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-37783-2.

[23] C. Gómez, F. Gilabert, M. Gomez, P. Lopez, and J. Duato. Deterministic versus adaptive routing in fat-trees. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 8, 2007. doi:10.1109/IPDPS.2007.370482.

[24] L. Hsu and C. Lin. *Graph Theory And Interconnection Networks*. CRC Press, 2009. ISBN 9781420044812. URL http://books.google.ie/books?id=vbxdqhDKOSYC.

[25] IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1.2):199 220, January 2008. ISSN 0018-8646. doi:10.1147/rd.521.0199.

[26] Infiniband. Iba. http://www.infinibandta.org/, 2011.

[27] R. Jain. Congestion control in computer networks: issues and trends. *IEEE Network*, 4(3):24 30, may 1990. ISSN 0890-8044. doi:10.1109/65.56532.

[28] G. Johnson, D. Kerbyson, and M. Lang. Optimization of infiniband for scientific applications. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1 8, 2008. doi:10.1109/IPDPS.2008.4536345.

[29] P. W. Jones. The los alamos parallel ocean program (pop) and coupled model on mpp and cluster smp computers. http://climate.lanl.gov/, June 1997. Making its Mark The Use of Parallel Processors in Meteorology.

[30] S. Kamil, J. Shalf, L. Oliker, and D. Skinner. Understanding ultra-scale application communication requirements. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 178 187, 2005. doi:10.1109/IISWC.2005.1526015.

[31] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf. Reconfigurable hybrid interconnection for static and dynamic scientific applications. In *Proceedings of the 4th international conference on Computing frontiers*, CF 07, pages 183 194, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-683-7. doi:10.1145/1242531.1242559. URL http://doi.acm.org/10.1145/1242531.1242559.

[32] S. Kamil, L. Oliker, A. Pinar, and J. Shalf. Communication requirements and interconnect optimization for high-end scientific applications. *IEEE Trans. Parallel Distrib. Syst.*, 21(2):188 202, Feb. 2010. ISSN 1045-9219. doi:10.1109/TPDS.2009.61. URL http://dx.doi.org/10.1109/TPDS.2009.61.

[33] K. Kandalla, H. Subramoni, A. Vishnu, and D. Panda. Designing topology-aware collective communication algorithms for large scale infiniband clusters: Case studies with scatter and gather. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1 8, 2010. doi:10.1109/IPDPSW.2010.5470853.

[34] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks (1976)*, 3(4):267 286, 1979. ISSN 0376-5075. doi:10.1016/0376-5075(79)90032-1. URL http://www.sciencedirect.com/science/article/pii/0376507579900321.

[35] M. Kinsy, M. H. Cho, K. S. Shim, M. Lis, G. Suh, and S. Devadas. Optimal and heuristic application-aware oblivious routing. *Computers, IEEE Transactions on*, 62 (1):59 73, 2013. ISSN 0018-9340. doi:10.1109/TC.2011.219.

[36] L. A. N. Laboratory. Asci sweep3d 3-dimensional discrete ordinates neutron transport benchmark. http://wwwc3.lanl.gov/pal/software/sweep3d/, May 1995. Los Alamos National Laboratory.

[37] S. Labs. Lammps molecular dynamics simulator. http://lammps.sandia.gov, January 2012. Sandia.

[38] D. Lugones, D. Franco, and E. Luque. Fast-response dynamic routing balancing for high-speed interconnection networks. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1 9, 2009. doi:10.1109/CLUSTR.2009.5289142.

[39] D. Lugones et al. Modeling adaptive routing protocols in high speed interconnection networks. *OPNETWORK 2008 Conf.*, 2008.

[40] D. Lugones et al. Dynamic routing balancing on infiniband networks. In *Journal. of Comp. Science & Tech.*, pages 104 110, 2008.

[41] D. Lugones et al. Dynamic and distributed multipath routing policy for high-speed cluster networks. In *CCGRID '09: Procs of the 2009 9th IEEE/ACM Int. Symp. on Cluster Comp. and the Grid*, pages 396 403, USA, 2009. ISBN 978-0-7695-3622-4. doi:http://dx.doi.org/10.1109/CCGRID.2009.13.

[42] D. Lugones et al. Dynamic and distributed multipath routing policy for high-speed cluster networks. In *Procs of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID 09, pages 396 403, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3622-4. doi:http://dx.doi.org/10.1109/CCGRID.2009.13. URL http://dx.doi.org/10.1109/CCGRID.2009.13.

[43] Myricom. Myricom. http://www.myricom.com/, 2013.

[44] L. Ni and P. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62 76, feb. 1993. ISSN 0018-9162. doi:10.1109/2.191995.

[45] C. Nunez, G. Zarza, D. Lugones, J. Navarro, D. Franco, and E. Luque. ClusterGUI, an Application to Launch OPNET Simulations within Resource Managed Environments. In *OPNETWORK Conference*, pages 1 7, 2011.

[46] L. Oliker, A. Canning, J. Carter, C. Iancu, M. Lijewski, S. Kamil, J. Shalf, H. Shan, E. Strohmaier, S. Ethier, and T. Goodale. Scientific application performance on candidate petascale platforms. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1 12, 2007. doi:10.1109/IPDPS.2007.370259.

[47] T. OPNET. Opnet modeler accelerating network r&d. http://www.opnet.com, June 2008. OPNET.

[48] OPNET Technologies. Veriying Statistical Validity of Discrete Event Simulations. WhitePaper, 2008. URL http://www.opnet.com/whitepapers/abstracts/vsvodes.html.

[49] F. Petrini and M. Vanneschi. A comparison of wormhole-routed interconnection networks. In *In Third International Conference on Computer Science and Informatics, Research Triangle Park, North*, 1997.

[50] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. KalA©, and K. Schulten. Scalable molecular dynamics with namd. *Journal of Computational Chemistry*, 26(16):1781 1802, 2005. ISSN 1096-987X. doi:10.1002/jcc.20289. URL http://dx.doi.org/10.1002/jcc.20289.

[51] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *JOURNAL OF COMPUTATIONAL PHYSICS*, 117:1 19, 1995.

[52] R. Riesen. Communication patterns. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS 06, pages 275 275, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 1-4244-0054-6. URL http://dl.acm.org/citation.cfm?id=1898699.1898795.

[53] G. Rodriguez, C. Minkenberg, R. Beivide, R. Luijten, J. Labarta, and M. Valero. Oblivious routing schemes in extended generalized fat tree networks. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1 8, 2009. doi:10.1109/CLUSTR.2009.5289145.

[54] G. Rodriguez et al. Exploring pattern-aware routing in generalized fat tree networks. In *ICS '09: Procs of the 23rd int. conf. on Supercomp.*, pages 276 285, USA, 2009. ACM. ISBN 978-1-60558-498-0. doi:http://doi.acm.org/10.1145/1542275.1542316.

[55] T. Sherwood et al. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *PACT '01: Procs of the 2001 Int. Conf. on Par. Arch. and Compil. Tech.*, pages 3 14, 2001. ISBN 0-7695-1363-8.

[56] A. Simulation and C. A. Purple. The asci purple benchmark codes. https://asc.llnl.gov/computingresources/purple/archive/benchmarks/ , May 2001. ASC.

[57] K. M. Su and K. H. Yum. Simple and effective adaptive routing algorithms in multi-layer wormhole networks. *IEEE International Performance, Computing and Communications Conference*, pages 176 184, 2008.

[58] TOP500 Supercomputing Sites. Architecture share for 11/2012, November 2012. URL http://www.top500.org.

[59] TOP500 Supercomputing Sites. Interconnect family share for 11/2012, November 2012. URL http://www.top500.org.

[60] J. Vetter and A. Yoo. An empirical performance evaluation of scalable scientific applications. In *Supercomputing, ACM/IEEE 2002 Conference*, pages 16 16, 2002. doi:10.1109/SC.2002.10036.

[61] A. Vishnu, M. Koop, A. Moody, A. Mamidala, S. Narravula, and D. Panda. Hot-spot avoidance with multi-pathing over infiniband: An mpi perspective. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 479 486, may 2007. doi:10.1109/CCGRID.2007.60.

[62] B. T. W. Dally. *Principles and Practices of Interconnection Networks*, volume 1. Morgan Kaufmann Publishers, 1 edition, 2004.

[63] M.-C. Wang, H. Siegel, M. Nichols, and S. Abraham. Using a multipath network for reducing the effects of hot spots. *Parallel and Distributed Systems, IEEE Transactions on*, 6(3):252 268, mar 1995. ISSN 1045-9219. doi:10.1109/71.372775.

[64] A. Wong et al. Parallel application signature. *CLUSTER '09. IEEE Int. Conf. on*, 1: 1 4, 2009.

[65] G. Zarza. *Multipath Fault-tolerant Routing Policies to deal with Dynamic Link Failures in High Speed Interconnection Networks*. PhD thesis, Universitat Autònoma de Barcelona. Departament d Arquitectura de Computadors i Sistemes Operatius, July 2011. URL http://hdl.handle.net/10803/51494.

[66] G. Zarza, D. Lugones, D. Franco, and E. Luque. Ft-drb: A method for tolerating dynamic faults in high-speed interconnection networks. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 77 84, 2010. doi:10.1109/PDP.2010.65.