Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computaciò d'Altes Prestacions

Políticas de encaminamiento tolerantes a fallos

Memoria del trabajo de investigación del "Máster en Computación de Altas Prestaciones", realizada por Gonzalo Zarza, bajo la dirección del Dr. Daniel Franco Puntes.
Presentada en la Escuela Técnica Superior de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

Barcelona, Julio de 2008

Trabajo de investigación

Máster en Computación de Altas Prestaciones

Curso 2007-08

Políticas de encaminamiento tolerantes a fallos

Autor

Gonzalo Zarza

Director

Daniel Franco Puntes

Departamento Arquitectura de Computadores y Sistemas Operativos

Escuela Técnica Superior de Ingeniería (ETSE)

Universidad Autónoma de Barcelona

Firma autor

Firma director

Resum

L'ús intensiu i perllongat de computadors d'altes prestacions per a executar aplicacions computacionalment intensives, sumat a l'elevat nombre d'elements que els componen, incrementen dràsticament la probabilitat d'ocurrència de fallades durant el seu funcionament.

L'objectiu del treball és resoldre el problema de tolerància a fallades per a xarxes d'interconnexió d'altes prestacions, partint del disseny de polítiques d'encaminament tolerants a fallades.

Busquem resoldre una determinada quantitat de fallades d'enllaços i nodes, considerant els seus factors d'impacte, probabilitat d'aparició. Per a això s'aprofita la redundància de camins de comunicació existents, partint des d'enfocaments d'encaminament adaptatius capaços de complir amb les quatre fases de la tolerància a fallades: detecció de l'error, contenció del dany, recuperació de l'error, i tractament de la fallada i continuïtat del servei. L'experimentació mostra una degradació de prestacions menor al 5 %. En el futur, es tractarà la pèrdua d'informació en trànsit.

Paraules claus: xarxes d'interconnexió, tolerància a fallades, encaminament adaptatiu.

Resumen

El uso intensivo y prolongado de computadores de altas prestaciones para ejecutar aplicaciones computacionalmente intensivas, sumado al elevado número de elementos que los componen, incrementan drásticamente la probabilidad de ocurrencia de fallos durante su funcionamiento.

El objetivo del trabajo es resolver el problema de tolerancia a fallos para redes de interconexión de altas prestaciones, partiendo del diseño de políticas de encaminamiento tolerantes a fallos.

Buscamos resolver una determinada cantidad de fallos de enlaces y nodos, considerando sus factores de impacto y probabilidad de aparición. Para ello aprovechamos la redundancia de caminos de comunicación existentes, partiendo desde enfoques de encaminamiento adaptativos capaces de cumplir con las cuatro fases de la tolerancia a fallos: detección del error, contención del daño, recuperación del error, y tratamiento del fallo y continuidad del servicio. La experimentación muestra una degradación de prestaciones menor al 5 %. En el futuro, se tratará la pérdida de información en tránsito.

Palabras clave: redes de interconexión, tolerancia a fallos, encaminamiento adaptativo.

Abstract

The intensive and continous use of high-performance computers to execute computationally intensive applications, coupled with the large number of elements that make them up, dramatically increase the likelihood of failures during their operation.

This works focuses on solving the problem of fault tolerance for high speed interconnection networks by means of designing fault tolerant routing policies.

The goal is to solve a determined number of link and node failures, considering its impact factor and occurrence probability. To acomplish this task we take advantage of the communication path redundancy, through adaptive routing approaches that fulfils with the four phases of the fault tolerance: error detection, damage confinement and assessment, error recovery, fault treatment and continuous service. The experiments shows performance's degradation under 5 %. In the future, we'll treat the loose of information in transit.

Keywords: interconnection networks, fault tolerance, adaptive routing.

A mi familia y amigos,
por todo el apoyo y los buenos momentos...

A Dani y a Diego,
por su invaluable ayuda en este proyecto...

Al CAOS,
por la confianza y la oportunidad...

¡Gracias!

Índice general

1. Introducción **1**

1.1. Contexto 1

1.1.1. Computación de Altas Prestaciones 2

1.1.2. Redes de Interconexión de Alta Velocidad 4

1.2. Problema 5

1.3. Motivación 6

1.4. Objetivos 8

1.5. Estructura de la memoria de la tesina 8

2. Marco teórico y estado del arte **11**

2.1. Redes de Interconexión 11

2.1.1. Definiciones 11

2.1.2. Ámbito de aplicación 13

2.1.3. Consideraciones de diseño 13

Parámetros de diseño 13

Factores que influyen en el diseño 14

Requerimientos de diseño de las redes de interconexión 15

2.1.4. Topologías 16

Redes de medio compartido 17

Redes directas 17

Redes indirectas 17

Redes híbridas 19

2.1.5. Control de flujo 20

2.1.6. Técnicas de conmutación 21

2.1.7. Encaminamiento 22

2.1.8. Balanceo Distribuido del Encaminamiento (DRB) 24

Mejoras al balanceo distribuido del encaminamiento (DRB-E) 27

2.2. Tolerancia a fallos 28

2.2.1. Requisitos generales 30

2.2.2. Fallo, error y avería 30

2.2.3. Fases de la tolerancia a fallos 31

2.2.4. Concepto e importancia de la redundancia . . . 32
2.2.5. Clasificación de fallos . . . 33
2.2.6. Modelo de fallos . . . 33
Definición de modelos de fallos . . . 34
Críticas a las definiciones de modelos de fallos . . . 34
2.2.7. Medidas de tolerancia a fallos para redes de interconexión . . . 35
Medidas tradicionales . . . 35
Medidas propias de la teoría de grafos . . . 36
Medidas propias de las redes de computadores . . . 36
2.3. Tolerancia a fallos orientada a redes de interconexión . . . 37
2.3.1. Diferencias entre red y encaminamiento tolerantes a fallos . . . 39
2.3.2. Material bibliográfico . . . 39
Libros . . . 39
Artículos . . . 40

3. Análisis **43**
3.1. Introducción . . . 43
3.2. Análisis de los fundamentos teóricos . . . 45
3.2.1. Modelos y tipos de fallos . . . 45
3.2.2. Estado de los enlaces . . . 46
3.2.3. Topologías . . . 46
3.2.4. Encaminamiento . . . 46
3.3. Análisis del trabajo . . . 47
3.3.1. Alcance . . . 47
3.3.2. Punto de partida y modificaciones . . . 49

4. Diseño **51**
4.1. Introducción . . . 51
4.2. Diseño orientado a la tolerancia a fallos . . . 52
4.2.1. Detección del error . . . 52
4.2.2. Contención del daño . . . 52
4.2.3. Recuperación del error . . . 53
4.2.4. Tratamiento del fallo y continuidad del servicio . . . 56
4.3. Políticas de encaminamiento tolerantes a fallos . . . 57
4.3.1. Fase 1: Monitorización del estado de los enlaces y la carga de tráfico . . . 60
4.3.2. Fase 2: Configuración dinámica de los metacaminos . . . 62
4.3.3. Fase 3: Selección del camino multipaso . . . 63

5. Implementación **67**
5.1. Introducción . . . 67

5.2. Entorno de simulación . . . 68
5.3. Modelos de red . . . 69
5.3.1. Implementación de los nodos de procesamiento . . . 71
5.3.2. Implementación de los encaminadores . . . 72
5.4. Conclusión . . . 74

6. Experimentación y análisis de resultados **77**
6.1. Introducción . . . 77
6.2. Diseño de los experimentos . . . 77
6.2.1. Diseño de las pruebas . . . 79
6.3. Análisis de resultados . . . 83
6.3.1. Comunicaciones punto a punto . . . 89
Escenario P . . . 93
Escenario R . . . 93
6.3.2. Comunicaciones colectivas all-to-all . . . 96
6.3.3. Comunicaciones colectivas one-to-all . . . 100
6.3.4. Comunicaciones colectivas all-to-one . . . 106
6.3.5. Resumen de la experimentación . . . 107

7. Conclusiones **115**
7.1. Conclusiones finales . . . 115
7.2. Conclusiones sobre los resultados . . . 116
7.3. Trabajo futuro y líneas abiertas . . . 117

Apéndices

A. Definiciones matemáticas **121**
A.1. Definiciones del encaminamiento de DRB-E . . . 121
A.1.1. Definiciones previas . . . 121
Nodos adyacentes . . . 121
Distancia . . . 121
Camino . . . 121
Longitud . . . 121
A.1.2. Supernodo . . . 122
A.1.3. Camino multipaso . . . 122
A.1.4. Metacamino . . . 122
A.1.5. Longitud del camino multipaso . . . 122
A.1.6. Latencia del camino multipaso . . . 123
A.1.7. Anchura del metacamino . . . 123
A.1.8. Latencia del metacamino . . . 123

Bibliografía **124**

Índice alfabético **128**

Índice de figuras

1.1. Clasificación de computadores paralelos 4
1.2. Ejemplo de fallos de enlace y dispositivo 7

2.1. Categorías básicas de redes de interconexión 17
2.2. Clasificación de redes de medio compartido 18
2.3. Clasificación de redes directas 18
2.4. Clasificación de redes indirectas 18
2.5. Topologías de redes de medio compartido 19
2.6. Topologías de redes directas ortogonales 19
2.7. Topología de red directas no ortogonal 20
2.8. Topologías de redes indirectas 20
2.9. Clasificación de redes híbridas 20
2.10. Topología de red híbrida 21
2.11. Funcionamiento de DRB para 3 caminos 27
2.12. Funcionamiento de DRB-E para 3 caminos 29
2.13. Relación entre fallo, error y avería 31
2.14. Tolerancia a fallos, encaminamiento y redes de interconexión 40

3.1. Ejemplo de redundancia de caminos 44
3.2. Conceptos gráficos y nomenclatura 45
3.3. Ejemplo del fallo de un nodo visto como fallos simultaneos de sus enlaces 45
3.4. Ejemplo de utilización del símbolo de camino nulo 50

4.1. Ejemplo de funcionamiento de la fase de contención del daño 53
4.2. Ejemplo de funcionamiento de la fase de recuperación del error 56
4.3. Ejemplo de funcionamiento de la fase de tratamiento del fallo y continuidad del servicio 58
4.4. Concepto de supernodo 59
4.5. Paquete DRB 61
4.6. Cabeceras del mensaje DRB 61

5.1. Estructura de un encaminador estándar 69
5.2. Modelo de red (malla 4x4) 70

5.3. Atributos del encaminador 70
5.4. Implementación del modelo de nodos de procesamiento 71
5.5. FSM del nodo de proceso de los nodos de procesamiento 72
5.6. FSM del *AMR_handler* de los nodos de procesamiento 73
5.7. FSM de la interfaz de red de lo nodos de procesamiento 74
5.8. Implementación del modelo de encaminadores 75
5.9. FSM del *crossbar* 76
5.10. FSM del *switch_info* 76
5.11. FSM de los módulos de encaminamiento y arbitraje 76

6.1. Relación entre los valores de tiempo y latencia (normalizada) 80
6.2. Dirección de los ejes de coordenadas 80
6.3. Numeración de enlaces en cada encaminador 81
6.4. Escenarios de pruebas de comunicaciones punto a punto 82
6.5. Escenarios de pruebas de comunicaciones colectivas 84
6.6. Escenarios de pruebas de comunicaciones colectivas adicionales 85
6.7. Resultados de pruebas de comunicaciones punto a punto (unidireccional) 90
6.8. Resultados de pruebas de comunicaciones punto a punto (bidireccional) 91
6.9. Formas de selección de las vías de escape 92
6.10. Comportamiento del escenario P 94
6.11. Comportamiento del escenario R 95
6.12. Resultados de las pruebas de comunicaciones all-to-all (por casos) 97
6.13. Resultados de las pruebas de comunicaciones all-to-all (por número de fallos) . . 98
6.14. Resultados de las pruebas de comunicaciones all-to-all (para 1 fallo) 98
6.15. Resultados de las pruebas de comunicaciones all-to-all (para 2 fallos) 99
6.16. Resultados de las pruebas de comunicaciones all-to-all (para 3 fallos) 99
6.17. Resultados de las pruebas de comunicaciones one-to-all (por casos) 102
6.18. Resultados de las pruebas de comunicaciones one-to-all (por número de fallos) . . 103
6.19. Resultados de las pruebas de comunicaciones one-to-all (para 1 fallo) 104
6.20. Resultados de las pruebas de comunicaciones one-to-all (para 3 fallos) 105
6.21. Resultados de las pruebas de comunicaciones all-to-one (por casos) 108
6.22. Resultados de las pruebas de comunicaciones all-to-one (por número de fallos) . . 109
6.23. Resultados de las pruebas de comunicaciones all-to-one(para 1 fallo) 110
6.24. Resultados de las pruebas de comunicaciones all-to-one (para 2 fallos) 111
6.25. Resultados de las pruebas de comunicaciones all-to-one (para 3 fallos) 112
6.26. Resultados de las pruebas de todas las comunicaciones (por casos) 113
6.27. Resultados de las pruebas de todas las comunicaciones (por número de fallos) . . 113
6.28. Resultados de las pruebas de todas las comunicaciones (promedio por fallos) . . . 114

Índice de tablas

1.1. Ejemplos de fallos en redes de interconexión 6

2.1. Parámetros de diseño estáticos de redes de interconexión 14
2.2. Parámetros de diseño dinámicos de redes de interconexión 14
2.3. Clasificación de los algoritmos de encaminamiento 25
2.4. Resumen de las técnicas de toleranca a fallos para redes. Parte 1 41
2.5. Resumen de las técnicas de toleranca a fallos para redes. Parte 2 41

3.1. Caracterización del trabajo a realizar 48
3.2. Ámbito de tratamiento de situaciones de fallos 49

6.1. Parámetros de simulación constantes en todos los escenarios 79
6.2. Caracterización de los escenarios de pruebas de comunicaciones punto a punto . . 81
6.3. Caracterización de las pruebas de comunicaciones punto a punto 82
6.4. Caracterización de los escenarios de pruebas de comunicaciones colectivas 86
6.5. Caracterización de las pruebas de comunicaciones colectivas *All-to-All* 86
6.6. Caracterización de las pruebas de comunicaciones colectivas *One-to-All* 87
6.7. Caracterización de las pruebas de comunicaciones colectivas *All-to-One* 88
6.8. Resultados de las pruebas de comunicaciones punto a punto 89
6.9. Resultados de las pruebas de comunicaciones colectivas all-to-all 96
6.10. Resultados de las pruebas de comunicaciones colectivas one-to-all 101
6.11. Resultados de las pruebas de comunicaciones colectivas all-to-one 106

Índice de ecuaciones

6.1 Latencia media 78
A.1 Longitud de un camino 121
A.2 Camino multipaso 122
A.3 Metacamino 122
A.4 Longitud de un camino multipaso 123
A.5 Latencia de un camino multipaso 123
A.6 Anchura del metacamino 123
A.7 Latencia de un camino multipaso 123

Índice de códigos fuente

4.1. Pseudocódigo de la metodología de tolerancia a fallos para DRB-E 60
4.2. Código de monitorización del estado de enlaces y de la carga de tráfico 62
4.3. Código de configuración dinámica de los metacaminos 64
4.4. Código de selección del camino multipaso 65

Capítulo 1

Introducción

"We choose to go to he moon in this decade and do he other things, not because they are easy, but because they are hard."

John F. Kennedy

1.1. Contexto

Uno de los rasgos que más caracteriza a la sociedad actual es la sinergia que se ha ido formando entre su entorno socio-cultural y los servicios tecnológicos de tiempo real. Sinergia que se fortalece día a día, debido a la progresiva incorporación de dispositivos tecnológicos digitales en casi todos los aspectos de la vida cotidiana.

La mayor parte de estos servicios, y sobre todo su disponibilidad, se sustentan en sistemas de cómputo de altas prestaciones; sistemas que fueron originalmente desarrollados con el fin de permitir la ejecución de aplicaciones de cálculo intensivo, valiéndose para esto de grandes capacidades de cómputo y procesadores de alto rendimiento. En la Sección 1.1.1 se describen brevemente las características de la computación de altas prestaciones (*HPC*, por sus siglas en inglés).

Más allá de estos orígenes ligados al cálculo intensivo, la computación de altas prestaciones ha ido experimentando una gran expansión a lo largo de los últimos años, partiendo desde ámbitos puramente científicos o académicos –y presupuestos millonarios– hacia ámbitos más globales, propios de estos servicios de respuesta inmediata.

En la actualidad, son muchas las disciplinas que requieren mayor capacidad de cómputo y resultados en tiempos menores. Este incremento en la demanda de cómputo y rendimiento, sumado a la mejora en la relación costo/beneficio, el perfeccionamiento y la simplificación de los sistemas de altas prestaciones, los sitúa como la solución más viable a estas demandas.

Entre los ejemplos emblemáticos de las disciplinas tradicionales que sustentan esta demanda se encuentran: el problema de descifrar el genoma humano, la simulación de la dinámica y turbulencia de fluidos, la predicción meteorológica y la prospección geológica.

Dentro de los denominados nuevos ámbitos, se pueden citar ejemplos como las aplicaciones de simulación científicas e ingenieriles, el procesamiento de imágenes médicas, los sistemas bancarios, los medios de comunicación, los proveedores de servicios multimedia (Internet, Video on Demand, etc.) y la recuperación de información de grandes bases de datos (Google, Amazon, etc.).

Pese a esta expansión, la computación de altas prestaciones no representa un sector mayoritario en el mundo de la computación. Sin embargo, se encuentra en la primera línea de investigación y desarrollo de la ciencia computacional. Esta posición de privilegio tiene su origen, y a su vez su finalidad, en los retos y desafíos propios de la computación de altas prestaciones.

En un principio, estos desafíos se relacionaban con la viabilidad de la realización de grandes volúmenes de cálculos matemáticos, pero a medida que su ámbito se ha ido diversificando y las aplicaciones se modernizaron, los retos fueron cambiando. Actualmente, las grandes fuentes de desafíos se relacionan con el gran volumen de comunicación de datos, los servicios en tiempo real y la interactividad con el usuario.

La computación de altas prestaciones, tal como se la entiende hoy en día, se basa en tres factores principales: las *unidades de procesamiento*, los *dispositivos de almacenamiento y de entada/salida*, y la *red de interconexión de alta velocidad* que los enlaza.

Es importante tener en cuenta que, si bien los avances tecnológicos logran incrementar periódicamente el rendimiento y las prestaciones, tanto de las unidades de procesamiento como de los dispositivos de almacenamiento y de la red de interconexión, este incremento no es equitativo. Las mejoras se producen en tiempos considerablementes menores para las unidades de procesamiento. Esto lleva a que el actual cuello de botella de los sistemas de cómputo de altas prestaciones sean las redes de interconexión. Este límite tecnológico es uno de los puntos de separación entre redes de interconexión estándar y redes de interconexión de alta velocidad (se explican en la Sección 1.1.2).

El otro gran punto de separación es la utilización de políticas de tolerancia a fallos (y su nivel de eficacia), ya que para alcanzar altas prestaciones, una red de interconexión debe estar dotada, casi obligatoriamente, de un sistema íntegro de tolerancia a fallos. De este modo la red puede continuar brindando una calidad de servicio aceptable, aún después de que se hayan producido fallos en la red.

1.1.1. Computación de Altas Prestaciones

Los computadores de altas prestaciones son aquellos capaces de ofrecer los requerimientos de cómputo de altas prestaciones que necesitan las aplicaciones hasta ahora mencionadas. Éstos son, básicamente, computadores paralelos; una evolución del computador convencional de un único procesador de Von Neumann.

Los computadores paralelos se basan en la replicación de unidades funcionales tales como procesadores y memorias, para aumentar la velocidad de cómputo manteniendo un costo aceptable. Los elementos replicados se conectan entre sí para formar el computador de altas prestaciones

mediante una red de interconexión, con el objetivo principal de ejecutar conjuntamente las tareas de cómputo de manera concurrente, lo que implica la operación simultánea de múltiples procesadores de cómputo, y paralela, cuando los procesadores ejecutan código de una misma aplicación.

En las primeras páginas de [4] se expone de una forma clara y concisa la razón de ser de los computadores paralelos:

> *"Single-processor supercomputers have achieved unheard of speeds and have been pushing hardware technology to the physical limit of chip manufacturing. However, this trend will soon come to an end, because there are physical and architectural bounds that limit the computational power that can be achieved with a single-processor system.*
>
> ...
>
> *The main argument for using multiprocessors is to create powerful computers by simply connecting multiple processors. A multiprocessor is expected to reach faster speed than the fastest single-processor system. In addition, a multiprocessor consisting of a number of single processors is expected to be more cost-effective than building a high-performance single processor."*

Los computadores paralelos pueden ser clasificados en diferentes categorías, en base a la presencia de flujos simples o múltiples de instrucciones y de datos. Esta clasificación se debe a Flynn [5], y da lugar a las siguientes cuatro categorías:

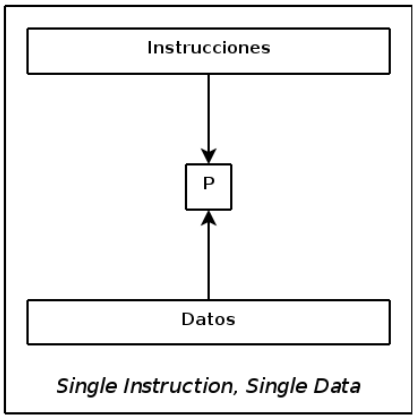
SISD (Single Instruction, Single Data stream). Define los computadores seriales de una única unidad de procesamiento (arquitectura tipo Von Neumann).

MISD (Multiple Instruction, Single Data stream). Implicaría múltiples procesadores aplicando diferentes instrucciones a un único dato; ésta posibilidad hipotética (aunque conceptualmente válida) es usualmente considerada como impracticable.

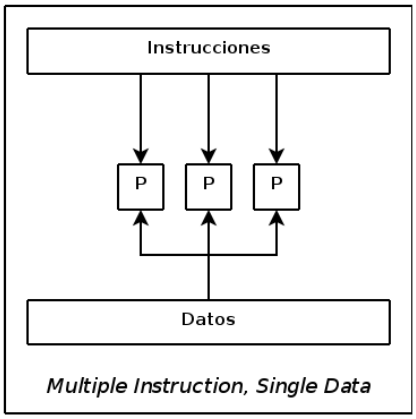
SIMD (Single Instruction, Multiple Data streams). Implica múltiples procesadores ejecutando simultáneamente la misma instrucción en datos diferentes.

MIMD (Multiple Instruction, Multiple Data streams). Implica múltiples procesadores ejecutando diversas instrucciones en datos diferentes de manera autónoma.

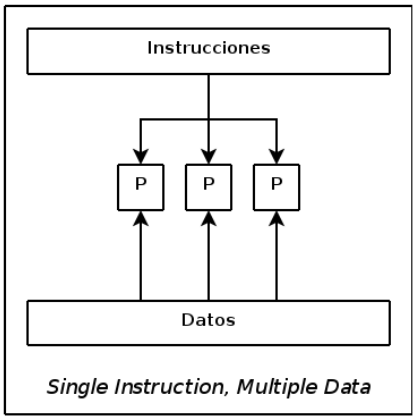
En las Figs. 1.1(a), 1.1(b), 1.1(c) y 1.1(d) se pueden observar, de manera gráfica, los conceptos de SISD, MISD, SIMD y MIMD, respectivamente.



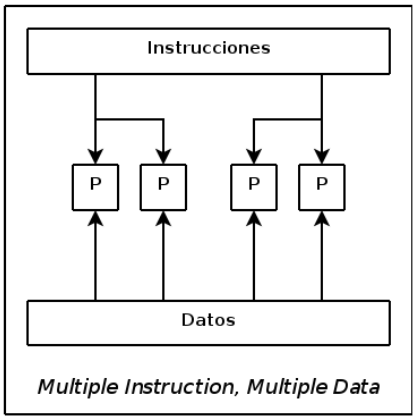
(a) SISD



(b) MISD



(c) SIMD



(d) MIMD

Figura 1.1: Clasificación de computadores paralelos

1.1.2. Redes de Interconexión de Alta Velocidad

Los avances tecnológicos de los últimos años en el campo de las redes de interconexión han permitido aumentar el ancho de banda de los enlaces, incrementar la eficiencia de las técnicas de conmutación y control de flujo, así como de los algoritmos de encaminamiento, y dotar de mayor flexibilidad a la topología. Todo esto, en conjunto, ha supuesto una gran mejora de las prestaciones de las redes, permitiendo la aparición de lo que se denominan redes de interconexión de alta velocidad (altas prestaciones).

Estas redes de alta velocidad se caracterizan por haber heredado algunas características tanto de las redes locales convencionales como de las redes de interconexión de grandes computadores paralelos; por emplear enlaces punto a punto de alta velocidad para conectar elementos; y por poseer cierta flexibilidad y variabilidad en cuanto a las topologías posibles [11].

Las redes de alta velocidad deben proveer:

- Las menores latencias de comunicación posibles (para evitar bloquear los procesos).

- Un elevado ancho de banda para permitir la comunicación concurrente de procesos.
- Un alto nivel de fiabilidad, y un método robusto y eficaz de tolerancia a fallos.
- Un nivel de escalabilidad aceptable, ya que es fundamental para el computador de altas prestaciones.

1.2. Problema

En la sección anterior se ha introducido el concepto de la computación de altas prestaciones, su razón de ser, y su cada vez mayor importancia en el mundo actual.

A partir de esto, surgen una serie de preguntas claves: ¿Cómo se ven afectados los sistemas de cómputo de altas prestaciones en presencia de fallos? ¿Son capaces de mantener sus niveles de funcionamiento y prestaciones a pesar de la ocurrencia de fallos? Si no es así ¿Cómo se soluciona? ¿Cuáles son las mejores opciones para lograrlo?

Si bien estos interrogantes acompañan a la ciencia de la computación desde sus orígenes a principios del siglo pasado, han adquirido una mayor importancia a medida que los sistemas de cómputo se fueron volviendo más complejos. Este aumento de la complejidad, sumado al incremento del número de elementos que componen estos sistemas, implican un incremento en la probabilidad global de fallos del sistema. Es verdad que no se puede pasar por alto el hecho de que las mejoras tecnológicas han permitido mejorar los dispositivos electrónicos de forma de hacerlos menos propensos a los fallos, sin embargo, el agrupamiento y aumento del número de estos dispositivos dentro de un mismo sistema, contribuyen al incremento de la probabilidad de fallo del sistema visto como un conjunto.

Realizando un análisis a grandes rasgos se puede apreciar que la tolerancia a fallos posee tres ramas bien definidas de tratamiento:

- Tolerancia a fallos orientada a las unidades de procesamiento.
- Tolerancia a fallos orientada a los dispositivos de almacenamiento y de entrada/salida (E/S).
- Tolerancia a fallos orientada a las redes de interconexión.

Como es de esperarse, estas ramas están relacionadas con los componentes principales de los sistemas de cómputo de altas prestaciones: las unidades de procesamiento, los dispositivos de almacenamiento y de E/S, y la red de interconexión que los enlaza.

La problemática aquí expuesta no puede ser pasada por alto o dejada de lado. El mejor ejemplo de la importancia de la misma se puede apreciar al exponer la situación del segundo supercomputador más poderoso del mundo, el *BlueGene/L* (compuesto por 212.992 procesadores, y con un rendimiento máximo real de 478.200 GFlops) [34]. El gran número de procesadores y dispositivos de interconexión de este supercomputador hacen que su tiempo medio entre fallos

(MTBF, Mean Time Between Failures) sea menor que el tiempo de ejecución de algunas aplicaciones que típicamente se ejecutan en él. Esto implica que durante el tiempo de ejecución de estas aplicaciones, se presenten uno o más fallos en el mismo [29], por lo cual resulta imprescindible dotarlo de un sistema de tolerancia a fallos efectivo y eficiente.

Lo expuesto anteriormente justifica el tratamiento de la tolerancia a fallos desde el punto de vista temporal, basándose en el tiempo de ocurrencia de los fallos y su relación con los tiempos medios de las aplicaciones. Sin embargo, es necesario ampliar y definir completamente los alcances de los fallos y sus consecuencias para los sistemas de cómputo de altas prestaciones.

Este proyecto de investigación se centrará en la tolerancia a fallos para redes de interconexión, por lo cual solamente se expondrán los fallos que se relacionen con esta temática y sus consecuencias.

Desde el punto de vista de las redes de interconexión, los fallos pueden generar problemas de diversa índole, repercusión y complejidad de tratamiento. En la Tabla 1.1 se listan algunos de estos fallos, a forma de ejemplo.

Fallo	Ámbito	Descripción/Efectos
Pérdida de caminos	Enlaces y/o dispositivos	Inhabilitan caminos entre pares de nodos origen-destino. Pueden incrementar sensiblemente la longitud de los caminos mínimos.
Caída de enlaces	Enlaces	Es un fallo difícil de tratar cuando la caída se produce con información en tránsito en el enlace.
Caída de dispositivos	Dispositivos	Es aún más complicado que el anterior ya que al caer un dispositivo se pueden perder todos los datos almacenados en sus buffers, tanto de entrada como de salida.
Todos	Enlaces y/o dispositivos	Cualquier fallo que se produzca en la red de interconexión genera congestión en el entorno próximo a dicho fallo.
Todos	Enlaces y/o dispositivos	Producen desconexiones en la red, aislando sectores y dejando inaccesibles componentes funcionales de la red.

Tabla 1.1: Ejemplos de fallos en redes de interconexión

En la Fig. 1.2 se ejemplifican gráficamente fallos de enlace y de dispositivo.

1.3. Motivación

Hasta aquí se han puesto en contexto los sistemas de cómputo de altas prestaciones y la importancia de las redes de interconexión de alta velocidad para estos sistemas; se ha puesto de

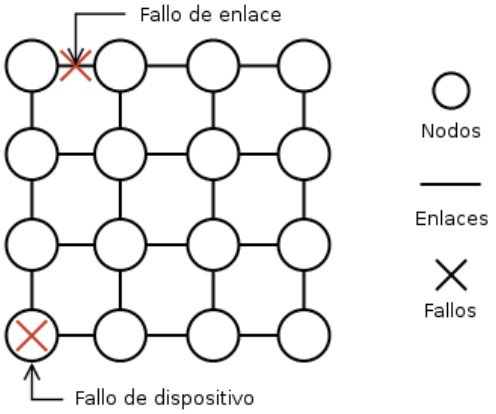


Figura 1.2: Ejemplo de fallos de enlace y dispositivo

manifiesto además la relevancia del problema de los fallos y la necesidad de tratarlos de manera adecuada.

A partir de esto, resulta simple y natural exponer y fundamentar el motivo que da origen al presente proyecto de investigación: lograr diseñar, desarrollar e implementar una red de interconexión tolerante a fallos.

Este no es un problema menor, dado que existen un gran número de posibles fallos que pueden afectar a las redes de interconexión y provocar desde desconexiones, hasta pérdidas totales de información en los dispositivos de red.

Cuando hacemos referencia a redes de interconexión hablamos de un sistema que sea tolerante a fallos en su conjunto, y no solo a lograr desarrollar algoritmos de encaminamiento tolerantes a fallos. El problema de las redes de interconexión tolerantes a fallos posee una complejidad mucho mayor que la del encaminamiento; incluye, no solo el encaminamiento tolerante a fallos, sino también la monitorización de enlaces, la detección y clasificación de los fallos y la distribución de la información relacionada con estos eventos a todos los dispositivos que puedan necesitarla. En la Sección 2.3.1 se explican con más detalle las diferencias entre los algoritmos de encaminamiento tolerantes a fallos y las redes de interconexión tolerantes a fallos.

Una red tolerante a fallos puede ser definida como aquella red que debe conseguir estándares de funcionamiento y prestaciones lo más homogéneos posibles, aún en presencia de múltiples fallos, encapsulando por completo la ocurrencia de estos a las capas superiores. Es decir, la ocurrencia y tratamiento de los fallos –junto con todos los procedimientos asociados a estos estados– deben ser totalmente transparentes a las capas superiores a las que da servicio la red. La transparencia debe ser tal que aún en el peor de los casos, el que implica pérdida de información por la caída de un dispositivo de encaminamiento, la red sea capaz de recuperar la información perdida sin solicitar la reinyección de mensajes a las capas superiores.

1.4. Objetivos

A pesar de que el objetivo fundamental de este proyecto de investigación, tal y como ya se ha explicado en los puntos anteriores, es lograr dotar a las redes de interconexión de las metodologías y procedimientos que le permitan tolerar fallos exitosamente, es posible desglosar este objetivo global en una serie de objetivos parciales que se detallan a continuación:

1. Estudiar las redes de interconexión, sus características tanto estáticas como dinámicas, y su marco de aplicación.
2. Estudiar la teoría general de tolerancia a fallos, analizando las posibilidades de aplicación en el ámbito de redes.
3. Realizar un análisis de los fallos que afectan las redes de interconexión, su probabilidad de ocurrencia y su factor de impacto.
4. Realizar un estudio de los modelos teóricos que relacionan la tolerancia a fallos y las redes de interconexión.
5. Realizar una evaluación de las soluciones existentes para tratar los fallos en las redes de interconexión.
6. Analizar las características de la técnica de balanceo distribuido del encaminamiento (DRB) y extenderla, para introducirle los conceptos de tolerancia a fallos.
   a) Adaptar DRB para incluir los identificadores de fallos y su tratamiento.
   b) Incorporación de las técnicas de monitorización de enlaces, detección de fallos y distribución de la información de relevancia.
   c) Ampliación de los modelos teóricos, para ajustarlos a los casos reales.
7. Seleccionar las herramientas de desarrollo y simulación adecuadas para el estudio, validación y experimentación de la propuesta establecida.
8. Realizar la experimentación y el análisis de los mecanismos introducidos, mediante la comparación con otras técnicas existentes a través de simulaciones.

1.5. Estructura de la memoria de la tesina

En este capítulo se han introducido tanto el entorno en que se sitúa el proyecto de investigación, así como también algunos conceptos y ejemplos básicos de la computación de altas prestaciones, las redes de alta velocidad y los fallos que las afectan. Los capítulos restantes del documento están dedicados a explicar con más detalle el trabajo que aquí se presenta:

Capítulo 2. Marco teórico y estado del arte. Está constituido por tres bloques fundamentales, dos de los cuales están enfocados a presentar el marco teórico del trabajo. Estos son: *Redes de Interconexión* y *Tolerancia a Fallos*. El tercer y último bloque es el que describe el estado del arte actual de la tolerancia a fallos para redes de interconexión, explicando el enfoque pasado y presente de los grupos de vanguardia en este campo.

Capítulo 3. Análisis. En este capítulo se hace un análisis detallado del problema de lograr una red de interconexión tolerante a fallos. Partiendo de los conceptos introducidos en el capítulo anterior se analizan parte por parte los retos y desafíos propios del problema, se lo enmarca, delimita y define. Se expone además el ámbito de desarrollo que se corresponde con el proyecto de la tesina del máster y se introduce el ámbito futuro de la tesis doctoral.

Capítulo 4. Diseño. Ya definido y delimitado el problema, en este capítulo se detallan y fundamentan las elecciones de diseño y los enfoques elegidos. Se detalla el diseño del sistema y las soluciones propuestas, como así también la hoja de ruta a seguir para llevarlas adelante.

Capítulo 5. Implementación. Está dedicado a explicar los detalles de implementación, introducir y detallar las características de modelado de la herramienta utilizada (OPNET), y el porqué de la elección de la misma.

Capítulo 6. Experimentación y análisis de resultados. En este capítulo se detallan las pruebas que fueron realizadas para validar los modelos y desarrollos propuestos en este trabajo. Se describen además las características de simulación de OPNET, y, para finalizar, se presentan y analizan los resultados obtenidos de la realización de dichos experimentos.

Capítulo 7. Conclusiones. Para finalizar, se recogen las conclusiones globales del estudio, junto con las ideas con las que se pretende continuar trabajando en esta línea en el futuro.

Capítulo 2

Marco teórico y estado del arte

Este capítulo está dedicado a introducir los conceptos de *redes de interconexión* y *tolerancia fallos* necesarios para enmarcar los desarrollos del presente trabajo. Para finalizar, se describen las tendencias actuales –estado del arte– en el ámbito de la tolerancia a fallos para redes de interconexión.

2.1. Redes de Interconexión

En esta sección se introducen las redes de interconexión, explicando los parámetros y conceptos fundamentales que las definen.

Entre los parámetros que intervienen en la definición de una red de interconexión, los más importantes son la *topología*, que es la forma física como están interconectados los diferentes nodos del computador paralelo, el *control de flujo*, que es la manera como se administra el avance de la información de un nodo al siguiente, y el *encaminamiento*, que se refiere a la manera de determinar el camino que sigue un mensaje desde el nodo fuente hasta el destino a través de varios nodos intermedios.

2.1.1. Definiciones

Antes de empezar a exponer y desarrollar los conceptos de las redes de interconexión, se deben introducir algunas definiciones formales. Se empieza definiendo el concepto de red de interconexión:

> **Red de interconexión.** Es un sistema programable que transporta datos entre terminales (nodos) [1].
>
> La red es un sistema porque está compuesta por numerosos componentes: buffers, canales, encaminadores, conmutadores, y controladores que trabajan en conjunto para enviar datos; y es programable en el sentido de que realiza diferentes conexiones en distintos momentos.

Definido con claridad el concepto de red de interconexión, es posible introducir la definición de los parámetros que la definen y que luego se explican en profundidad:

> **Topología.** En el ámbito de redes, el término topología haca referencia al patrón de conexión y organización del conjunto de nodos y canales que forman la red de interconexión.
>
> **Encaminamiento.** Se refiere al método que emplea una red para determinar la trayectoria (*path*) utilizada por un paquete para ir de un nodo origen a un nodo destino.
>
> **Control de flujo.** Es el método mediante el cual se notifica al emisor la disponibilidad de espacio dentro del receptor en función de la cantidad máxima de información que puede ser envidada ininterrumpidamente.

Para poder estudiar el rendimiento de la red de interconexión, y analizar el efecto del tráfico de red y el impacto de los factores de diseño anteriores, es necesario contar con métricas estandarizadas. En ausencia de fallos, las medidas de rendimiento más importantes son la *latencia* y el *throughput* [1]:

> **Latencia.** Tiempo necesario para que un paquete atraviese la red, desde el momento en que la cabecera (*head*) del paquete llega al puerto de entrada hasta el momento en que la cola del paquete sale del puerto de salida.
>
> **Throughput.** Es la velocidad de transmisión de datos en bits por segundo que la red acepta por puerto de entrada. Se puede medir como una fracción de la capacidad máxima de la red.
>
> También se lo puede definir como el máximo tráfico que acepta la red, donde tráfico, o tráfico aceptado es la cantidad de información emitida por unidad de tiempo.

Los términos *camino* y *flit* son muy utilizados en las explicaciones de técnicas de encaminamiento y conmutación, por lo que se definen a continuación:

> **Camino (path).** Es un conjunto ordenado de canales $P = \{c_1, c_2, ..., c_k\}$, donde el nodo de salida del canal c_i es igual al nodo de entrada del canal c_{i+1}, el origen es la entrada al canal c_1, y el destino es la salida del canal c_k.
>
> **Flit.** Es la unidad de control de flujo de mensajes. Normalmente los buffers de entrada y salida de un encaminador son lo suficientemente grandes para almacenar varios flits.

2.1.2. Ámbito de aplicación

En la actualidad, las redes de interconexión de altas prestaciones se utilizan en diferentes sistemas dedicados tanto a la computación como a la comunicación. Debido a sus altas prestaciones, el rango en el cual estos sistemas pueden ser aplicados es muy amplio, aún así es posible diferenciar tres grandes grupos:

1. **Clusters.** Es básicamente un conjunto de computadores personales o de *workstations* conectados a través de una red de interconexión de alta velocidad.

   Debido a su relación coste/beneficio se popularizaron como alternativa a los grandes sistemas multiprocesadores en la década de los noventa. En la actualidad son utilizados en una amplia variedad de aplicaciones como por ejemplo en redes de almacenamiento, servidores de Internet, o gigantes como *Google*, *Hotmail* y *Amazon*.

2. **Computadores masivamente paralelos.** Más allá del avance de los nuevos tipos de sistemas de cómputo de altas prestaciones, los computadores masivamente paralelos (multicomputadores y multiprocesadores) no se han dejado de utilizar, ni mucho menos. En estos sistemas la red debe ofrecer unas latencias de comunicación reducidas, a fin de evitar el uso ineficiente de procesadores y demás recursos.

   Ejemplos de estos sistemas son los *BlueGene/L* y *BlueGene/P* de IBM, y el *Jaguar* y *RedStorm* de Cray.

3. **Encaminadores de altas prestaciones para IP.** El gran crecimiento de Internet en los últimos años explica el creciente interés que existe en la construcción de encaminadores de altas prestaciones para tráfico IP. El crecimiento del número de usuarios y la modernización de las aplicaciones (vídeo-conferencias, gaming, etc.) ha provocado una enorme demanda de ancho de banda.

2.1.3. Consideraciones de diseño

Como se ha expuesto en la Sección 1.1, las redes de interconexión constituyen una parte fundamental de los sistemas de cómputo de altas prestaciones actuales. Por este motivo, para poder estudiarlas y analizarlas se deben tener en consideración los parámetros, factores y requerimientos de diseño que se exponen en las siguientes secciones.

Parámetros de diseño

El diseño de las redes de interconexión se encuentra definido por un conjunto de parámetros estáticos y dinámicos que constituyen el llamado *espacio de diseño en redes de interconexión*.

En las Tablas 2.1 y 2.2 se listan algunos de estos parámetros y sus ejemplos, que serán introducidos en las próximas secciones.

Parámetros estáticos o físicos	
Parámetros	*Ejemplos*
Topología	Toro, malla, fat-tree, etc.
Componentes de red	Enlaces, encaminadores, conmutadores, etc.

Tabla 2.1: Parámetros de diseño estáticos de redes de interconexión

Parámetros dinámicos o de funcionamiento	
Parámetros	*Ejemplos*
Control de flujo	Basado en créditos, basado en marcas.
Técnicas de conmutación	Store-and-forward, wormhole, etc.
Encaminamiento	Determinístico, adaptativo, etc.

Tabla 2.2: Parámetros de diseño dinámicos de redes de interconexión

Factores que influyen en el diseño

Existen numerosos factores que tienen influencia en el diseño de una red de interconexión y que deben ser tomados en cuenta. Los más importantes son:

1. Requerimientos de prestaciones. Debido a que tanto la sincronización como la comunicación entre los procesos de los sistemas de computo de altas prestaciones se realizan a través de la red de interconexión, los requerimientos de prestaciones de la misma son de vital importancia. Los factores como la *latencia*, el *throughput* y el nivel de *saturación* son algunos de los mejores indicadores de performance.

2. Escalabilidad. Una arquitectura escalable es aquella en la que al agregar procesadores aumenta de manera proporcional su ancho de banda de memoria y de entrada/salida.

3. Posibilidad de expansión. Se refiere a que el tamaño o las capacidades del sistema puedan ser expandidos sin necesidad de realizar grandes modificaciones y gastos para lograrlo.

   Un ejemplo de redes que presentan dificultades de expansión son las que se basan en topologías de tipo hipercubo.

4. Particionabilidad. Se refiere a la posibilidad de particionar la red en subsistemas funcionales menores. Esto puede ser por motivos de prestaciones y/o seguridad.

5. Simplicidad. La simplicidad en el diseño de una red facilita el aprovechamiento y la maximización de sus recursos y capacidades, y también su entendimiento por parte de los clientes/usuarios. Puede determinar también las prestaciones (mas simple $\Rightarrow$ mas rápido).

6. Variación de las distancias. Este factor se relaciona con la variación de las distancias entre nodos según la implementación de la red. Estas variaciones pueden generar problemas de ruido electromagnético, acoplamientos, etc.

7. Restricciones físicas. Las tecnologías de empaquetamiento, cableado, y mantenimiento son las que presentan las principales restricciones físicas a tener en cuenta. Éstas tienen relación con la longitud de los cables, el control de las temperaturas de operación de los componentes, etc.

8. Fiabilidad y reparabilidad. Una red de interconexión debe ser capaz de enviar información de forma fiable. Además, debe poseer un diseño modular, para permitir actualizaciones "en caliente" y reparaciones.

9. Cargas de trabajo esperadas. El diseño de la red debe ser robusto para poder adaptarse y ofrecer un rendimiento "aceptable" ante diferentes tipos y niveles de cargas de trabajo.

10. Restricciones de costo. Las decisiones de diseño normalmente son compromisos entre el costo económico y los demás factores de diseño.

Requerimientos de diseño de las redes de interconexión

La red de interconexión debe permitir a un procesador enviar mensajes a cualquier otro procesador durante la ejecución de cualquier aplicación. Debe proveer por lo tanto, un mecanismo de encaminamiento capaz de hacer avanzar cada mensaje entre los diferentes nodos de la red de interconexión, hasta alcanzar el destino deseado. Este mecanismo, según su propio funcionamiento, seleccionará una o más trayectorias posibles para establecer la conexión entre el par de nodos origen-destino.

Lo mencionado anteriormente nos permite establecer una serie de requerimientos [1], [3] que deben satisfacerse en el diseño de una red de interconexión para cumplir con las características impuestas por los computadores paralelos de altas prestaciones.

- Requerimientos funcionales:
  - El protocolo de encaminamiento debe ser libre de *deadlock*, es decir que no provoque una situación indefinida en la que ningún mensaje de la red pueda avanzar hacia su destino.
  - Ningún paquete debe permanecer indefinidamente en la red, es decir que se debe evitar que un mensaje deambule eternamente por la red sin llegar nunca a su destino. En resumen, se debe evitar el *livelock*.
  - Un nodo no debe denegar indefinidamente la aceptación de un mensaje de un usuario. Esta situación se llama *starvation*.
- Requerimientos de prestaciones:
  - Un paquete siempre debe tomar la ruta más corta a su destino. La definición de ruta más corta puede entenderse como un concepto de distancia física topológica o de tiempo de comunicación.

- Se debe conseguir la mínima latencia posible. La latencia puede variar según las condiciones de tráfico en la red y a la aparición de fallos en la misma.
- Se debe conseguir el mayor throughput posible, que solo estará limitado por el ancho de banda disponible en el sistema.
- El mecanismo de encaminamiento debe adaptarse a las condiciones de tráfico y fallos de la red, y explotar el máximo ancho de banda de comunicaciones disponible en el sistema. Es decir, debe ser un mecanismo robusto y, además, explotar eficientemente los enlaces de la red.

2.1.4. Topologías

Cuando se habla de la topología de una red se hace referencia a la disposición estática de canales y nodos que la componen, es decir, los caminos por los que viajan los paquetes.

La elección de la topología es el primer paso que se debe tomar al momento de diseñar una red, debido a que, tanto las estrategias de encaminamiento, así como los métodos de control de flujo dependen fuertemente de ella. Esta elección está guiada por factores técnicos como el número y ancho de banda efectivo de los puertos, la longitud de cables requeridos, etc.; y por su relación costo/beneficio.

La topología de una red de interconexión se especifica en base a un conjunto de nodos N^* conectados por una serie de canales C. Los mensajes se originan y terminan en una serie de nodos terminales N donde $N \subseteq N^*$. Cada canal, $c = (x, y) \in C$, conecta un nodo origen, x, con un nodo destino y, donde x, $y \in N^*$. Esta definición de topología es equivalente a la de un grafo dirigido [1, cap. 3].

Los nodos de la red pueden actuar como fuentes (*source*) y sumideros (*sink*) de paquetes, como nodos encaminadores que se encargan de mover los paquetes desde los puertos de entrada a los puertos de salida, o como ambos.

Algunas de las propiedades básicas de las topologías son [3, cap. 1]:

Node degree (grado). Número de canales que conectan a un nodo con sus vecinos.

Diameter (diámetro). La distancia máxima entre dos nodos en la red.

Regularity (regularidad). Una red es regular cuando todos sus nodos tienen el mismo grado.

Symmetry (simetría). Una red es simétrica u homogénea si todos los nodos de la red tienen la misma visión del resto de la ella.

Las topologías que se utilizan en redes de interconexión se pueden clasificar en cuatro categorías básicas, según su estructura (Fig. 2.1): redes de *medio compartido*, redes *directas*, redes *indirectas* y redes *híbridas*.

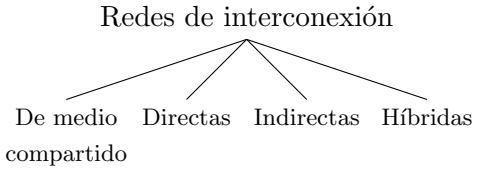


Figura 2.1: Categorías básicas de redes de interconexión

Redes de medio compartido

Las redes de medio compartido son las que poseen la estructura de interconexión más simple. En estas redes el medio de transmisión es compartido por todos los dispositivos de comunicación, y solamente un dispositivo puede utilizar la red a la vez [3, cap. 1].

Estas redes no soy muy utilizadas en computadores paralelos por sus bajas prestaciones. Se las pueden clasificar en redes de área local tradicionales o en "bus", que es un único medio compartido al que se conectan todos los nodos de la red.

En la Fig. 2.2 se puede observar la clasificación de las redes de medio compartido, y algunos ejemplos de las mismas en las Figs. 2.5(a), y 2.5(b)

Redes directas

Las redes directas están formadas por un conjunto de nodos, donde cada uno se encuentra directamente conectado a un subconjunto (normalmente pequeño) de otros nodos de la red.

Estas redes se caracterizan por la topología que forman los nodos, un algoritmo de encaminamiento que determina el camino a seguir entre cada par de nodos y la técnica de control de flujo que utilizan.

Como se observa en la Fig. 2.3, las redes directas se pueden dividir en ortogonales, es decir, simétricas, y en redes no ortogonales [3, cap. 1].

Dentro de las redes ortogonales o simétricas se encuentras las mallas abiertas, los toros y los hipercubos. Las dos primeras son las topologías de red más utilizadas en la actualidad, mientras que los hipercubos han caído en desuso por su elevado costo de escalabilidad.

En las Figs. 2.6(a) a 2.6(g) se pueden observar gráficamente varias de las topologías de redes directas ortogonales, mientras que en la Fig. 2.7(a) se observa un ejemplo de las no ortogonales.

Redes indirectas

Las redes indirectas o basadas en conmutadores en lugar de proveer conexiones directas entre nodos, establecen la comunicación entre cualesquiera dos nodos a través de varios conmutadores (*switchs*) [3]. Cada nodo de la red contiene un adaptador que lo conecta a un switch de la red, y cada switch tiene varios puertos bidireccionales.

En la Fig. 2.4 se puede observar la clasificación completa de las redes indirectas.

La interconexión de esos switchs definen varias topologías de red. Por un lado están las topologías regulares, entre las que se encuentran los *crossbar* (Fig. 2.8(a)) y las *MINs* (Fig.

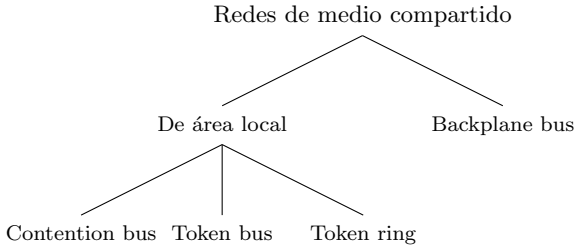


Figura 2.2: Clasificación de redes de medio compartido

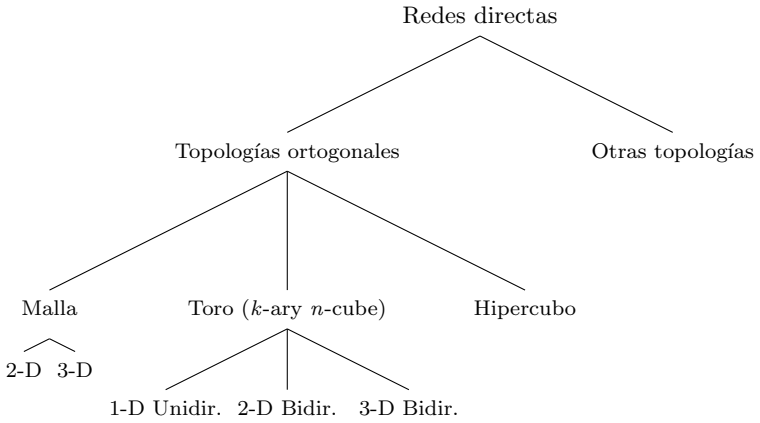


Figura 2.3: Clasificación de redes directas

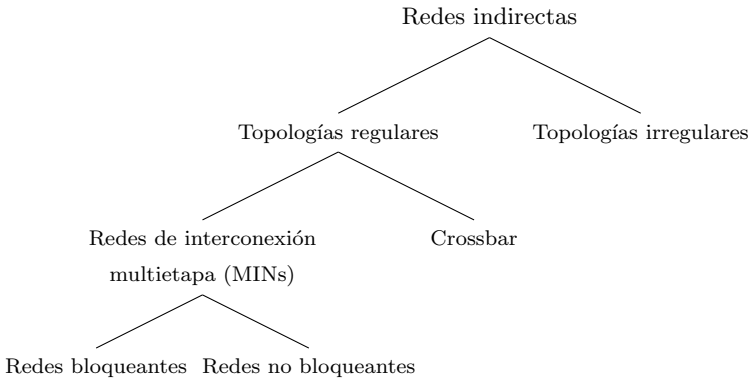
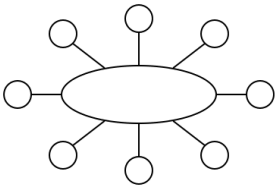
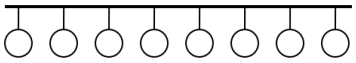


Figura 2.4: Clasificación de redes indirectas

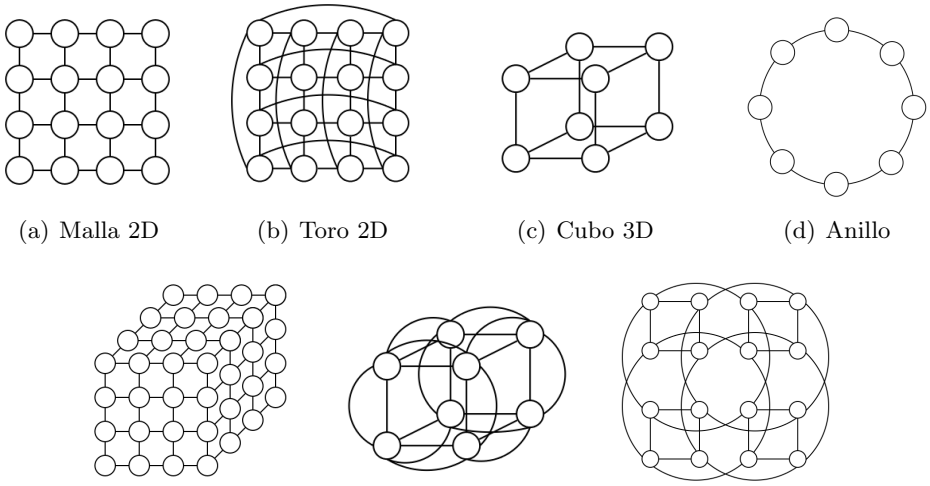


(a) Red de área local



(b) Red tipo bus

Figura 2.5: Topologías de redes de medio compartido



(a) Malla 2D (b) Toro 2D (c) Cubo 3D (d) Anillo

(e) Malla 3D (f) Toro 3D (g) Cubo 4D

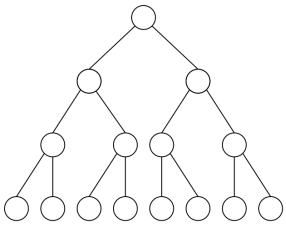
Figura 2.6: Topologías de redes directas ortogonales

2.8(b)), y por el otro las topologías irregulares.

Las redes de interconexión multiepata (MINs), se dividen en redes bloqueantes y redes no bloqueantes. Dentro de las últimas, la más conocida es sin dudas la *red de Clos*, mientras que por el lado de las bloqueantes se desataca la topología en forma de árbol conocida como *fat-tree* (Fig. 2.8(c)), cuya principal característica es el incremento de la cantidad de enlaces cerca de la raíz.

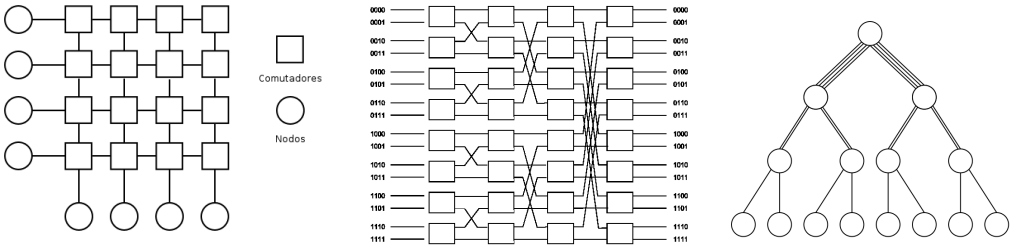
Redes híbridas

En general las redes híbridas combinan mecanismos de las redes de medio compartido, de las directas y de las indirectas. Por lo tanto, estas redes incrementan el ancho de banda respecto a las redes de medio compartido y reducen la distancia entre nodos respecto a las redes directas e indirectas.



(a) Árbol

Figura 2.7: Topología de red directas no ortogonal



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

Figura 2.8: Topologías de redes indirectas

Estas redes han vuelto a ganar aceptación en el último tiempo, gracias a las posibilidades de implementación que ofrecen las tecnologías ópticas. Entre estas nuevas posibilidades se encuentra la implementación de buses de alto rendimiento.

La clasificación de las redes híbridas se puede observar en la Fig. 2.9, y un ejemplo gráfico en la Fig. 2.10(a).

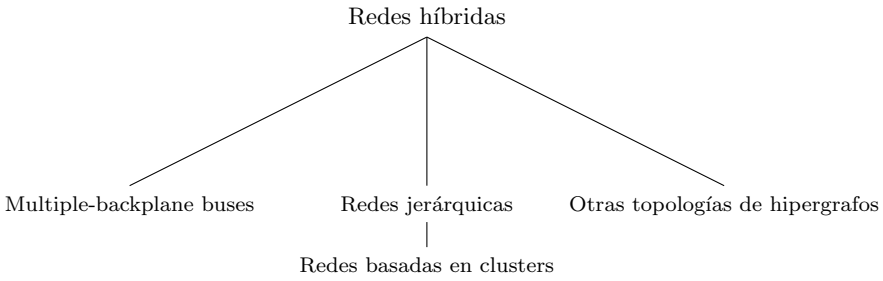
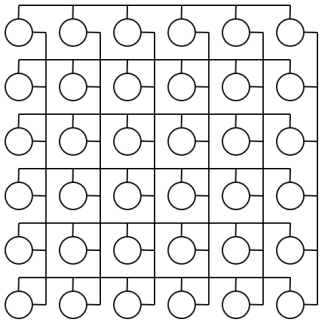


Figura 2.9: Clasificación de redes híbridas

2.1.5. Control de flujo

Actualmente, la gran mayoría de los dispositivos de red incorporan algún tipo de buffer a ambos extremos de un enlace (tanto emisor como receptor) para contener los datos enviados y recibidos, por lo que es necesario implementar algún mecanismo de control de flujo para garantizar que la capacidad del receptor no se desborde.

Mediante el control de flujo se notifica al emisor la disponibilidad o no de espacio en el



(a) Hipermalla 2D

Figura 2.10: Topología de red híbrida

receptor, en función de la cantidad máxima de información que puede ser enviada ininterrumpidamente. Los mecanismos más utilizados para este propósito se dividen en dos grupos: los basado en créditos y los basados en marcas.

- **Control de flujo basado en créditos.** En este mecanismo cada conmutador recibe inicialmente varios créditos para enviar mensajes a un conmutador vecino. En cada transmisión de datos, el emisor consume un crédito, deteniendo la emisión cuando ha consumido todos sus créditos. Cuando se libera espacio en el receptor, se envían nuevos créditos al emisor.

- **Control de flujo basado en marcas.** En este mecanismo el receptor detecta que su buffer de entrada está próximo a desbordarse, debido a que su nivel excede cierto valor, y envía una señal al nodo emisor para que éste detenga el envío de paquetes. Cuando el receptor tiene espacio suficiente en su buffer, envía al emisor otra señal para que restablezca el envío.

2.1.6. Técnicas de conmutación

Las técnicas de conmutación determinan como se realiza el avance de los paquetes desde el nodo origen hasta el nodo destino. Concretamente, definen cómo se establece el camino entre el nodo origen y el destino, y cómo se regula el avance de la información en cada nodo de la red. Las técnicas más aceptadas en el ámbito de los computadores de altas prestaciones son: *store-and-forward*, *virtual cut-through* y *wormhole*.

- **Store-and-forward.** En esta técnica el paquete se almacena completamente en cada nodo y entonces se transmite al siguiente nodo cuando el enlace de salida está libre.

  La principal desventaja de éste método es su alta latencia en el transporte de un paquete individual de un nodo a otro. Esta latencia es proporcional al producto de la longitud del paquete por el número de enlaces atravesados.

- **Virtual cut-through.** En esta técnica el paquete se transmite hacia el siguiente conmutador tan pronto como se recibe y se interpreta su cabecera, sin la necesidad de esperar a que se reciba el paquete completo. No obstante, el buffer debe disponer de espacio suficiente para almacenar todo el paquete debido a que existe la posibilidad de que el canal de salida esté ocupado. Por este motivo, en caso de ocupación se comporta de la misma manera que el *store-and-forward*.

  La ventaja de esta técnica radica en que permite el rápido avance de los paquetes una vez que se procesa su cabecera.

- **Wormhole.** Esta técnica, al igual que en la de *virtual cut-through*, reenvía inmediatamente el paquete haca el puerto de salida antes de recibirlo por completo. Sin embargo, en caso de que el puerto de salida esté ocupado, el paquete puede no almacenarse completamente en el buffer asociado al puerto de entrada, sino que puede quedar almacenado a lo largo de todos los buffers de los conmutadores visitados. Conforme los buffers visitados queden sin espacio, el protocolo de control de flujo bloqueará el avance del paquete. Con este mecanismo, el tamaño del paquete no esa limitado, requiriendo además poco espacio de almacenamiento en los buffers asociados a los puertos de entrada de los conmutadores.

  Esta técnica posee la desventaja de que como los flits que componen el mensaje, excepto el flit de cabecera, no contienen información del destino, no pueden ser intercalados con los flits de otros mensajes. Por lo que cuando un mensaje se bloquea, todos los flits del mensaje dejan de avanzar y bloquean el progreso de todos los mensajes de necesitan los enlaces que están ocupando.

  Como ventaja se puede decir que esta técnica es capaz de reducir significativamente la latencia de los mensajes y requiere capacidades de almacenamiento temporal mínimas.

2.1.7. Encaminamiento

El mecanismo de encaminamiento (*routing*) de una red es el que determina el camino que toman los paquetes desde un nodo fuente a un nodo destino. Es por esto que el mecanismo de encaminamiento deberá obtener información de la topología de red para poder seleccionar el o los caminos apropiados.

Dicha información se debe generar y procesar siguiendo cierta planificación o estrategia definida en un *algoritmo de encaminamiento*, que establezca el mecanismo mediante el cual se guían los paquetes a sus destinos a través de la red. Por lo tanto, el principal objetivo del algoritmo de encaminamiento es seleccionar el camino que represente el mínimo retardo total para cada paquete, intentando seleccionar las rutas de manera que se eviten las sobrecargas y los fallos en algunas de las lineas de comunicación, a partir de la utilización de otras rutas inactivas.

La elección del método de encaminamiento es uno de los puntos críticos en el diseño de redes de interconexión. Se espera que un buen método de encaminamiento sea capaz de balancear la

carga a través de los diferentes caminos de la red, aún en presencia de patrones de tráfico no uniformes. A medida que este balanceo de carga mejora, el *trhoughput* de la red se aproximará más al ideal. Por otra parte, un mecanismo de encaminamiento bien diseñado siempre ha de mantener las longitudes de los caminos lo más cortas posibles, reduciendo el número de saltos y la latencia total de los mensajes. Sin embargo, a menudo, el objetivo de lograr un encaminamiento mínimo (escoger siempre el menor camino) se contradice con los objetivos de balancear la carga y maximizar el *trhoughput*.

Varias de las propiedades de las redes de interconexión son una consecuencia directa de los algoritmos de encaminamiento que éstas utilizan. Algunas de las propiedades más importantes son:

- **Conectividad.** Es la habilidad de encaminar los paquetes desde cualquier nodo origen a cualquier nodo destino.
- **Adaptividad.** Es la habilidad de encaminar los paquetes a través de caminos alternativos ante la presencia de contenciones o fallos en los componentes.
- **Libertad de deadlock y livelock.** Es la habilidad de garantizar que los paquetes no se van a bloquear o vagar eternamente por la red.
- **Tolerancia a fallos.** Es la habilidad para encaminar paquetes en presencia de componentes fallidos. Aunque parezca que la tolerancia a fallos implica adaptividad, esto no es necesariamente cierto. La tolerancia a fallos se puede lograr sin adaptividad mediante el encaminamiento de paquetes en dos o más fases, almacenandolos en nodos intermedios.

Hasta aquí se han introducido los algoritmos de encaminamiento y se han expuesto sus propiedades más importante. Ahora es posible clasificar estos algoritmos.

Los algoritmos de encaminamiento se pueden clasificar en diferentes categorías según una serie de criterios [3, cap. 4]. La Tabla 2.3 muestra las diferentes alternativas posibles. El primer criterio es el **número de destinos** a los que se envía el mensaje, que puede ser uno solo, en cuyo caso tenemos comunicación *unicast*, o varios, en cuyo caso hablaremos de comunicaciones *multicast* o *broadcast*, según se destine a un subconjunto de varios nodos o a todos los nodos de la red.

El segundo criterio es quién y donde se toman las **decisiones de encaminamiento**, es decir, la determinación del camino que seguirán los mensajes. Se puede hacer de manera *centralizada*, en el caso que exista un nodo especial que centralice las decisiones, o *no centralizado*, donde las decisiones se toman localmente entre todos los nodos de la red. En este caso de encaminamiento no centralizado, existen dos posibilidades, dependiendo de si la determinación del camino a seguir la toma el nodo fuente que envía el mensaje o si son los nodos que recorre el mensaje los que deciden de manera distribuida. La alternativa multifase es una combinación de las dos anteriores. En ella el camino se divide en diversas fases determinadas por destinos intermedios

decididos por el nodo fuente y en cada una de las fases se utiliza encaminamiento distribuido. La **implementación** de la determinación del camino puede realizarse mediante una *tabla* que almacena los caminos o mediante un *algoritmo* que calcula el camino a recorrer.

La categoría más importante de clasificación de los algoritmos de encaminamiento es la **adaptabilidad**. Este aspecto hace referencia a si se tiene en cuenta el tráfico presente en la red, la ocupación de los enlaces y/o los fallos para determinar los caminos, o no se tiene en cuenta éste estado. En el primer caso tenemos algoritmos *adaptativos* y en el segundo caso tenemos algoritmos *deterministas o estáticos*, en los cuales las decisiones de encaminamiento no se basan en mediciones o estimacines del tráfico existente en un instante dado, sino que sólo dependen de los nodos fuente y destino, en el sentido de que dado un par de nodos se determina el camino de manera estática según una topología determinada.

Los algoritmos *adaptativos* son capaces de cambiar o adaptar las rutas a las condiciones de tráfico de la red. Dentro de éstos algoritmos, pueden darse varias alternativas respecto de la información de la red, la progresividad, la minimalidad y el número de caminos que utilizan. Si la determinación del tráfico de la red se realiza utilizando información únicamente del nodo en tránsito, tenemos encaminamiento aislado; en caso de que se use información de una vecindad de nodos, tenemos el caso de encaminamiento con información local. En los algoritmos progresivos, la cabecera siempre avanza reservando nuevos enlaces, en los algoritmos de backtracking es posible que la cabecera desande parte del camino realizado para tomar nuevos caminos. Respecto de la minimalidad, los algoritmos adaptativos provechosos siempre utilizan caminos de longitud mínima de manera de que cada paso acerca los mensajes al destino, mientras que los algoritmos no mínimos permiten alargar los caminos mediante un alejamiento respecto del destino. Finalmente, existen algoritmos que permiten configurar caminos utilizando todos los enlaces de la red, mientras que otros sólo permiten utilizar un cierto subconjunto parcial de los enlaces de la red.

2.1.8. Balanceo Distribuido del Encaminamiento (DRB)

Tal y como se explicó en los objetivos (Sección 1.4), nuestro trabajo tiene como punto de partida el *método de balanceo distribuido del encaminamiento* (DRB, por sus siglas en inglés) [7], [10], [9], [8], [6], [12].

DRB es un método que persigue distribuir los caminos que usan los canales (definidos como conexiones entre procesos) en la red de interconexión. Para conseguirlo, usa una técnica basada en la expansión de caminos controlada por la propia carga de la red de interconexión. El principal objetivo del método es balancear uniformemente el tráfico de comunicaciones sobre todos los caminos de la red de interconexión completa.

El método se basa en la creación de caminos alternativos simultáneos entre cada par origen y destino con objeto de mantener una baja latencia de los mensajes. Originalmente DRB define cómo crear los caminos alternativos para expandir los caminos simples originales, y cuándo y cómo usarlos dependiendo de la carga de tráfico de la red de interconexión. Esta distribución

Algoritmos de encaminamiento		
Número de destinos		
Unico		
Múltiple		
Decisiones de encaminamiento		
Centralizadas		
No centralizadas	Nodo fuente	
	Distribuida	
	Multifase	
Implementación		
Consulta en tabla		
Cálculo algorítmico		
Adaptabilidad		
Determinísticos o estáticos		
Adaptativos	Información de la red	Aislado
		Local
	Progresividad	Progresivos
		Backtracking
	Minimalidad	Provechosos
		No mínimos
	Número de caminos	Todos los enlaces
		Subconjunto de enlaces

Tabla 2.3: Clasificación de los algoritmos de encaminamiento

está controlada por el nivel de carga del camino "multicarril" (conjunto de caminos simples). Estos nuevos caminos (que pueden ser de distancia mínima o no) harán uso de los enlaces disponibles de los encaminadores. Se produce entonces un efecto colectivo, pues esta expansión se realiza para todos los pares origen-destino que también interaccionan entre si.

En el párrafo anterior se ha descrito el objetivo original de DRB, que se enfoca en la resolución de los problemas de congestión de las redes de interconexión (*hot-spots*). Nuestro objetivo es extender éste método para desarrollar políticas de encaminamiento tolerantes a fallos a partir de él.

Para introducir los conceptos y métodos propios de la tolerancia a fallos, se necesita adaptar DRB de forma de que sea capaz de crear y utilizar caminos alternativos, no solo ante la aparición de congestión en la red, sino también ante la presencia de fallos en la misma.

Se debe señalar que el comportamiento de DRB:

- Es un comportamiento para todos los pares origen-destino.
- Es dinámico, es decir, en función de las altas latencias se dispara el mecanismo de crear nuevos caminos para los canales saturados, lo que influencia a los no saturados.

El objetivo original de DRB se puede desglosar en varios objetivos menores complementarios:

- La reducción de la latencia de los mensajes bajo un cierto valor umbral por medio de la adaptación dinámica del número de caminos que utilizan los canales, mientras se mantiene una latencia uniforme para todos los mensajes. Este objetivo se consigue optimizando el uso de los recursos de la red de interconexión con objeto de minimizar los retardos de comunicación.

- La minimización del alargamiento del camino, de manera que se ajuste lo más posible al camino de distancia mínima. Este punto es importante para las técnicas de control del flujo tipo *wormhole* y *cut-through* porque alargar el camino supone incrementar el uso de ancho de banda y el número de puntos de colisión, que es un factor muy importante. Para redes controladas bajo *store-and-forward* también es muy importante pues el retardo de transmisión depende directamente de la longitud del camino que recorre el mensaje.

- La optimización del uso de los recursos de la red, de todos los recursos disponibles en todo momento, en general, y de los enlaces de los encaminadores que inyectan y reciben mensajes, en particular, mediante la distribución equitativa de los mensajes entre todos ellos.

Conceptualmente, este método mide el estado de la carga en todas las conexiones entre pares origen-destino (monitorizando la latencia); al detectar congestión se notifica al(los) nodo(s) que inyectan mensajes para que configuren nuevas trayectorias posibles y redistribuyan el tráfico según su estado de carga.

En concordancia con lo anteriormente expuesto, el encaminamiento DRB se divide en tres fases:

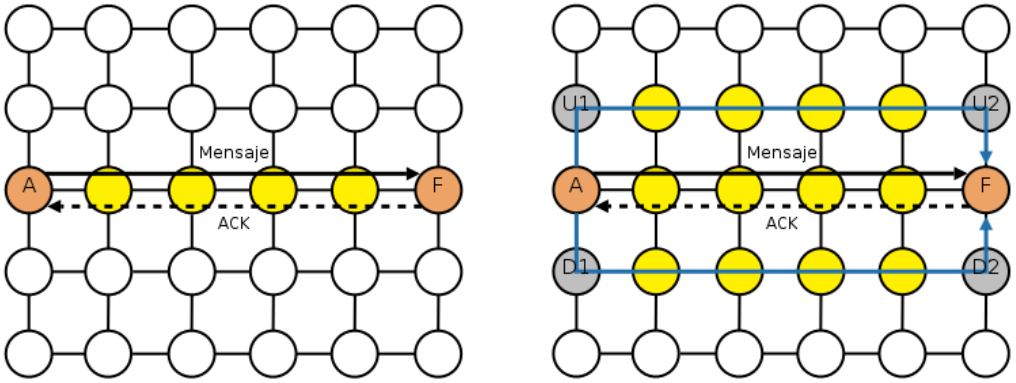
1. Monitorización de la carga de tráfico de la aplicación. Los mensajes almacenan la latencia que experimentan en su camino y esta información se reenvía hacia atrás hasta el nodo origen mediante un mensaje de reconocimiento.

2. Configuración dinámica de las trayectorias alternativas. Cuando el nodo origen recibe la información de latencia se modifica el número de posibles trayectorias si ésta supera un umbral.

3. Selección de caminos múltiples. Teniendo en cuenta la latencia en los caminos y su valor se genera un distribución de probabilidad para seleccionar las trayectorias más convenientes para el envío.

En las Figs. 2.11(a) y 2.11(b) se puede observar un ejemplo del funcionamiento de DRB para el par de nodos origen-destino *A-F*. En este ejemplo, se abren 2 caminos adicionales: el *A-U1-U2-F* y el *A-D1-D2-F*.

La Fig. 2.11(a) representa la primer fase del método: monitorización de latencia y notificación al nodo origen; mientras que las Fig. 2.11(b) representa las dos fases siguientes: configuración y selección de trayectorias alternativas.



(a) Monitorización de latencias y notificación (b) Configuración y selección de trayectorias

Figura 2.11: Funcionamiento de DRB para 3 caminos

Mejoras al balanceo distribuido del encaminamiento (DRB-E)

El balanceo distribuido del encaminamiento utiliza información del par origen-destino, por lo que depende de la información del estado de los caminos que el nodo destino envía hacia atrás al nodo origen. Esta dependencia implica la existencia de restricciones de tipo temporal, relacionadas con la recepción y disponibilidad de la información de latencia de estos caminos.

Estas restricciones son las responsables de los puntos débiles de DRB:

- Si la longitud del camino es considerablemente grande, y los problemas de congestión se encuentran relativamente cerca del nodo origen, se generan atascamientos y rápidas perdidas de rendimiento. Esto se debe a que el tiempo que se debe esperar a que la información de congestión del camino llegue al nodo origen es demasiado grande.
- Durante el tiempo que el nodo origen tarda en recibir la información del estado del camino y "abrir" otros nuevos caminos alternativos, la comunicación se sigue realizando por el camino original, y por lo tanto se siguen enviando mensajes a la zona congestionada, contribuyendo a la congestión.
- Si el tráfico de la red responde a patrones de tipo de envíos a ráfagas, puede suceder que el nodo origen reciba la información sobre la congestión demasiado tarde, y abra caminos para solucionar problemas que ya han desaparecido.

La solución a estos problemas o puntos débiles de DRB da origen a un conjunto de mejoras a DRB que se agrupan bajo el nombre de *Enhanced DRB* (DRB-E).

Las mejoras se basan principalmente en la distribución de las funcionalidades de DRB. Más especificamente se permite a los encaminadores intermedios:

1. Analizar la latencia a medida que el mensaje avanza por la red.

2. Abrir “vías de escape” en tiempo real e *in situ*.

3. Notificar a los nodos origen de la congestión.

Al analizar la latencia a medida que el mensaje avanza por la red, en lugar de en el nodo destino, es posible mejorar los tiempos de detección de la congestión, y reaccionar más rápidamente. Esta reacción rápida se basa en la posibilidad de notificar la congestión a los nodos origen a partir de los encaminadores intermedios.

El abrir vías de escape *in situ*, permite dar salida a los primeros mensajes que se encuentran con la congestión y a los que el enfoque tradicional de DRB no puede brindarles una solución.

Estas mejoras de DRB-E hacen posible:

- Minimizar el tiempo de reacción.
- Hacer más rápidas la detección y notificación de la congestión.
- Migrar de manera inmediata los caminos en los encaminadores.
- Mejorar el rendimiento en situaciones de patrones de tráfico con envíos de ráfagas de mensajes.

En las Figs. 2.12(a), 2.12(b) y 2.12(c) se muestra un ejemplo del funcionamiento de DRB-E ante la presencia de congestión temprana entre los nodos origen-destino *A*-*F*. El símbolo **X** representa la congestión del camino en el puerto de salida del encaminador *In*.

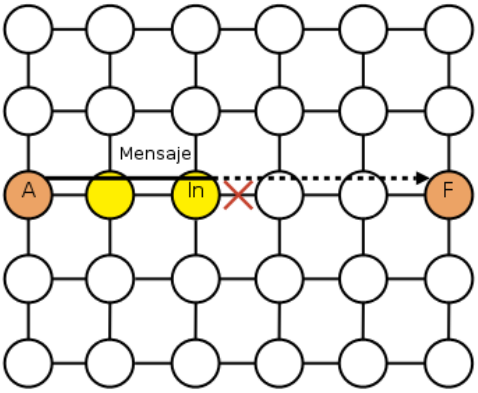
Si bien ambos métodos, tanto de DRB como de DRB-E, están orientados a tratar los problemas de congestión en redes de interconexión, los mismos pueden ser adaptados para incluir los métodos de tolerancia a fallos. Concretamente, DRB-E es el punto de partida *real* del trabajo de investigación aquí presentado. Esto se debe a que el DRB original no fue diseñado para tratar la problemática de los fallos en las redes de interconexión; más concretamente, al ser dependiente de la información que el nodo destino envía hacia atrás al origen, un fallo en el camino entre estos inhabilita el método. Mientras que las mejoras de DRB-E, más precisamente la apertura de vías de escape y notificación temprana al origen, introducen los mecanismos necesarios para implementar políticas de tolerancia a fallos.

En los capítulos siguientes se analizan y tratan con más profundidad estas temáticas.

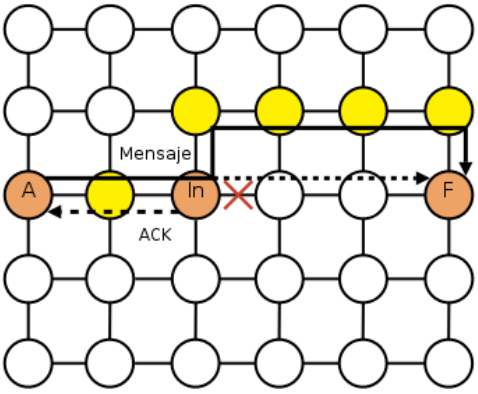
2.2. Tolerancia a fallos

Las computadoras –hardware y software trabajando en conjunto– son, sin lugar a dudas, los sistemas más complejos creados por los seres humanos hasta el momento, y esta complejidad crece día a día [20].

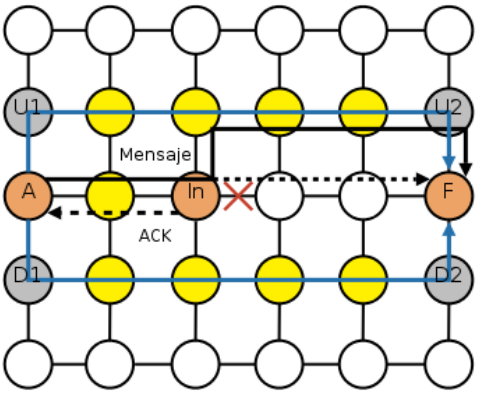
Tal como se ha explicado en la Sección 1.2, la creciente complejidad de los sistemas de cómputo y el incremento del número de dispositivos que los componen, lleva a que en algunas



(a) Detección temprana de la congestión



(b) Apertura de la vía de escape e informe al origen



(c) Configuración y selección de trayectorias

Figura 2.12: Funcionamiento de DRB-E para 3 caminos

ocasiones el *Mean Time Between Failures (MTBF)* de los computadores de altas prestaciones sea menor que el tiempo medio de ejecución de algunas aplicaciones actuales. Es por esto que es necesario diseñar y desarrollar procedimientos y técnicas que permitan que las redes de interconexión brinden un nivel de servicio aceptable aún en presencia de uno o más fallos. Se dice que una metodología o un sistema es n-fault tolerant si es capaz de tolerar cualquier combinación de n fallos.

La tolerancia a fallos tiene como objetivo principal dotar a los sistemas de la capacidad de proveer las funcionalidades y servicios para los cuales fueron diseñados, a pesar de la presencia de fallos en los mismos. Por lo tanto, se dice que un sistema es tolerante a fallos si es capaz de enmascarar la ocurrencia de fallos, valiéndose para esto de la adecuada utilización de la redundancia, en cualquiera de sus variantes. El concepto e importancia de la redundancia en el mundo de la tolerancia a fallos se explica con detalle en la Sección 2.2.4.

Como no es posible anticipar ni eliminar por completo todos los fallos que pueden ocurrir durante el funcionamiento de un sistema, se debe asumir que algunas averías (*failures*) ocurrirán ocasionalmente en los dispositivos que lo componen. La avería de uno o más componentes se traducen en fallos del sistema, y es aquí donde se hace evidente la necesidad de contar con métodos de tolerancia a fallos.

La tolerancia a fallos puede, y es, aplicada a varios niveles en un sistema de cómputo. Se tratan a estos como sistemas por capas, con las aplicaciones en las capas superiores y el hardware en las inferiores.

2.2.1. Requisitos generales

El diseño de cualquier metodología o procedimiento que pretenda resolver el problema de la tolerancia a fallos debe cumplir con los siguientes requisitos:

- Tolerar el mayor número y combinación posible de fallos, sin descuidar los dos puntos siguientes.
- En ausencia de fallos: no degradar de ninguna manera el rendimiento del sistema.
- En presencia de fallos: lograr la menor degradación posible en el rendimiento del sistema, según los tipos y relevancia de los mismos.
- Que la solución no represente un coste adicional demasiado grande (en recursos, tiempo, etc.).
- Que la solución no implique, en la medida de lo posible, sacrificar recursos funcionales operativos (es decir, que no presenten fallos).

2.2.2. Fallo, error y avería

Si bien en el lenguaje cotidiano se utilizan de manera indistinta, es importante definir correctamente los términos *fallo*, *error* y *avería* y la diferencia que existe entre ellos (ver Fig. 2.13).

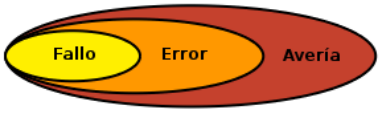


Figura 2.13: Relación entre fallo, error y avería

Fault (fallo). Se asocia con la noción de defecto. Se definen los *fallos* como los defectos que tienen el potencial de generar *errores*.

A pesar de que tienen el potencial de generar *errores*, puede que no generen ningún *error* durante el periodo de observación. En otras palabras, la presencia de *fallos* no asegura la ocurrencia de *errores*.

Error. Es la manifestación de un *fallo*; la causa de un *error* es un *fallo*. Si existe un *error* en el estado del sistema, existe una secuencia de acciones que puede ser ejecutada por el sistema y ocasionar una *avería (failure)*.

Failure (avería). Ocurre cuando el comportamiento del sistema se desvía de lo definido en su especificación. Es decir, un sistema presenta una *avería* cuando no puede proveer el servicio deseado o esperado.

2.2.3. Fases de la tolerancia a fallos

Como se ha mencionado anteriormente, un sistema tolerante a fallos intenta prevenir la avería del sistema a pesar de que se averíen algunos de los dispositivos que lo componen.

Así como el diseño de un sistema depende de los requerimientos del mismo, diseñar un sistema tolerante a fallos también depende de las necesidades y funcionalidad del sistema. Por este motivo, no existe una técnica general para "agregar" tolerancia a fallos a un sistema. Sin embargo, se pueden identificar algunos principios que pueden ser de utilidad en el momento de realizar el diseño del mismo [19].

Dentro de la tolerancia a fallos se pueden identificar cuatro fases:

1. **Detección del error.** El primer paso de cualquier actividad de tolerancia a fallos es detectar el error. Debido a que los errores se encuentran definidos por el estado de un sistema, es posible realizar controles (monitorización) para verificar si existe o no un error. A partir de la presencia de errores se pueden deducir las averías y fallos (ver Fig. 2.13).

   Los controles para detectar errores pueden ser de diversos tipos. Entre los más utilizados se encuentran: de *replicación*, de *timing*, de *razonabilidad*, de *diagnóstico*, y *estructurales y de control de código*.

2. **Contención del daño.** Según el método de detección de errores y el intervalo de monitorización utilizados, pueden haber retrasos en la detección de los errores y estos se pueden propagar a otras partes del sistema provocando la corrupción de parte del sistema. Antes de corregir el estado erróneo del sistema, se debe determinar hasta donde se ha extendido la corrupción. Ese es el objetivos de ésta fase.

3. **Recuperación del error.** Una vez detectado el error y su extensión, es tiempo de solucionarlo o tratarlo adecuadamente. En caso contrario, el estado erróneo puede ocasionar la avería del sistema en el futuro. Existen dos técnicas generales de recuperación: *backward* y *forward*.

   En la detección de tipo *backward* se restaura el estado del sistema a un estado anterior, con la esperanza de que en el estado anterior no hayan errores.

   En la detección de tipo *forward* se intenta "ir hacia adelante" intentando llevar al sistema a un estado libre de errores aplicando las acciones correctivas necesarias.

4. **Tratamiento del fallo y continuidad del servicio.** Las tres fases anteriores se centran en los errores. Se detecta, contiene y remueve el error. Esto puede ser suficiente para fallos de tipo transitorio, pero no para fallos permanentes ya que estos aún permanecen en el sistema y pueden volver a ocasionar errores e incluso averías. Esta fase se encarga de las importantes tareas de identificar y aislar los componentes fallidos.

2.2.4. Concepto e importancia de la redundancia

La redundancia es el pilar fundamental de la tolerancia a fallos: **no puede haber tolerancia a fallos sin redundancia**.

> **Redundancia.** Es la propiedad de tener más recursos de los que son mínimamente necesarios para realizar normalmente una tarea (en ausencia de averías).

Es a través del aprovechamiento de la *redundancia* como se logra mantener un nivel de funcionalidad "aceptable" en un sistema luego de que se produzcan los fallos en el mismo (normalmente generados por averías o en uno o más componentes del sistema).

Existen cuatro tipos fundamentales de redundancia:

1. **De hardware.** Comprende los componentes de hardware que se agregan al sistema para soportar la tolerancia a fallos.

2. **De software.** Incluye todos los programas e instrucciones que se utilizan para soportar la tolerancia a fallos. Se utiliza principalmente para tratar los fallos de software.

3. **De información.** Se basa en la inclusión de información extra, como por ejemplo los códigos de detección y corrección de errores.

4. **De tiempo.** Se refiere al tiempo extra necesario para realizar tareas propias de la tolerancia a fallos, como ejecutar una (o varias) instrucciones varias veces.

Los fallos de hardware normalmente son atacados mediante el uso de las redundancias de hardware, de información o de tiempo, mientras que los fallos de software se tratan con la redundancia de software.

2.2.5. Clasificación de fallos

Los fallos de hardware pueden ser clasificados de acuerdo a varios aspectos. Según su duración, pueden ser: *permanentes (permanent)*, *transitorios (transient)* o *intermitentes (intermittent)* [20].

> **Fallo permanente.** Refleja la caída (salida de servicio) permanente de un componente.
>
> **Fallo transitorio.** Es el que causa el mal funcionamiento de un componente durante un tiempo; luego de ese tiempo desaparece y el funcionamiento del componente vuelve a ser correcto.
>
> **Fallo intermitente.** Nunca desaparece por completo; oscila entre estados de latencia y actividad. Cuando el fallo se encuentra en estado latente, el componente funciona con normalidad; cuando el fallo está activo, el componente funciona de manera errónea.
>
> Se podrían considerar a los fallos *intermitentes* como manifestaciones crónicas de los fallos *transitorios*.

Otra clasificación de los fallos de hardware es la que distingue entre fallos *benignos (benigns)* y *maliciosos (malicious)* [20].

> **Fallo benigno.** Es el fallo que simplemente causa la caída o "muerte" de un componente. Son más simples de tratar (relativamente hablando). Son también llamados fallos de *fail-stop*.
>
> **Fallo malicioso.** Es el fallo que hace que el componente produzca resultados o salidas similares a las reales, pero incorrectas. Es decir, logra que los componentes actúen de manera "maliciosa". Son también llamados fallos *bizantinos (byzantine)*.

2.2.6. Modelo de fallos

Un *modelo de fallos* especifica la información sobre los patrones de fallos de los componentes y, en nuestro caso, el comportamiento que se espera que tengan los procesadores y dispositivos de red cuando se produzcan dichos fallos [3].

La definición de los *modelos de fallos* es uno de los puntos fundamentales –quizás el más importante– al momento de atacar el problema de la tolerancia a fallos en redes de interconexión, debido a que las estrategias utilizadas para resolver situaciones como *deadlock, livelock y starvation* dependen en gran medida de estas definiciones.

Definición de modelos de fallos

Desde el punto de vista del nivel en que se diagnostican los fallos en los componentes, la clasificación de modelos de fallos es la siguiente [3]:

> **Node failure.** Los fallos de elementos de procesamiento y los encaminadores asociados a ellos son considerados como *node failures*. Cuando se producen este tipo de fallos, se deben marcar como fallidos (en los encaminadores adyacentes) todos los enlaces físicos conectados al nodo que presenta el fallo.
>
> También son considerados como *node failures* los fallos que se producen en la unidad de control del encaminador y/o en el nodo de procesamiento asociado a dicho encaminador. Incluso se pueden considerar algunos errores graves de software de la capa de mensajes como *node failures*.
>
> **Link failure.** Los fallos de enlaces se corresponden con los fallos de los canales de comunicación. Cuando un canal físico falla, todos los canales virtuales de ese canal físico en particular son marcados como fallidos.
>
> Los fallos de los controladores de los enlaces y/o los *buffers* de los canales virtuales también son considerados como *link failures*.

Críticas a las definiciones de modelos de fallos

Las definiciones de modelos de fallos anteriormente expuestas representan, sin dudas, un buen punto de partida y la mayor parte –por no decir todos– los artículos recientes del área de tolerancia a fallos en redes de interconexión se basan en ellas.

Vale aclarar que dichos artículos, en su amplia mayoría, tratan el problema desde el punto de vista de los algoritmos de encaminamiento tolerantes a fallos, y no abordan el problema de una red tolerante a fallos.

Otro aspecto importante a tener en cuenta es que prácticamente todos los artículos y trabajos que utilizan estas definiciones pertenecen al grupo del autor de las mismas, el GAP de J. Duato, y al Simula Lab noruego (que mantiene una estrecha colaboración con el anterior).

Sin embargo, estas definiciones de modelos de fallos no son del todo completas y realizan una simplificación quizás excesiva de los fallos y su comportamiento.

Críticas a la definición típica del modelo de fallos de nodo:

1. Manejo de buffers: no se toma en cuenta la información almacenada en los *buffers* de los encaminadores en los que se producen los fallos. Resulta ilógico y muy simplista considerar que en el momento de producirse un fallo en un nodo, éste no tenga información almacenada en algunos de sus *buffers*.

2. Mensajes en tránsito: otro de los problemas que no se consideran es qué hacer con los mensajes que ya estaban en tránsito hacia los nodos donde se ha producido un fallo, y también los nuevos mensajes que se puedan generar (en el caso en que la información de los fallos no sea compartida globalmente por todos los nodos de procesamiento y dispositivos de la red). Esto puede generar *deadlocks* y sobre todo *livelocks*.

3. Mensajes erroneos: otro aspecto que se ha dejado de lado en esta definición es la posibilidad de que un nodo que presenta fallos genere mensajes erroneos y pueda llegar a introducir un gran número de los mismos llegando incluso a saturar enlaces o la red misma.

4. Mensajes corruptos: en relación con la crítica anterior, se encuentra la posibilidad de que los nodos corrompan los mensajes durante el periodo de "manipulación" de los mismos.

Críticas a la definición típica del modelo de fallos de enlaces:

1. Pérdida total o parcial: no considera el caso en que los fallos se produzcan con información en tránsito en los enlaces.

2. Corrupción de información: no considera el caso en que los fallos no eliminen (ni pierdan) la información en tránsito, sino que por su mal funcionamiente corrompan la información en tránsito.

2.2.7. Medidas de tolerancia a fallos para redes de interconexión

Para poder medir la robustez de una red de interconexión frente a distintos tipos y combinaciones de fallos, y realizar comparaciones de dicha red con y sin los métodos de tolerancia a fallos se necesitan medidas. Estas medidas deben permitir medir no solo la robustez de la red, sino también la degradación que sufre la misma cuando se producen los fallos.

A continuación se introducen varias medidas que pueden ser de gran utilidad [20, cap. 4].

Medidas tradicionales

MTTF. *Mean Time to Failure*, tiempo medio hasta la ocurrencia de un fallo.

MTBF. *Mean Time Between Failures*, tiempo medio entre la ocurrencia de dos fallos consecutivos. $MTBF = MTTF + MTTR$, donde MTTR es el tiempo medio de reparación (*Mean Time to Repair*).

Availability. $A(t)$ es la fracción promedio de tiempo en el intervalo $[0, t]$ en que el sistema esta funcionando. La *disponibilidad de largo plazo* (*long-term availability*) se puede definir como la probabilidad de que el sistema esté funcionando en un punto de tiempo aleatorio $A = MTTF/MTBF$

Medidas propias de la teoría de grafos

Si se representa la red como un grafo, donde los procesadores y dispositivos de red serían los nodos (vértices) y los enlaces las aristas (arcos), se pueden utilizar las siguientes medidas propias de la teoría de grafos [20]:

Node and Link Connectivity. Se define como el mínimo número de nodos (o enlaces) que deben ser removidos para desconectar el grafo (cuando se remueve un nodo, todos los enlaces incidentes en él son también removidos). Claramente, mientras mayor es la conectividad, más resistente es la red a fallos.

Diameter Stability. La *distancia* entre nodos origen y destino de una red se define como el menor número de enlaces que debe atravesar un mensaje desde el origen al destino. El *diámetro* de una red es la mayor distancia entre dos nodos cualesquiera. *Diameter Stability* hace referencia a cuan rápido se incrementa el *diámetro* de la red cuando se produce un fallo. Una instancia determinística de dicha medida es la *persistencia*, que es el menor número de nodos que deben fallar para que se incremente el *diámetro* de la red.

Medidas propias de las redes de computadores

Las medidas propias de las redes de computadoras permiten expresar el nivel de degradación del rendimiento (performance) y la fiabilidad de la red de interconexión en presencia de fallos [20]:

Reliability. La *fiabilidad* $R(t)$ de la red en el tiempo t se define como la probabilidad de que todos los nodos estén operativos, y se puedan comunicar con todos los demás nodos a lo largo de todo el intervalo de tiempo $[0, t]$. Si se tiene especial interés en un par origen-destino, se define la *path reliability* (fiabilidad de camino) como la probabilidad de que haya existido un camino operacional entre este par origen-destino durante todo el intervalo de tiempo $[0, t]$.

Bandwidth. En el ámbito de las redes de interconexión, el *ancho de banda* hace referencia a la máxima tasa en la que los mensajes pueden circular por la red. Esta tasa normalmente se degrada a medida que se producen fallos en los nodos y/o enlaces de la red.

Connectability. La *conectabilidad* $Q(t)$ en un tiempo t, se define como el número de pares origen-destino que se espera permanezcan conectados en presencia de fallos de nodos en un tiempo t. Es importante remarcar que *Connectability* $\neq$ *Connectivity*.

2.3. Tolerancia a fallos orientada a redes de interconexión

En el ámbito de las redes de interconexión, la tolerancia a fallos se entiende como la capacidad de la red para funcionar en presencia de averías en uno o más de sus componentes [3, cap. 6]

Normalmente, las técnicas utilizadas para lograr esta tolerancia a fallos producen considerables degradaciones en el rendimiento de las redes. Por lo tanto, desarrollar técnicas de comunicación de altas prestaciones que sean resistentes a fallos conlleva una serie de problemas de difícil solución.

La gran mayoría de las técnicas actuales de tolerancia a fallos para redes presuponen la existencia de técnicas de diagnóstico, y se enfocan en cómo la disponibilidad de la información obtenida a partir de estos diagnósticos puede ser utilizada para desarrollar mecanismos de comunicación robustos y fiables. La presunción de los diagnósticos implica que esta problemática no es abordada por estas técnicas. Sin embargo, la habilidad de diagnosticar fallos, especialmente en modo *on-line*, es un problema que presenta grandes desafíos y merece especial atención. La dificultad del diagnóstico *on-line* depende de la técnica de conmutación utilizada.

La presencia de fallos tiende a hacer que las soluciones a los problemas de *deadlock* y *livelock* se vuelvan ineficaces. Incluso el fallo de un solo enlace es capaz de destruir las propiedades de libertad de *deadlock* de los algoritmos de encaminamiento adaptativos.

En general, si el fallo es permanente, cualquier mensaje que se encuentre esperando para utilizar un componente fallido (sea un enlace o un nodo de red) esperará indefinidamente, acaparando recursos como buffers y canales. El efecto del fallo se propagará entonces a través de las dependencias, afectando a mensajes que puede que no tengan que utilizar los componentes fallidos.

Los problemas pueden resultar aún peores si se considera el caso de los fallos dinámicos interrumpiendo la comunicación de mensajes en curso. Por ejemplo, el fallo de un enlace durante la transmisión de un flit de datos: los siguientes flits de datos permanecerán bloqueados en los buffers sin información de encaminamiento; esos flits permanecen indefinidamente en la red a no ser que sean explícitamente removidos. Cualquier otro mensaje que permanezca bloqueado esperando para utilizar esos buffers constituye el inicio de una *cadena de espera* (*wait chain*) [13] que puede conducir a *deadlocks*.

Algunas definiciones importantes en el mundo de la tolerancia a fallos para redes de interconexión son:

Redundancia de canal. Se dice que un canal es redundante si y solo si, luego de removerlo, la función de encaminamiento resultante aún permanece

conectada y libre de *deadlock*.

Algoritmo tolerante a f fallos. Se dice que un algoritmo es capaz de tolerar f fallos si, para cualesquiera f componentes fallidos en la red, la función de encaminamiento aún permanece conectada y libre de *deadlock*.

La definición de redundancia de canal es importante para establecer las condiciones de redundancia necesarias para tratar los *single point of failure*. Se dice que un sistema posee un *single poinf ot failure* cuando existe la posibilidad de que un solo fallo pueda provocar una avería en el sistema. Un ejemplo de *single point failure* para una red de interconexión podría ser un par de nodos conectados únicamente por un solo enlace. En este caso no existe redundancia (pilar fundamental de la tolerancia a fallos) por lo cual si este enlace falla, la comunicación entre ese par de nodos no se puede llevar a cabo.

Existen varias formas de tolerar fallos en las redes de interconexión, entre las más importantes se encuentran:

- Redundancia de componentes.
- Reconfiguración.
- Algoritmos de encaminamiento tolerantes a fallos.

La replicación de componentes es el enfoque preferido por los sistemas comerciales. Al producirse una avería, los componentes de recambio (*spare*) son puestos en funcionamiento en reemplazo de los componentes que presentan fallos. La principal desventaja de este enfoque es el alto coste adicional de los componentes de recambio.

La reconfiguración es una de las técnicas mas potentes y se basa en la reconfiguración de las tablas de encaminamiento ante la presencia de averías, adaptándolas a la nueva topología resultante. Esta técnica es extremadamente flexible, pero a costa de grandes perjuicios en el rendimiento.

La mayor parte de las soluciones propuestas en la literatura se basan en el diseño de algoritmos de encaminamiento tolerantes a fallos capaces de encontrar un camino alternativo cuando un paquete encuentra un fallo a lo largo del camino a su destino. Muchas de estas estrategias de encaminamiento tolerantes a fallos requieren una significativa cantidad de recursos de hardware adicional para encaminar los paquetes alrededor de los componentes fallidos, dependiendo del número de fallos tolerados o del número de dimensiones de la topología. Alternativamente, existen algunas estrategias de encaminamiento tolerantes a fallos que necesitan muy pocos o ningún recurso adicional para tratar los fallos a expensas de lograr menores grados de tolerancia a fallos, deshabilitar un cierto número de nodos sanos (*healthy*), impedir el encaminamiento adaptativo de los paquetes, o incrementar drásticamente la latencia de los paquetes [27].

2.3.1. Diferencias entre red y encaminamiento tolerantes a fallos

Actualmente en el campo de la tolerancia a fallos para redes de interconexión se publican trabajos que hacen uso de los conceptos de redes y algoritmos de encaminamiento tolerantes a fallos de manera indistinta, prácticamente como sinónimos. No es extraño incluso hallar trabajos de investigación publicados bajo el título de redes de interconexión tolerantes a fallos, cuando en realidad desarrollan algoritmos de encaminamiento capaces de tolerar fallos. Estos conceptos no pueden ni deberían ser utilizados como sinónimos porque, sencillamente, no lo son. Entre ellos existe una relación de inclusión y ordenación jerárquica, ya que el concepto de redes de interconexión tolerantes a fallos es de orden superior al del encaminamiento tolerante a fallos.

Por definición, una red tolerante a fallos debe incluir algún tipo de encaminamiento capaz de tolerar fallos, además de los mecanismos necesarios para implementar las fases propias de la tolerancia a fallos, de manera de que sea capaz de detectar los errores, contener los daños, tratar los fallos y continuar con su normal funcionamiento; todo esto de manera transparente a las capas superiores.

Es posible expresar matemáticamente esta relación entre las redes y el encaminamiento tolerantes a fallos:

$$\text{RITF} \neq \text{AETF}$$

$$\text{AETF} \subset \text{RITF}$$

$$\text{RITF} = \text{Redes de Interconexión Tolerantes a Fallos}$$

$$\text{AETF} = \text{Algoritmos de Encaminamiento Tolerantes a Fallos}$$

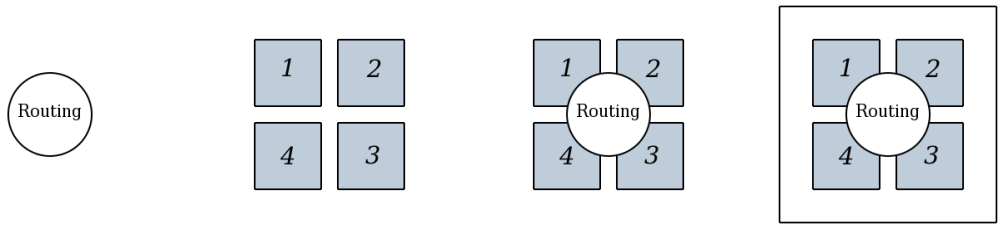
En las Figs. 2.14(a), 2.14(b), 2.14(c) y 2.14(d) se puede observar la relación que existe entre los conceptos de tolerancia a fallos, encaminamiento y redes de interconexión tolerantes a fallos. La Fig. 2.14(b) representa las cuatro fases de la tolerancia a fallos (ver Sección 2.2.3). El encaminamiento basado en las fases anteriores es el verdadero encaminamiento tolerante a fallos (Fig. 2.14(c)). Por último, en la Fig. 2.14(d) se puede ver ya la red de interconexión tolerante a fallos completa, compuesta por: las fases de la tolerancia a fallos, el mecanismo de encaminamiento, y los métodos necesarios para asegurar la integridad de los datos como el CRC, checksum, etc.

Gran parte de los desarrollos actuales atacan directamente el problema del encaminamiento, dando por supuesto el hecho de que se conoce la información sobre la ubicación de los fallos y la correcta distribución de la misma a todos los nodos de la red. De esta manera, en realidad están haciendo uso de un enfoque de distribución de información de tipo global.

2.3.2. Material bibliográfico

Libros

Si bien la problemática de la tolerancia a fallos para redes de interconexión ha sido ampliamente tratada a lo largo de las últimas tres décadas, no se han editado libros dedicados



(a) Encaminamiento (b) Tolerancia a fallos (c) Encaminamiento tolerante a fallos (d) Red de interconexión tolerante a fallos

Figura 2.14: Tolerancia a fallos, encaminamiento y redes de interconexión

específicamente al tema.

El libro *"Interconnection networks. An Engineering Approach"* [3], dedica el capítulo 6 al estudio del encaminamiento tolerante a fallos. Se presentan los conceptos de redundancia, tolerancia a fallos, y modelos de fallos. Además se analizan los casos de tolerancia a fallos para diferentes tipos de redes y se finaliza con un estudio de la recuperación de fallos dinámicos.

En el 2007 fue publicado el libro *"Fault-tolerant systems"* [20], cuyo capítulo 4 trata las redes tolerantes a fallos. En este libro los autores presentan una serie de medidas para caracterizar la tolerancia a fallos de las redes de interconexión. Luego se encargan de hacer un estudio, en base a estas medidas, de las características de las topologías más comunes de redes de interconexión, para finalizar con una breve referencia al encaminamiento tolerante a fallos.

Artículos

A lo largo de las últimas décadas se han publicado gran cantidad de trabajos de investigación sobre la temática de tolerancia a fallos para redes de interconexión, tanto en revistas, como en conferencias especializadas.

A modo de resumen, en las Tablas 2.4 y 2.5 se caracterizan los trabajos más relevantes de los últimos años. A modo de facilitar la legibilidad y comprensión de las tablas, no se incluyen los nombres de los trabajos directamente en ellas, sino que se utilizan identificadores. Las relaciones entre los nombres de los trabajos y sus identificadores se listan a continuación:

- **Artículo 1:** *"A Theory of Fault-Tolerant Routing in Wormhole Networks"* [2].
- **Artículo 2:** *"An Efficient Fault-Tolerant Routing Methodology for Fat-Tree Interconnection Networks"* [14].
- **Artículo 3:** *"A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers"* [17].
- **Artículo 4:** *"Evaluating the Performance of Adaptive Fault-Tolerant Routing Algorithms for Wormhole-Switched Mesh Interconnect Networks"* [31].
- **Artículo 5:** *"Siamese-Twin: A Dynamically Fault-Tolerant Fat-Tree"* [32].
- **Artículo 6:** *"A routing methodology for achieving fault tolerance in direct networks"* [15].

- **Artículo 7:** *"Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori"* [25].
- **Artículo 8:** *"A Fully Adaptive Fault-Tolerant Routing Methodology Based on Intermediate Nodes"* [27].
- **Artículo 9:** *"FROOTS – Fault Handling in Up*/Down* Routed Networks with Multiple Roots"* [33].
- **Artículo 10:** *"Reachability-based fault-tolerant routing"* [26].
- **Artículo 11:** *"Fast recovery from link failures using resilient routing layers"* [21].
- **Artículo 12:** *"Resilient routing layers for recovery in packet networks"* [16].
- **Artículo 13:** *"Immucube: Scalable Fault-Tolerant Routing for k-ary n-cube Networks"* [29].
- **Artículo 14:** *"Immunet: a cheap and robust fault-tolerant packet routing mechanism"* [30].

Artículo	1	2	3	4	5	6	7
Reconfiguración	-	-	-	-	-	-	√
Requisitos de hardware	-	-	√	-	√	√	-
Encaminamiento							
adaptativo	√	√	√	√	√	√	-
estático	-	-	-	-	-	√	√
por tablas	-	-	-	-	-	-	√
Tolerancia a fallos							
detección del error	-	-	-	-	-	-	-
contención del daño	-	-	-	-	-	-	-
recuperación del error	√	√	√	√	-	√	√
tratamiento del fallo y continuidad del servicio	-	-	-	-	-	-	-

Tabla 2.4: Resumen de las técnicas de toleranca a fallos para redes. Parte 1

Artículo	8	9	10	11	12	13	14
Reconfiguración	-	√	-	-	-	√	√
Requisitos de hardware	√	√	-	-	-	√	√
Encaminamiento							
adaptativo	√	-	-	√	-	-	-
estático	-	√	√	-	-	-	-
por tablas	-	√	-	-	-	√	√
Tolerancia a fallos							
detección del error	-	-	-	-	-	√	√
contención del daño	-	-	√	-	-	-	-
recuperación del error	√	√	√	√	√	√	√
tratamiento del fallo y continuidad del servicio	-	-	-	-	-	-	-

Tabla 2.5: Resumen de las técnicas de toleranca a fallos para redes. Parte 2

La primer conclusión que se puede obtener observando los datos que se muestran en las tablas es que, de las fases de tolerancia a fallos, todos los trabajos implementan la de recuperación del error, mientras que pasan por alto la última de las fases (tratamiento del fallo y continuidad del servicio).

Exceptuando [16], todos estos desarrollos utilizan algún tipo de algoritmo de encaminamiento, de los cuales [2], [14], [17], [31], [32], [15], [27] y [21] son algoritmos de encaminamiento tolerantes a fallos. En [15] se combinan el encaminamiento adaptativo con el estático para solucionar algunos casos de fallos excepcionales.

Los trabajos [25], [33], [29] y [30] utilizan algoritmos de encaminamiento basados en tablas. Esto se debe a que atacan la problemática de la tolerancia a fallos a través del enfoque de reconfiguración.

La mitad de todos los trabajos aquí presentados se vale de algunos requerimientos especiales de hardware como caminos virtuales adicionales [15], modificaciones de topología (agregado de enlaces) [32] y deshabilitación de nodos sanos [17], o dispositivos de red modificados [29], [30]. Los dos últimos son los únicos que resuelven la fase de detección de errores.

Uno de los artículos que más aportes introduce, desde el punto de vista teórico, es [2]. En él se definen y expresan (muchas veces en base a formulaciones matemáticas) gran parte de los conceptos fundamentales de la tolerancia a fallos para redes de interconexión, como son la redundancia de caminos y canales y, además, ofrece una propuesta metodológica para desarrollar algoritmos de encaminamiento tolerantes a fallos.

La orientación de las soluciones es muy variada. Hay desarrollos para *fat-tree* [14], [32], para *mallas* [17], [31], para *k-ary n-cube* [15], [27], para redes irregulares [33], y para clusters de computadores [26].

Capítulo 3

Análisis

3.1. Introducción

En el primer capítulo fueron expuestas las motivaciones y el objetivo final del trabajo, *dotar a las redes de interconexión de tolerancia a fallos*. Luego, en el capítulo 2, se introdujeron los conceptos teóricos necesarios para desarrollar y fundamentar este trabajo.

A partir de estas definiciones se hace posible analizar en detalle los fundamentos del trabajo, y delimitar de manera clara y concisa su alcance. Este análisis nos permitirá tener un conocimiento profundo de las características, requerimientos funcionales y riesgos de nuestro problema y nos posibilitará realizar el diseño de nuestra propuesta de manera eficaz.

Al momento de plantear qué pasos seguir para lograr que las redes de interconexión sean tolerantes a fallos, se debe tener presente el hecho de que las redes forman parte de sistemas mayores. En nuestro caso, estos sistemas son los computadores de altas prestaciones, sobre los que se ejecutan una gran variedad de aplicaciones que poseen elevados niveles de exigencia en cuanto a prestaciones y calidad de servicio.

Ésto implica que la red resultante deber ser capaz de proveer los servicios que necesite el sistema en sus capas superiores, ya sea en presencia o ausencia de fallos, de manera totalmente transparente.

Para ser tolerante a fallos, la red de debe cumplir con las cuatro fases que se han presentado en la sección 2.2.3, por lo que es necesario contar con algún tipo de redundancia, ya que *sin redundancia no existe la tolerancia a fallos*.

El enfoque del trabajo se basa en el aprovechamiento de la *redundancia de caminos* que ofrecen las topologías de redes modernas, particularmente las de tipo *k-ary n-cube*. En la Fig. 3.1 se puede observar la redundancia de caminos entre el par de nodos *A*-*F*.

Los fallos en la red inhabilitan recursos por periodos de tiempo variables, en los que esos recursos no pueden ser utilizados. Como consecuencia de esta situación, aparecen congestiones en las áreas de influencia de dichos recursos. Es aquí donde se hacen evidentes los beneficios de tomar a DRB-E como punto de partida para el desarrollo, ya que este fue originalmente diseñado para lidiar con los problemas de congestión en la red. Ésto brinda la posibilidad de

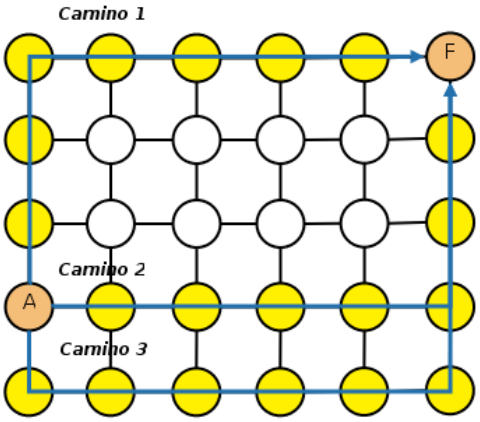


Figura 3.1: Ejemplo de redundancia de caminos

obtener soluciones más robustas, capaces de tratar la problemática de la tolerancia a fallos y a la vez atacar los problemas secundarios que aparecen como consecuencia de los mismos.

El trabajo se ve influenciado por dos áreas del mundo de la ciencia computacional que, *a priori*, no poseen muchos puntos en común, ya que tradicionalmente se han tratado por separado. Basta con el simple hecho de leer el título del trabajo para evidenciar las influencias, tanto de las redes de interconexión, como de la tolerancia a fallos. Esta dualidad implica tener en cuenta e intentar satisfacer los objetivos, requerimientos y métodos propios de las dos áreas, pero esto no siempre es posible. Vale recordar que incluso dentro de las áreas mismas los objetivos son contradictorios, por lo que encontrar un punto de equilibro entre ambos mundos es un problema no trivial.

De los objetivos propios del mundo de las redes de interconexión, debemos prestar especial atención a los *requerimientos de prestaciones*, especialmente a la *latencia* y al *throughput*,a sí como respetar los requerimientos funcionales de conectividad, ausencia de *deadlock*, etc.

Por el mundo de la tolerancia a fallos los más importantes son: no degradar de ninguna manera el rendimiento del sistema en ausencia de fallos; lograr la menor degradación posible en presencia de fallos (según el tipo y relevancia de estos); que la solución no implique un coste adicional muy elevado (en recursos, tiempo, etc.).

En la Fig. 3.2 se representan gráficamente algunos conceptos importantes para el análisis del trabajo. Se incluye además parte de la nomenclatura utilizada. En la imagen se pueden ver ejemplos de fallos de enlaces y nodos (izquierda) y la agrupación de varias unidades de procesamiento en torno a un mismo nodo de red (derecha). En la actualidad, es una práctica usual utilizar agrupaciones de varias unidades de procesamiento en torno a un mismo nodo de red.

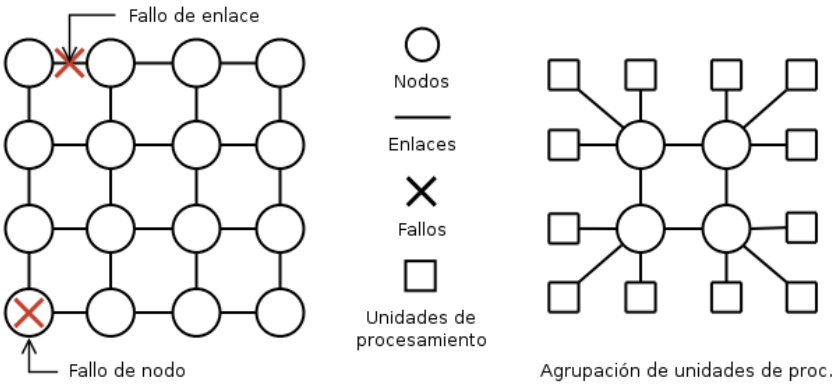


Figura 3.2: Conceptos gráficos y nomenclatura

3.2. Análisis de los fundamentos teóricos

Un punto muy importante a tener en cuenta al momento de realizar el análisis es el hecho de que los fallos son un *fenómeno dinámico*, por lo que pueden ocurrir durante las etapas de configuración de las trayectorias de encaminamiento, de transmisión de mensajes, o de notificación de recepción del mensaje.

En esta sección se analizan los conceptos teóricos que se han presentado en el capítulo anterior y se valora su utilidad para el desarrollo del trabajo.

3.2.1. Modelos y tipos de fallos

En la sección 2.2.6 fue definido el modelo de fallo, tal y como se encuentra en la literatura específica [3, cap. 6], y las críticas que se le realizan al mismo.

Para llevar a cabo los desarrollos del trabajo se utiliza este modelo tal cual han sido definido, considerando fallos de enlaces (*link failures*) y fallos de nodos (*node failures*). De acuerdo a este modelo, los fallos de nodos pueden ser expresados como fallos simultáneos de todos los enlaces del nodo (ver Fig. 3.3).

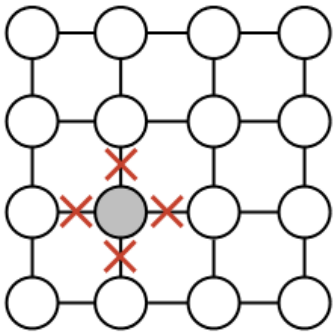


Figura 3.3: Ejemplo del fallo de un nodo visto como fallos simultaneos de sus enlaces

Este modelo se va a utilizar bajo condiciones tanto estáticas como dinámicas. En las primeras los fallos están presentes al momento de iniciar la ejecución del sistema. En la segunda, en cambio, los fallos se pueden producir aleatoriamente.

Si bien excede el ámbito de aplicación de este trabajo, vale mencionar que a futuro se pretende ampliar esta definición de fallos, para poder resolver la problemática relacionada con la posible pérdida de información en tránsito en los enlace y/o almacenada en los nodos.

Las soluciones van a ser diseñadas teniendo en cuenta las tres categorías de fallos según su duración, que han sido definidas en el capítulo anterior: permanentes, transitorios e intermitentes.

Teniendo en cuenta la clasificación de fallos según su comportamiento, se van a atacar los fallos benignos, pero no los fallos maliciosos o bizantinos porque sus efectos son resueltos en capas superiores (si se envían mensajes a nodos caídos) o en capas inferiores (si se generan mensajes erroneos).

3.2.2. Estado de los enlaces

Un punto importante a tener en cuenta es el de la monitorización del estado de los enlaces. Como se ha explicado en los párrafos anteriores, los modelos a utilizar se basan en gran parte en fallos de enlaces, por lo que es crucial poder saber el estado de los mismos en un momento dado.

Los dispositivos de red actuales incorporan técnicas de control del estado de sus puertos y enlaces, a través de mediciones de parámetros físicos tales como la diferencia de potencial, la impedancia, etc. [18, cap. 7]. Si bien esta información se obtiene en los niveles más bajos del sistema (a nivel de hardware), se encuentra disponible a las capas superiores en las que operan los métodos de encaminamiento de las redes de interconexión, y por lo tanto puede ser utilizada para la toma de decisiones e implementación de las fases de la tolerancia a fallos.

3.2.3. Topologías

Actualmente existen una gran diversidad de topologías de red que poseen redundancia de caminos. Más allá de esto, las redes de tipo *k-ary n-cube* y las mallas, poseen un conjunto de características que las hacen más interesantes desde el punto de vista del trabajo: permiten realizar un buen aprovechamiento de las características de localidad en las comunicaciones, proveen un gran número de caminos alternativos entre pares de nodos origen-destino (muchos de los cuales poseen la misma longitud, como los caminos 1 y 2 de la Fig. 3.1), y las *k-ary n-cube* son simétricas. Además, son unas de las topologías más utilizadas en la actualidad. Ejemplo de ello es la red de interconexión del *BlueGener/L*, un toro 3D de 32x32x64 nodos de red [34].

3.2.4. Encaminamiento

Como ya se ha mencionado en más de una oportunidad, el punto de partida de nuestro trabajo es el método mejorado de balanceo distribuido del encaminamiento, DRB-E. Vale recordar que

este método se encuentra dentro del grupo de los algoritmos de encaminamiento adaptativos. Nuestro trabajo seguirá la línea de este algoritmo.

Esto implica que el enfoque de tratamiento de la tolerancia a fallos para redes de interconexión elegido cae dentro del grupo de los *algoritmos de encaminamiento tolerantes a fallos*.

3.3. Análisis del trabajo

Hasta aquí se han estudiado y expuesto los fundamentos teóricos sobre los que se basa la realización del trabajo. Ahora se lo debe definir correctamente y decidir el alcance del mismo.

3.3.1. Alcance

El trabajo busca dotar a las redes de interconexión de tolerancia a fallos, para lo cual se ha elegido utilizar el enfoque de algoritmos de encaminamiento tolerantes a fallos.

Se toma a DRB-E como base de desarrollo del algoritmo de encaminamiento, aprovechando sus características de diseño para tratar los problemas de congestión derivados de la aparición de fallos. En base a esto, se realizaron las modificaciones necesarias para hacer que DRB-E pase a ser un algoritmo de encaminamiento tolerante a fallos completo. Estas modificaciones se comentan en la sección 3.3.2, y luego se detallan en el capítulo siguiente.

En la tabla 3.1 se puede observar la caracterización del trabajo. Si se contrasta esta tabla con las presentadas en el estudio del estado del arte del capítulo anterior (tablas 2.4 y 2.5), se puede ver la diferencia de enfoques con los demás métodos. Hasta el momento, nuestro trabajo es el único que resuelve las cuatro fases de la tolerancia a fallos.

Al basarse en un método de encaminamiento adaptativo que aprovecha la redundancia de caminos, no se precisan requisitos de hardware adicional ni hacer reconfiguraciones de tablas de encaminamiento.

La fase de detección del error se sustenta en la información del estado de enlaces puesta a disposición por los dispositivos de encaminamiento actuales, tal y como se explicó en anteriormente.

Ya han sido expuestos los tipo de fallos tratados y los modelos de fallos que se utilizan. A partir de esto, es posible realizar un análisis más exhaustivo de lo que representan estas definiciones.

En los fallos de enlaces se pueden dar varias situaciones distintas tales como:

- Fallos de los canales físicos, sus controladores o buffers.
- Corrupción de mensajes debido a su mal funcionamiento.
- Atascamiento de mensajes en los buffers de salida de enlaces caídos.
- Pérdida de mensajes en tránsito.

Trabajo	Desarrollo propio
Reconfiguración	-
Requisitos de hardware	-
Encaminamiento	
adaptativo	√
estático	-
por tablas	-
Tolerancia a fallos	
detección del error	√
contención del daño	√
recuperación del error	√
tratamiento del fallo y continuidad del servicio	√

Tabla 3.1: Caracterización del trabajo a realizar

Estas cuatro situaciones han sido estudiadas a fin de analizar su ámbito de aplicación y la forma de resolver los problemas que ocasionan. De estas tres, es la primera la que resuelve nuestro trabajo, mientras que la segunda ya se ha resuelto en capas inferiores del sistema mediante funcionalidades de control como *checksums*, *CRC*, etc.

La pérdida de mensajes en tránsito y el atascamiento de mensajes han sido analizadas, pero debido a que poseen un elevado nivel de complejidad serán tratados en trabajos futuros. Para el caso del atascamiento de mensajes, la solución sería convertir esos buffers de salida asociados a enlaces fallidos en buffers de entrada y, reinyectarlos a los "árbitros" o procesos de selección de los encaminadores como si fueran buffers de entrada normales. El problema de pérdida de mensajes en tránsito requiere de soluciones combinadas con replicación de datos y *checkpoints*.

Por el lado de los fallos de nodos, se pueden dar:

- Fallos del dispositivo. Expresado como el fallo simultáneo de todos sus enlaces.
- Envíos de mensajes a nodos desconectados o caídos.
- Pérdida de los datos almacenados en los buffers del dispositivo.

Al igual que para los fallos de enlaces, aquí se han estudiado y analizado el ámbito y los problemas ocasionados por los fallos. Nuevamente, el primero de ellos es el que se resuelve en este trabajo, mientras que el tratamiento del envío de mensajes a nodos desconectados o caídos corresponde a las capas superiores del sistema (a la aplicación).

El último de los problemas, el de la pérdida de datos al caer los dispositivos, es el de más difícil solución y representa un gran desafío. Este problema ha sido analizado, pero será tratado en trabajos futuros (tesis doctoral). La solución al mismo debería estar basada en duplicación de la información en nodos vecinos, en conjunción con *checkpoints* y otras técnicas propias de la tolerancia a fallos propia de sistemas de almacenamiento.

En la Tabla 3.2 se resumen el ámbito en el que son tratadas cada una de la situaciones.

Ámbito de solución	Este trabajo	Trabajos futuros	Otros niveles
Fallos de enlace			
fallos físicos	√	-	-
corrupción de mensajes	-	-	√
atascamiento de mensajes	-	√	-
pérdida de mensajes	-	√	-
Fallos de nodos			
fallos de dispositivos	√	-	-
envíos a nodos caídos	-	-	√
pérdida de datos	-	√	-
Tipos de fallos			
permanentes	√	-	-
transitorios	√	-	-
intermitentes	√	-	-
benignos	√	-	-
maliciosos	-	-	√

Tabla 3.2: Ámbito de tratamiento de situaciones de fallos

3.3.2. Punto de partida y modificaciones

En reiteradas oportunidades se ha expuesto que se utilizan los desarrollos de DRB-E como punto de partida de este trabajo y los motivos de esta decisión, por lo que no se incluye aquí la argumentación de la elección.

A fin de permitir el desarrollo del algoritmo de encaminamiento tolerante a fallos a partir de DRB-E, fue necesario introducirle una serie de modificaciones.

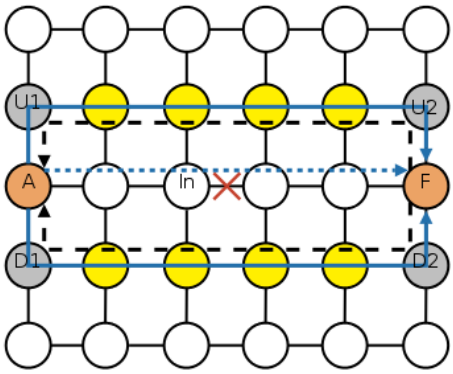
La primera de estas modificaciones es la inclusión de un nuevo elemento dentro de la simbología de demarcación de caminos que utiliza DRB-E (heredada de DRB), específicamente el símbolo de "*camino nulo*".

Se necesita este nuevo símbolo de manera de poder marcar y luego identificar los caminos –entre pares de nodos origen-destino– que presenten fallos, para así evitar su utilización. Es decir, evitar que los nodos origen continúen enviando mensajes por caminos que presentan fallos y/o que estos caminos puedan ser utilizados como caminos alternativos cuando DRB-E abre trayectorias adicionales ante la presencia de problemas de congestión.

Debido a que DRB-E basa su lógica de funcionamiento (apertura y utilización de trayectorias alternativas) en los niveles de congestión que presentan los caminos, es decir en la latencia, el nuevo símbolo debe ser coherente con esta lógica, por lo que se representa como un valor de latencia infinito. En términos prácticos, como un valor de latencia muy grande (varios niveles de magnitud superiores a la media de latencias).

Las Figs. 3.4(a) y 3.4(b) ejemplifican el modo de utilización del símbolo de camino nulo. En la Fig. 3.4(a) se muestra una red con tres caminos entre los nodos *A*-*F*, uno de los cuales presenta un fallo (*A*-*In*-*F*). La Fig. 3.4(b) muestra la tabla de caminos del nodo origen *A* con los valores de latencia de estos tres caminos. Obsérvese que al camino *A*-*In*-*F* se le ha asignado

un valor de latencia infinito (*INF.*).



(a) Red de interconexión

Camino	Latencia
(A)-(In)-(F)	INF.
(A)-(U1)-(U2)-(F)	17
(A)-(D1)-(D2)-(F)	23
. . .	

(b) Tabla de latencias del nodo origen

Figura 3.4: Ejemplo de utilización del símbolo de camino nulo

A partir de los análisis presentados en este capítulo y a modo de conclusión, se presenta el esquema de relación entre los desarrollos del trabajo y los conceptos teóricos.

Algoritmo de encaminamiento tolerante a fallos (*DRB-E modificado*):

- **Detección del error.** Control del estado de los enlaces.
- **Contención del daño.** Utilización de las vías de escape de DRB-E aplicando la información anterior.
- **Recuperación del error.** Notificación reactiva (en respuesta al fallo) hacia los nodos origen.
- **Tratamiento del fallo y continuidad del servicio.** Monitorización reactiva (se inicia al detectarse el fallo) y actualización del estado del enlace a los orígenes en caso de que el fallo desaparezca (podría ser el caso de fallos transitorios).

En el próximo capítulo se explican en detalle cada una de estas relaciones.

Capítulo 4

Diseño

4.1. Introducción

En este capítulo se presenta en detalle el diseño de nuestra propuesta, desarrollada en base al análisis de las características, requerimientos funcionales y riesgos propios del problema de la tolerancia a fallos para redes de interconexión presentado en el capítulo anterior.

El diseño de nuestro trabajo se centra en el desarrollo de políticas de encaminamiento tolerantes a fallos a partir del método de encaminamiento distribuido mejorado (DRB-E) que ha sido presentado y estudiado en capítulos anteriores.

Nuestro objetivo es desarrollar un algoritmo de encaminamiento que cumpla con las cuatro fases propias de la tolerancia a fallos, y que a su vez sea capaz de tratar, de forma eficiente y eficaz, con las diferentes manifestaciones temporales de los fallos.

En este ámbito, una solución eficaz es aquella capaz de tolerar fallos en una red de interconexión, mientras que una solución eficiente es la que logra el mismo resultado pero con un mínimo coste y ocupación de recursos.

Numerosos esfuerzos han sido dedicados durante la etapa de diseño en pos de la eficiencia. Como punto de partida, nuestra propuesta no precisa de recursos de *hardware* adicionales y evita solicitar la retransmisión de mensajes a las capas superiores del sistema. Por otra parte, se tratan de forma adecuada los fallos intermitentes y transitorios, es decir, los no permanentes a fin de maximizar el nivel de aprovechamiento de los recursos en los momentos en que no presentan fallos. Con ésto se evita la aparición de "falsos positivos" en la detección y clasificación de fallos permanentes. Todo ello se traduce en un mejor aprovechamiento de los recursos del sistema y por lo tanto en un incremento de las prestaciones, sobre todo en el caso de los fallos intermitentes, donde la oscilación entre estados de latencia y actividad los hacen sumamente difíciles de tratar por la mayoría de los enfoques actuales.

El capítulo se ha dividido en dos partes. En la primera se detallan por separado, y prácticamente de forma independiente, las soluciones de diseño orientadas a cada una de las cuatro fases de la tolerancia a fallos. Éstas soluciones son luego utilizadas en el diseño de las políticas de encaminamiento tolerantes a fallos que se presentan en la segunda parte del capítulo.

4.2. Diseño orientado a la tolerancia a fallos

En base al análisis realizado en el capítulo anterior, fue posible diseñar soluciones para las fases de la tolerancia a fallos para luego incluirlas como parte fundamental de las políticas de encaminamiento tolerantes a fallos.

4.2.1. Detección del error

En nuestra propuesta esta fase se basa en el control del estado de los enlaces físicos que conectan a los encaminadores entre sí. Actualmente ésta información ya se encuentra disponible en los dispositivos de red que constituyen el sistema (a nivel de las capas de *hardware*), por lo que puede ser utilizada en las capas de red a nivel local sin ningún coste adicional.

Es importante tener en cuenta que éste no es el único enfoque posible para tratar la fase de detección de errores. Otra opción viable sería enviar mensajes de control simples para medir el estado de los enlaces. De este modo, si el paquete de *acknowledge* de estos mensajes no retorna en un tiempo dado (medido por un umbral de tolerancia), se considera el enlace como fallido.

Si bien este enfoque resulta igual de válido que el nuestro, implicaría introducir algún valor de *overhead* innecesario, relacionado con el envío y recepción de los mensajes de control, mientras que el enfoque basado en la medición física de los enlaces es gratuito en cuanto a *overhead*.

Por otra parte, nuestro enfoque posee la ventaja adicional de que, al formar parte de las funcionalidades propias de los encaminadores, permite tener un método de detección "no orientado a la comunicación". En otras palabras, la detección no depende de que un mensaje necesite utilizar el enlace, sino que se puede hacer de forma permanente e independiente del contexto de encaminamiento. Ésto permite que la información de estado se encuentre disponible de manera inmediata y pueda ser utilizada por los mecanismos de arbitraje, permitiéndoles conocer de antemano el estado de un enlace. Gracias a esto, se pueden obtener mejores tiempos de decisión y encaminamiento debido a que no se necesita verificar explícitamente el estado de un enlace antes de utilizarlo porque es información que ya se encuentra disponible.

Ésto evita además que los mensajes puedan llegar a almacenarse en los *buffers* de salida de enlaces que presentan fallos, solucionando en parte el problema del atascamiento de mensajes mencionado en el capítulo anterior. Lo que queda por resolver de este problema es la situación en la que el fallo de un enlace de salida se produce luego de que se han almacenado mensajes en su *buffer*. Ésto es parte de los trabajos futuros más allá de este trabajo con los que se pretende continuar la investigación.

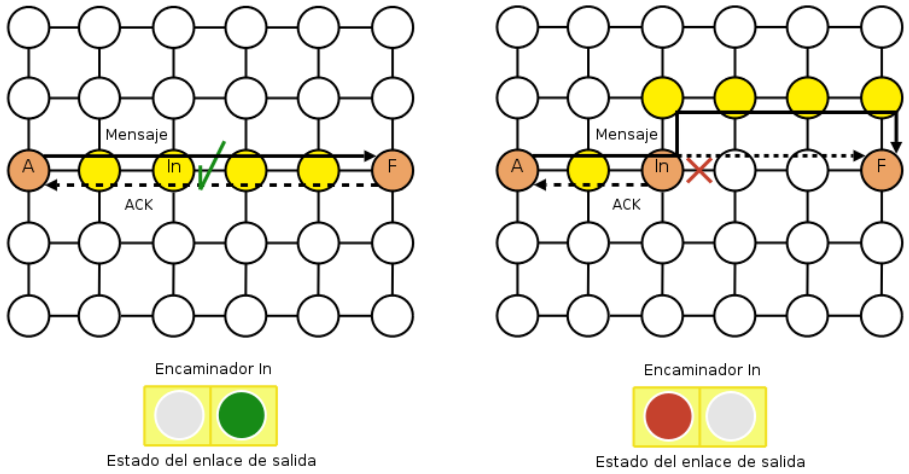
4.2.2. Contención del daño

El diseño de esta fase se basa principalmente en la utilización de las "vías de escape" que introduce DRB-E. Esto hace posible que un encaminador intermedio pueda desviar los mensajes por caminos alternativos, de manera inmediata, si el camino original entre un par de nodos origen-destino presenta fallos.

Originalmente, las vías de escape de DRB-E fueron pensadas para dar salida, por caminos alternativos, a los mensajes que se encontraban con problemas de congestión. Para ello se utilizan umbrales y acumulación de latencias a lo largo del camino, por lo que la decisión de apertura de las vías de escape utilizaba un mecanismo de decisión a partir de dichos valores. Vale la pena mencionar que este mecanismo de decisión se basaba en latencias y umbrales debido a que fue ideado para resolver problemas de congestión, no para fallos. En el caso de fallos, la toma de decisiones es intrínsecamente binaria, es decir, se puede utilizar el enlace o no. Mientras que en el problema de congestión requiere mecanismos más complejos y variables.

Para poder utilizar esta característica en algoritmos de encaminamiento tolerantes a fallos, es necesario modificar el mecanismo de decisión original de DRB-E. La decisión en base a latencias fue reemplazada por una decisión basada en la información del estado de los enlaces que se explicó en la sección anterior. A modo de ejemplo, se podría hacer una analogía de este mecanismo de decisión con el funcionamiento de un semáforo; si el semáforo se encuentra en verde, el enlace se puede utilizar, en caso contrario, se debe utilizar una vía de escape.

En las Figs. 4.1(a), 4.1(b) se ejemplifica el funcionamiento de la propuesta de diseño para esta fase y la analogía con el sistema del semáforo. En la Fig. 4.1(a) el camino entre el par de nodos *A*-*F* no presenta fallos, y la Fig. 4.1(b) muestra la utilización de la vía de escape en el encaminador intermedio *In* debido a que el enlace de salida asignado originalmente (el de la derecha) presenta fallos.



(a) Enlace de salida funcional

(b) Enlace de salida fallido

Figura 4.1: Ejemplo de funcionamiento de la fase de contención del daño

4.2.3. Recuperación del error

Tal y como se adelantó en el capítulo anterior, el diseño de la fase de recuperación del error se basa en un método de notificación "reactiva" hacia los nodos origen. Se habla de notificación reactiva debido a que ésta se genera únicamente si un mensaje debe utilizar un camino alternativo

para alcanzar su destino. Ésto ocurrirá en el caso de que el enlace de salida que debía ser utilizado originalmente por el mensaje presente fallos.

En esta fase se producen tres acciones fundamentales:

1. Se notifica al origen sobre el fallo. En términos prácticos, y siguiendo la lógica de funcionamiento de DRB, esto se realiza a través del envío de un mensaje de reconocimiento (*acknowledge*) al nodo origen en el que se especifica como latencia del camino un valor lógico que represente infinito. Con esto se evita que el nodo origen vuelva a enviar información por el camino fallido.

2. Se evita que el nodo destino envíe un mensaje de *acknowledge* duplicado. Para esto basta con incluir un *bit* de notificación especial en el cuerpo del mensaje que pueda ser activado en el caso de la utilización de las vías de escape, para que el nodo destino al recibir el mensaje y leerlo no envíe el *acknowledge*.

3. Se registran cuales son los orígenes a los que se notificó del fallo. Esta acción se lleva a cabo para tener un registro de cuales son los nodos origen a los cuales se ha notificado del fallo, y por lo tanto no volverán a utilizar el camino fallido. El objetivo de esto es poder tratar eficientemente los fallos no permanentes, ya que si se detecta la desaparición del fallo, es posible notificar a los nodos orígenes de que el camino vuelve a estar disponible y que lo pueden utilizar.

El diseño del registro de nodos origen implica que se consideren cuestiones tales como el tamaño del registro y la cantidad de nodos a registrar. Debido a que en sistemas reales el tamaño del registro es finito, es necesario considerar con que criterios se seleccionan la información que debe ser removida del registro cuando se alcance dicho límite.

En términos prácticos, el registro se implementa a través de una tabla en la que se debe almacenar información del nodo origen así como también información relacionada con el destino final del mensaje. Ésto último se puede realizar de dos maneras distintas:

- Enfoque global. En este caso se almacena sobre cuáles son los nodos origen y destino en una única tabla global por cada encaminador. Se almacena la información de ambos nodos debido a que un mismo nodo origen puede enviar información a más de un nodo destino mediante un mismo encaminador intermedio y no necesariamente a través del mismo enlace de salida.

- Enfoque de enlaces. En este enfoque se asocia una tabla a cada enlace de salida donde se almacena la información de los nodos origen cuyos mensajes se encontraron con el fallo del enlace. Esto permite tener tablas de menor tamaño ya que solamente se guarda la información de los nodos origen.

Ambos enfoques tienen pros y contras. El enfoque global es más simple debido a que solamente se utiliza una única tabla global para todos los enlaces, mientras que el enfoque de

enlaces, al ser dedicado, posee menos requerimientos de memoria. Este último además, facilita la notificación a los nodos origen en caso de desaparición de fallos, ya que no es necesario buscar los pares de nodos origen-destino a los que se deben enviar los mensajes de acualización del estado de un enlace porque éstos se encuentran almacenados en tablas separadas por enlaces.

Para ambos enfoques, la tabla en la que se almacena la información recibe el nombre de "tabla de caminos fallidos".

En lo que respecta a los métodos de selección de la información que debe ser removida de la tabla (al alcanzar el límite de espacio), se pueden utilizar enfoques del tipo *First In First Out* (FIFO), *Least Frequently Used* (LFU), etc.

Las restricciones de tamaño de los registros, y la posibilidad de tener que remover elementos de él provocan un problema adicional: en el caso de un fallo no permanente ¿cómo notificar la desaparición del fallo a los nodos cuyo identificador se borra del registro?

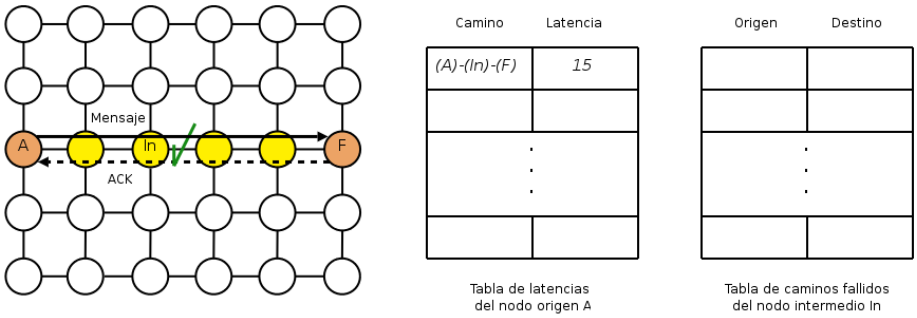
La solución a este problema consite en implementar en los nodos origen un temporizador que se active en base a una "notificación de baja" por parte de los encaminadores intermedios. Ésta notificación será enviada por lo encaminadores intermedios hacia atrás a los nodos origen en el caso de que, por motivos de espacio, su información deba ser removida de alguna de las tablas de caminos fallidos. Al momento de recibir esta notificación, se activa en los nodos origen el temporizador que, transcurrido un tiempo dado, cambia los valores de latencia infinitos de los caminos fallidos por un valor de latencia menor. Como consecuencia de éste cambio, los nodos origen que hayan sido removidos, y por tanto no recibiran las posibles actualizaciones del estado de los enlaces fallidos, intentarán utilizar ocasionalmente estos caminos enviando algún mensaje por ellos. El envío de estos mensajes puede ser visto como un método de sondeo, mediante el cual se prueba la disponibilidad de un camino.

A partir de éste envío, se pueden dar dos situaciones:

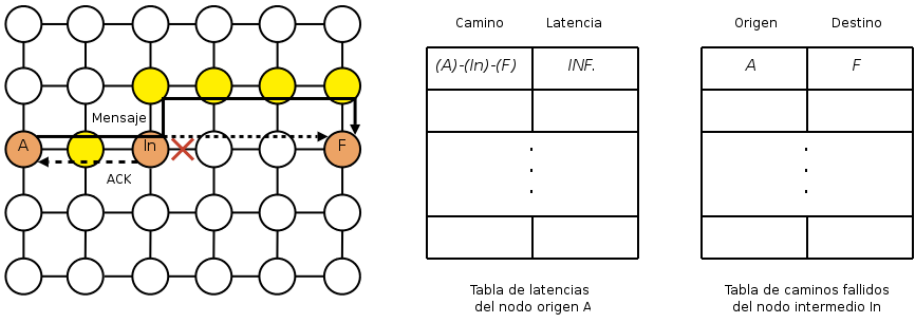
- Que el mensaje se vuelva a encontrar con el fallo. En éste caso el encaminador se volverán a repetir las tres acciones propias de esta fase, por lo que el mensaje llegará a destino a través de una vía de escape y la latencia del camino volverá a tomar el valor infinito (porque el encaminador intermedio enviará al nodo origen ese valor).
- Que el fallo haya desaparecido. En este caso el encaminamiento se realizará de acuerdo a las normas de funcionamiento normal de DRB-E (heredadas de DRB), mediante las cuales se acumulará la latencia real del camino y al llegar al nodo destino este enviará dicho valor al nodo origen, restableciendo la disponibilidad del camino.

Continuando con el ejemplo de la sección anterior, en las Figs. 4.2(a) y 4.2(b) se pueden observar las acciones de esta fase para el mismo ejemplo. En la Fig. 4.2(a) se observa el proceso de envío de mensajes entre el par de nodos *A*-*F* y las tabla de latencias del nodo origen *A* y de caminos fallidos del nodo intermedio *In* (utilizando el enfoque de tablas global). En la Fig. 4.2(b) se observa el envío del mensaje a través de la vía de escape al producirse el fallo en el enlace de salida del encaminador intermedio *In*, la inclusión del identificador del nodo origen *A*

en la tabla de caminos fallidos del encaminador intermedio *In* y la modificación de los valores de latencia para el camino original entre los nodos *A*-*F*.



(a) Enlace de salida funcional



(b) Enlace de salida fallido

Figura 4.2: Ejemplo de funcionamiento de la fase de recuperación del error

4.2.4. Tratamiento del fallo y continuidad del servicio

Ésta fase de basa en los procesos de monitorización reactiva del estado del enlace y de actualización del estado del enlace a los orígenes, en caso de que se detecte la desaparición del fallo. Esto último es de especial utilidad en el caso de los fallos no permanentes. Un ejemplo de este tipo de fallos puede ser el reinicio de algún dispositivo de red, ocasionado por problemas en la alimentación eléctrica del mismo. En este caso los enlaces conectados a dicho dispositivo aparecerán como fallidos solamente durante el intervalo de tiempo que tarde el mismo en reiniciarse y volver a un estado funcional completo.

Esta fase tiene su origen en la misma situación que las dos fases anteriores, es decir, cuando un mensaje intenta utilizar un enlace que presenta un fallo. Es aquí donde se inicia el proceso de monitorización reactiva del estado de dicho enlace. Este proceso se encarga de verificar el estado del enlace fallido a fin de detectar cambios en él. Si el estado del enlace vuelve a ser funcional, es decir, desaparece el fallo en el mismo, se da inicio al segundo proceso que de esta fase, la actualización del estado del enlace a los orígenes.

En este segundo proceso, se leen los datos almacenados en la tabla de caminos fallidos, a fin

de identificar los nodos origen que deben ser notificados del cambio de estado del enlace y, por ende, la disponibilidad del camino entre el par de nodos origen-destino. Una vez identificados, se les envía un mensaje de *acknowledge* con un valor de latencia promedio predefinido, a fin de reemplazar el valor infinito que se les había enviado en la fase anterior, y se elimina la información de este par origen destino de la tabla de caminos fallidos. Esta actualización del valor de latencia hace que los nodos origen permite que éstos puedan volver a utilizar el camino.

En las Figs. 4.3(a), 4.3(b), 4.3(c) y 4.3(d) se pueden observar las cuatro etapas de esta fase. En la primera el estado del enlace sigue siendo fallido, por lo que el nodo origen A utiliza caminos alternativos para llegar al nodo destino F. En la segunda se descubre, en el nodo intermedio In, el cambio de estado del enlace. En la tercera el nodo In notifica el cambio al nodo origen A, y en la última el nodo origen vuelve a utilizar el camino original.

4.3. Políticas de encaminamiento tolerantes a fallos

Tal y como se ha comentado anteriormente, nuestra propuesta se basa en el diseño de políticas de encaminamiento tolerantes a fallos, partiendo del mejorado método distribuido del encaminamiento (DRB-E) y de los diseños de las fases propias de la tolerancia a fallos presentadas en las secciones anteriores.

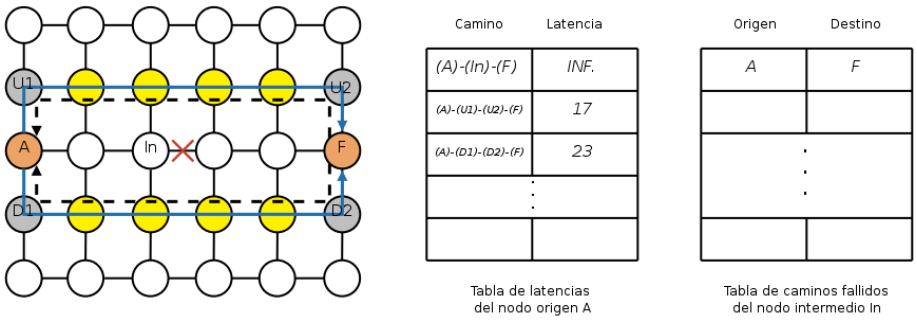
Debido a que nuestra propuesta se basa en la incorporación de los métodos de tolerancia a fallos a DRB-E, es necesario exponer el funcionamiento del encaminamiento propio de DRB-E y las fases que lo componen. Para ello se establecen una serie de definiciones que corresponden a los componentes de DRB-E, agrupadas en el Apéndice A (para facilitar la legibilidad del documento).

DRB-E es un método para crear caminos alternativos entre pares de nodos origen-destino en la red de interconexión. A partir de esto distribuye la carga de mensajes de cada par origen-destino entre un camino "multicarril" (metacamino) constituido por varios caminos simples.

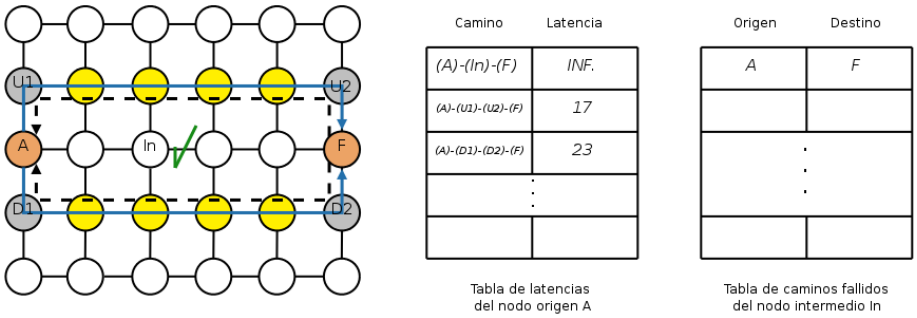
La forma de crear caminos alternativos, entre los nodos origen y destino, se basa en enviar los mensajes a nodos intermedios de la red antes de enviarlos al nodo destino final. De esta forma, el camino del mensaje se recorre en diversas fases o etapas. Para este trabajo, consideramos dos destinos intermedios, de manera que el camino queda dividido en tres segmentos y se realiza en tres etapas: desde el nodo origen al primer destino intermedio, de ahí al segundo nodo intermedio, y, finalmente, desde este punto al destino final. Ésta camino de tres segmentos toma el nombre de camino multipaso. Cada uno de los pasos individuales se realiza utilizando encaminamiento estático mínimo propio de la red en cuestión.

Este sistema de pasos intermedios pretende distribuir uniformemente por toda la red de interconexión todo el tráfico de mensajes.

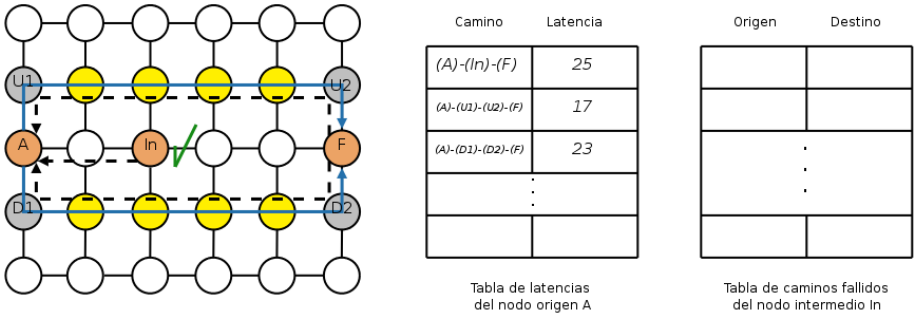
El primero de los nodos intermedios se escoge de entre un subconjunto de nodos "cercano a", o "centrado en", el nodo origen, mientras que el segundo de los nodos intermedios se elige de un subconjunto "cercano a", o "centrado en", el nodo destino.



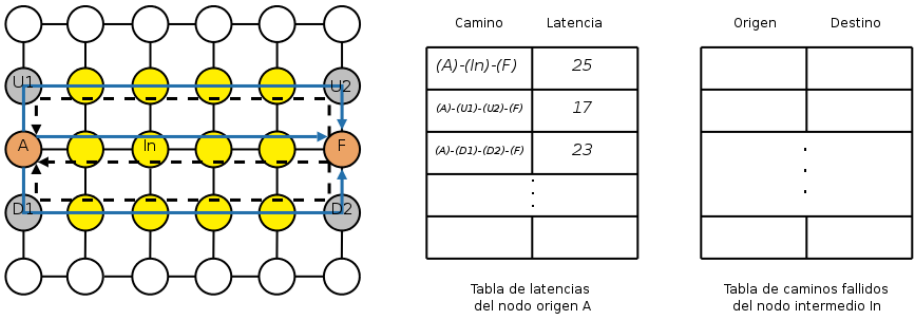
(a) Enlace de salida fallido



(b) Enlace de salida funcional



(c) Notificación del cambio de estado



(d) Utilización del camino

Figura 4.3: Ejemplo de funcionamiento de la fase de tratamiento del fallo y continuidad del servicio

Estos subconjuntos de nodos, que siempre incluyen al nodo fuente o destino se llaman *supernodos*. En la Fig. 4.4 se puede observar gráficamente el concepto de supernodo.

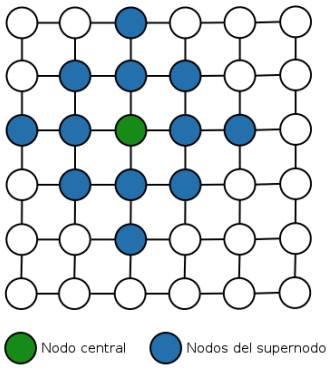


Figura 4.4: Concepto de supernodo

El encaminamiento de DRB-E se encarga de configurar dinámicamente los caminos y de distribuir los mensajes entre los caminos multipaso del metacamino, con el objetivo de minimizar la latencia y utilizar uniformemente los recursos de la red de interconexión. Los fundamentos del encaminamiento son:

- Detección de las condiciones del tráfico en la red de interconexión mediante la monitorización de la latencia experimentada por lo mensajes en tránsito por la red.
- Configuración dinámica de los metacaminos dependiendo de esa latencia detectada.
- Distribución de los mensajes entre los caminos multipaso del encaminamiento.

Consecuentemente, el encaminamiento de DRB-E se divide en tres fases:

- Fase 1: Monitorización de la carga de tráfico de la aplicación.
- Fase 2: Configuración dinámica de los metacaminos.
- Fase 3: Selección de caminos multipaso.

Estas fases son unidades independientes entre ellas y son ejecutadas individualmente para cada canal de la aplicación en curso, considerando un canal como la conexión lógica que conecta procesos. Inicialmente, todo los canales se configuran mediante metacaminos canónicos, es decir, usando los caminos estáticos mínimos.

La actividad de monitorización de la latencia es llevada a cabo por los propios mensajes en tránsito por la red y su objetivo es registrar la latencia que el mensaje experimenta en su viaje hasta el destino. Cuando el mensaje llega a su destino con la información de latencia, ésta se envía con un mensaje de reconocimiento hacia atrás hasta el nodo origen del mensaje. La configuración de los metacaminos se realiza a nivel del canal cada vez que una información de

latencia llega al origen. Esta latencia se usa para configurar el nuevo metacamino. La selección del camino multipaso también se realiza a nivel del canal cada vez que se inyecta un mensaje, tratando de elegir el camino multipaso de menor latencia [6, cap. 4].

Hasta aquí se ha explicado el comportamiento original de DRB-E. En el listado de código fuente 4.1 se expone, en forma de pseudocódigo, como sería el funcionamiento de la metodología que nuestra propuesta incorpora a éste. Lo que se expone en el listado ha sido incluido en las tres fases propias de DRB-E, de acuerdo al ámbito de aplicación de cada acción, dando como resultado la política de encaminamiento tolerante a fallos que hemos propuesto.

En las próxima tres secciones se explican en detalle las modificaciones introducidas a cada una de estas fases y su estructura final.

```
1  DRB-E-Fault-tolerant(mensaje M, camino multipaso MSP)
2  /* pseudocodigo de tolerancia a fallos para DRB-E */
3
4  Inicio
5
6  Para cada paso del mensaje M,
7      Verificar el estado del enlace de salida a utilizar
8          Si el enlace presenta un fallo
9              Se redirige el mensaje por la via de salida
10             Se notifica al nodo origen de la presencia del fallo
11             Se almacena la informacion del nodo al que se notifico
12              del fallo
13             Se activa el proceso de monitorizacion del estado del
14              enlace
15             Si se detecta la desaparicion del fallo
16                 Se notifica a los nodos origen almacenados del cambio
17                 Se borra la informacion del nodo origen del registro
18             Fin Si
19         Fin Si
20     Continuar hacia el proximo destino intermedio o hacia el destino final
21 Fin Para
22
23 Fin Pseudocodigo
```

Código fuente 4.1: Pseudocódigo de la metodología de tolerancia a fallos para DRB-E

4.3.1. Fase 1: Monitorización del estado de los enlaces y la carga de tráfico

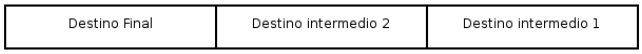
La monitorización de la carga de tráfico es realizada sobre los propios mensajes por parte de los encaminadores DRB, mientras que la monitorización del estado de los enlaces se hace sobre los canales físicos conectados a estos encaminadores.

En ausencia de fallos, es en los mismos mensajes donde se registra y transporta la información sobre la latencia sufrida durante su viaje hacia el destino. El mensaje registra información sobre la latencia que él experimenta en cada encaminador que atraviesa cuando se bloquea por causa de contención con otros mensajes.

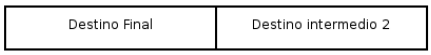
Un mensaje, cuando es inyectado, lleva información en la cabecera de los destinos intermedios que debe visitar. En la Fig. 4.5 se muestra el formato del paquete DRB-E. A la información a transmitir del usuario se le añade una cabecera formada por distintos campos. Esta cabecera se forma por la concatenación de los diferentes destinos a seguir. Este es el formato definido para la gestión de la función de encaminamiento por el encaminador DRB.

Figura 4.5: Paquete DRB

El protocolo para homogeneizar y simplificar la funcionalidad del encaminador DRB es el siguiente. El mensaje viaja hacia el destino 1 siguiendo el encaminamiento estático mínimo como muestra la Fig. 4.6(a), cuando llega a él se elimina esa parte de la cabecera quedando como en la Fig. 4.6(b), y el mensaje continúa hacia el destino 2. Cuando llega a él, se repite la operación de eliminar la cabecera y se continúa hacia el destino final (Fig. 4.6(c)). El encaminador DRB siempre decide el siguiente enlace a utilizar en función de un destino que encuentra en el mismo lugar de la cabecera: al principio de todo.



(a) Primer paso



(b) Segundo paso

(c) Tercer paso

Figura 4.6: Cabeceras del mensaje DRB

Concretamente, cuando un mensaje llega a un encaminador y el siguiente enlace de salida que debe tomar está siendo usado por otro mensaje, el primero debe ser encolado y esperar hasta que se libere el enlace que desea utilizar. El tiempo de permanencia en la cola es el que se registra y se va acumulando en un campo en la cabecera del mensaje. Si este valor acumulado supera un cierto valor umbral que indica que se ha detectado un problema, se envía hacia atrás de nuevo en un mensaje de reconocimiento. Este mensaje de reconocimiento debe tener la máxima prioridad en la red. De esta forma los nodos fuentes tienen información de latencia, es decir, el grado de ocupación, de todos los caminos multipaso por los que están enviando mensajes. Con este enfoque, el algoritmo de configuración de caminos dispone de información actualizada en cuanto se produce el aumento de latencia.

Al detectarse la presencia de un fallo en un enlace que debe ser utilizado por un mensaje, el encaminador envía hacia atrás al nodo origen un mensaje de reconocimiento. En lugar de enviar la latencia acumulada hasta el momento, envía un valor de latencia infinito, de modo que el nodo origen no utilice dicho camino mientras se mantenga este estado. De esta forma es como se realiza la monitorización del estado de los enlaces a través de mensajes de reconocimiento.

En el listado de código fuente 4.2 se detalla el funcionamiento de esta fase en base a la especificación original de DRB-E y las modificaciones que hemos introducido en base a nuestra propuesta. En este listado solamente se detallan las tareas de monitorización del estado de los enlaces y la carga de tráfico que se realiza en los encaminadores intermedios, mientras que en el listado 4.1 se muestra el pseudocódigo de las tres fases de la propuesta.

```
1  Monitorizacion(mensaje M, latencia umbral L, camino multipaso MSP)
2  /* realizado en cada encaminador intermedio */
3
4  Inicio
5
6  Para cada paso del mensaje M
7     Verificar el estado del enlace de salida a utilizar
8        Si el enlace presenta un fallo
9           Se envia hacia atras hasta el origen el valor de
10           latencia = infinito en un mensaje de reconocimiento
11       Sino si el enlace no presenta un fallo
12          Acumular la latencia para calcular la latencia(MSP)
13           (tiempo de transimicion mas el tiempo de espera en
14           las colas de los encaminadores)
15          Si se llega a un destino intermedio
16             Continuar hacia el proximo destino intermedio o hacia el final
17          Fin Si
18          Si latencia(MSP) supera latencia umbral L
19             Enviar mensaje de reconocimiento al origen
20          Fin Si
21          Cuando el mensaje llega al destino final
22             La latencia(MSP) se envia hacia atras hasta el origen  en un
23              mensaje de reconocimiento
24       Fin Si
25 Fin Para
26 Cuando el mensaje de reconocimiento llega al origen
27    Latencia(MSP) se entrega a la funcion de  Configuracion(MSP, latencia(MSP))
28
29 Fin Monitorizacion
```

Código fuente 4.2: Código de monitorización del estado de enlaces y de la carga de tráfico

4.3.2. Fase 2: Configuración dinámica de los metacaminos

El objetivo de esta fase es determinar, para un par origen-destino, el tipo y tamaño del metacamino dependiendo de la latencia medida por los mensajes entre origen y destino (en ausencia de fallos, o entre origen y el encaminador que haya detectado un fallo), mediante la

configuración de unos supernodos determinados a partir de sus parámetros de tipo y tamaño.

Cuando un nodo fuente recibe una latencia de un camino multipaso, calcula la latencia del metacamino del que forma parte ese MSP. Según esta latencia decide incrementar o reducir el tamaño de los supernodos del metacamino dependiendo de si la latencia del metacamino está dentro o fuera del intervalo definido por $[LatUmbral - Tol, LatUmbral + Tol]$.

Si la latencia aumenta, se debe aumentar el metacamino para utilizar un mayor ancho de banda. Si la latencia disminuye y sale fuera del intervalo es porque ese canal dispone de un metacamino configurado demasiado grande y está usando unos recursos que debe liberar para otros canales. Por esta razón se debe disminuir su metacamino.

Por lo tanto, se busca incrementar o decrementar el metacamino para que su ancho de banda esté dentro del intervalo definido por Tol. Suponiendo que cada camino multipaso añadido al metacamino aporta un ancho de banda equivalente al ancho de banda canónico, se busca en cuántos caminos se debe aumentar el metacamino para que su ancho de banda esté dentro del intervalo deseado.

Ésta configuración de supernodos aquí expuesta toma en consideración los valores de latencia en cada momento, así como las características topológicas de la red de interconexión y la distancia física entre origen y destino, para balancear el ancho de banda que configura para el metacamino y su alargamiento correspondiente.

Además de la latencia de un camino multipaso, un nodo origen puede recibir por parte de un encaminador intermedio un mensaje de "notificación de baja". La recepción de este mensaje implica que el encaminador intermedio que lo envió ha tenido que dar de baja la información del nodo fuente de la "tabla de caminos fallidos" por cuestinos de espacio, por lo que no se recibirá una actualización de la latencia del camino fallido (a la que se le asigna un valor infinito al producirse el fallo) ante una eventual desaparición del fallo. Por este motivo, al recibir un mensaje de notificación de baja se inicializa un temporizador a partir del cual, pasado un cierto tiempo, se disminuye el valor de latencia del camino fallido de infinito a un valor dado de modo que, eventualmente, ese camino vuelva a ser utilizado. En la sección 4.2.3 ha sido explicado en detalle este procedimiento.

En el listado de código fuente 4.3 se muestran los procedimientos propios de la fase de configuración del metacamino.

4.3.3. Fase 3: Selección del camino multipaso

Esta fase se encarga de seleccionar un camino multipaso para cada mensaje para:

- Distribuir equilibradamente la carga de comunicaciones entre los caminos multipaso de un metacamino.
- Evitar la utilización de caminos multipaso que presentan fallos.

Para cada mensaje que se envía se selecciona un MSP dependiendo de sus anchos de banda en una relación donde el MSP de mayor ancho de banda es el más frecuentemente utilizado.

```
1  Configuracion(latencia umbral LatU, tolerancia Tol)
2  /* ejecutado en los nodos origen cada vez que llega una Latencia(MSP) */
3
4  Inicio
5
6  Si se recibe una latencia(MSP)
7      Calcular la latencia(P*) usando la ecuacion
8          Si (latencia(P*) > LatU + Tol)
9              Incrementar el tamanio de los supernodos
10         Sino si (latencia(P*) < LatU + Tol)
11             Decrementar el tamanio de los supernodos
12         Fin Si
13 Fin Si
14 Si se recibe un mensaje de baja
15     Iniciar el temporizador de modificacion de latencia
16     Si temporizador = tiempo_umbral
17         Reducir el valor de latencia del camino fallido
18          correspondiente
19     Fin Si
20 Fin Si
21
22 Fin Configuracion
```

Código fuente 4.3: Código de configuración dinámica de los metacaminos

Así pues los mensajes se distribuyen entre los caminos multipaso en proporción al ancho de banda de cada uno de ellos. En el caso de los caminos que presenten fallos, estos no se utilizarán debido a que se les asigna un valor de latencia infinito, por lo que su ancho de banda, definido como la inversa de la latencia, es:

$$\lim_{x \to +\infty} \left(\frac{1}{x} \right) = 0$$

Por consiguiente, la carga se distribuye entre todos los caminos multipaso del metacamino, exceptuando los de ancho de banda cero (que corresponden a los caminos fallidos), pero de manera que los que tienen mayor capacidad disponible reciben un mayor número de mensajes.

Entonces, se usan los anchos de banda para ordenar los MSP mediante el método de construir una función probabilística de distribución acumulativa directa. Para ello, se toma el ancho de banda de cada MSP como valor de una distribución de probabilidad discreta de los caminos multipaso.

En el listado de código fuente 4.4 se muestran los procedimientos propios de la fase selección del camino multipaso.

```
1  Seleccion()
2  /* ejecutado en los nodos origen cada vez que se inyecta un mensaje */
3
4  Inicio
5
6  Construir la funcion acumulativa de distribuciones sumando y normalizando
7   los anchos de banda de los caminos multipaso
8  Generar un numero aleatorio entre [0,1)
9  Seleccionar un MSP usando la funcion de distribucion acumulativa
10 Inyectar el mensaje en la red
11     Construir una cabecera multiple con los destinos intermedios que
12      forman el MSP y el destino final
13     Concatenar los datos con la cabecera
14     Inyectar el mensaje con formato DRB
15
16 Fin Seleccion
```

Código fuente 4.4: Código de selección del camino multipaso

Capítulo 5

Implementación

5.1. Introducción

En este capítulo proporcionamos una descripción de los detalles de implementación de nuestra propuesta, a partir del diseño presentado en el capítulo anterior y prestando especial atención a los detalles de simulación.

Un modelo de simulación proporciona una forma simple, pero efectiva, de predecir y estudiar el rendimiento de un sistema o comparar distintas configuraciones de él, sobre todo en las etapas de diseño cuando no se dispone del sistema. Por otra parte, incluso si el sistema se encuentra disponible para su estudio, un modelo de simulación puede ser la mejor opción porque permite comparar diferentes alternativas del sistema bajo una gran variedad de entornos y cargas de trabajo. El modelado y simulación son necesarios para ganar conocimiento y comprensión sobre el funcionamiento del sistema [24].

Para la implementación de nuestro trabajo, utilizamos la plataforma de software de modelado y simulación *OPNET Modeler* [28]. Ésta herramienta permite realizar el modelado y la simulación de protocolos y tecnologías de red, y provee las herramientas necesarias para la experimentación y el análisis de diferentes escenarios y configuraciones. Asimismo, permite realizar simulaciones robustas en todo tipo de redes, a través del modelado simple e intuitivo de sus componentes. Permite además analizar eficientemente el funcionamiento de estos modelos a una escala realista, mediante una colección de mecanismos que proporcionan la recolección y presentación de métricas y resultados.

Sus principales características, desde el punto de vista técnico son:

- Es un motor de simulación orientado a eventos discretos, rápido y eficiente.
- Permite el modelado mediante técnicas de programación orientada a objetos.
- Presenta un entorno de modelado jerárquico e interfaces gráficas para depuración y análisis.
- Brinda soporte para simulaciones paralelas con soporte para 32 y 64 bits.

- Consta de soporte para simulaciones distribuidas.
- Es un entorno diseñado para modelar protocolos y componentes mediante la técnica de estado (*Finite State Machines*), con los lenguajes de programación C/C++ y un conjunto de librerías propias.

El motor de simulación permite la ejecución eficiente y escalable de diferentes aspectos de la red a través de diversas técnicas de aceleración. El modelado del comportamiento de cada objeto que forma parte de la red se realiza mediante un paradigma de modelado sumamente simple y claro, las entidades que conforman un elemento de la red son modeladas a nivel de proceso y la interconexión entre estas entidades se realiza a nivel de nodo, a su vez los elementos de la red se conectan mediante enlaces a nivel de red. El comportamiento de los enlaces es programable y permite establecer retardos precisos, diversos niveles de error, características de utilización, etc.

La viabilidad y ventajas de la utilización de OPNET para la simulación del método de encaminamiento distribuido original (DRB) han sido puestas de manifiesto en los trabajos [22] y [23], en los cuales ha sido utilizado para modelar, analizar y realizar pruebas de DRB sobre InfiniBand.

5.2. Entorno de simulación

El entorno de simulación y modelado de OPNET proporciona soporte para el modelado de redes de comunicación y sistemas distribuidos. Esta herramienta es adecuada para diseñar y analizar equipamiento de comunicaciones y protocolos de red, permitiendo mejorar la fiabilidad y el rendimiento de los mismos.

OPNET está dotado de un sistema de modelado jerárquico de tres niveles:

- Nivel de red. Incluye nodos, enlaces y subredes interconectados entre sí, lo que permite establecer una gran cantidad de topologías diferentes. A este nivel se definen los atributos de los modelos y se configuran los parámetros de la simulación.
- Nivel de nodo. En este nivel se representan los componentes de red a través de módulos que poseen características como: procesamiento de mensajes (creación, transmisión, recepción y almacenamiento), encaminamiento interno, análisis de contenido, etc. Estos módulos normalmente representan aplicaciones, capas de protocolos y recursos físicos como buffers y puertos, entre otros.
- Nivel de proceso. En este nivel se programa el comportamiento de los módulos del nivel anterior en base a modelos de proceso. Estos modelos consisten en máquinas de estado finito (*Finite State Machines, FSM*) que contienen bloques de código de usuario en C/C++ y procedimientos del kernel de OPNET. Estas máquinas de estado responden a interrupciones generadas por el kernel de simulación y buscan representar la lógica de procesos del mundo real.

Estas características de OPNET lo convierten en una herramienta sumamente útil para modelar redes de interconexión de alta velocidad debido a que mantiene una buena relación entre nivel de abstracción y velocidad de simulación, provee una adecuada representación del hardware real, permite simulaciones de redes de gran tamaño y la inyección de trazas de aplicaciones paralelas reales.

Los modelos de OPNET que hemos diseñado y utilizado para realizar las simulaciones de nuestro trabajo se describen en las próximas secciones.

5.3. Modelos de red

Una red de interconexión de alta velocidad está compuesta por elementos de interconexión llamados encaminadores, conectados entre sí a través de enlaces formando topologías de red; y por nodos de procesamiento que poseen una interfaz que permite conectar el nodo a la red.

Los encaminadores son los bloques básicos de una red de interconexión y su función es transmitir los paquetes hacia sus destinos. Están provistos de una serie de canales de entrada para aceptar paquetes, y de canales de salida para transmitirlos. Cada canal dispone de un Control de Enlace de Entrada (CEE) o de un Control de Enlace de Salida (CES) dependiendo de si es de entrada o salida, respectivamente, y un buffer. Además poseen canales de entrada y de salida para conectar los nodos de procesamiento. La interconexión interna entre los canales de entrada y salida es realizada mediante un *crossbar*, y se administra a través de una unidad de encaminamiento y arbitraje. En la Fig. 5.1 se puede observar la estructura básica de un encaminador estándar.

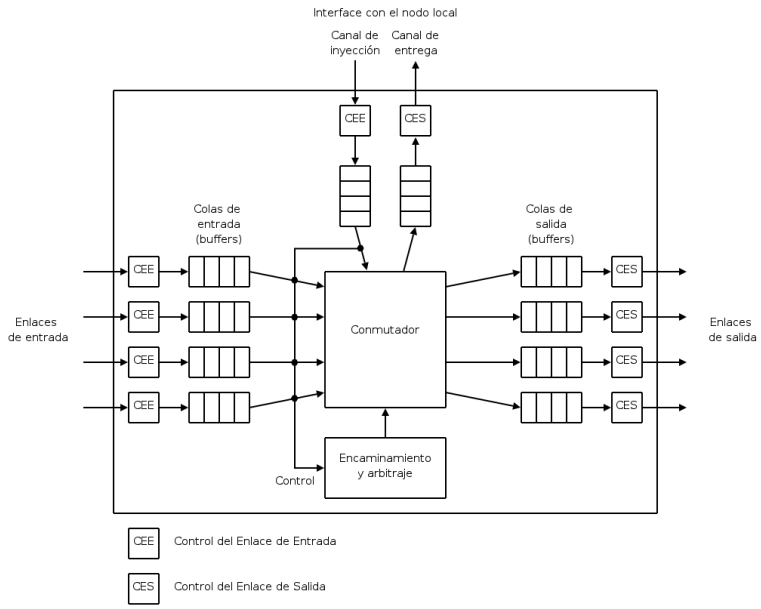


Figura 5.1: Estructura de un encaminador estándar

Los nodos de procesamiento, por su parte, incluyen un nodo de proceso que ejecuta las aplicaciones a partir del envío y recepción de mensajes, y una interfaz de red que permite conectar los nodos de proceso con los nodos de red (encaminadores).

En la Fig. 5.2 se observa un ejemplo del modelo de una red directa de tipo malla de 16 nodos (4x4). En ella se pueden observar los nodos de procesamiento, los enlaces y los encaminadores (los puntos de unión entre los enlaces).

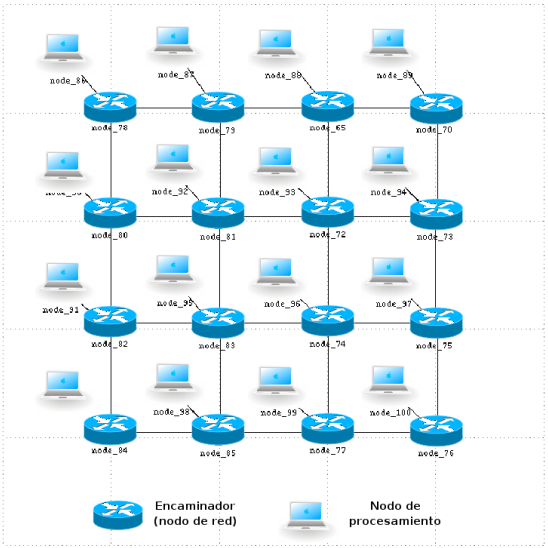


Figura 5.2: Modelo de red (malla 4x4)

Los elementos que constituyen los modelos de red poseen una serie de atributos que definen su funcionamiento y permiten parametrizar la simulación. En la Fig. 5.3 se puede observar el menú de atributos de uno de los encaminadores del modelo de la Fig. 5.2. En este menú es posible configurar, entre otros atributos, el nombre lógico del encaminador (*name*), a cuál de los puertos se quiere introducir el fallo (*fail_port*), el tiempo de inicio del fallo (*fail_start*) y el número de puertos físicos (*physical_ports*).

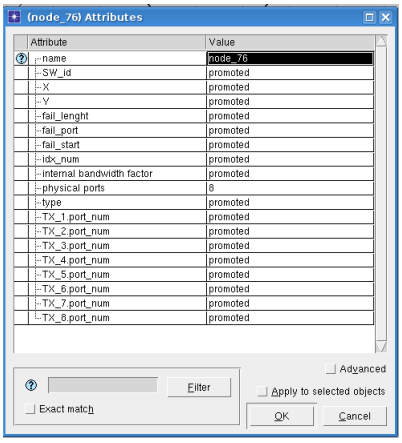


Figura 5.3: Atributos del encaminador

5.3.1. Implementación de los nodos de procesamiento

El modelo de nodo de procesamiento que se ha implementado incluye un nodo de proceso que simula el patrón de comunicaciones de una aplicación o una carga de trabajo sintética, y una interfaz de red para conectar los nodos con los encaminadores, que constituirán la topología de red. En la Fig. 5.4 se muestra de forma gráfica éste modelo. A la izquierda se muestra el nodo de proceso y a la derecha la interfaz de red.

Cada uno de los elementos que componen los modelos de nodos se diseñan en base a estados y transiciones, propios del enfoque de máquinas de estados finitos, donde los círculos representan los diferentes estados y los arcos las transiciones.

El nodo de proceso está compuesto por un módulo generador de paquetes (*src*), que genera paquetes de la capa de enlace de acuerdo al patrón de tráfico utilizado, por una función de probabilidad o a la traza de una aplicación, y por un módulo consumidor de paquetes (*dst*) encargado de analizar los paquetes recibidos para actualizar las estadísticas de tráfico y las métricas de rendimiento (latencia, throughput, etc.). Además, los nodos de proceso poseen varios atributos relacionados con la generación de paquetes como la tasa de inyección, los tiempos de inicio y finalización del intervalo de envío de paquetes, etc.

En las Figs. 5.5(a) y 5.5(b) se pueden observar gráficamente las máquinas de estado finito (FSM) correspondientes a los módulos generador y consumidor de paquetes.

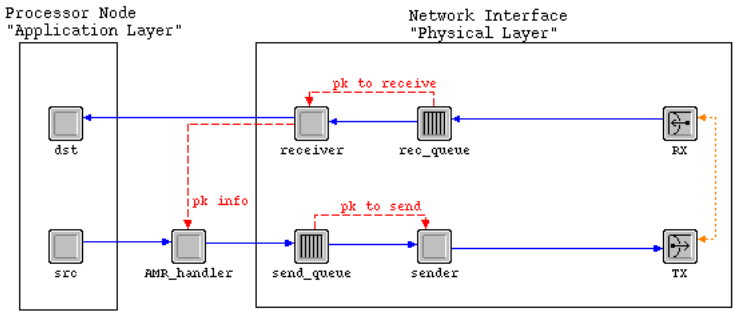
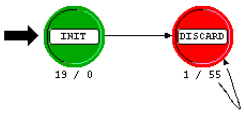


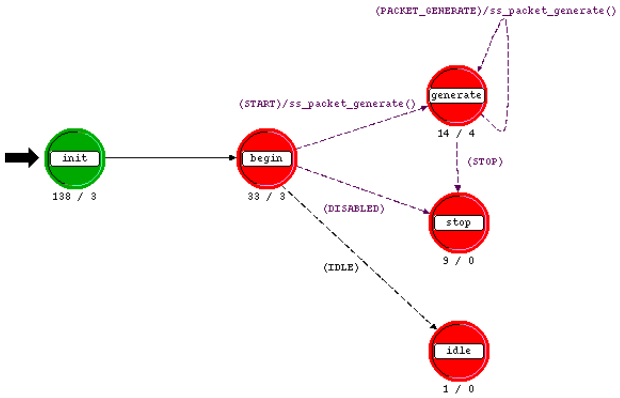
Figura 5.4: Implementación del modelo de nodos de procesamiento

El módulo generador de paquetes (Fig. 5.5(b)) se encarga de leer los parámetros de simulación propios de la generación de paquetes como el tamaño de los paquetes, el tiempo de inicio y fin de la generación, etc., a fin de determinar el modo de operación del nodo final. Luego de leídos los parámetros, la FSM se "moverá" hacia alguno de los tres estados posibles: generación de paquetes, fin de generación de paquetes o inactividad.

Entre el nodo de proceso y la interfaz de red del nodo final se ubica un módulo de administración que sirve de interfaz entre estos dos. Este módulo, denominado *AMR_handler*, y cuya FSM se muestra en la Fig. 5.6(a), se encarga de inicializar los parámetros necesarios para el envío de datos, de seleccionar el path de salida de los paquetes y de generar los mensajes de reconocimiento (*acknowledge*).



(a) Módulo consumidor de paquetes *dst*



(b) Módulo generador de paquetes *src*

Figura 5.5: FSM del nodo de proceso de los nodos de procesamiento

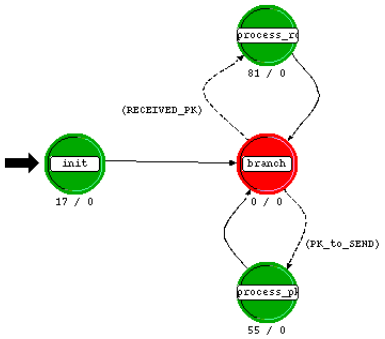
La interfaz de red del nodo final está compuesta por los buffers de entrada y de salida, *send_queue* y *rec_queue* respectivamente, por los controladores de envío y recepción, *sender* y *receiver*, y los módulos de envío y recepción puerto a puerto *TX* y *RX*. Las FSM de estos módulos se muestran en las Figs. 5.7(a) a 5.7(d).

El módulo *sender* es el encargado de recibir el mensaje enviado por el nodo de proceso y dividirlo en varios paquetes de acuerdo con el MTU de la interfaz de red. De manera similar, el módulo *receiver* es el responsable de analizar el orden de los paquetes, reunificarlos en un solo mensaje y finalmente entregarlo al nodo de proceso.

La lógica de los buffers se representa con las FSM de las Figs. 5.7(a) y 5.7(b). Estos modelos de proceso simulan colas asíncronas de tipo FIFO. En su estado inicial se inicializan las variables de estado y se procesa el parámetro que especifica el tamaño de la cola. Luego de la inicialización, el modelo espera en el estado *BRANCH* por la llegada de paquetes a la cola. Cuando un paquete llega se produce la transición al estado *INS_TAIL* en el cual se inserta el paquete a la cola. Luego se produce el envío del paquete, a un nodo vecino en el caso del buffer de salida (Fig. 5.7(a)) o al nodo de proceso en el caso del buffer de entrada (Fig. 5.7(b)).

5.3.2. Implementación de los encaminadores

El modelo de encaminador que hemos implementado está compuesto por los módulos *switch_info*, *crossbar*, *arbitration* y *routing*, que definen su lógica de comportamiento, y por los módulos *buffer_inX*, *buffer_outX*, *forwardX*, *RX_X* y *TX_X* (donde la *X* final corresponde a un



(a) Módulo de administración intermedio *AMR_handler*

Figura 5.6: FSM del *AMR_handler* de los nodos de procesamiento

número entre 1 y 8, asociado al número de puertos físicos del encaminador) que representan los puertos de entrada y salida con sus respectivos buffers. Es importante aclarar que si bien el encaminador posee 8 puertos físicos, estos son bidireccionales por lo que se tienen 8 de entrada y 8 de salida.

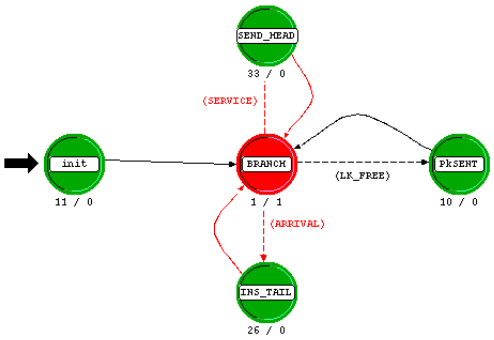
La estructura interna del modelo de encaminador de 8 puertos que hemos implementado se muestra en la Fig. 5.8. Comparándola con la Fig. 5.1 es posible observar las similitudes entre los diseños de ambos. En la Fig. 5.9(a) se puede observar el módulo *crossbar* que interconecta los 8 puertos de entrada/salida entre sí en nuestro encaminador. Éste módulo utiliza a su vez información de los módulos de *routing* y *switch_info* para realizar las conexiones entre los puertos.

El módulo *switch_info* es el que se encarga de la configuración inicial de la red (que se ejecuta una única vez por simulación). Es en este módulo donde son asignados los identificadores a cada uno de los elementos de la red, como por ejemplo los identificadores de puertos de los encaminadores. El diagrama de la FSM de este módulo se muestra en la Fig. 5.10(a).

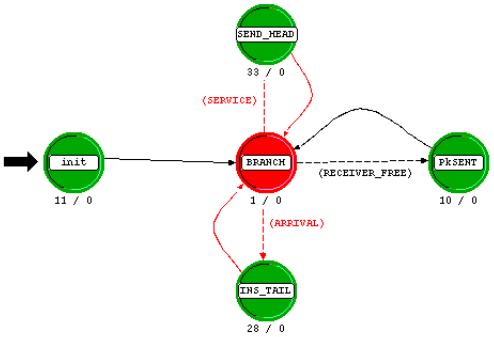
Las Figs. 5.11(a) y 5.11(b) representan los módulos de *routing* y *arbitration*, respectivamente. El módulo de *routing* asigna un puerto de salida a cada paquete entrante, mientras que el módulo *arbitration* procesa solicitudes simultáneas aplicando una política de tipo *round-robin*. Los paquetes que compiten por un canal de salida dado se almacenan en una estructura y son atendidos de forma secuencial.

Luego de que un paquete ha sido encaminado con éxito, se realiza un pedido al módulo *crossbar* para que permita que los paquetes pasen desde los buffers de entrada a los buffers de salida.

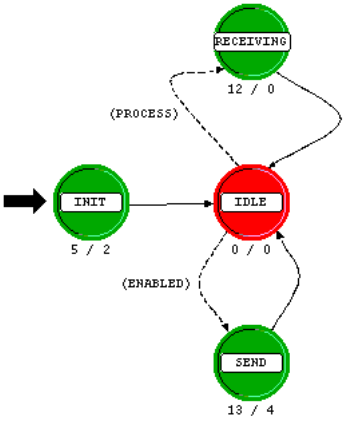
En resumen, hemos modelado un encaminador de 8 puertos completo en el cual pueden ser implementados diferentes algoritmos de encaminamiento de manera muy simple, lo que permite realizar comparaciones entre diferentes enfoques con relativa facilidad.



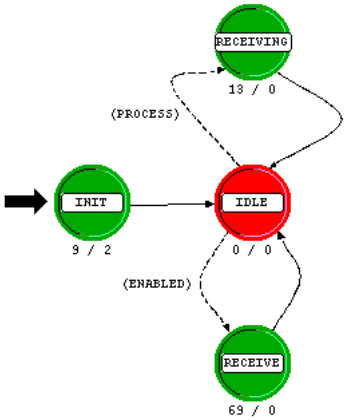
(a) Módulo del buffer de salida *send_queue*



(b) Módulo del buffer de entrada *rec_queue*



(c) Módulo del controlador de envío *sender*



(d) Módulo del controlador de recepción *receiver*

Figura 5.7: FSM de la interfaz de red de lo nodos de procesamiento

5.4. Conclusión

Hasta aquí hemos descrito en detalle los modelos que constituyen la implementación de nuestra propuesta. A partir de éstos modelos es posible realizar simulaciones completas de los diversos aspectos de nuestras políticas de encaminamiento tolerantes a fallos. En el capítulo siguiente se describen los experimentos realizados en base a éstos modelos, y se presentan y analizan los resultados obtenidos a partir de ellos.

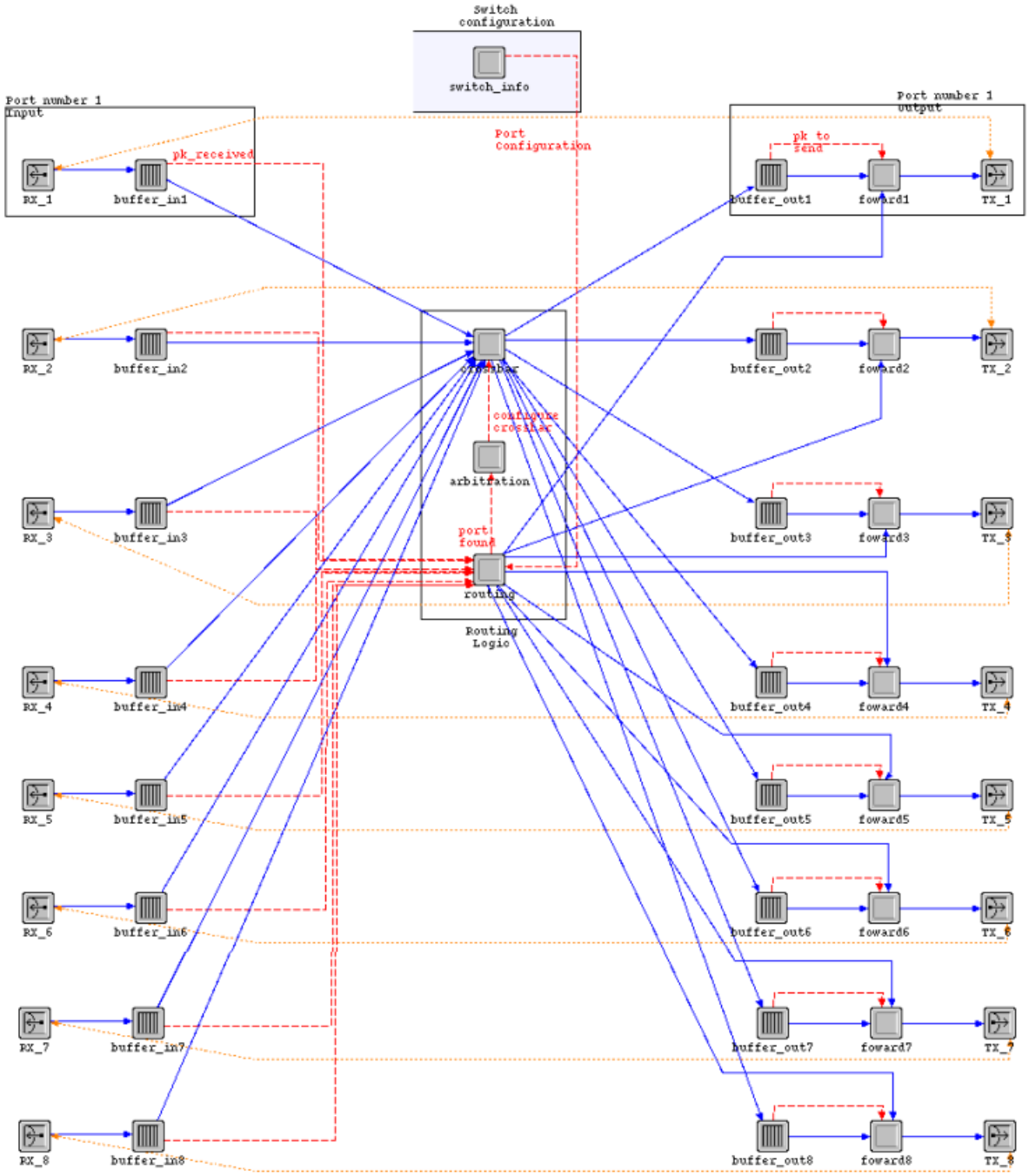
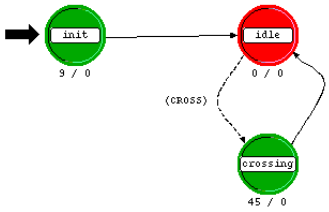
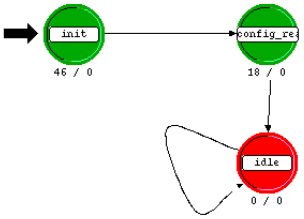


Figura 5.8: Implementación del modelo de encaminadores



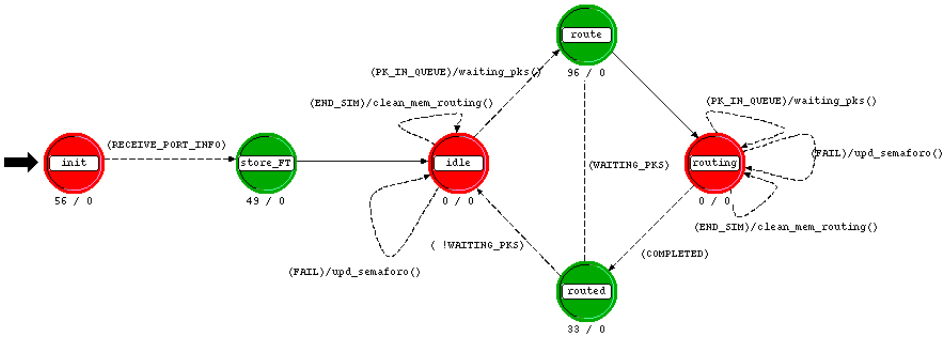
(a) Módulo *crossbar*

Figura 5.9: FSM del *crossbar*

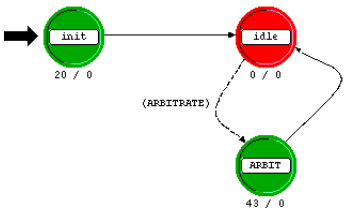


(a) Módulo *switch_info*

Figura 5.10: FSM del *switch_info*



(a) Módulo *routing*



(b) Módulo *arbitration*

Figura 5.11: FSM de los módulos de encaminamiento y arbitraje

Capítulo 6

Experimentación y análisis de resultados

6.1. Introducción

En este capítulo describimos los experimentos que han sido realizados a fin de estudiar el comportamiento de nuestra propuesta, y analizamos los resultados obtenidos a partir de los mismos.

El objetivo de esta experimentación es evaluar, por medio de la simulación, el comportamiento de nuestros desarrollos ante diferentes escenarios de fallos en la red. Esta experimentación nos permite conocer las capacidades y las limitaciones de nuestra propuesta.

En este nivel de desarrollo de la propuesta, la experimentación se encuentra principalmente orientada a comprobar el comportamiento de nuestras políticas de encaminamiento tolerantes a fallos, es decir, se busca demostrar experimentalmente el correcto funcionamiento de las mismas y que ha sido posible desarrollar un método de balanceo distribuido del encaminamiento tolerante a fallos.

Por este motivo, la experimentación aquí presentada se centra en la comparación del nivel de rendimiento de las redes en ausencia de fallos con las mismas redes en presencia de un número variable de fallos.

6.2. Diseño de los experimentos

Actualmente existe una gran variedad de topologías de red, tal y como se ha expuesto en el capítulo 2. De todas ellas, las mallas poseen características que las hacen muy interesantes, desde nuestro punto de vista, como por ejemplo los beneficios que presenta en cuanto al aprovechamiento de la localidad en las comunicaciones y la disponibilidad de un gran número de caminos alternativos entre pares de nodos origen-destino, además de ser una de las topologías más utilizadas en la actualidad. Por éstos motivos hemos decidido realizar la primera etapa

de nuestra experimentación sobre esta topología y, a corto plazo, ampliar la experimentación a otras topologías como *k-ary n-cube* y *fat-tree* ya que son muy utilizadas en máquinas reales.

Habiendo elegido la topología para evaluar el método, es posible definir los valores de los parámetros a medir, y la forma de lograrlo mediante el diseño de pruebas específicas.

Debido a que en esta primera etapa el objetivo de las pruebas es comprobar el correcto funcionamiento del método propuesto, la pruebas se realizan utilizando:

- Patrones de comunicación basados en una distribución aleatoria uniforme para la selección de los pares origen destino.
- Patrones de comunicación basados en el envío hacia un único nodo destino.
- Patrones de comunicación basados en el envío desde un único nodo origen.
- Patrones de comunicación punto a punto (nodo a nodo).

Con el objetivo de verificar el nivel de degradación del funcionamiento de las redes en presencia de fallos, medimos la latencia media de las comunicaciones de la red de interconexión.

La "latencia de comunicación" se mide como el tiempo total transcurrido desde que el mensaje viaja desde el nodo origen hasta el nodo destino, incluyendo el tiempo que el mensaje espera por ser inyectado en la red. Para un conjunto de valores de latencia Li medios, la "latencia media" $\bar{L}$ se calcula promediando todas las latencias de todos los mensajes enviados y se mide en segundos:

$$\bar{L} = \frac{1}{n} \sum_{i=1}^{n} Li \tag{6.1}$$

En base a esta medida de latencia media, se obtienen los porcentajes de degradación de las prestaciones de las redes de interconexión en presencia de fallos respecto a la latencia en ausencia de fallos.

Todos los experimentos han sido llevados a cabo utilizando la plataforma de simulación de redes de interconexión *OPNET Modeler*. Por este motivo se han incorporado a la plataforma mencionada todos los aspectos de funcionamiento del balanceo distribuido del encaminamiento mejorado, así como las modificaciones y funcionalidades adicionales en las que se basa nuestra propuesta, como se ha descrito en el capítulo anterior.

Entre las herramientas de simulación provistas por *OPNET* se encuentra el visualizador de animaciones, que utiliza la información de las simulaciones para mostrar gráficamente el comportamiento de las mismas. A partir de esta herramienta hemos podido realizar detallados estudios del comportamiento de nuestra propuesta, siguiendo la evolución paso a paso.

6.2.1. Diseño de las pruebas

La experimentación realizada está definida por la caracterización de los patrones de fallos introducidos, las características que conforman el espacio de diseño de la red de interconexión, las herramientas utilizadas para simular y evaluar el funcionamiento, y la metodología utilizada en la experimentación con el fin de obtener resultados, procesarlos y presentarlos en forma adecuada.

En virtud de lo mencionado, al momento de diseñar las pruebas se han elegido una serie de casos representativos para evaluar el comportamiento de nuestra propuesta. Vale mencionar que aplicar un enfoque de pruebas exhaustivas no representa ningún beneficio en esta situación, ya que la existencia de fallos en ciertos caminos de la red de interconexión no tiene ningún efecto a no ser que éstos deban ser utilizados.

Las pruebas fueron realizadas sobre una malla bidimensional de 64 nodos de red (8x8) y 1 nodo de procesamiento por cada encaminador, ejecutando simulaciones de diferentes escenarios de prueba, variando la ubicacion de los fallos y los patrones de comunicación, a fin de evaluar la degradación del rendimiento de la red en presencia de fallos.

Si bien los modelos que hemos desarrollado permiten realizar simulaciones de fallos de enlaces tanto unidireccionales como bidireccionales, nos hemos limitado a la simulación de fallos de enlaces bidireccionales.

Los parámetros de simulación comunes a todos los escenarios de prueba se detallan en la tabla 6.1.

Parámetro	Valor	Descripción
Topología	Malla 2D	Topología utilizada en la simulación
Tamaño de red	8x8 (64 nodos)	Tamaño de la red simulada
Tiempo de simulación	2400 seg (40 min)	Tiempo total simulado
Inicio de inyección de paquetes	10 seg	Tiempo en que se empiezan a inyectar paquetes a la red
Fin de inyección de paquetes	2400 seg	Tiempo en que se dejan de inyectar paquetes a la red
Tasa de inyección de paquetes	Constante: 1 paq/seg	Tasa con la que cada nodo de procesamiento inyecta paquetes en la red
Tamaño de paquetes	1024 bytes	Tamaño de los paquetes utilizados

Tabla 6.1: Parámetros de simulación constantes en todos los escenarios

Para la elección del tiempo de simulación hemos tenido en cuenta el periodo inicial de "*warm-up*" de la red, en el cual la relación entre la latencia y el tiempo presenta un crecimiento casi lineal, y que en el caso de nuestros experimentos va desde el inicio de la simulación hasta aproximadamente los 1100 segundos de la simulación. Luego de esta etapa inicial, la relación se mantiene constante a lo largo de la simulación (en ausencia de fallos) debido a que estamos utilizando una tasa de inyección de paquetes constante y no estamos saturando la red. En la Fig. 6.1 se pueden observar estos valores (con los valores de latencia normalizados en base al

valor en que se hace constante).

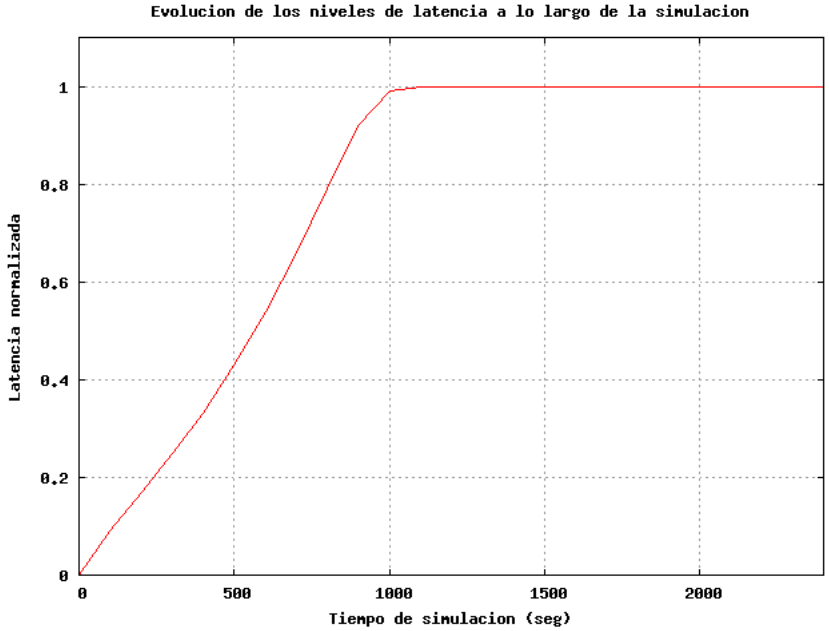


Figura 6.1: Relación entre los valores de tiempo y latencia (normalizada)

Antes de definir los escenarios de prueba debemos especificar la forma como nombramos los nodos de la red utilizando los ejes cartesianos. Como se puede ver en la Fig. 6.2, consideramos como punto *(0,0)* la esquina superior izquierda de la red, con los valores del *eje x* creciendo hacia la derecha y los valores del *eje y* creciendo hacia abajo.

Otra aclaración necesaria sobre la notación utilizada es la forma de identificar los enlaces que se utiliza para describir la ubicación de los fallos en los escenarios de pruebas. El método que hemos utilizado se basa en enumerar los enlaces siguiendo una orientación horaria, tal y como se muestra en la Fig. 6.3, de forma de utilizar esta numeración en conjunto con el identificador de los encaminadores de la forma *(x,y)-z*, donde *(x,y)* identifica la posición del encaminador y *z* el número del enlace relativo a ese encaminador. Por ejemplo el identificador *(1,2)-1* hace referencia al enlace superior del encaminador *(1,2)*.

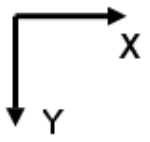


Figura 6.2: Dirección de los ejes de coordenadas

El enfoque que hemos utilizado para definir los experimentos se basa en la simulación de comunicaciones punto a punto y comunicaciones colectivas, entendidas como las operaciones en las que intervienen el movimiento de datos y control globales [3, cap. 5].

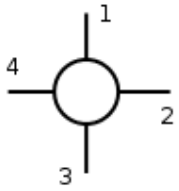


Figura 6.3: Numeración de enlaces en cada encaminador

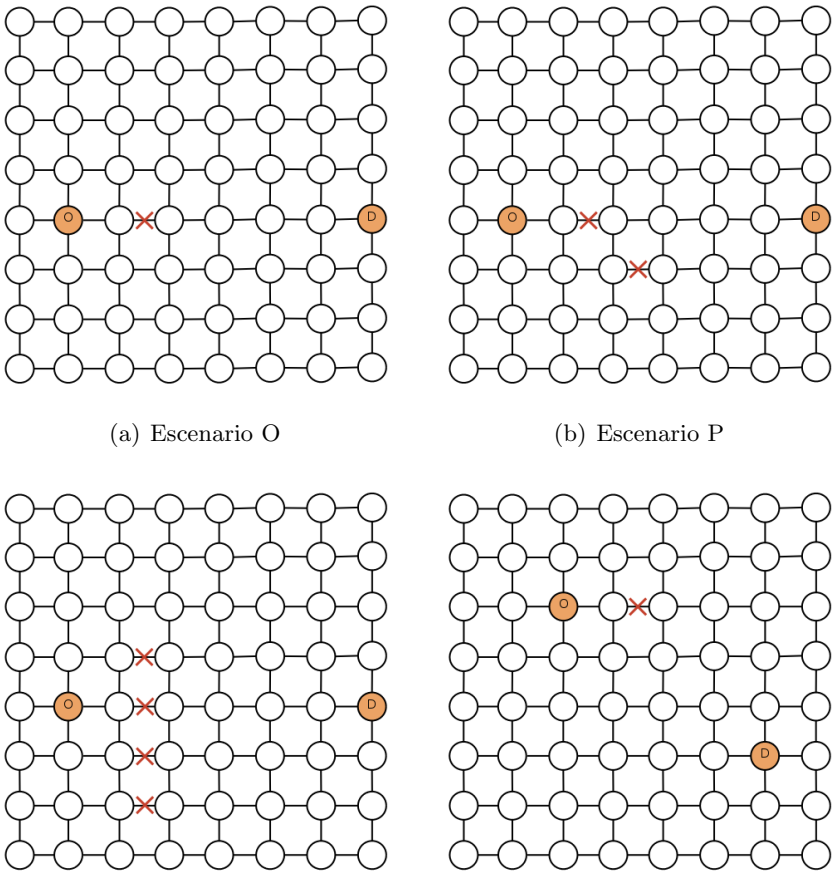
Para la realización de las pruebas de comunicación punto a punto, fueron definidos cuatro escenarios de pruebas distintos, tres de los cuales se basan en un mismo par de nodos origen-destino (*O-D*) colineales respecto al *eje y*, mientras que el escenario restante se basa en un par de nodos origen-destino (*O-D*) cuya ubicación no coincide en ninguna de las dos dimesiones de la malla. En las Figs. 6.4(a), 6.4(b), 6.4(c) y 6.4(d) se muestran estos cuatro escenarios.

Para estos escenarios de comunicación punto a punto se realizaron pruebas de comunicamación tanto unidireccional como bidireccional entre los pares de nodos origen-destino *O-D*. Esta referencia a los términos unidireccional y bidireccional no se refiere a las posibilidades de comunicación de los enlaces sino al envío de información desde el origen hacia el destino en el caso unidireccional, y en el caso bidireccional también se envía desde el destino al origen.

En la tabla 6.2 se describen las características de estos cuatro escenarios, y en la tabla 6.3 se describe el conjunto de pruebas realizadas a partir de ellos.

Escenarios de pruebas de comunicaciones punto a punto		
Nombre	*Tipo de prueba*	*Ubicación de los fallos*
Escenario Base O,P,Q	Comunicacion punto a punto. Origen y destino colineales	0 fallos
Escenario O	Comunicacion punto a punto. Origen y destino colineales	1 fallo: (2,4)-2
Escenario P	Comunicacion punto a punto. Origen y destino colineales	2 fallos: (2,4)-2; (3,5)-2
Escenario Q	Comunicacion punto a punto. Origen y destino colineales	4 fallos: (2,3)-2; (2,4)-2; (2,5)-2; (2,6)-2
Escenario Base R	Comunicacion punto a punto. Origen y destino no coincidentes	0 fallos
Escenario R	Comunicacion punto a punto. Origen y destino no coincidentes	1 fallo: (3,2)-2

Tabla 6.2: Caracterización de los escenarios de pruebas de comunicaciones punto a punto



(a) Escenario O

(b) Escenario P

(c) Escenario Q

(d) Escenario R

Figura 6.4: Escenarios de pruebas de comunicaciones punto a punto

Pruebas de comunicaciones punto a punto			
Escenario	*Comunicación*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base O,P,Q	Unidireccional	11 seg	2400 seg
Base R	Unidireccional	11 seg	2400 seg
O	Unidireccional	11 seg	2400 seg
P	Unidireccional	11 seg	2400 seg
Q	Unidireccional	11 seg	2400 seg
R	Unidireccional	11 seg	2400 seg
Base O,P,Q	Bidireccional	11 seg	2400 seg
Base R	Bidireccional	11 seg	2400 seg
O	Bidireccional	11 seg	2400 seg
P	Bidireccional	11 seg	2400 seg
Q	Bidireccional	11 seg	2400 seg
R	Bidireccional	11 seg	2400 seg

Tabla 6.3: Caracterización de las pruebas de comunicaciones punto a punto

La experimentación sobre las comunicaciones colectivas que hemos realizado son del tipo:

- Comunicaciones All-to-All de tipo All-scatter. Todos los nodos de procesamiento envían diferentes mensajes a todos los demás nodos de procesamiento.
- Comunicaciones One-to-All de tipo Scatter. Un solo nodo de procesamiento envía mensajes a todos los demás nodos de procesamiento.
- Comunicaciones All-to-One de tipo Gather. Todos los nodos de procesamiento (excepto el receptor) envían mensajes a un solo nodo de procesamiento receptor.

Los experimentos realizados sobre las comunicaciones colectivas se basan en nueve escenarios de pruebas comunes a las tres clasificaciones expuestas anteriormente. Estos escenarios presentan diferentes disposiciones y combinaciones de 1, 2, 3 y 12 fallos. Además, a fin de realizar estudios más profundos sobre algunas de las características específicas de estas comunicaciones, se realizaron experimentos con algunos escenarios adicionales.

En las Figs. 6.5(a) a 6.5(i) se pueden observar los 9 escenarios comunes, y en las Figs. 6.6(a) a 6.6(e) los escenarios adicionales.

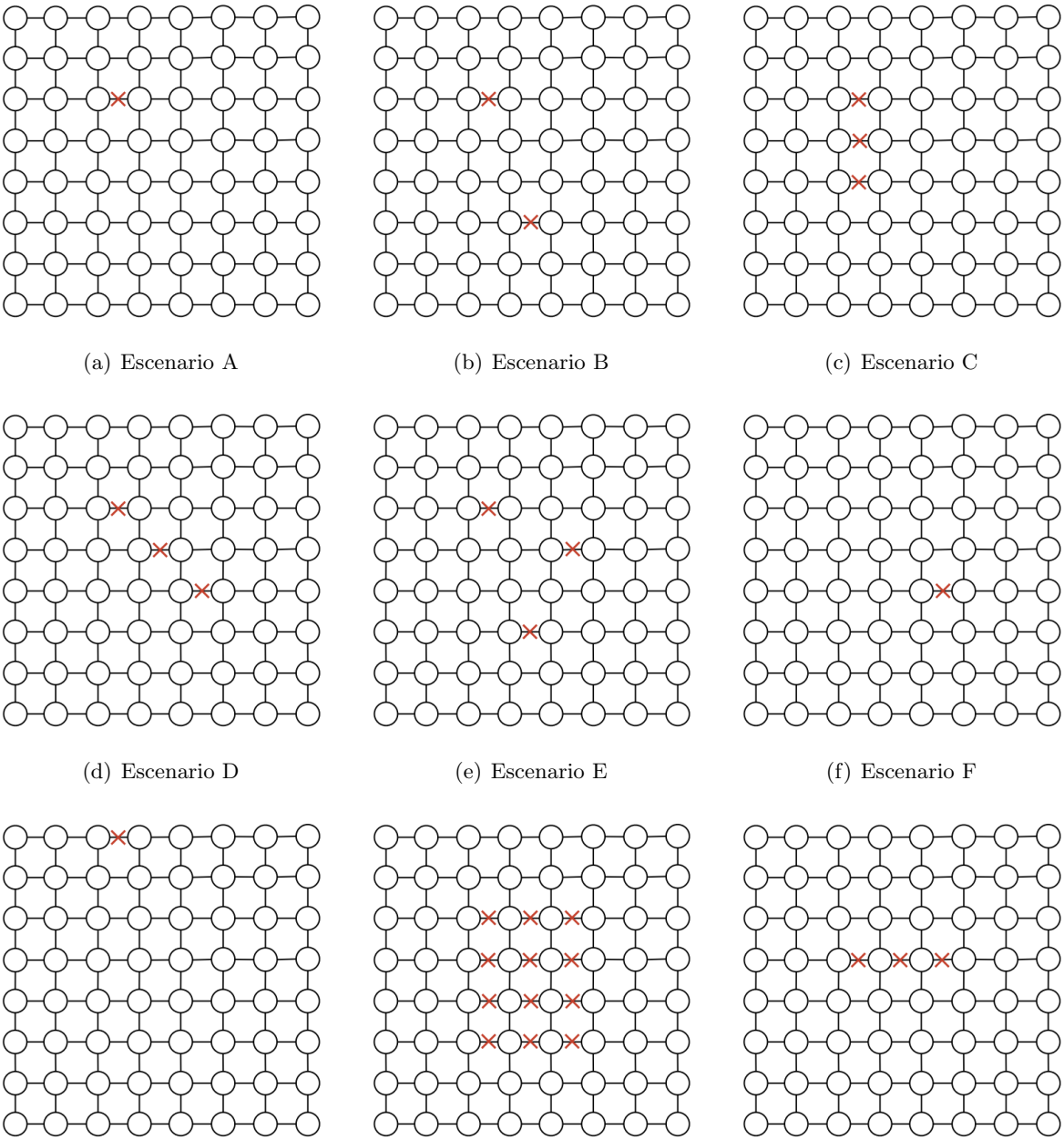
Para el caso de las comunicaciones *One-to-All* y *All-to-One*, hemos realizado dos conjuntos de experimentos, utilizando dos nodos origen o destino diferentes, según corresponda, a fin de observar las posibles diferencias en el rendimiento de la red de acuerdo a la ubicación relativa de los fallos respecto de dichos nodos. El primer nodo seleccionado se encuentra en la zona central de la red, en la cuarta posición en el *eje x* y tercera en el *eje y* (vale aclarar que se los enumera desde cero), mientras que el segundo nodo se encuentra en la esquina inferior derecha de la red *(0,7)*.

En la tabla 6.4 se describen las características de los 14 escenarios (los comunes a todos los patrones más los adicionales), y en la tablas 6.5, 6.6 y 6.7 se describe el conjunto de pruebas realizadas a partir de ellos, discriminadas por tipo de comunicación.

Los resultados de estos experimentos se presentan y analizan en la siguiente sección.

6.3. Análisis de resultados

En este punto se presentan los resultados extraídos de toda la experimentación realizada en base a lo que hemos descrito en la sección anterior. A fin de facilitar la exposición de los resultados, hemos dividido la presentación de los mismos a partir de la clasificación del tipo de comunicación en *punto a punto*, *colectivas all-to-all*, *colectivas one-to-all*, *colectivas all-to-one*. Para finalizar presentamos un resumen de todo el conjunto de experimentos realizados.

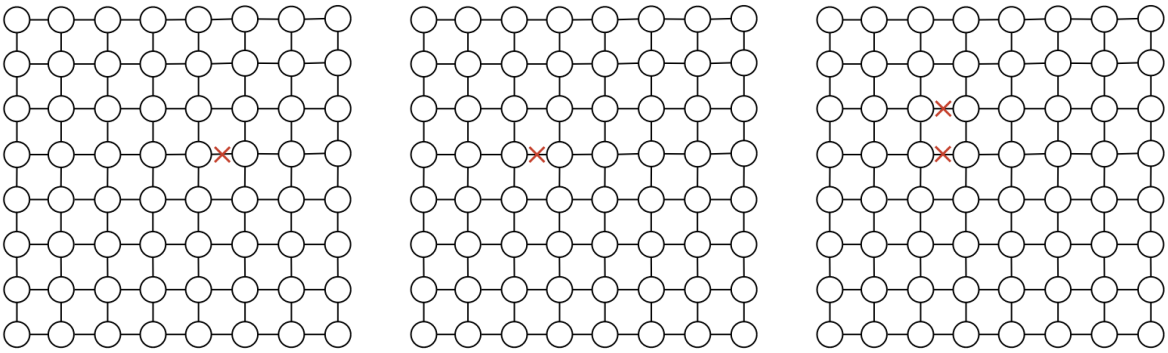


(a) Escenario A (b) Escenario B (c) Escenario C

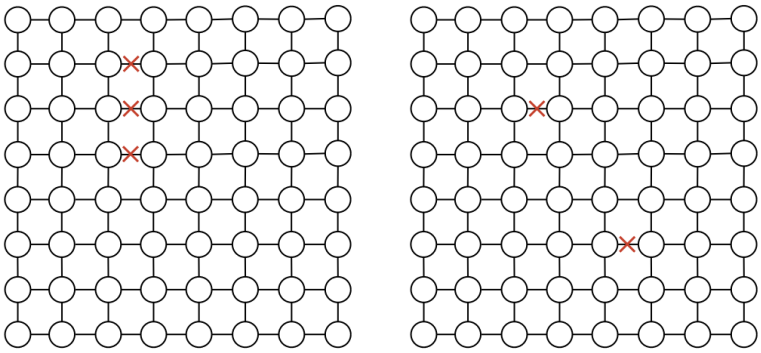
(d) Escenario D (e) Escenario E (f) Escenario F

(g) Escenario G (h) Escenario H (i) Escenario I

Figura 6.5: Escenarios de pruebas de comunicaciones colectivas



(a) Escenario J (b) Escenario K (c) Escenario L



(d) Escenario M (e) Escenario N

Figura 6.6: Escenarios de pruebas de comunicaciones colectivas adicionales

Escenarios de pruebas de comunicaciones colectivas		
Escenarios de pruebas comunes		
Nombre	*Tipo de prueba*	*Ubicación de los fallos*
Escenario Base	Comunicacion colectiva.	0 fallos
Escenario A	Comunicacion colectiva.	1 fallo: (2,2)-2
Escenario B	Comunicacion colectiva.	2 fallos: (2,2)-2; (3,5)-2
Escenario C	Comunicacion colectiva.	3 fallos: (2,2)-2; (2,3)-2; (2,4)-2
Escenario D	Comunicacion colectiva.	3 fallos: (2,2)-2; (3,3)-2; (4,4)-2
Escenario E	Comunicacion colectiva.	3 fallos: (2,2)-2; (4,3)-2; (3,5)-2
Escenario F	Comunicacion colectiva.	1 fallo: (4,4)-2
Escenario G	Comunicacion colectiva.	1 fallo: (2,0)-2
Escenario H	Comunicacion colectiva.	12 fallos: (2,2)-2; (3,2)-2; (4,2)-2; (2,3)-2; (3,3)-2; (4,3)-2; (2,4)-2; (3,4)-2; (4,4)-2; (2,5)-2; (3,5)-2; (4,5)-2
Escenario I	Comunicacion colectiva.	3 fallos: (2,3)-2; (3,3)-2; (4,3)-2
Escenarios de pruebas adicionales		
Nombre	*Tipo de prueba*	*Ubicación de los fallos*
Escenario J	Comunicacion colectiva.	1 fallo: (4,3)-2
Escenario K	Comunicacion colectiva.	1 fallo: (2,3)-2
Escenario L	Comunicacion colectiva.	2 fallos: (2,2)-2; (2,3)-2
Escenario M	Comunicacion colectiva.	3 fallos: (2,1)-2; (2,2)-2; (2;3)-2
Escenario N	Comunicacion colectiva.	2 fallos: (2,2)-2; (4,5)-2

Tabla 6.4: Caracterización de los escenarios de pruebas de comunicaciones colectivas

Pruebas de comunicaciones colectivas All-to-All			
Escenario	*Distribución*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base	Uniforme	11 seg	2400 seg
A	Uniforme	11 seg	2400 seg
B	Uniforme	11 seg	2400 seg
C	Uniforme	11 seg	2400 seg
D	Uniforme	11 seg	2400 seg
E	Uniforme	11 seg	2400 seg
F	Uniforme	11 seg	2400 seg
G	Uniforme	11 seg	2400 seg
H	Uniforme	11 seg	2400 seg
I	Uniforme	11 seg	2400 seg
K	Uniforme	11 seg	2400 seg
L	Uniforme	11 seg	2400 seg
M	Uniforme	11 seg	2400 seg

Tabla 6.5: Caracterización de las pruebas de comunicaciones colectivas *All-to-All*

Pruebas de comunicaciones colectivas One-to-All			
Nodo origen (4,3)			
Escenario	*Distribución*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base	Uniforme	11 seg	2400 seg
A	Uniforme	11 seg	2400 seg
B	Uniforme	11 seg	2400 seg
C	Uniforme	11 seg	2400 seg
D	Uniforme	11 seg	2400 seg
E	Uniforme	11 seg	2400 seg
F	Uniforme	11 seg	2400 seg
G	Uniforme	11 seg	2400 seg
H	Uniforme	11 seg	2400 seg
I	Uniforme	11 seg	2400 seg
Nodo origen (0,7)			
Escenario	*Distribución*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base	Uniforme	11 seg	2400 seg
A	Uniforme	11 seg	2400 seg
B	Uniforme	11 seg	2400 seg
C	Uniforme	11 seg	2400 seg
D	Uniforme	11 seg	2400 seg
E	Uniforme	11 seg	2400 seg
F	Uniforme	11 seg	2400 seg
G	Uniforme	11 seg	2400 seg
H	Uniforme	11 seg	2400 seg
I	Uniforme	11 seg	2400 seg
J	Uniforme	11 seg	2400 seg
K	Uniforme	11 seg	2400 seg

Tabla 6.6: Caracterización de las pruebas de comunicaciones colectivas *One-to-All*

Pruebas de comunicaciones colectivas All-to-One			
Nodo destino (4,3)			
Escenario	*Distribución*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base	Todos al nodo (4,3)	11 seg	2400 seg
A	Todos al nodo (4,3)	11 seg	2400 seg
B	Todos al nodo (4,3)	11 seg	2400 seg
C	Todos al nodo (4,3)	11 seg	2400 seg
D	Todos al nodo (4,3)	11 seg	2400 seg
E	Todos al nodo (4,3)	11 seg	2400 seg
F	Todos al nodo (4,3)	11 seg	2400 seg
G	Todos al nodo (4,3)	11 seg	2400 seg
H	Todos al nodo (4,3)	11 seg	2400 seg
I	Todos al nodo (4,3)	11 seg	2400 seg
N	Todos al nodo (4,3)	11 seg	2400 seg
Nodo destino (0,7)			
Escenario	*Distribución*	*Tiempo inicio fallos*	*Tiempo fin fallos*
Base	Todos al nodo (0,7)	11 seg	2400 seg
A	Todos al nodo (0,7)	11 seg	2400 seg
B	Todos al nodo (0,7)	11 seg	2400 seg
C	Todos al nodo (0,7)	11 seg	2400 seg
D	Todos al nodo (0,7)	11 seg	2400 seg
E	Todos al nodo (0,7)	11 seg	2400 seg
F	Todos al nodo (0,7)	11 seg	2400 seg
G	Todos al nodo (0,7)	11 seg	2400 seg
H	Todos al nodo (0,7)	11 seg	2400 seg
I	Todos al nodo (0,7)	11 seg	2400 seg
J	Todos al nodo (0,7)	11 seg	2400 seg

Tabla 6.7: Caracterización de las pruebas de comunicaciones colectivas *All-to-One*

6.3.1. Comunicaciones punto a punto

El principal objetivo de la experimentación de las comunicaciones punto a punto no es la obtención de valores de rendimiento significativos sino analizar paso a paso, con ayuda de la herramienta de visualización de animaciones de *OPNET*, el correcto funcionamiento de nuestra propuesta.

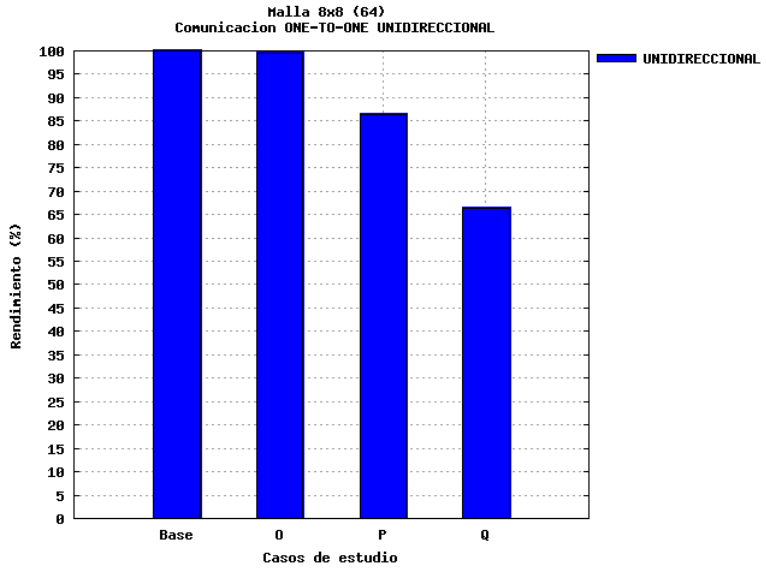
Los resultados numéricos obtenidos a partir de la simulación de los 4 escenarios de prueba, para comunicaciones tanto unidireccionales y bidireccionales, se detallan en la tabla 6.8, en la que se exponen, para cada escenario, el número de fallos que presentan, el rendimiento obtenido (calculado en base a la latencia media) y el nivel de degradación (diferencia con el rendimiento del caso base). Éstos mismos valores de rendimiento se muestran gráficamente en las Figs. 6.7(a) y 6.7(b), para el caso unidireccional, y 6.8(a) y 6.8(b) para el caso bidireccional. En ambos casos, se muestran las valores de rendimiento de los escenarios de prueba respecto de los escenarios base (es decir, sin fallos) correspondientes. Es importante recordar el hecho de que los escenarios *O, P y Q* utilizan un par de nodos origen-destino diferente al del escenario *R*, motivo por el cual en estas experimentaciones se utilizan dos escenarios bases distintos.

Resultados de las pruebas de comunicaciones punto a punto			
Comunicación unidireccional			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base O,P,Q	0	100 %	0 %
O	1	99,608 %	0,392 %
P	2	86,408 %	13,592 %
Q	4	66,405 %	33,595 %
Rendimiento promedio (escenarios O,P,Q): 84.14 %			
Base R	0	100 %	0 %
R	1	100 %	0 %
Comunicación bidireccional			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base O,P,Q	0	100 %	0 %
O	1	75,575 %	24,425 %
P	2	59,532 %	40,468 %
Q	4	32,376 %	67,624 %
Rendimiento promedio (escenarios O,P,Q): 53,83 %			
Base R	0	100 %	0 %
R	1	88,802 %	11,198 %

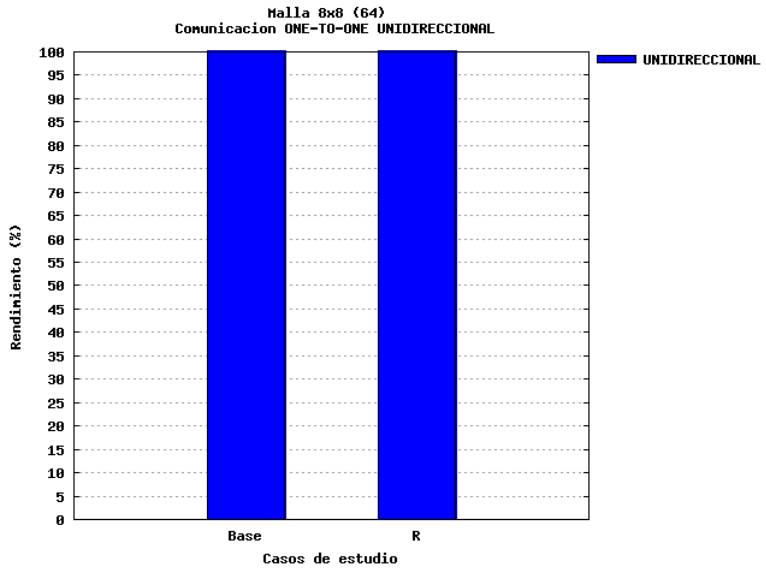
Tabla 6.8: Resultados de las pruebas de comunicaciones punto a punto

A simple vista parecería que los valores de rendimiento y degradación obtenidos para los escenarios *O, P y Q* no son muy satisfactorios, pero este análisis no es correcto. El error en éste análisis se puede verificar tomando en cuenta la longitud de los caminos entre los pares de nodos origen-destino que se deben utilizar para evitar los fallos presentes en éstos escenarios de prueba. El incremento de la longitud de los caminos es de alrededor del 30 % para el escenario *O* y llega

a un 50 % en el escenario Q, mientras que la degradación de las prestaciones es de 0,392 % y 33,595 % respectivamente por lo que los valores de degradación obtenidos son muy satisfactorios. Los valores de rendimiento disminuyen considerablemente en el caso de las comunicaciones bidireccionales, pero se debe considerar que el tráfico es dos veces el del caso unidireccional. Para el caso específico del escenario Q de las pruebas de comunicación bidireccional, se debe tener en cuenta el hecho de que el incremento de la longitud del camino, desde el nodo destino hacia el nodo origen, es de cerca del 90 % al momento de encontrar el fallo por primera vez.

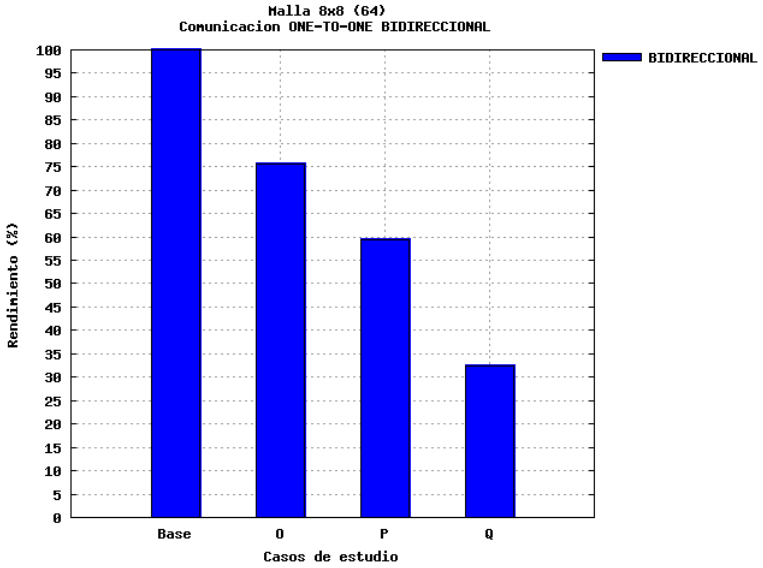


(a) Comunicación unidireccional - Escenarios O, P, Q

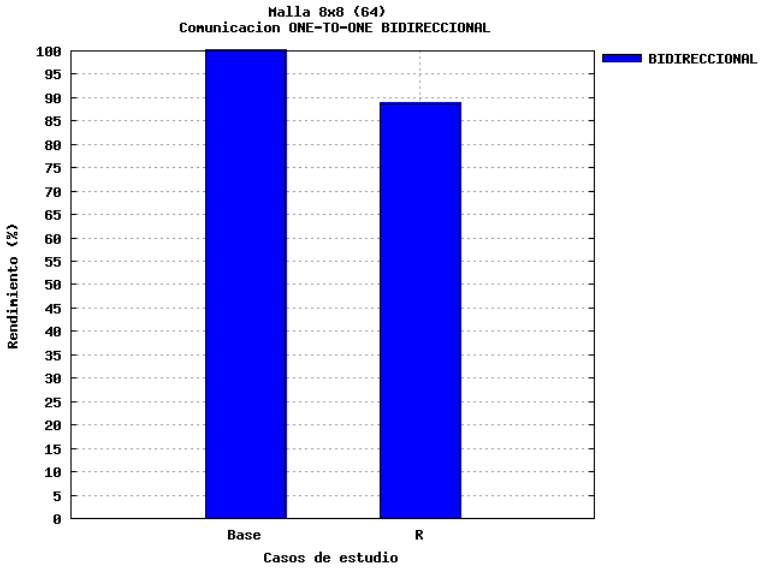


(b) Comunicación unidireccional - Escenario R

Figura 6.7: Resultados de pruebas de comunicaciones punto a punto (unidireccional)



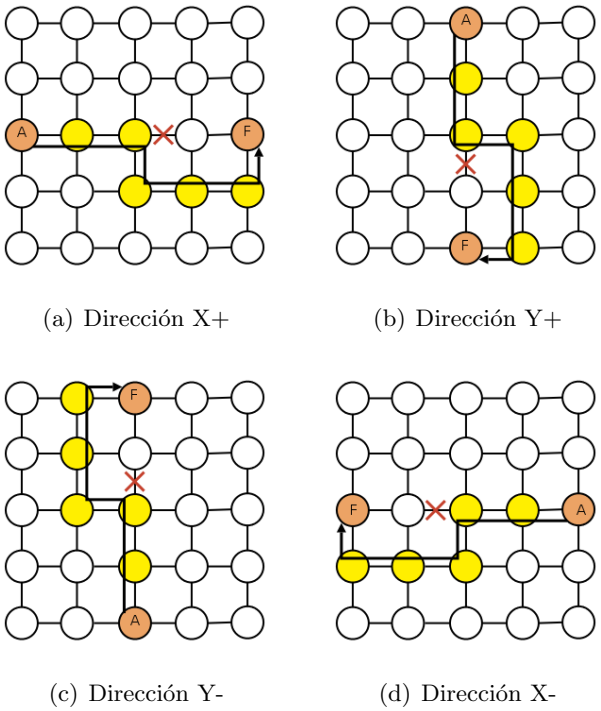
(a) Comunicación bidireccional - Escenarios O, P, Q



(b) Comunicación bidireccional - Escenario R

Figura 6.8: Resultados de pruebas de comunicaciones punto a punto (bidireccional)

Éste incremento de las longitudes se debe al hecho de que la selección de las vías de escape en los encaminadores intermedios, para el caso de pares origen-destino colineales, se realiza en base a reglas predefinidas de direccionamiento, ya que no se conoce *a priori* el número y la disposición de los fallos. En las Figs. 6.9(a) a 6.9(d) se pueden observar estas formas de elección de vías de escape para las cuatro direcciones posibles en una malla 2D.



(a) Dirección X+ (b) Dirección Y+

(c) Dirección Y- (d) Dirección X-

Figura 6.9: Formas de selección de las vías de escape

Para el escenario R en el caso de comunicación unidireccional el rendimiento es del 100 % debido a dos factores. El primero de ellos es que al encontrar el fallo en el camino original, la elección de la vía de escape se realiza en la dirección $Y+$, de manera que el nuevo camino hacia el destino sigue siendo de longitud mínima. El segundo es que al ser un par de nodos origen destino no colineales existen más de un camino mínimo entre ellos, por ejemplo el camino XY original y el YX. Éste segundo camino es que el que elige DRB-E al momento de abrir caminos alternativos en este escenario. Para el caso de la comunicación bidireccional en este escenario, el rendimiento disminuye debido a que al encontrar el fallo, cualquiera de las opciones de vías de escape elegidas incrementa la longitud del camino.

A continuación explicaremos brevemente el funcionamiento de nuestras políticas de encaminamiento tolerantes a fallos para los escenarios P y R en el caso de comunicaciones unidireccionales.

Escenario P

La operativa del método en este escenario puede ser dividida en cuatro pasos tal y como se muestra en las Figs. 6.10(a) a 6.10(d):

- Paso 1. El nodo origen O intenta enviar un mensaje al nodo destino D pero se encuentra con un fallo en el camino original. El nodo intermedio, al detectar el intento de utilización de un camino fallido, envía un mensaje de notificación al nodo origen sobre dicho fallo (Fig. 6.10(a)).

- Paso 2. El nodo intermedio abre una vía de escape y envía el mensaje a través de ella. Al encontrar el segundo fallo, se repite el mismo paso, pero esta vez no se envía la notificación al nodo origen (Fig. 6.10(b)).

- Paso 3. El nodo origen, al haber recibido la notificación del fallo en el camino original, abre un camino alternativo en la dirección $Y+$. El nodo intermedio que detecta el intento de utilización de este nuevo camino fallido envía un mensaje de notificación al nodo origen sobre el fallo, y abre una vía de escape para enviar el mensaje al nodo destino (Fig. 6.10(c)).

- Paso 4. El nodo origen, en lugar de continuar abriendo caminos en la dirección $Y+$, decide abrir un nuevo camino en la dirección $Y-$ debido a que éste camino, en principio, presenta una longitud inferior (Fig. 6.10(d)).

Escenario R

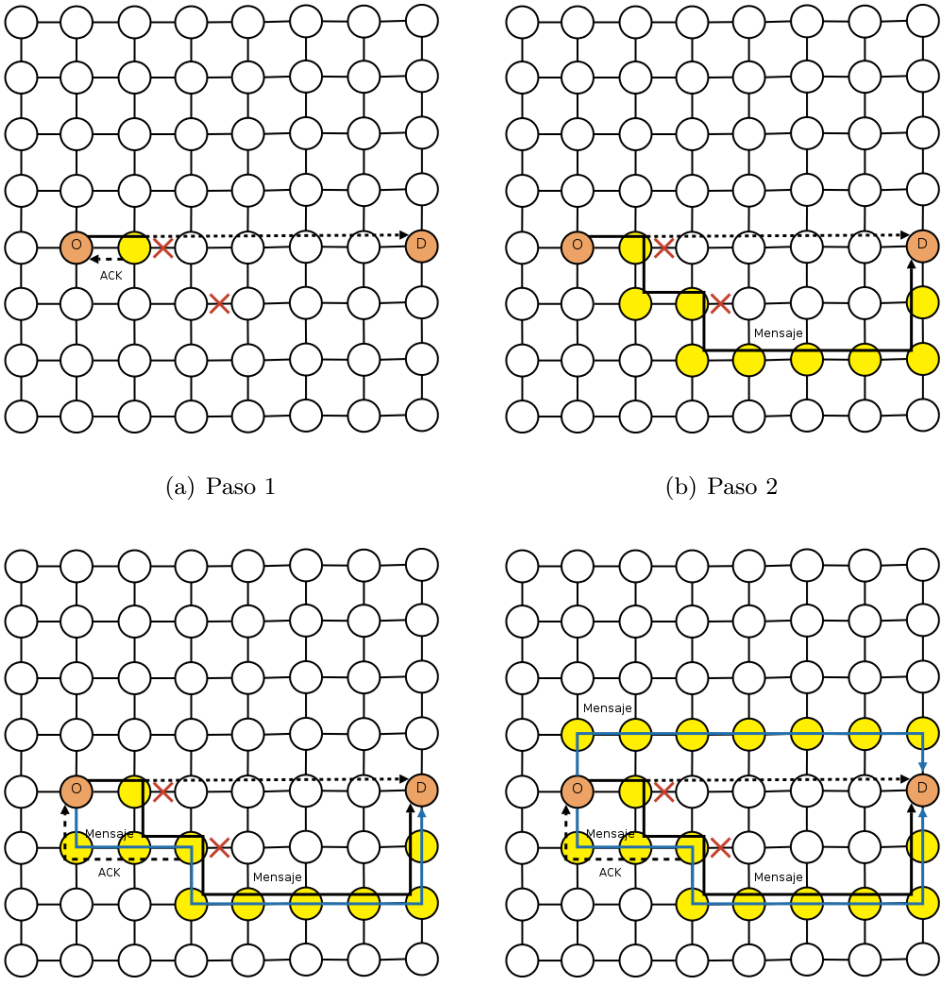
Debido a que en el escenario anterior ya se ha explicado el modo de utilización de los mensajes de notificación de caminos fallidos, explicaremos el funcionamiento del método para el caso del escenario R (Fig. 6.11(a)) en base a tres pasos (Figs. 6.11(b) a 6.11(d)):

- Paso 1. El nodo origen O intenta enviar un mensaje al nodo destino D a través del camino XY original que presenta un fallo (Fig. 6.11(b)).

- Paso 2. En este paso el nodo intermedio que detecta el fallo abre una vía de escape para enviar el mensaje hacia el nodo destino. En la Fig. 6.11(c) se puede observar que éste nuevo camino posee la misma longitud que el camino original.

- Paso 3. En este paso el nodo origen abre un camino alternativo utilizando el encaminamiento YX. Este camino sigue siendo de longitud mínima (Fig. 6.11(d)).

Es aquí donde se pueden apreciar los beneficios de la ventaja que presentan las redes de tipo malla de contar con más de un camino de longitud mínima. En el caso de utilizar redes de tipo toro, la cantidad de caminos alternativos sería aún mayor.



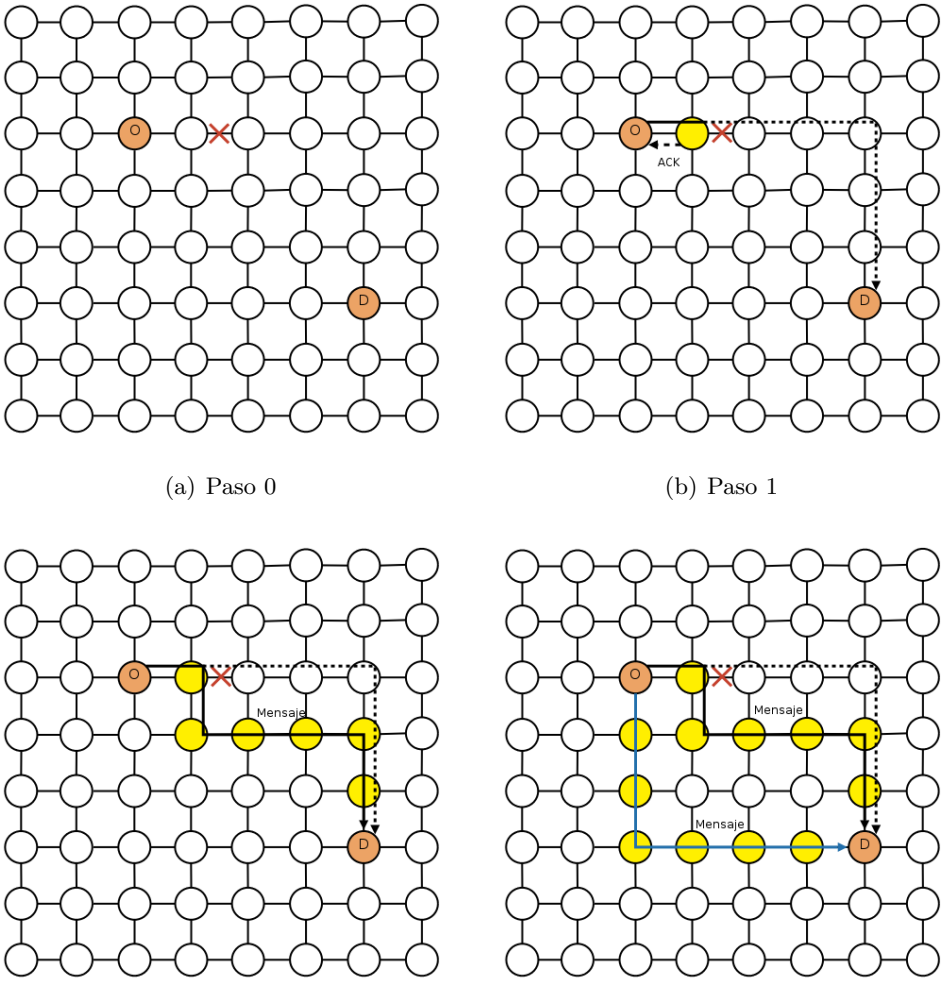
(a) Paso 1

(b) Paso 2

(c) Paso 3

(d) Paso 4

Figura 6.10: Comportamiento del escenario P



(a) Paso 0

(b) Paso 1

(c) Paso 2

(d) Paso 3

Figura 6.11: Comportamiento del escenario R

6.3.2. Comunicaciones colectivas all-to-all

A diferencia del caso de las comunicaciones punto a punto, las comunicaciones colectivas representan escenarios de comunicación más realistas y operaciones comúnmente utilizadas en los sistemas de cómputo de altas prestaciones.

En la tabla 6.9 se presentan los resultados de las pruebas realizadas para las comunicaciones colectivas de tipo *all-to-all*, más especificamente las de tipo *all-scatter*.

Resultados de las pruebas de comunicaciones all-to-all			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base	0	100 %	0 %
A	1	95,703 %	4,297 %
B	2	91,246 %	8,754 %
C	3	97,651 %	2,349 %
D	3	91,558 %	8,442 %
E	3	87,912 %	12,088 %
F	1	93,885 %	6,115 %
G	1	97,278 %	2,722 %
H	12	94,356 %	5,644 %
I	3	94,585 %	5,155 %
K	1	94,845 %	5,415 %
L	2	96,308 %	3,692 %
M	3	96,952 %	3,048 %
Rendimiento promedio: 94,34 %			

Tabla 6.9: Resultados de las pruebas de comunicaciones colectivas all-to-all

En la Fig. 6.12 se muestran gráficamente los valores de rendimiento obtenidos de la experimentación, mientras que en la Fig. 6.13 se muestra la misma información pero agrupada y promediada por número de fallos, en lugar de por escenarios. Las gráficas de comparación de rendimientos individuales agrupadas por fallos se muestran en las Figs. 6.14, 6.15 y 6.16, para 1, 2 y 3 fallos, respectivamente.

Para los escenarios de prueba de 1 y 2 fallos los rendimientos obtenidos son superiores al 90 %, llegando a alcanzarse un 97,65 % de rendimiento en el escenario C (de 3 fallos). En todos los escenarios de 3 fallos, a excepción del *E*, los rendimientos también superan el 90 %. Además, para el caso de prueba de 12 fallos se ha obtenido un rendimiento de más del 94 %.

El rendimiento promedio para este tipo de comunicaciones, en el que se tienen 64 nodos de procesamiento enviando mensajes a los demás 63 nodos, es del 94,34 %, resultando más que satisfactorio para este patrón de comunicaciones intensivo.

Al observar los valores de rendimiento de la tabla 6.9 nos encontramos con el hecho de que el rendimiento del escenario *C*, de 3 fallos, es mejor que el de los escenarios *A, G* e *I*, de un solo fallo. Incluso los escenarios *L* y *M* poseen mejores rendimientos que el *A*. La explicación de las diferencias de rendimiento entre los escenarios de un solo fallo (*A, F, G, K*) se relaciona con la posición de los fallos dentro de la red y el hecho de que la comunicación sea de tipo *all-to-all*. A

medida que los fallos se van situando más cerca de los extremos, éstos interfieren en un menor número de caminos entre pares de nodos origen-destino por lo que esto mejora los valores de latencia obtenidos. Por ejemplo, el rendimiento del escenario G es mejor que el del A debido a que en el primero el fallo se ubica en un enlace que se encuentra en el límite superior de la malla.

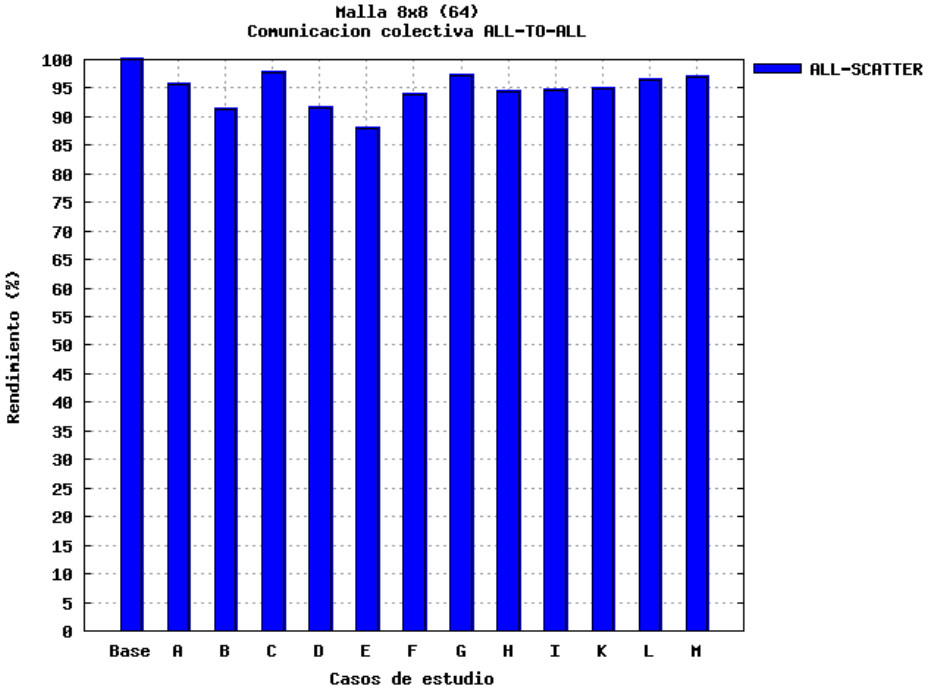


Figura 6.12: Resultados de las pruebas de comunicaciones all-to-all (por casos)

La explicación de los mejores rendimientos de algunos escenarios con mayor cantidad de fallos respecto a otros con menor cantidad es más complicada y tiene relación con el nivel de carga de la red y las características de distribución de carga de DRB-E.

Antes que nada debemos recordar dos puntos fundamentales: primero, que nuestra experimentación fue realizada con patrones de inyección de mensajes uniformes y hemos evitado congestionar y saturar la red, a fin de analizar el comportamiento de nuestra propuesta respecto a la aparición de fallos sin que se vea afectado por éstas situaciones; y segundo, que en esta topología de red puede existir más de un camino de longitud mínima.

Si bien en nuestros experimentos no se produjeron grandes problemas de congestión, los niveles de latencia de la red, junto con los valores del umbral de apertura de caminos de DRB-E, hacen que los nodos origen tiendan a abrir más de un camino entre pares de nodos origen destino antes de que se sature la red (situación que no se produce debido a que la inyección de paquetes en la red es constante). En presencia de fallos, algunos de éstos caminos son de mayor longitud debido a que se debe rodear la zona del fallo, lo que incrementa los valores de latencia, aunque según la longitud total del camino y el número de fallos, éstos valores de latencia pueden permanecer dentro del umbral de utilización de DRB-E. En el caso de que el número de fallos se incremente, estas zonas de fallos serán de mayor dimensión por lo que rodearlas implica

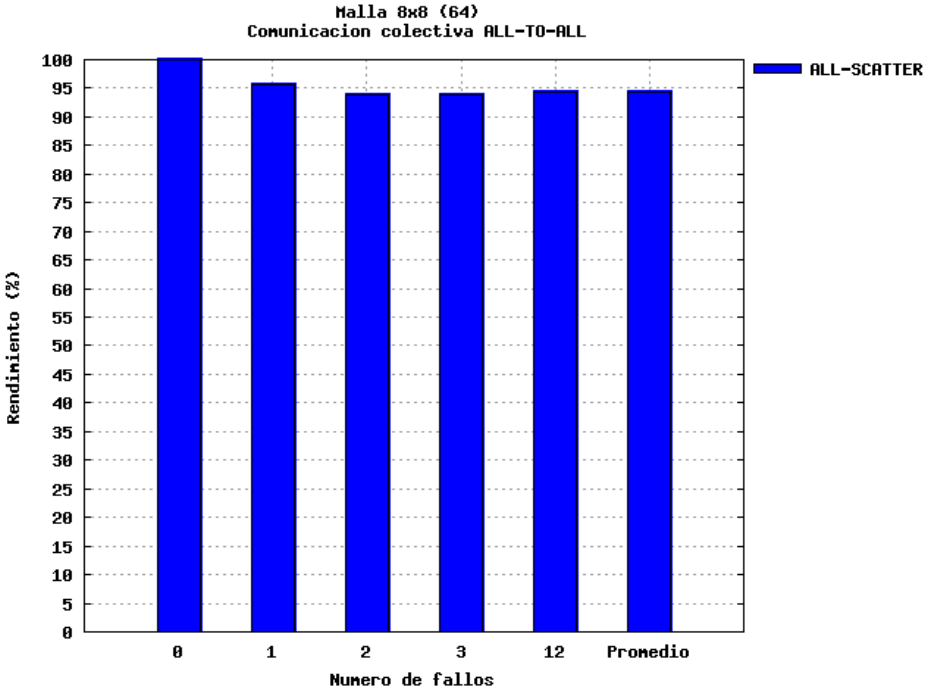


Figura 6.13: Resultados de las pruebas de comunicaciones all-to-all (por número de fallos)

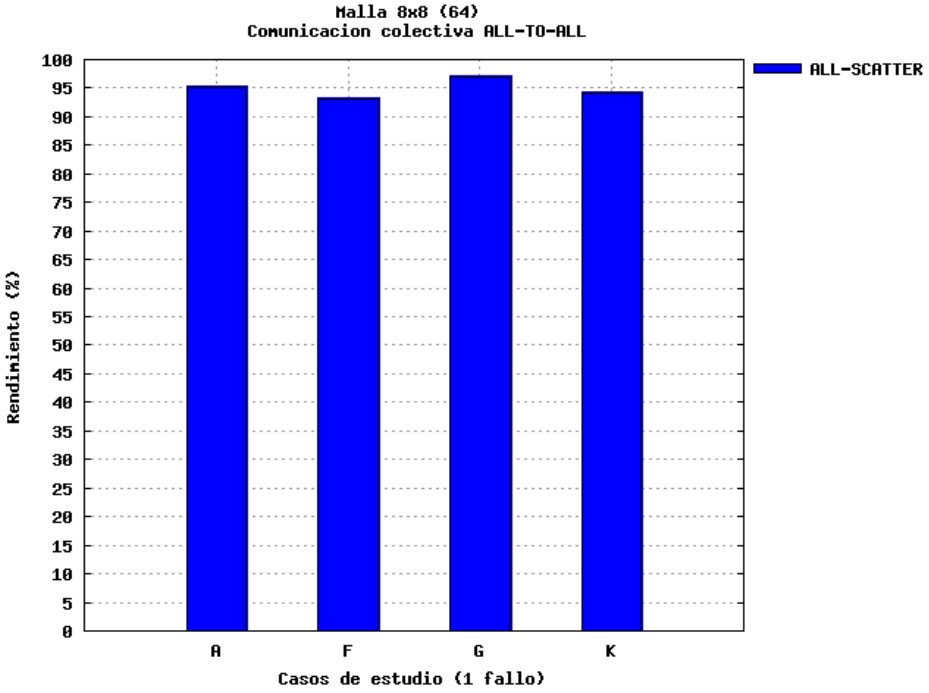


Figura 6.14: Resultados de las pruebas de comunicaciones all-to-all (para 1 fallo)

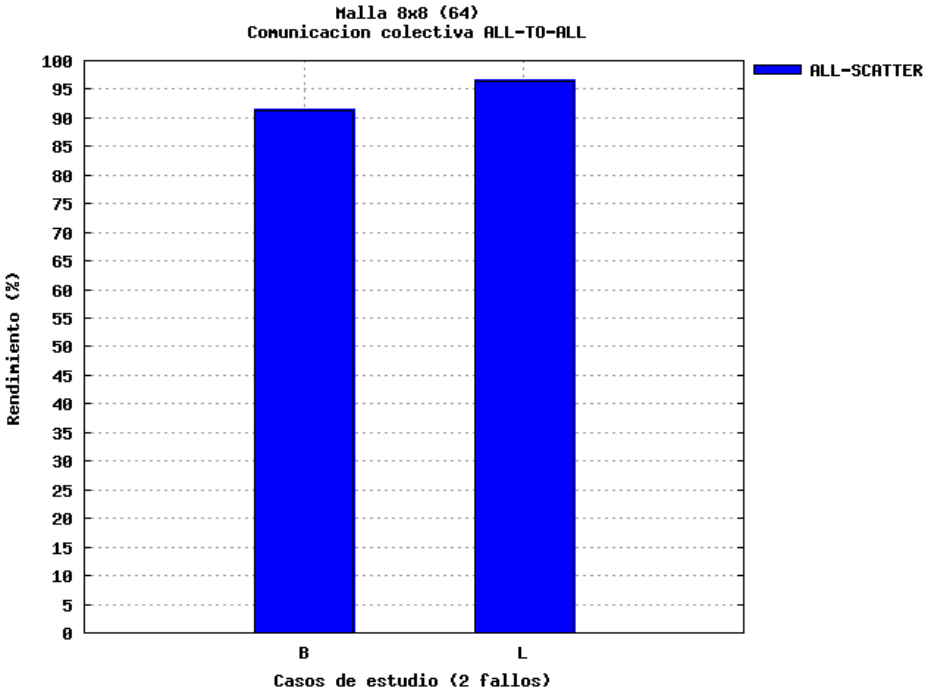


Figura 6.15: Resultados de las pruebas de comunicaciones all-to-all (para 2 fallos)

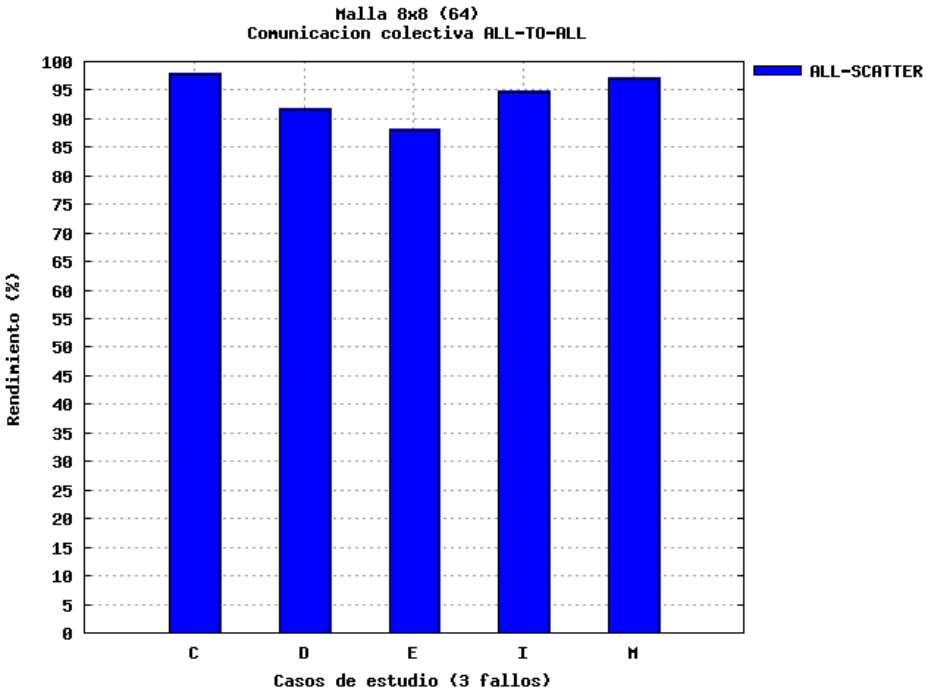


Figura 6.16: Resultados de las pruebas de comunicaciones all-to-all (para 3 fallos)

aumentar aún más la longitud de los caminos, motivo por el cual DRB-E deja de utilizar algunos de éstos caminos alternativos y mantiene abiertos solo los de longitud mínima. Éste cierre de los caminos de mayor longitud, debido al incremento del número de fallos, es el responsable de que el rendimiento en escenarios como el C sea mejor que el del A.

En el caso de que no utilizáramos una tasa de inyección de paquetes constante y se produjeran situaciones de congestión, la utilización de caminos alternativos sería en verdad provechosa, y el rendimiento del escenario A sería considerablemente mejor que el del escenario C.

A partir de estas situaciones hemos descubierto que, al momento de abrir caminos en los nodos origen, es necesario diferenciar el motivo que da origen a la apertura: congestión o fallos.

6.3.3. Comunicaciones colectivas one-to-all

Éste tipo de comunicaciones colectivas es el que se utiliza en frameworks de tipo *master-worker* cuando el proceso *master* envía información a los procesos *workers*.

Los resultados de los experimentos para este tipo de comunicaciones se muestran en la tabla 6.10. En esta se presentan por separado las dos series de experimentos realizados, una tomando como origen el nodo *(3,4)* y la otra tomando como origen el nodo *(0,7)*.

En las Figs. 6.17(a) y 6.17(b) se muestran gráficamente los valores de rendimiento obtenidos de la experimentación para cada una de estas series, separados por escenarios, mientras que en las Figs. 6.18(a) y 6.18(b) se muestran los mismos valores pero agrupados y promediados por número de fallos.

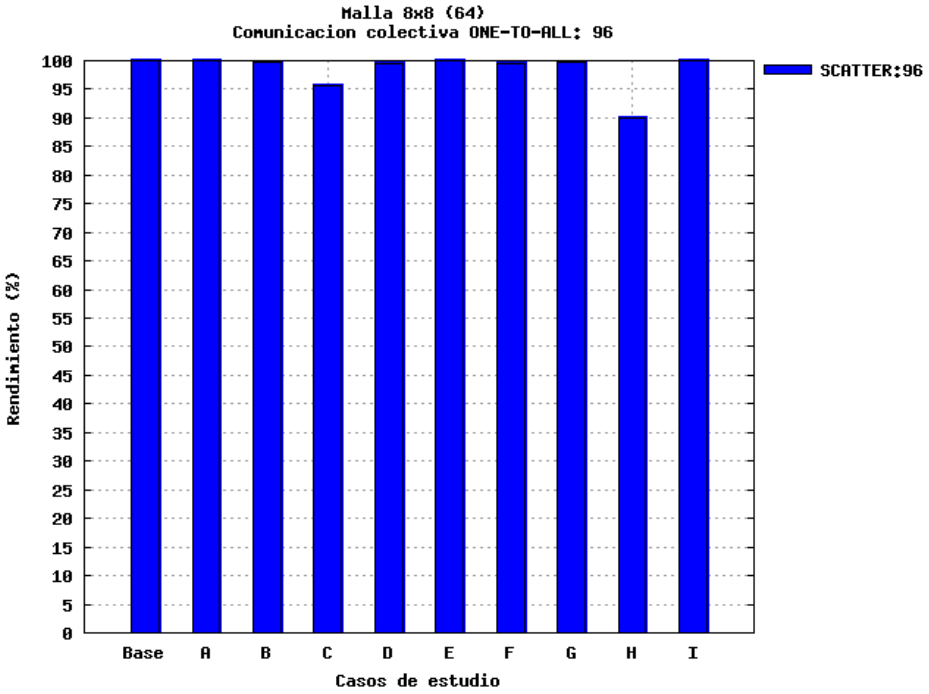
Los valores de rendimientos por número de fallos se muestran en las Figs. 6.19(a), 6.19(b) para los casos de 1 fallo, y 6.20(a) y 6.20(b) para los casos de 3 fallos.

Al igual que para las comunicaciones *all-to-all*, en el caso de comunicaciones *one-to-all* tomando como origen el nodo *(3,4)* hemos obtenidos mejores rendimientos en escenarios de 3 fallos (escenario I) respecto a escenarios de 1 solo fallo (escenario A) e incluso variaciones en los rendimientos de los escenarios de un solo fallo. Las mejoras para escenarios con mayor número de fallos se deben al mismo motivo que para las comunicaciones *all-to-all*, es decir, el nivel de apertura de caminos de DRB-E y las variaciones de las longitudes de los caminos utilizados. Por su parte, las diferencias entre escenarios de un solo fallo, se deben a las distintas ubicaciones de los fallos y la cantidad de caminos que éstos afectan.

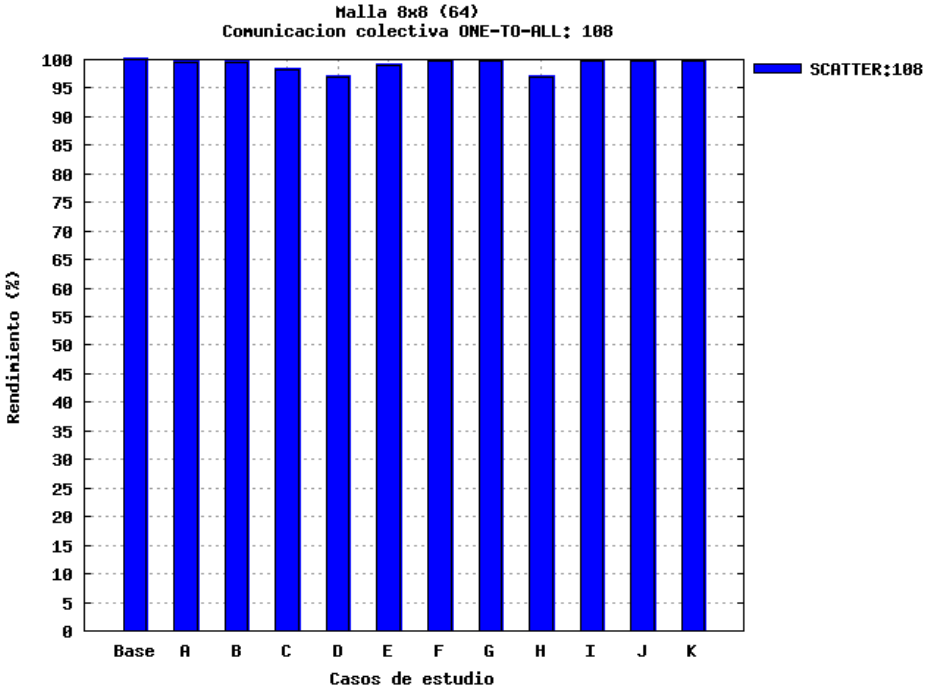
Al utilizar como nodo origen el nodo *(0,7)* se hicieron aún más evidentes las relaciones entre la ubicación, no solo de los fallos, sino también de los nodos origen y destino de mensajes. Debido a que el nodo *(0,7)* se ubica en la esquina inferior derecha de la red, a la utilización del encaminamiento XY, y debido también a las distancias entre éste nodo y los posibles destinos, el número de caminos de longitud mínima entre pares de nodos fuente destino es mayor que al utilizar el nodo origen *(3,4)*. En la mayor parte de las situaciones en que se encuentra un fallo a lo largo del camino entre el par de nodos origen destino, el comportamiento es muy similar al presentado en la explicación del escenario R para las comunicaciones punto a punto unidireccionales (pero con las direcciones de desplazamiento invertidas).

Resultados de las pruebas de comunicaciones one-to-all			
Nodo origen (3,4)			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base	0	100 %	0 %
A	1	99,910 %	0,090 %
B	2	99,830 %	0,170 %
C	3	95,718 %	4,282 %
D	3	99,421 %	0,579 %
E	3	99,950 %	0,050 %
F	1	99,586 %	0,414 %
G	1	99,801 %	0,199 %
H	12	90,090 %	9,910 %
I	3	99,994 %	0,006 %
Rendimiento promedio: 98,26 %			
Nodo origen (0,7)			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base	0	100 %	0 %
A	1	99,585 %	0,415 %
B	2	99,585 %	0,415 %
C	3	98,092 %	1,908 %
D	3	97,020 %	2,980 %
E	3	98,848 %	1,152 %
F	1	99,855 %	0,145 %
G	1	99,709 %	0,291 %
H	12	97,020 %	2,980 %
I	3	99,709 %	0,291 %
J	3	99,792 %	0,208 %
K	3	99,709 %	0,291 %
Rendimiento promedio: 98,99 %			

Tabla 6.10: Resultados de las pruebas de comunicaciones colectivas one-to-all

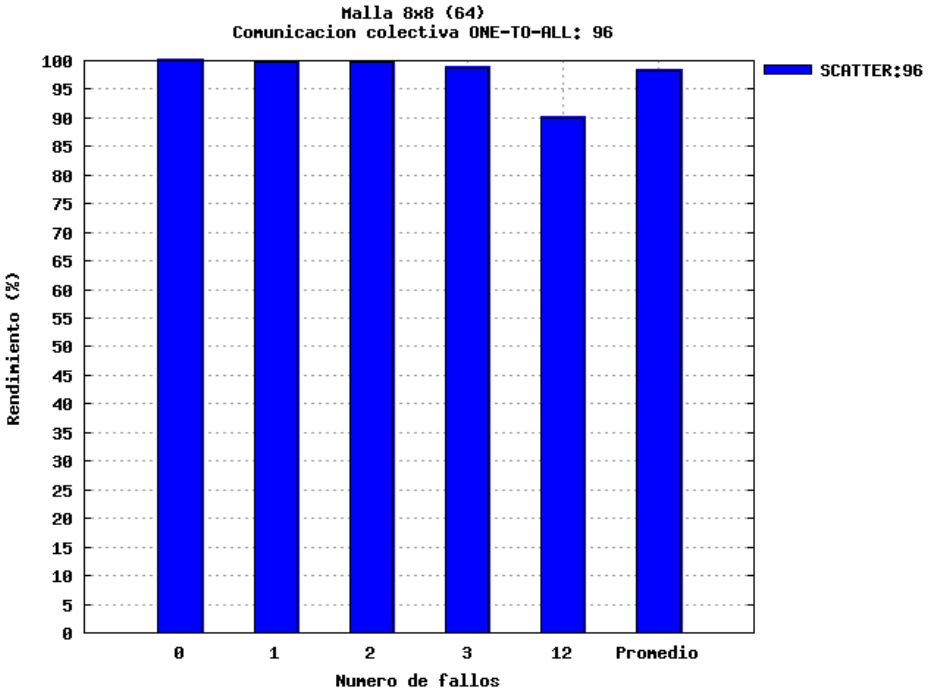


(a) Nodo origen (3,4)

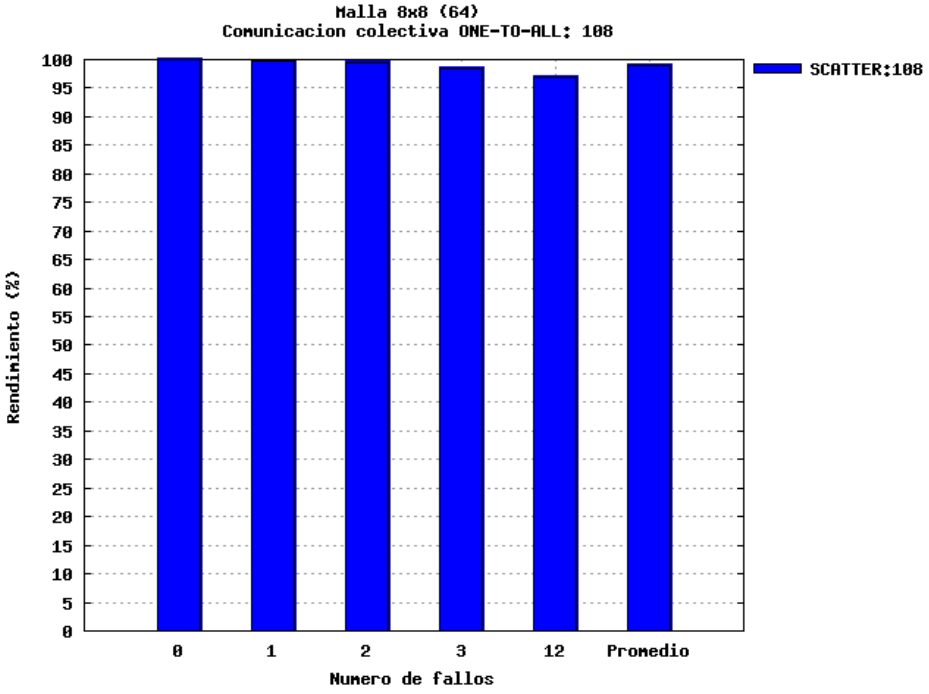


(b) Nodo origen (0,7)

Figura 6.17: Resultados de las pruebas de comunicaciones one-to-all (por casos)

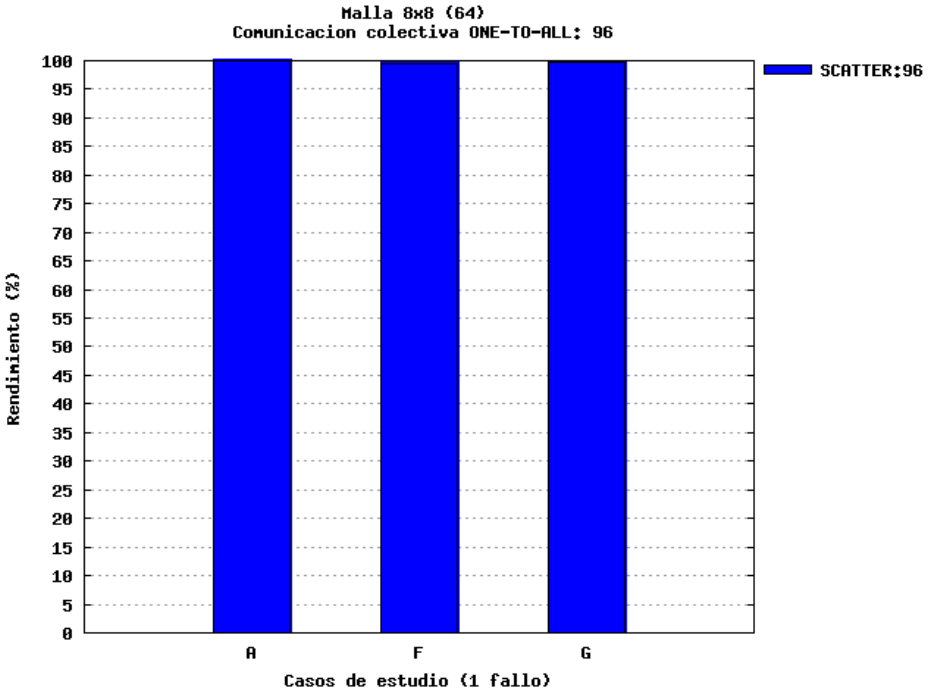


(a) Nodo origen (3,4)

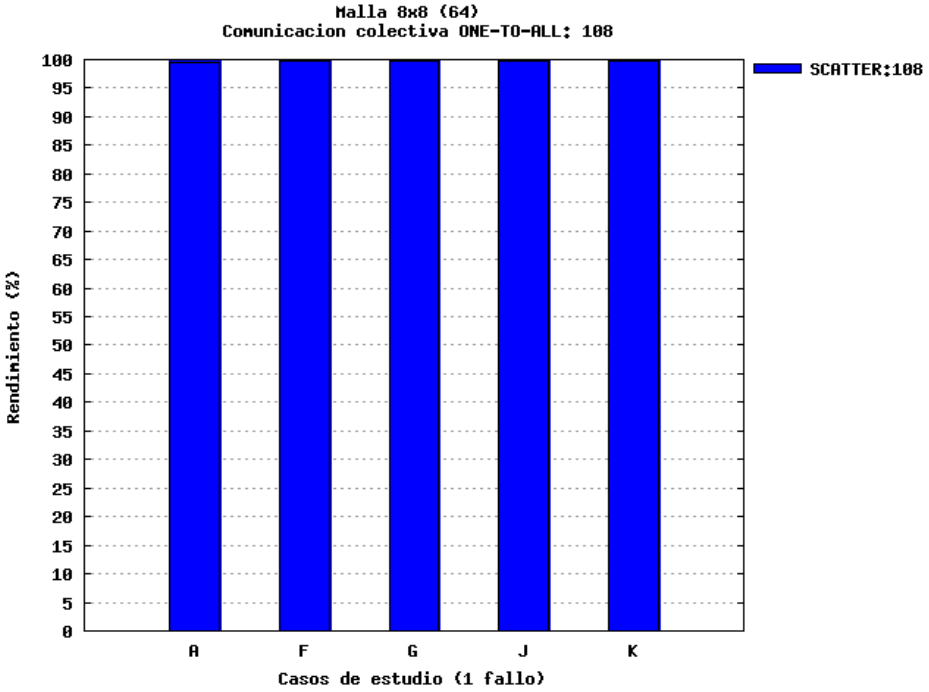


(b) Nodo origen (0,7)

Figura 6.18: Resultados de las pruebas de comunicaciones one-to-all (por número de fallos)

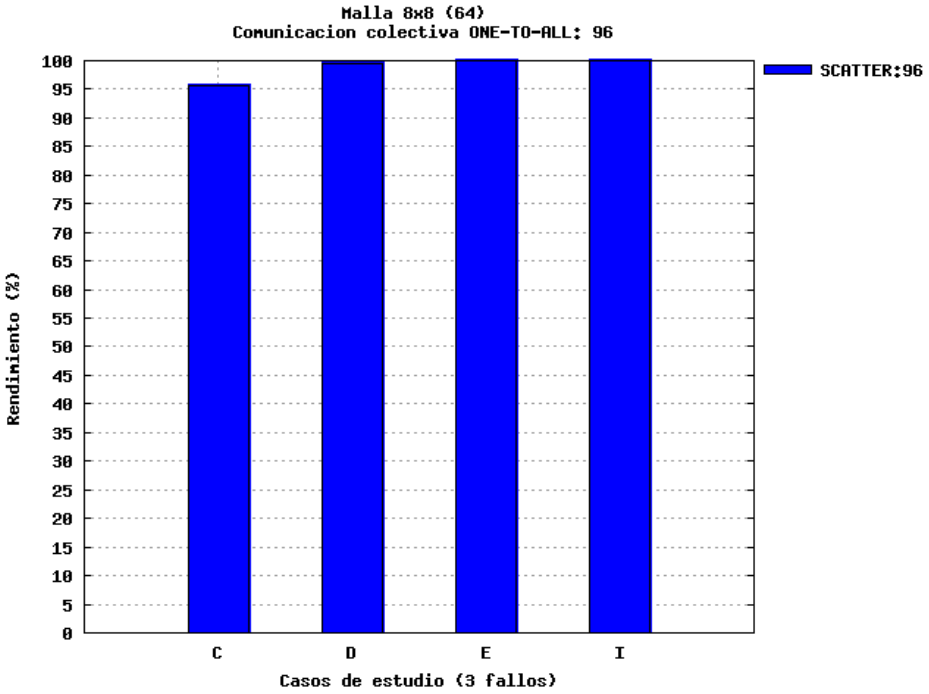


(a) Nodo origen (3,4)

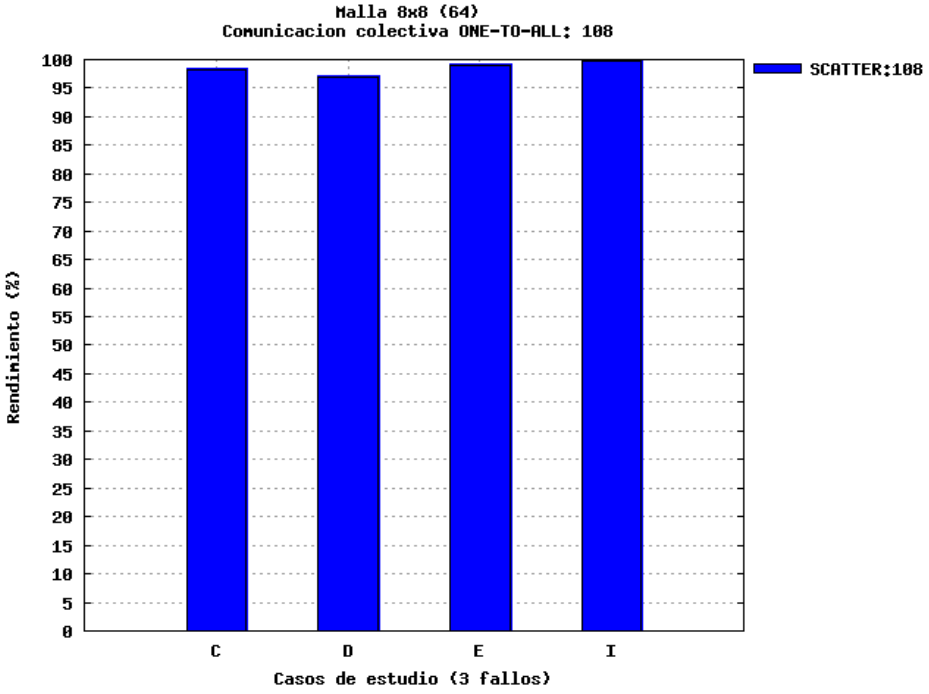


(b) Nodo origen (0,7)

Figura 6.19: Resultados de las pruebas de comunicaciones one-to-all (para 1 fallo)



(a) Nodo origen (3,4)



(b) Nodo origen (0,7)

Figura 6.20: Resultados de las pruebas de comunicaciones one-to-all (para 3 fallos)

6.3.4. Comunicaciones colectivas all-to-one

Al igual que en el caso anterior, éste tipo de comunicaciones colectivas se utiliza en frameworks de tipo *master-worker*, pero en éste cuando los procesos *workers* envían información al *master*.

Los resultados de los experimentos para este tipo de comunicaciones se muestran en la tabla 6.11. En esta se presentan por separado las dos series de experimentos realizados, una tomando como destino el nodo *(3,4)* y la otra tomando como destino el nodo *(0,7)*.

Resultados de las pruebas de comunicaciones all-to-one			
Nodo destino (3,4)			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base	0	100 %	0 %
A	1	99,889 %	0,11 %
B	2	99,999 %	0 %
C	3	99,926 %	0,07 %
D	3	99,985 %	0,01 %
E	3	99,948 %	0,05 %
F	1	99,936 %	0,06 %
G	1	99,912 %	0,09 %
H	12	99,853 %	0,15 %
I	3	99,317 %	0,68 %
N	3	99,937 %	0,06 %
Rendimiento promedio: 99,87 %			
Nodo destino (0,7)			
Escenario	*Num. de fallos*	*Rendimiento (%)*	*Degradación (%)*
Base	0	100 %	0 %
A	1	99,957 %	0,04 %
B	2	99,968 %	0,03 %
C	3	99,962 %	0,04 %
D	3	99,958 %	0,04 %
E	3	99,968 %	0,03 %
F	1	100 %	0 %
G	1	99,956 %	0,04 %
H	12	99,996 %	0 %
I	3	100 %	0 %
J	3	100 %	0 %
Rendimiento promedio: 99,98 %			

Tabla 6.11: Resultados de las pruebas de comunicaciones colectivas all-to-one

Para las dos series de experimentos los valores de rendimiento promedio son superiores al 99 %, lo que implica que para este tipo de comunicaciones nuestro método permite alcanzar resultados muy satistactorios.

En las Figs. 6.21(a) y 6.21(b) se muestran gráficamente los valores de rendimiento obtenidos de la experimentación para cada una de estas series separados por escenarios, mientras que en

las Figs. 6.22(a) y 6.22(b) se muestran los mismos valores pero agrupados y promediados por número de fallos.

Los valores de rendimiento agrupados por número de fallos se muestran en las Figs. 6.23(a), 6.23(b) para los casos de 1 fallo, y 6.25(a) y 6.25(b) para los casos de 3 fallos y 6.24(a) para los casos de 2 fallos del nodo destino *(3,4)*.

El punto a destacar de toda esta serie de experimentos, además de la relevancia de la ubicación de los fallos y los nodos origen y destino, es el nivel de rendimiento obtenido en el caso de que se utilice como nodo destino el *(0,7)*. Estos niveles de rendimiento, que inclusive son del 100 % para los escenarios *F*, *I* y *J*, se basan en la ubicación del nodo destino *(0,7)*, que al estar en una de las esquinas de la malla permite tener un gran número de caminos de longitud mínima y, sobre todo, el hecho de posibilitar la evasión de los fallos a través de vías de escape sin incrementar las longitudes de los caminos en la mayoría de los casos. Al igual que para el caso de la comunicación de tipo *one-to-all* con origen en este mismo nodo, el comportamiento en esta serie de experimentos es muy similar a lo explicado para el escenario *R* de la comunicación punto a punto unidireccional.

6.3.5. Resumen de la experimentación

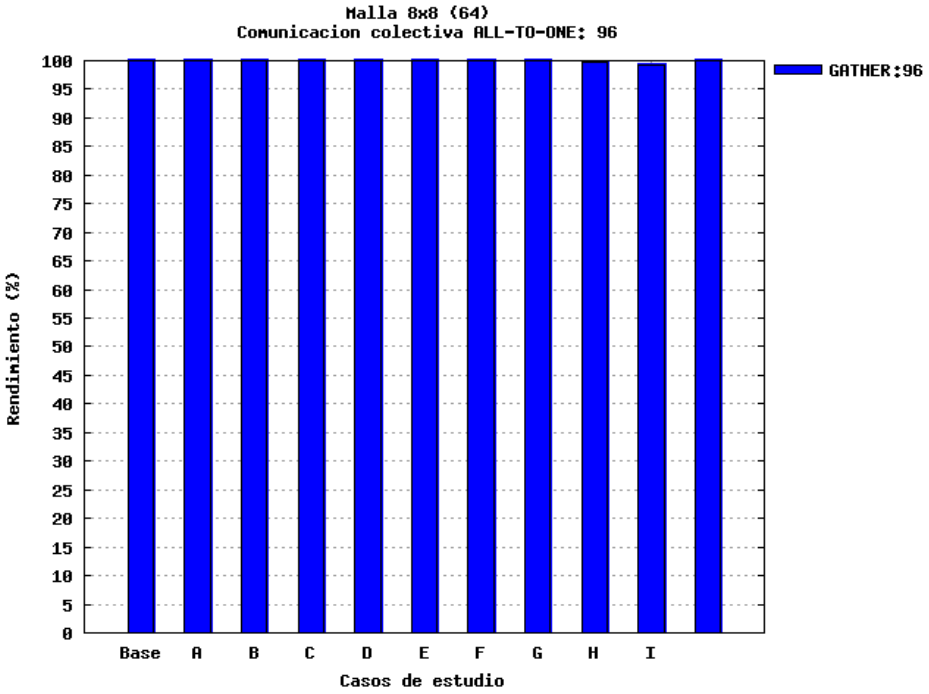
En la Fig. 6.26 se muestran todos los resultados de los 5 casos de comunicaciones colectivas que hemos utilizado en la experimentación agrupados por escenarios. Los identificadores que se utilizan en esta gráfica son (por color y nombre):

- All-to-all: rojo.
- One-to-all con nodo origen (3,4): verde.
- One-to-all con nodo origen (0,7): celeste.
- All-to-one con nodo destino (3,4): violeta.
- All-to-one con nodo destino (0,7): azul.

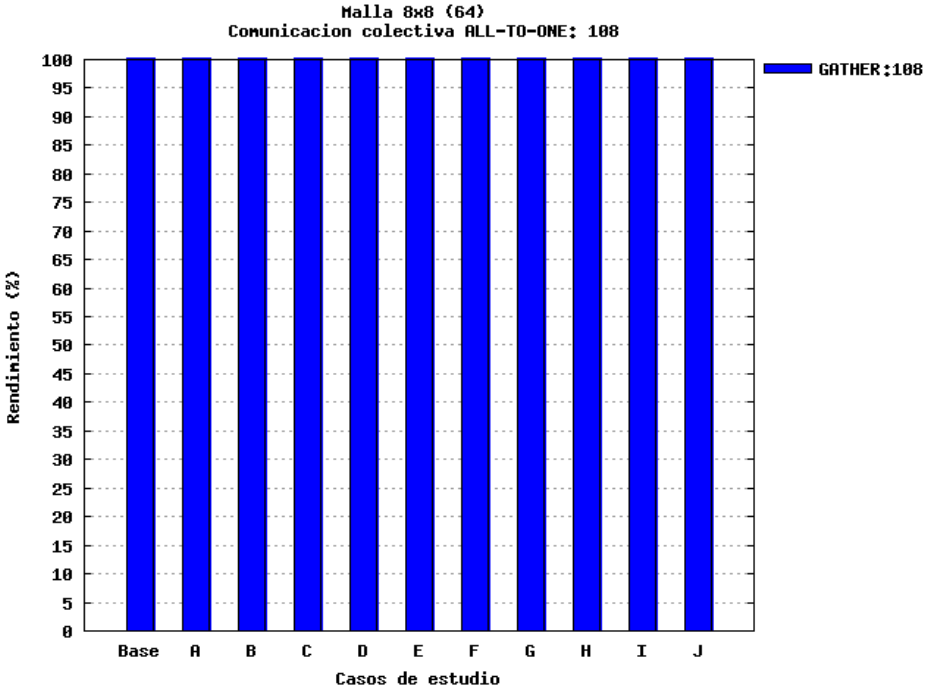
En esta gráfica se observan los niveles de rendimiento que alcanza nuestra propuesta, situados por encima del 95 % en la mayoría de los casos y nunca por debajo del 80 %.

La misma información se puede ver en la Fig. 6.27 pero agrupada y promediada por número de fallos. En este caso, todos los promedios de rendimiento se sitúan por encima del 90 %, por lo que podemos concluir, en base a los escenarios que hemos estudiado, que nuestra propuesta permite obtener degradaciones menores al 10 % con combinaciones de hasta 12 fallos.

En la Fig. 6.28 se muestran los valores promedio agrupados por número de fallos y es posible observar que el promedio global de rendimiento para los experimentos realizados es superior al 98 %.

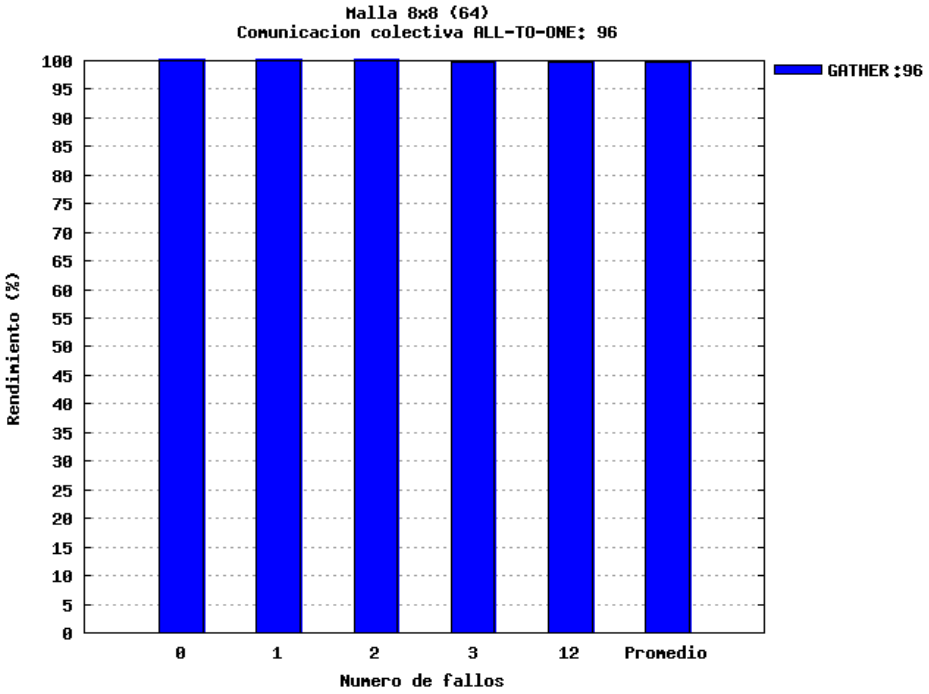


(a) Nodo destino (3,4)

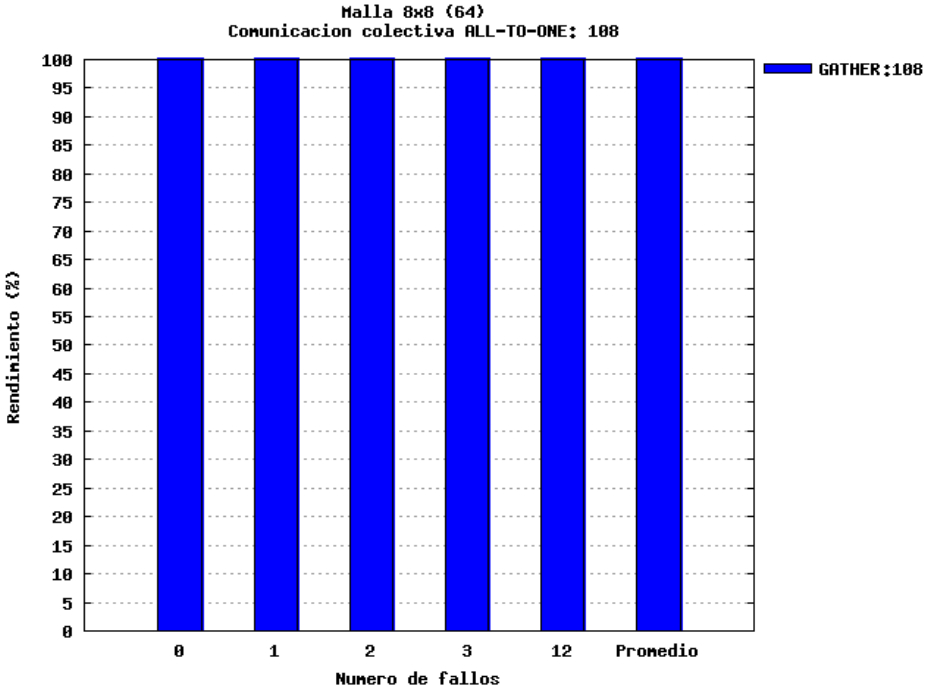


(b) Nodo destino (0,7)

Figura 6.21: Resultados de las pruebas de comunicaciones all-to-one (por casos)

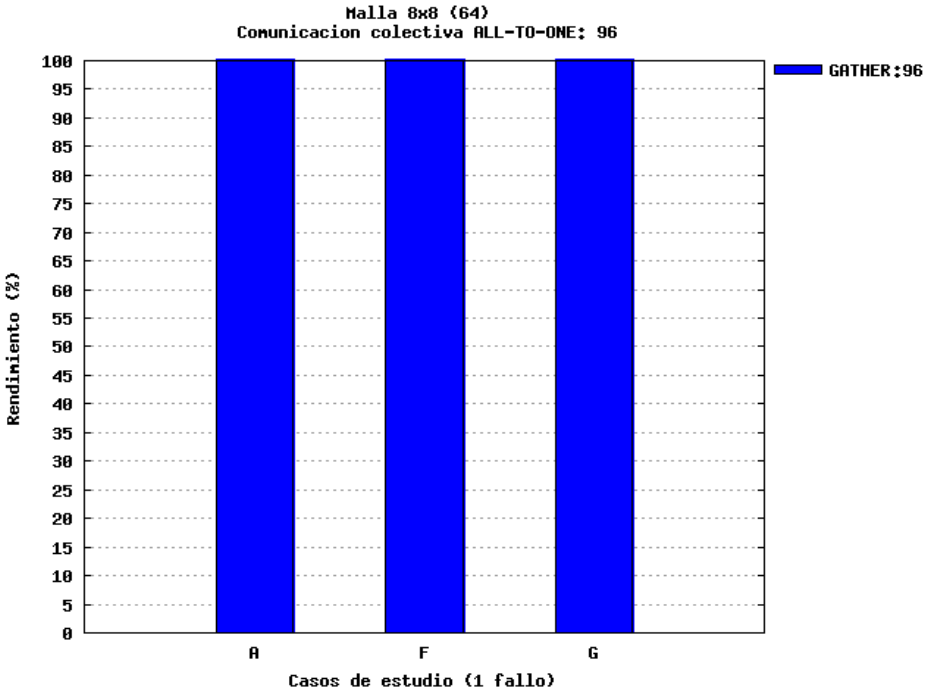


(a) Nodo destino (3,4)

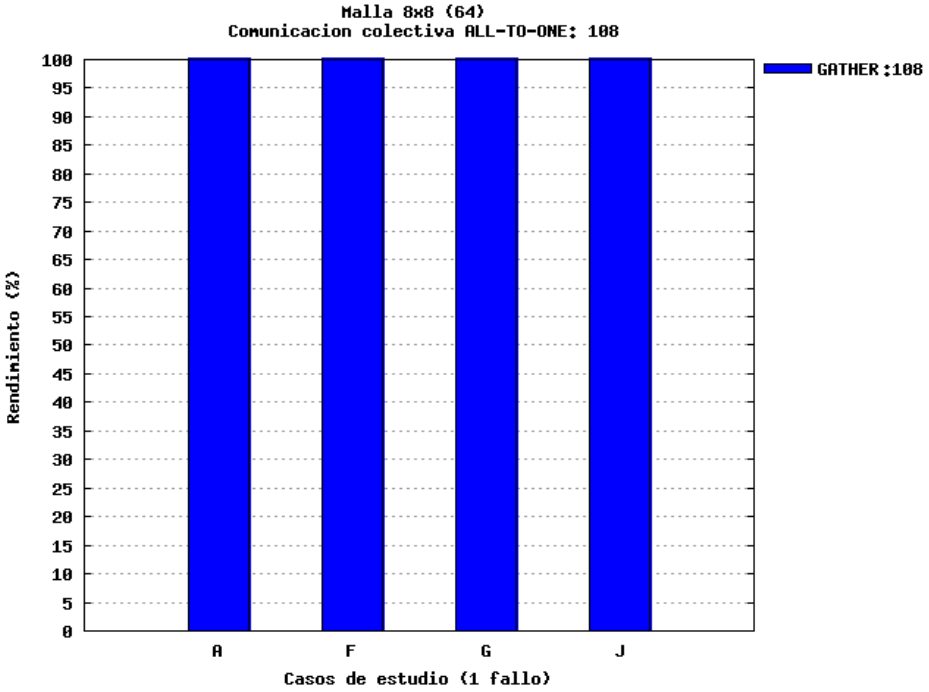


(b) Nodo destino (0,7)

Figura 6.22: Resultados de las pruebas de comunicaciones all-to-one (por número de fallos)

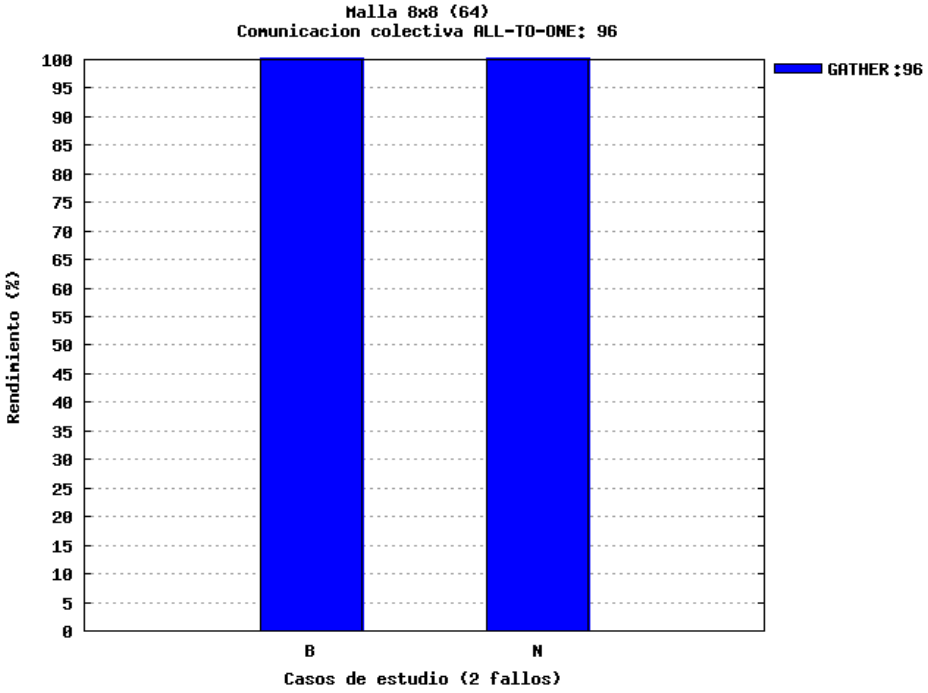


(a) Nodo destino (3,4)



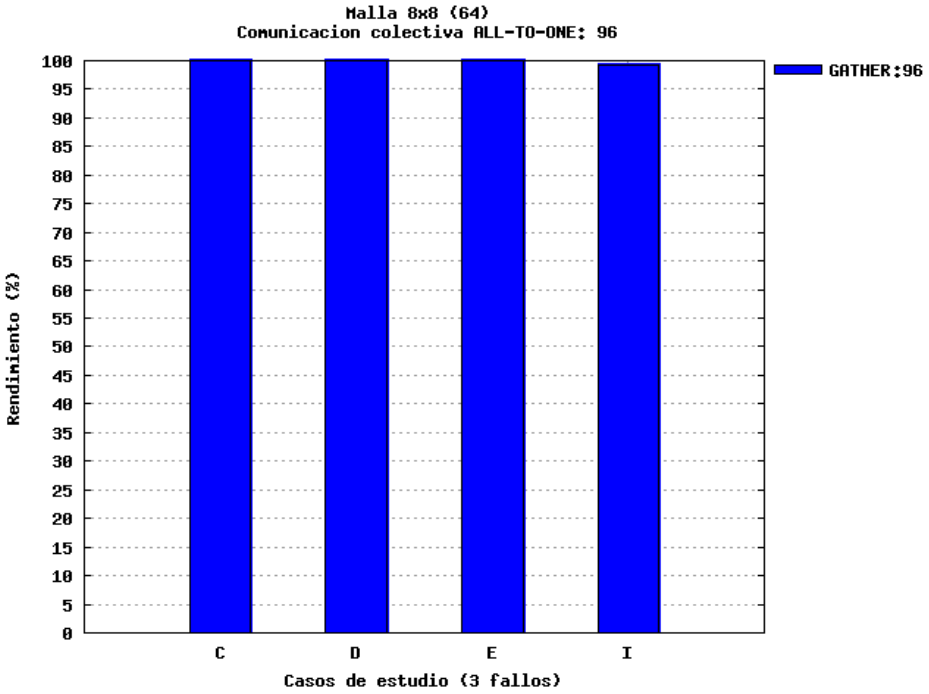
(b) Nodo destino (0,7)

Figura 6.23: Resultados de las pruebas de comunicaciones all-to-one(para 1 fallo)

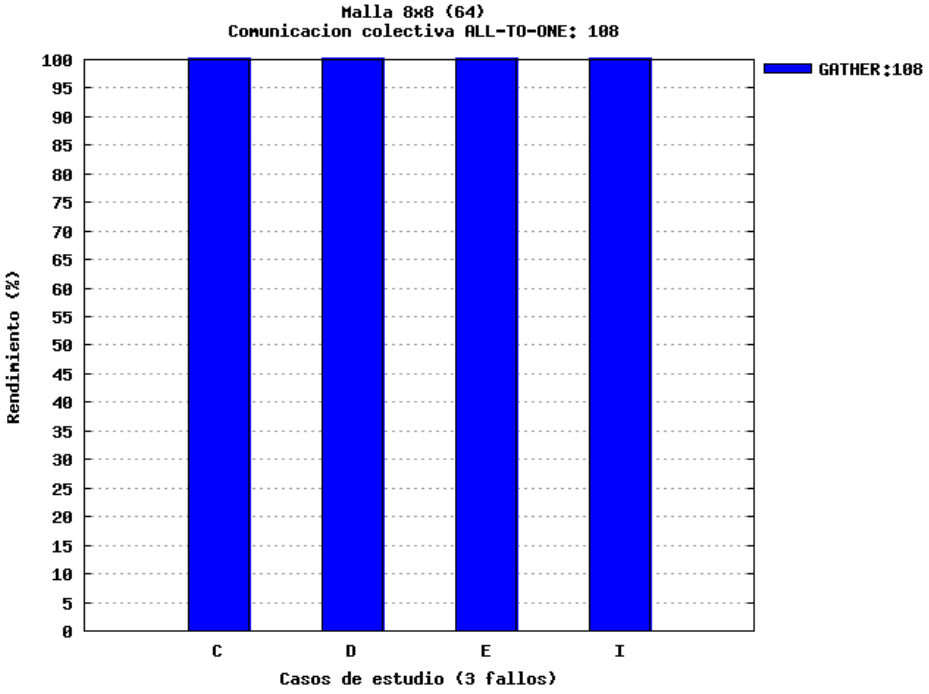


(a) Nodo destino (3,4)

Figura 6.24: Resultados de las pruebas de comunicaciones all-to-one (para 2 fallos)



(a) Nodo destino (3,4)



(b) Nodo destino (0,7)

Figura 6.25: Resultados de las pruebas de comunicaciones all-to-one (para 3 fallos)

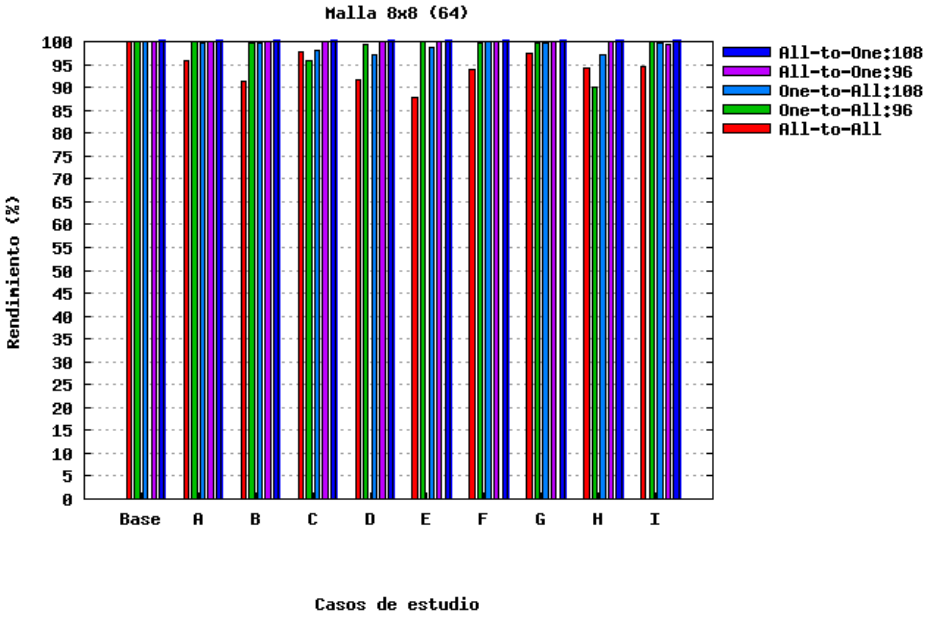


Figura 6.26: Resultados de las pruebas de todas las comunicaciones (por casos)

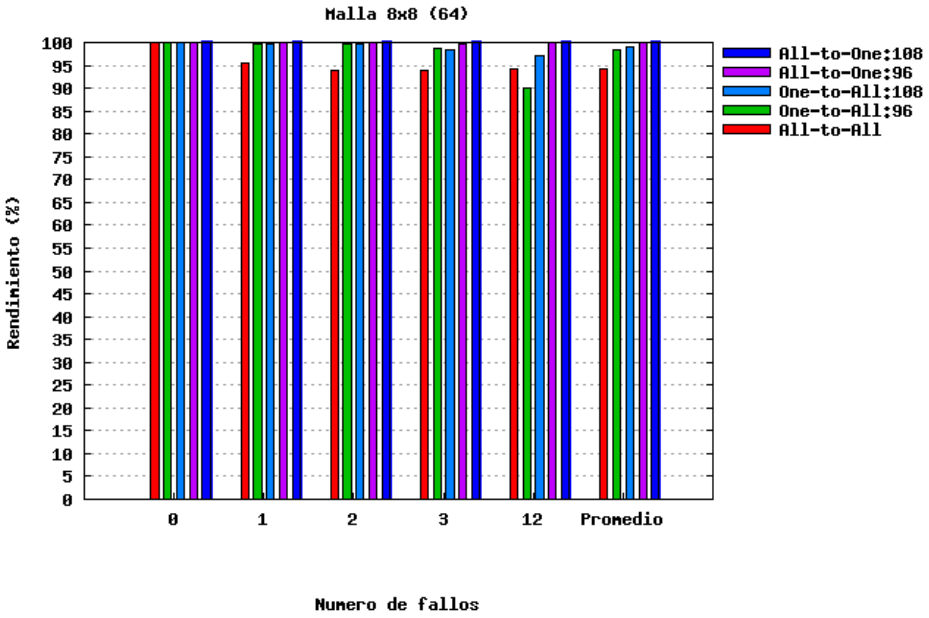


Figura 6.27: Resultados de las pruebas de todas las comunicaciones (por número de fallos)

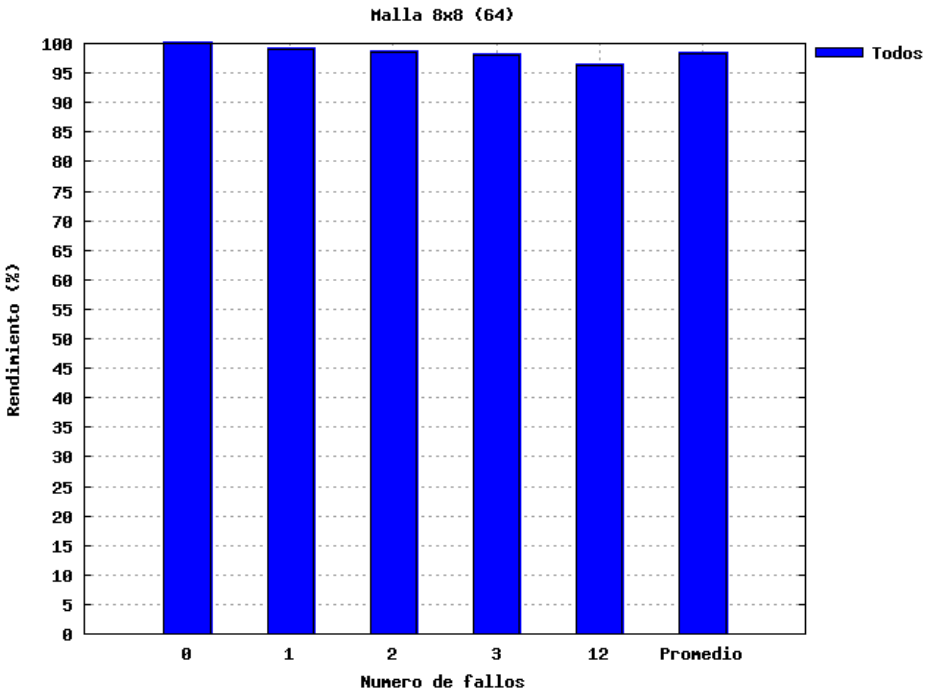


Figura 6.28: Resultados de las pruebas de todas las comunicaciones (promedio por fallos)

Capítulo 7

Conclusiones

7.1. Conclusiones finales

Con este capítulo finalizamos la presentación del trabajo de investigación que hemos realizado en el ámbito de la tolerancia a fallos para redes de interconexión y en el contexto de los sistemas de cómputo de altas prestaciones.

Desde el primer capítulo hemos ido introduciendo los objetivos y la motivación que condujeron al desarrollo de este trabajo, partiendo de la explicación del contexto en que ha sido desarrollado y los desafíos que éste implica. Se empezó por la introducción teórica de los dos marcos de referencia que delimitan el trabajo: las redes de interconexión y la tolerancia a fallos. A partir de esta introducción fue posible realizar el análisis de las cuestiones que tienen influencia, directa o indirecta, en su desarrollo, poniendo especial énfasis en las cuatro fases de la tolerancia a fallos (detección del error, contención del daño, recuperación del error, y tratamiento y continuidad del servicio) debido a que estas constituyen el núcleo central de nuestra propuesta. De éste análisis se establecieron los requerimientos funcionales propios del problema de las redes de interconexión tolerantes a fallos y se determinaron los condicionantes que influyen en el diseño y desarrollo de las posibles soluciones a esta problemática.

Este enfoque de presentación de contenidos nos ha permitido plantear la etapa de diseño de manera totalmente independiente de la etapa de implementación, logrando así un diseño flexible, libre de las restricciones que impone la etapa de implementación y las tecnologías asociadas a ella. Sin embargo, el precio a pagar por esta flexibilidad es el distanciamiento de las tecnologías que, en último caso, son necesarias para implementar cualquier tipo de diseño, motivo por el cual hemos incluido un capítulo completo dedicado al modelado y la implementación de la propuesta, utilizando herramientas de modelado y simulación.

Las pruebas y la validación de la propuesta han sido realizadas en base a simulaciones de eventos discretos, llevadas a cabo a partir de modelos y experimentos implementados en OPNET.

Habiendo finalizado todas las fases que componen el proyecto de investigación, desde el planteo y discusión del problema hasta las pruebas y validación de las soluciones, nos encontramos en condiciones de describir los resultados obtenidos.

Los objetivos originalmente planteados al momento de definir el problema han sido cumplidos a lo largo del desarrollo de nuestro trabajo. Para empezar, se han logrado extender las funcionalidades de DRB, convirtiéndolo en un algoritmo de encaminamiento tolerante a fallos sin que haya sido necesario sacrificar sus características de balanceo del encaminamiento. Para lograrlo fue necesario hacer un estudio profundo de varios aspectos propios de la tolerancia a fallos y buscar la mejor manera de adaptar las metodologías de la tolerancia a fallos a los requerimientos de las redes de interconexión, obteniendo como resultado las políticas de encaminamiento tolerantes a fallos propuestas en este trabajo.

A continuación presentamos las conclusiones a las que hemos llegado a partir del análisis de los resultados obtenidos en la etapa de experimentación presentada en el capítulo anterior.

7.2. Conclusiones sobre los resultados

Los resultados obtenidos a partir de los modelos y la simulación muestran la validez de nuestra propuesta como políticas de encaminamiento tolerantes a fallos. Esto nos permite concluir que hemos sido capaces de cumplir con éxito los objetivos inicialmente planteados. Se ha resuelto cómo encaminar los mensajes en presencia de fallos, es decir, el algoritmo es capaz de encontrar caminos alternativos a los que contienen el o los fallos y dirigir el mensaje hasta su destino. Esto se hace mediante el recálculo de la ruta en el encaminador en el que se detecta el fallo, lo que corresponde a las dos primeras fases de la tolerancia a fallos: de detección del error y contención del daño. También se han dado soluciones a las dos fases restantes: la de recuperación del error y la de tratamiento del fallo y continuidad del servicio. La solución de la fase de recuperación del error se basa en la notificación de la existencia de fallos a los nodos origen para que éstos no utilicen más ese camino, mientras dure el fallo, y abran caminos alternativos. En la última de las cuatro fases de la tolerancia a fallos, la fase de tratamiento del fallo y continuidad del servicio, se ha implementado un sistema de monitorización reactiva del enlace en presencia de un fallo y actualización del estado de los caminos a los orígenes en caso de que el fallo desaparezca.

En base a los diferentes experimentos realizados hemos podido comprobar que la degradación de las prestaciones de una red de interconexión depende de diversos factores, además del número de fallos, como la disposición espacial y temporal de fallos, el tipo de comunicación (colectiva, punto a punto, etc.) y la ubicación de los nodos inyectores y receptores de mensajes. Hemos observado además el efecto producido por el fenómeno que hemos denominado "ocultamiento de fallos", a partir del cuál es posible que ciertos escenarios que contengan combinaciones de múltiples fallos obtengan rendimientos similares, o incluso mejores, que escenarios con menor cantidad de fallos (incluso escenarios de un solo fallo).

Los resultados numéricos globales muestran que las políticas de encaminamiento que hemos diseñado son capaces de ofrecer niveles de rendimientos muy alentadores, ya que han sido capaces de tolerar escenarios con más de diez fallos en redes directas, obteniendo niveles de degradación de prestaciones inferiores al 5 % (en promedio) respecto de los mismos escenarios en ausencia de

fallos.

Ha sido posible además cuantificar la influencia de la política de selección de la vía de escape en los encaminadores intermedios ante la ocurrencia de fallos. La elección de una vía de escape favorable, es decir, que no incremente la longitud del camino entre nodos, puede implicar mejoras en el rendimiento de la red de hasta un 3 % respecto a una vía que sí incremente el camino, según hemos podido observar en los resultados de nuestra experimentación.

Debido a que este trabajo representa solamente la primer etapa de desarrollo del sistema, no resulta ilógico pensar que es posible mejorar los índices de degradación hasta aquí obtenidos y que se cuenta con un horizonte de desarrollos y líneas abiertas muy prometedores.

7.3. Trabajo futuro y líneas abiertas

En cuanto a las líneas abiertas, el principal punto de desarrollo se encuentra en la modificación y extensión de los modelos de fallos. Estos son utilizados como punto de partida por la gran mayoría de los trabajos actuales en el área de la tolerancia a fallos para redes de interconexión, incluido el nuestro.

A partir de la ampliación de estos modelos resulta posible abordar la resolución de los mayores desafíos que hoy por hoy existen en el área, según nuestro punto de vista: la resolución de los problemas de perdida de información por la ocurrencia de fallos. Estos problemas, como ya se ha expuesto a lo largo del trabajo, tienen su principal dificultad en la solución de la pérdida de la información almacenada en los buffers de los encaminadores en caso de que algún fallo los afecte.

Éste no es un problema menor debido principalmente a que la actual tendencia a incrementar la velocidad de comunicación en las redes de interconexión va en dirección contraria a las posibles soluciones a estos problemas de pérdidas de información, usualmente basados en replicación de datos e incrementos de la seguridad en los protocolos de comunicación y encaminamiento, lo que tiene incidencias directas y negativas en términos de velocidad.

Actualmente estamos trabajando en los detalles de implementación necesarios para realizar simulaciones sobre redes de tipo *toro*.

Las líneas abiertas pueden ser agrupadas en dos categorias: *trabajos en base a los desarrollos obtenidos en este trabajo* y *ampliación de los desarrollos de este trabajo.*

Líneas abiertas en base al trabajo:

- Realizar experimentos sobre las topologías de tipo *k-ary n-cube* y *fat-tree*, y sobre mallas tridimensionales.
- Realizar experimentos de comportamiento de las políticas de encaminamiento en situaciones de congestión de la red.
- Estudiar de forma adecuada la verdadera influencia de la disposición tanto espacial como temporal de los fallos en las políticas de encaminamiento tolerantes a fallos.

- Analizar las ventajas y desventajas de incluir más de un nodo de procesamiento por cada encaminador de la red.
  - Estudiar las posibles ventajas de enviar las notificaciones de caminos fallidos a todos los nodos de procesamiento conectados al encaminador que lo recibe.
  - Estudiar el efecto de los envíos simultaneos de mensajes por parte de más de un nodo de procesamiento y el nivel de variación que esto genera en los patrones de comunicación sintéticos (*butterfly, complement, etc.*).
  - Estudiar los efectos de la comunicación local, es decir, entre los nodos de procesamiento conectados al mismo encaminador y su influencia en el encaminamiento de la comunicación externa.
- Realizar un estudio más profundo sobre las ventajas y desventajas de los dos enfoques de implementación de las tablas de caminos fallidos que hemos propuesto (global y de enlace).
- Estudiar la política de eliminación de datos de la tabla de caminos fallidos, analizando las diferentes posibilidades para seleccionar que nodo origen eliminar de la tabla, y estudiar la influencia de la notificación de baja de la tabla al nodo origen.
- Utilizar la experiencia adquirida sobre la elección de vías de escape para mejorar el rendimiento de las políticas de encaminamiento tolerantes a fallos que hemos propuesto.

Líneas abiertas de ampliación del trabajo:

- Extender los modelos de fallos de la literatura de modo de posibilitar el tratamiento de la pérdida de información a causa de fallos.
- Tratar los problemas de pérdida de información a causa de fallos en la red de interconexión (ya sea en tránsito o almacenada en los encaminadores).
- Dar solución al problema de atascamiento de mensajes que ya estaban encolados en los buffers de salida cuando ocurre el fallo.
- Modelar matemáticamente el comportamiento de los fallos no permanentes, a fin de estudiar de mejor manera su comportamiento ya que, a nuestro entender, serán los fallos más usuales en los próximos años.

Apéndices

Apéndice A

Definiciones matemáticas

A.1. Definiciones del encaminamiento de DRB-E

A.1.1. Definiciones previas

Nodos adyacentes

Se dice que dos nodos N_i y N_j son *nodos adyacentes* si están directamente conectados por un enlace.

Distancia

Se define $Distancia(N_i, N_j)$ como el mínimo número de enlaces que deben ser atravesados para viajar desde N_i hasta N_j.

Camino

Un $CaminoP(N_i, N_j)$ entre dos nodos N_i y N_j es el conjunto formado por los nodos atravesados seleccionados al ir de N_i a N_j de acuerdo al encaminamiento mínimo estático definido para la red de interconexión. N_i es el nodo fuente y N_j es el nodo destino del camino.

Longitud

La *longitud* de un camino P, $Long(P)$ es el número de enlaces entre N_i y N_j siguiendo el camino P. En el caso de encaminamiento estático mínimo:

$$Long(P(N_i, N_j)) = Distancia(N_i, N_j) \tag{A.1}$$

A.1.2. Supernodo

Un Supernodo $S(tipo, tamaño, N_0^S) = \bigcup_{i=0}^{l} N_i^S$, se define como una región de la red de interconexión formada por el conjunto de l nodos adyacentes N_i^S alrededor de un nodo "central" N_0^S, que cumplen las siguientes condiciones:

- N_i^S, $\forall i$ cumple una propiedad dada en el "tipo" especificado
- $Distancia(N_i^S, N_0^S) \leq$ tamaño

A.1.3. Camino multipaso

Un camino multipaso "MultiStepPath" $MSP(SOrigen, N_i^{SOrigen}, N_j^{SDest}, SDest)$ es el camino generado entre dos Supernodos, $SOrigen$ y $SDest$, de la siguiente manera (donde el símbolo $*$ significa concatenación y $P1$, $P2$ y $P3$ son caminos simples):

$$\begin{aligned} MSP &= \prod(N_0^{SOrigen}, N_i^{SOrigen}, N_j^{SDest}, N_0^{SDest}) \\ &= P1(N_0^{SOrigen}, N_i^{SOrigen}) * P2(N_i^{SOrigen}, N_j^{SDest}) * P3(N_j^{SDest}, N_0^{SDest}) \end{aligned} \quad \text{(A.2)}$$

Este camino multipaso está compuesto por los siguientes pasos:

- Paso 1: Desde el nodo central $N_0^{SOrigen}$ del Supernodo $SOrigen$ al nodo $N_i^{SOrigen}$ perteneciente al Supernodo $SOrigen$.
- Paso 2: Desde el nodo $N_i^{SOrigen}$ al nodo N_j^{SDest} perteneciente al Supernodo $SDest$.
- Paso 3: Desde el nodo N_j^{SDest} al nodo central central N_0^{SDest} del Supernodo $SDest$.

A.1.4. Metacamino

Un metacamino $P*(Supernodo\_Origen, Supernodo\_Destino)$ es el conjunto de todos los caminos multipaso que se pueden generar entre los Supernodos $Supernodo\_Origen$ y $Supernodo\_Destino$ mediante el producto cartesiano entre los conjuntos formados por los nodos que componen los supernodos:

$$P* = \bigcup_{\forall i,j} MSP(N_0^{SOrigen}, N_i^{SOrigen}, N_j^{SDest}, N_0^{SDest}) \quad \text{(A.3)}$$

A.1.5. Longitud del camino multipaso

Se define la longitud de un camino multipaso $long(MSP)$ como la suma de las longitudes determinadas siguiendo encaminamiento estático de cada uno de los pasos individuales que lo componen.

$$\begin{aligned} Long(MSP) = \quad & Long(P1(N_0^{SOrigen}, N_i^{SOrigen})) + \\ & Long(P2(N_i^{SOrigen}, N_j^{SDest})) + \\ & Long(P3(N_j^{SDest}, N_0^{SDest})) \end{aligned} \tag{A.4}$$

A.1.6. Latencia del camino multipaso

La latencia del camino multipaso $Latencia(MSP)$ se define como la suma del tiempo de transmisión, que es la $long(MSP) * Te$ (Te es el tiempo de transmitir por un enlace), más el tiempo de espera gastado por el mensaje para viajar desde $N_0^{SOrigen}$ hasta N_0^{SDest} en las colas de los encaminadores debido a la contención entre mensajes. Es, por lo tanto, el tiempo que se tarda en enviar un mensaje por el camino multipaso.

$$\begin{aligned} Latencia(MSP) = \quad & Tiempo\ de\ transmision + \\ & \textstyle\sum_{\forall nodos \in MSP} RetardoDeEncolamiento(Nodo) \end{aligned} \tag{A.5}$$

A.1.7. Anchura del metacamino

Definimos anchura del metacamino s como el número de caminos multipaso que componen un metacamino. Estos caminos pueden compartir una cierta parte.

$$AnchuraDelMetacamino = s = l * k \tag{A.6}$$

donde l es el número de nodos del supernodo origen y k el número de nodos del supernodo destino.

A.1.8. Latencia del metacamino

Definimos latencia del metacamino como la inversa de la suma de las inversas de las latencias de cada uno de los caminos multipaso que componen el metacamino. Estas latencias son las sufridas por los mensajes que viajan por los caminos multipaso. Estas latencias inversas son, de hecho, anchos de banda, y su suma es el ancho de banda del metacamino.

$$Latencia(P*) = (\sum_{\forall s} Latencia(MSPs)^{-1})^{-1} \tag{A.7}$$

Bibliografía

[1] W. J. Dally and B. Towles, *Principles and practices of interconnection networks.* Amsterdam; San Francisco: Morgan Kaufmann Publishers, 2004, iD: 52902442.

[2] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 8, pp. 790–802, 1997. [Online]. Available: http://dx.doi.org/10.1109/71.605766

[3] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks. An Engineering Approach.* San Francisco, CA: Morgan Kaufmann, 2003, iD: 52616057.

[4] H. El-Rewini and M. Abd-El-Barr, *Advanced computer architecture and parallel processing.* Hoboken, N.J.: Wiley, 2005, iD: 55797866.

[5] M. J. Flynn, "Very high-speed computing systems," *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1901–1909, 1966.

[6] D. Franco, "Balanceo distribuido del encaminamiento de redes de interconexión de computadores paralelos," Ph.D. dissertation, Universitat Autònoma de Barcelona, 2001.

[7] D. Franco, I. Garcés, and E. Luque, "Distributed routing balancing for interconnection network communication," in *HIPC '98. 5th International Conference On High Performance Computing*, 1998, pp. 253–261.

[8] ——, "Avoiding communication hot-spots in interconnection networks," in *HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 1999.

[9] ——, "A new method to make communication latency uniform: distributed routing balancing," in *ICS '99: Proceedings of the 13th international conference on Supercomputing.* New York, NY, USA: ACM, 1999, pp. 210–219. [Online]. Available: http://doi.acm.org/10.1145/305138.305195

[10] ——, "Dynamic routing balancing in parallel computer interconnection networks," in *VECPAR '98: Selected Papers and Invited Talks from the Third International Conference on Vector and Parallel Processing.* London, UK: Springer-Verlag, 1999, pp. 494–507.

[11] P. J. Garcia, "Diseño de una técnica escalable y eficiente para el control de congestión en redes de interconexión," Ph.D. dissertation, Universidad de Castilla-La Mancha, 2006.

[12] I. Garcés and D. Franco, "Analysis of distributed routing balancing behavior," in *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2002, pp. 817–824. [Online]. Available: http://doi.acm.org/10.1145/508791.508951

[13] P. T. Gaughan, B. V. Dao, S. Yalamanchili, and D. E. Schimmel, "Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks," *IEEE Transactions on Computers*, vol. 45, no. 6, pp. 651–665, 1996.

[14] C. Gómez, M. Gómez, P. López, and J. Duato, "An efficient fault-tolerant routing methodology for fat-tree interconnection networks," in *Parallel and Distributed Processing and Applications*. Springer Berlin/Heidelberg, 2007, vol. 4742/2007, pp. 509–522.

[15] M. E. Gómez, "A routing methodology for achieving fault tolerance in direct networks," *IEEE Transactions on Computers*, vol. 55, no. 4, pp. 400–415, 2006.

[16] A. F. Hansen, "Resilient routing layers for recovery in packet networks," in *Proceedings of International Conference on Dependable Systems and Networks*, 2005, pp. 238–247.

[17] C. T. Ho, "A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers," *IEEE Transactions on Computers*, vol. 53, no. 4, pp. 427–438, 2004.

[18] InfiniBand Trade Association, *InfiniBand architecture specification: release 1.2.* Portland, OR: InfiniBand Trade Association, 2004, vol. 1.

[19] P. Jalote, *Fault tolerance in distributed systems.* Englewood Cliffs, N.J.: PTR Prentice Hall, 1994, iD: 29359323.

[20] I. Koren and C. M. Krishna, *Fault-tolerant systems.* Amsterdam; Boston: Elsevier/Morgan Kaufmann, 2007, iD: 73741844.

[21] A. Kvalbein, "Fast recovery from link failures using resilient routing layers," in *Proceedings of the 10th IEEE Symposium on Computers and Communications*, 2005, pp. 554–560.

[22] D. Lugones, "Control de congestión adaptativo en redes inifiniband," Master's thesis, Universitat Autònoma de Barcelona, 2007.

[23] D. Lugones, D. Franco, and E. Luque, "Dynamic routing balancing on infiniband networks," *Journal on Computer Science and Technology*, July 2008.

[24] ——, "Modeling adaptive routing protocols in high speed interconnection networks," in *OPNETWORK 2008 Conference*, 2008. [Online]. Available: http://www.opnet.com/opnetwork2008/

[25] A. Mejia, J. Flich, J. Duato, S. A. Reinemo, and T. Skeie, "Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori," in *20th International Parallel and Distributed Processing Symposium*, 2006.

[26] J. M. Montañana, "Reachability-based fault-tolerant routing," in *12th International Conference on Parallel and Distributed Systems*, vol. 1, 2006.

[27] N. A. Nordbotten, M. E. Gómez, J. Flich, P. López, A. Robles, T. Skeie, O. Lysne, and J. Duato, "A fully adaptive fault-tolerant routing methodology based on intermediate nodes," in *Network and Parallel Computing*. Springer Berlin/Heidelberg, 2004, pp. 341–356.

[28] OPNET Technologies, "Opnet modeler accelerating network R&D," June 2008. [Online]. Available: http://www.opnet.com/

[29] V. Puente and J. A. Gregorio, "Immucube: Scalable fault-tolerant routing for k-ary n-cube networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 776–788, 2007.

[30] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: a cheap and robust fault-tolerant packet routing mechanism," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, 2004, pp. 198–209.

[31] F. Safaei, "Evaluating the performance of adaptive fault-tolerant routing algorithms for wormhole-switched mesh interconnect networks," in *IEEE International Parallel and Distributed Processing Symposium*, 2007, pp. 1–8.

[32] F. O. Sem-Jacobsen, "Siamese-twin: A dynamically fault-tolerant fat-tree," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[33] I. Theiss and O. Lysne, "Froots – fault handling in up*/down* routed networks with multiple roots," in *HiPC 2003. 10th International Conference On High Performance Computing*, ser. Lecture Notes in Computer Science. Springer, 2003, pp. 106–117.

[34] TOP500 Supercomputing Site, "TOP500 List - June 2008," June 2008. [Online]. Available: http://www.top500.org/lists/2008/06

Índice alfabético

Adaptividad, 23
All-scatter, 83
All-to-All, 83
All-to-One, 83
Availability, 36
Avería, 31

Bandwidth, 36

Camino, 12
    nulo, 49
Camino multipaso, 57
Cargas de trabajo, 15
Clusters, 13
Computadores masivamente paralelos, 13
Computadores paralelos, 2
Conectividad, 23
Connectability, 37
Control de flujo, 12

Diámetro, 16
Diameter, 16
Diameter Stability, 36

Encaminadores de altas prestaciones para IP, 13
Encaminamiento, 12
    Adaptativo, 24
    Broadcast, 23
    Centralizado, 23
    Determinista, 24
    Distribuido, 23
    Estático, 24
    Multicast, 23
    Multifase, 23
    No centralizado, 23
    Nodo fuente, 23
    Por algoritmo, 24
    Por tabla, 24
    Unicast, 23
Enfoque de enlaces, 54
Enfoque global, 54
Entrada/Salida, 2
Error, 31
Escalabilidad, 14

Failure, 31
    Link, 34
    Node, 34
Fallo, 31
    Benigno, 33
    Bizantino, 33
    Enlace, 34
    Intermitente, 33
    Malicioso, 33
    Nodo, 34
    Permanente, 33
    Transitorio, 33
Fault, 31
    Benign, 33
    Byzantine, 33
    Fail-stop, 33
    Intermittent, 33
    Malicious, 33
    Permanent, 33
    Transient, 33
Fiabilidad, 15

Flit, 12

Gather, 83
Grado, 16

Hot-spot, 25
HPC, 1

Latencia, 12
Libertad de deadlock y livelock, 23

Mean Time Between Failures, 35
Mean Time to Failure, 35
Metacamino, 57
MIMD, 3
MISD, 3
MTBF, 35
MTTF, 35
Multicarril, 57
Multicomputadores, 13
Multiple Instruction, Multiple Data streams, 3
Multiple Instruction, Single Data stream, 3
Multiprocesadores, 13

Node and Link Connectivity, 36
Node degree, 16
Notificación de baja, 55
Notificación reactiva, 53

One-to-All, 83
OPNET, 67
    Nivel de nodo, 68
    Nivel de proceso, 68
    Nivel de red, 68

Particionabilidad, 14
Path, 12
Posibilidad de expansión, 14

Red de interconexión, 11
Redes de interconexión de alta velocidad, 4
Redundancia, 32
    De canal, 37
    De hardware, 32
    De información, 32
    De software, 32
    De tiempo, 33
Regularidad, 16
Regularity, 16
Reliability, 36
Reparabilidad, 15
Requerimientos de prestaciones, 14
Restricciones de costo, 15
Restricciones físicas, 15

Scatter, 83
SIMD, 3
Simetría, 16
Simplicidad, 14
Single Instruction, Multiple Data streams, 3
Single Instruction, Single Data stream, 3
SISD, 3
Store-and-forward, 21
Supernodos, 59
Symmetry, 16

Tabla de caminos fallidos, 55
Throughput, 12
Tolerancia a fallos, 23
    Contención del daño, 32
    Detección del error, 31
    Recuperación del error, 32
    Tratamiento del fallo y continuidad del servicio, 32
Topología, 12

Unidades de procesamiento, 2

Vías de escape, 52
Variación de las distancias, 14
Virtual cut-through, 22

Wormhole, 22