Facultad de Ciencias

Análisis y Desarrollo de Herramientas de Simulación para Redes de Sistema en Supercomputadores Exascale

(Analysis and Development of Simulation Tools for Exascale Supercomputers System Networks)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN COMPUTACIÓN

Autor: Mariano Benito Hoz

Directores: Enrique Vallejo Gutiérrez
Ramón Beivide Palacio

Octubre - 2014

Agradecimientos

Ahora que toca a su fin este Máster con la presentación de este Trabajo, me gustaría dedicar unas lineas a todas las personas, que de un modo u otro, han estado presentes en este año.

En primer lugar, me gustaría darles las gracias a los directores de mi proyecto, por su dedicación, por sus consejos, por su profesionalidad y por haberme sabido guiar hasta este punto. Además, me gustaría agradecerles también la puerta que me han abierto recientemente.

En segundo lugar, quisiera expresarle mi afecto a mi padre, porque aunque el camino vivido, tanto en la cercanía como en la lejanía nunca ha sido fácil, sé que siempre contaré con todo su apoyo y más ahora que vas a estar un poco más lejos.

En tercer lugar, quisiera agradecer a Sandra su apoyo en esto y en tantas otras cosas. Sigámonos tomándonos las cosas con la misma filosofía que hasta ahora y continuaremos andando este camino juntos... a base de “mas uno”.

Por ultimo, y más importante, mis gracias más sinceras a mi abuela, por haberme enseñado casi todo lo que sé y por convencerme inicialmente para que tomase este camino.

Muchas gracias a todos.

Resumen

Tanto los nuevos sistemas *High-Performance Computing* (HPC) que se desarrollarán en un futuro como el prototipo construido en el seno del proyecto europeo `Mont-Blanc` necesitarán redes de alto rendimiento. Estas redes deben ser capaces de dotar de conectividad a centenares de miles o incluso millones de nodos de cómputo para lograr el rendimiento que se obtendrá con los futuros supercomputadores *Exascale*.

Para presentar aportaciones al diseño de estas redes es necesaria una herramienta con la que poder evaluarlas, y que permita analizar su coste, rendimiento y consumo energético. La herramienta a utilizar debe admitir nuevos desarrollos de forma ágil pero a la vez tiene que ser fiable y, debido a los tamaños que se van a manejar en el futuro, tiene que ser escalable y permitir abordar experimentos con redes de gran tamaño.

En este trabajo se analiza el simulador `BookSim` y se comprueban sus capacidades analizando soluciones planteadas recientemente para redes de alto grado. Durante la realización de los experimentos se han obtenido datos que permiten analizar la viabilidad de este simulador como herramienta futura para el análisis de redes para los mencionados supercomputadores *Exascale*.

Del trabajo realizado se concluye principalmente que el simulador ha permitido la evaluación de los casos de uso propuestos y que a priori puede ser la herramienta elegida para futuros desarrollos, puesto que permite llegar a simular redes con un millón de nodos y la incorporación de nuevas redes a evaluar en un futuro. Asimismo, durante la realización de los mismos se ha concluido que el uso del *trunking* para balancear redes `Flattened Butterly` (FB) asimétricas y redes `Dragonfly` (DF) desbalanceadas es válido. Concretamente en este último caso, aunque existen dos tipos de *trunking* posibles, se concluye que desde un punto de vista tecnológico no tiene sentido plantearse uno de ellos. Aunque se da por válido el simulador para su posterior uso en futuros proyectos, se desea destacar que la ausencia de documentación sobre el mismo y la inactividad de su lista de usuarios no facilita el uso ni el desarrollo sobre el mismo.

Palabras clave: Redes de sistema, Dragonfly, Flattened Butterfly, Trunking, Simulación de red.

Abstract

The new *High-Performance Computing* (HPC) systems that will be developed in the future, as well as the prototype built in the bosom of the `Mont-Blanc` european project will need high performance networks. These networks must be able to connect hundreds of thousands or even millions of nodes to achieve the performance which will be reached with the future Exascale supercomputers.

In order to submit contributions to the design of these networks, a tool to evaluate them and that allows to analyse their cost, performance and power cost is needed. The tool to be used must allow new developments easily, but at the same time it must be reliable and, due to the sizes that are going to be handled in the future, it must be scalable and allow carrying out experiments with big-size networks.

In this paper, the `Booksim` simulator is analysed, checking its abilities analysing solutions which have been recently suggested for high-radix networks. During the realization of the experiments, data has been obtained, which allows to analyse the viability of this simulator as a future tool for the analysis of networks for the mentioned Exascale supercomputers.

As a main conclusion, it can be said that the simulator has allowed the evaluation of the use cases suggested and that it can a priori be the tool chosen for future developments, since it lets simulate networks with a million nodes and the incorporation of new networks to be evaluated in the future. Furthermore, during the realization of those use cases, it has been concluded that the use of trunking to balance asymmetric `Flattened Butterly` (FB) networks and unbalanced `Dragonfly` (DF) networks is valid. Specifically in this last case, although there are two type of possible trunking, it is concluded that from a technological point of view it does not make sense to think of one of them. Although the simulator is considered valid for its use in future projects, it is worth mentioning that the lack of documentation about it and the inactivity of its users'list does not facilitate to use it or develop on it.

Keywords: System networks, Dragonfly, Flattened Butterfly, Trunking, Network Simulation

Índice general

Página

Índice de ilustraciones

Página

Índice de tablas

Página

1 Introducción

Los sistemas HPC y el proyecto europeo Mont-Blanc son referencias punteras actualmente en la industria informática. Respectivamente, los primeros siempre han sido la punta de lanza de la computación y por otra parte, el objetivo perseguido por el segundo plantea un cambio disruptivo con las tecnologías empleadas en los primeros en los últimos años. La red de interconexión empleada en ambos debe ser estudiada con precisión, para ello se pueden emplear herramientas de simulación que aporten datos para el análisis de las propuestas a realizar.

1.1. Antecedentes y motivación

Las redes de interconexión juegan hoy en día un papel clave en el desarrollo de las Tecnologías de la información y las comunicaciones (TIC). Las redes *Metropolitan Area Network* (MAN) y *Wide Area Network* (WAN) típicamente han estado bajo la responsabilidad de los grandes operadores de telecomunicaciones mientras que las redes de sistema y las *networks on chip* (NOCs) han sido desarrolladas por diferentes comunidades de la industria informática. Las redes *Local Area Network* (LAN) se encuentran en un punto intermedio de ambos dominios, en el cual existe un fuerte grado de convergencia.

Los computadores más avanzados actualmente son los que forman parte de los sistemas de HPC. Los microprocesadores de un sistema HPC basan su diseño en el uso de múltiples *cores* en el mismo *chip* (*Chip Multiprocessor* (CMP)) interconectados mediante una NOC. A la conjunción de las placas de circuito impreso que suelen soportar varios CMPs implementando una arquitectura *Cache Coherent Non-Uniform Memory Access* (CC-NUMA) y otros componentes se le suele denominar servidor, en muchos casos, estos servidores (o nodos de cómputo) disponen de aceleradores *hardware* o *Graphics Processing Units* (GPUs). Los servidores, en un número variable que depende del tamaño del propio servidor, se agrupan en cabinas (o *racks*) que se interconectan entre sí; dependiendo del tamaño requerido del sistema se usarán más o menos cabinas.

La evolución del rendimiento de los supercomputadores utilizados en HPC ha evolucionado de forma acelerada, en el año 1985 se superó la barrera "*Gigascale*" (10^9 *FLoating-point Operations Per Second* (FLOPS)) con el `Cray 2` [CR85], en 1997 la "*Terascale*" (10^{12} FLOPS) con el `Intel ASCI Red System` [TOPa] y en 2008 fue la "*Petascale*" (10^{15} FLOPS) por el `IBM Roadrunner` [TOPb]. En base a esto, en el año 2019 se espera ver sistemas "*Exascale*" (10^{18} FLOPS) [Meu09]. Recientemente `Intel` ha anunciado su próximo paso hacia la computacion `exascale`, es decir, la comercialización para la segunda mitad de 2015 de sus `Knight's Landing` y los 3 trillones de operaciones en punto flotante de doble precision (3 teraflops) en un sólo *socket*.

Los retos que se plantea la computación *exascale* se pueden agrupar en cuatro categorias: energía, memoria y capacidad de almacenamiento, concurrencia y localidad y tolerancia a fallos [BBC+08]. En [BBC+08] se fija el objetivo de contar con la tecnología base para soportar el desarrollo y despliegue de sistemas *exascale* en 2015 y se fija un limite recomendado de que los sistemas más grandes no deberian superar los 500 *racks* ni los 20 MWatios.

En cuanto a las redes de interconexión, desde principios de los años 2000, los sistemas HPC han implementado redes con *full bisection bandwidth*. Sin embargo, a medida que los supercomputadores han crecido en número de nodos de cómputo, llegando a decenas o cientos de miles de nodos, las redes empleadas han pasado a ser muy costosas en términos de complejidad, consumo energético y coste económico [KOPS10]. Como posible solución se ha planteado el uso de topologías como la Dragonfly [KDSA08], ésta presenta buenas características de escalabilidad-coste para sistemas *exascale* pero puede presentar casos patológicos en los que el rendimiento se degrada sustancialmente, para los que hace falta emplear mecanismos de encaminamiento adaptativo relativamente complejos. En cuanto a la tecnología de red, se espera que en 2015 los *routers* lleguen a contar con entre 128 y 256 puertos con un ancho de bando por puerto de entre 60 y 120 Gb/s [BBC+08].

La eficiencia energética es actualmente una de las prioridades en el diseño de computadores y más en los futuros sistemas *exascale*. En este sentido, el proyecto `Mont-Blanc`[1] tiene como objetivo diseñar una nueva arquitectura capaz de definir los estándares futuros de los sistemas HPC, integrando soluciones energéticamente eficientes usadas hasta la fecha en sistemas embebidos y dispositivos móviles. Este proyecto europeo liderado por el `Barcelona Supercomputing Center` (BSC), cuenta con la colaboración de la `Universidad de Cantabria` (UC) [Mon14], la cual tiene como principal cometido en el mismo el diseño de la red de interconexión que comunicará los millones de dispositivos que compondrán el futuro prototipo de supercomputador.

La principal motivación del trabajo ha sido un primer acercamiento al trabajo que se va a realizar dentro de este marco de colaboración firmado entre la UC y el BSC para el proyecto `Mont-Blanc`. Para concluir con el objetivo que se le ha impuesto a la UC se ha de disponer de una herramienta que permita simular la interconexión del órden de un millon de nodos de cómputo mediante la red que se propondrá. Por otra parte, las topologías estudiadas en este trabajo se acercan a las empleadas en el `Cray XC30` [Wic13].

1.2. Objetivo

Como se ha comentado previamente, este trabajo surge en el seno de un acuerdo de colaboración entre la UC y el BSC para el diseño de la futura red a desplegar en el prototipo del supercomputador construido en el proyecto `Mont-Blanc`.

El objetivo principal del presente trabajo es evaluar el simulador de red `BookSim` y analizar si puede servir como herramienta para evaluar las redes a proponer para el citado proyecto. Las propuestas de red a realizar se evaluarán mediante simulaciones realizadas con dicha herramienta y

[1] Se puede obtener más información sobre el proyecto en su página web: `http://www.montblanc-project.eu`.

se obtendrá de la misma la información necesaria para analizar el coste, rendimiento y consumo de las propuestas. Para alcanzar rendimientos del orden de *Exascale* es necesario emplear sistemas muy grandes, y por tanto interesa que las herramientas de simulación sean capaces de abordar redes de gran tamaño, alcanzando hasta un millón de nodos de cómputo.

Como segundo objetivo del trabajo, se evaluarán diferentes redes de alto grado y mecanismos de balanceo de las mismas en situaciones no estándar. Esta evaluación permitirá profundizar en el análisis de la herramienta de simulación.

1.3. Evaluación mediante simulación

Teniendo en cuenta factores temporales y econónimos, la evaluación por simulación se toma como la única herramienta posible para realizar análisis del rendimiento alcanzado por las redes de interconexión.

Una taxonomía bien conocida de las herramientas de simulación es si estas están conducidas por tiempo o por eventos. Respectivamente, las primeras avanzan el tiempo con un delta fijo y en cada actualización del reloj evaluan y actualizan el estado del sistema completo y las segundas, avanzan el tiempo hasta el momento en el que se produce el próximo evento y evaluan y actualizan en ese momento el estado del sistema tras el evento ocurrido.

Asímismo, hay tres posibilidades para proporcionar una carga de trabajo a estas herramientas: simulacion completa, uso de trazas y tráfico sintético. Las dos primeras representan de forma fiel la realidad de las comunicaciones de un sistema pero la carga computacional necesaria para llevar a cabo estas evaluaciones es elevada.

En este trabajo se ha usado una herramienta conducida por tiempo, ya que aunque da la posibilidad de trabajar con parte del sistema evaluado por eventos, no está recomendado y no se ha evaluado. Respecto a las cargas usadas, se ha trabajado con tráficos sintéticos modelados con la propia herramienta.

1.4. Herramientas actuales

Como ya se ha comentado, este trabajo se centra en el estudio del simulador `BookSim`, aunque en esta sección se presentan una serie de alternativas y un resumen comparativo de características.

Para realizar investigación en redes de interconexión han sido desarrolladas una serie de herramientas o simuladores con ciertas peculiaridades que los hacen diferentes entre si. A continuación se enumeran una serie de herramientas que se podrían evaluar como alternativas al simulador estudiado:

- **Garnet**: simulador de red detallado incluido en el simulador de sistema completo GEM5, aunque es posible su uso de forma individual [AKPJ09].

- **Noxim**: simulador centrado en NOCs desarrollado en la Universidad de Catania [FPP, STN+14].
- **TOPAZ**: simulador de redes de interconexión de propósito general desarrollado en la UC [APM+12].
- **INSEE**: *framework* de simulación que incluye el simulador funcional de redes de interconexión FSIN y un modulo de generación de tráfico [NMAPR11].
- **DragonFly Simulator**: simulador de redes Dragonfly desarrollado en la UC [GVB+13].
- **Venus**: simulador de `IBM` basado en el motor de OMNeT++ [MR09].

A continuación se muestra una tabla que permite comparar las principales características de `BookSim` y los simuladores presentados previamente.

	BookSim	Garnet	TOPAZ	INSEE	Dragonfly Simulator	Noxim	Venus
Simula Redes de sistema	✔	✘	✔	✔	✔	✘	✔
Simula NOCs	✔	✔	✔	✘	✘	✔	✘
Admite múltiples topologías	✔	✔	✔	✔	✘	✘	✔
Admite múltiples *routers*	✔	✔	✔	✘	✔	✘	✔
Integrado con algún *full-system simulator* (FSS)	✔	✔	✔	✘	✘	✔	✘
Paralelo	✘	✘	✔	✘	✘	✘	✔
Admite trazas	✔	✔	✔	✔	✔	✘	✔
Datos sobre potencia	✘	✔	✔	✘	✘	✔	✘
Código abierto	✔	✔	✔	✔	✘	✔	✘
Nivel de detalle	*flit*	*flit*	*flit*	*phit*	*phit*	*flit*	*byte*
Tipo de *router*: {M=monociclo,S=segmentado}	{S}	{M,S}	{M,S}	{M}	{M,S}		{S}
Uso extendido (Nº. de citas)	35	223	$19 + 86$	74	7	78	40

Tabla 1.1: Comparativa de características de los diferentes simuladores presentados.

1.5. Estructura del documento

El resto del presente documento se ha estructurado como se explica a continuación. En el capítulo 2 se analiza el simulador `BookSim` y se estudia la viabilidad del mismo para los objetivos planteados a largo plazo. A lo largo del capítulo 3 se presenta y analiza la topología de red FB y se trata el uso del *trunking* sobre la misma. De igual forma, se realiza en el capítulo 4 para la topología DF. Finalmente, en el capítulo 5 se presentan una serie de conclusiones extraidas y unas lineas de trabajo abiertas al futuro.

2 BookSim

En este capítulo se presenta el simulador a estudiar y se realiza un análisis de viabilidad muy básico que tiene por objetivo responder a la pregunta de si es una herramienta apta para simular hasta millón de nodos. Posteriormente, se comenta de forma breve el desarrollo sobre el mismo.

`BookSim` es un simulador de redes de interconexión desarrollado originalmente como parte del libro *Principles and Practices of Interconnection Networks* [DT03b]. Se concibió y usó en un principio para generar las gráficas de rendimiento mostradas en el citado libro de texto.

2.1. Historia

La primera versión del simulador, distribuida como código libre, se encuentra disponible [DT03a] en la página web del autor. Esta versión del simulador no se centró inicialmente en ningún entorno en concreto, por lo que se diseñó de forma generalista para poder estudiar las redes de interconexión en general.

Entre sus principales metas se encuentran la flexibilidad y la precisión a la hora de modelar los componentes clave de una red. Es por ello, que los autores destacan su diseño modular y la "facilidad" para ser modificado y poder añadir nuevas características como algoritmos de encaminamiento, topologías o micro-arquitecturas de *router*. El simulador está escrito en C++ y está compuesto por una jerarquía de módulos codificados en clases separadas que implementan las diferentes funcionalidades de la red y del entorno de simulación. Cada uno de estos módulos cuenta con una interfaz bien definida que permite su reemplazo o customización sin afectar a otras partes del simulador. Una visión jerárquica de los principales módulos que lo forman se muestra a continuación.

`BookSim` ha sido incorporado en otros simuladores para modelar la red de estos, como es el caso de GPGPU-sim [BYF+09, BYF+], un simulador que proporciona un modelo detallado de las GPUs actuales.

La red a modelar con el simulador se compone de una colección de *routers* y enlaces, los cuales se conectan tal como especifica la topología. Todas las comunicaciones entre *routers* conectados se realizan a traves de operaciones de *send* y *receive* explícitas. Esto es así puesto que los autores pretenden que los usuarios piensen en objetos físicos cuando trabajan con el simulador y con esto reflejar de la forma más precisa posible el diseño de redes actual a la hora de implementar nuevas características en el simulador.

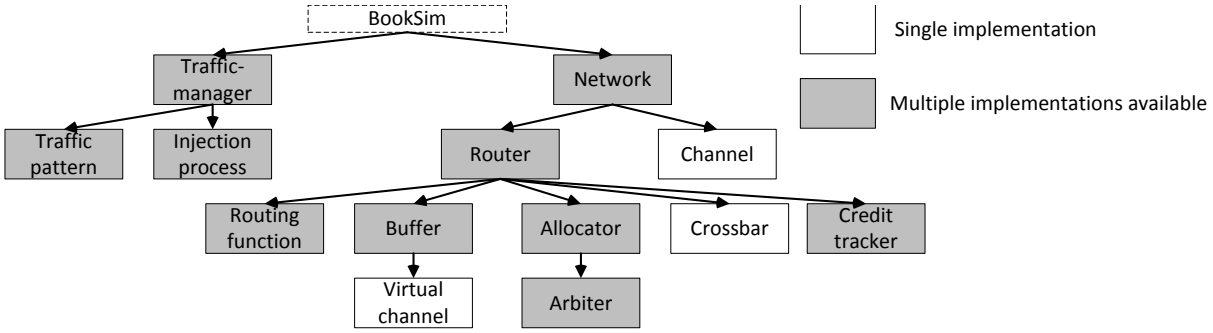


Ilustración 2.1: Jerarquia de módulos que componen el simulador. Fuente: [JBM+13]

`BookSim` simula la red a nivel de *flow control digits* (flits) y ciclos de reloj. Un paquete consiste en uno o más flits, la mínima unidad que se puede reservar en un *buffer* o en un canal. La anchura de todos los canales y de las etapas del *router* corresponde con la anchura de un *flit*. Durante un ciclo de reloj, un canal puede leer un *flit* de su entrada y escribir también otro *flit* en la salida.

`BookSim 2` [JBM+13] surgió tras actualizar la primera versión para que soportase características y topologías que se proponen en el contexto de las NOCs. La actualización incluye cambios que le permiten reflejar mejor el estado del arte en la investigación de NOCs. Concretamente, han incluido un modelo de la micro-arquitectura del *router* más detallado, el modelado de los retardos producidos en los canales internos al *router*, proporcionando soporte para nuevos modelos de tráfico y la implementación de nuevas topologías, como la “FlatFly-On-Chip”. A lo largo de este trabajo se hace uso de esta versión del simulador, puesto que aunque no se ha trabajado sobre NOCs, es una versión evolucionada que sigue permitiendo realizar simulaciones de redes de sistema.

Una vez presentado el simulador de forma breve, en la siguiente sección se trata la manera de lanzar un experimento básico con el mismo.

2.2. Simulación con BookSim

Comenzar a trabajar con BookSim es tan sencillo como descargar el código fuente del simulador disponible en su repositorio[1], disponer de las herramientas necesarias para su compilación [JMB+13] y compilar el código.

El primer parámetro que recibe el simulador es un fichero de configuración en texto plano que contiene información de la simulación a realizar. Esta configuración puede ser sobre-escrita pasando más parámetros por consola al simulador. Una serie de parámetros usados como base para todas las simulaciones realizadas a lo largo del trabajo y que se consideran relevantes se exponen en la siguiente tabla. Cualquier parámetro no especificado por el usuario toma un valor por defecto que se encuentra especificado en el fichero `booksim_config.cpp`.

[1] Repositorio GitHub: `http://github.com/booksim/`.

Parámetro	Valor	Unidades / Explicación
injection_rate	$\{0{,}05 \,..\, 1{,}00\}$ con paso $0{,}05$	*flits/node/cycle*
internal_speedup	$10{,}0$	*speedup* del *router* sobre la tasa de transmisión de los enlaces
vc_buff_size	64	profundidad del canal virtual en *flits*
num_vcs	1	número de canales virtuales por canal físico
vc_allocator	islip	tipo de *allocator*
sw_allocator	islip	tipo de *allocator*
sample_period	10000	ciclos de simulación
warmup_periods	3	longitud de la fase de calentamiento expresado como múltiplo de `sample_period`
sim_count	5	número de simulaciones promediadas
packet_size	1	*flits*
sim_type	throughput	tipo de simulación
routing_function	{min, valiant, ugal}	algoritmo de encaminamiento

Tabla 2.1: Parámetros generales usados en las simulaciones realizadas.

El parámetro `sim_type` permite especificar si se aborta una simulación superado un cierto límite de latencia fijándolo como "`latency`" o continuar la simulación pasado el punto de saturación de la red simulada hasta que 3 simulaciones converjan a un valor de *throughput*.

2.3. Viabilidad de las simulaciones

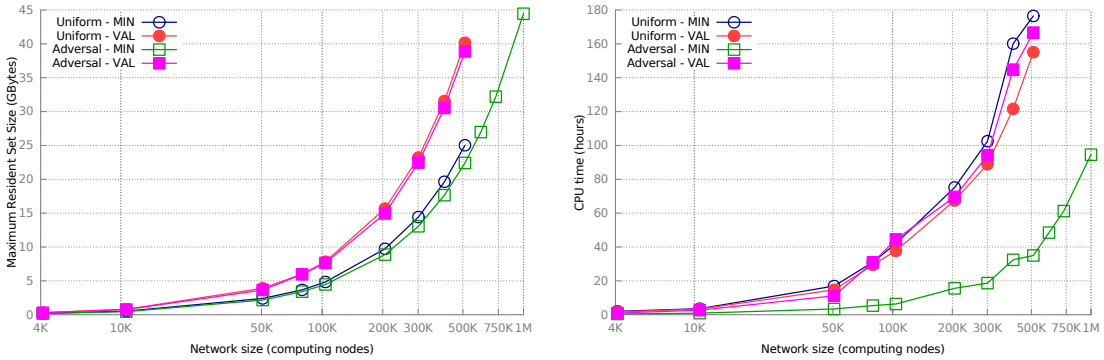
En esta sección se pretende mostrar si el simulador permite llegar a evaluar los sistemas objetivo a día de hoy y los recursos que necesita para lograr este objetivo. De forma rápida se puede estimar que estos sistemas alcanzan el millón de nodos, debido a que se pretende llegar a un rendimiento *Exascale* en base a nodos *Terascale*.

Para poder llegar a esta conclusión se ha medido el consumo de memoria y el tiempo de CPU que implican las simulaciones realizadas de la red FB a lo largo del presente trabajo y otras adicionales, llevadas a cabo con el objetivo de tener una aproximación de los recursos necesarios para evaluar un número determinado de nodos.

Obtener estos datos ha supuesto un problema en un primer momento debido a los entornos[2] en los que se ha trabajado, puesto que los datos que arrojaban las simulaciones eran muy dispares entre ambos. Esto viene propiciado porque uno de los sistemas empleados tiene un *bug* [Pis11, Nau12, Nie13] en la herramienta usada para medir el consumo de memoria. Concretamente, el fallo

[2]Se ha trabajado simultáneamente con los *clusters* `Calderon` y `Altamira`.

cometido por la herramienta `GNU time` produce que la información resultante esté multiplicada por el parámetro `PAGE_SIZE` del `kernel` del sistema usado. La versión del sistema operativo con la que cuenta el *cluster* que proporcionaba información errónea es `Scientific Linux 6.5`, un derivado de `Red Hat` que no ha parcheado el código de la herramienta para solucionar este problema. La forma de solucionar el problema ha sido obtener el valor de dicha variable en el sistema mediante `getconf PAGE_SIZE`, expresarla en `Kbytes` y dividir los valores obtenidos por ese factor.



(a) Consumo de memoria

(b) Tiempo de CPU

Ilustración 2.2: Consumo de memoria y tiempo de CPU empleado en la red FB en función del número de nodos y del tráfico-encaminamiento.

En base a las gráficas presentadas, se puede concluir que una simulación de una red con un millón de nodos de cómputo es viable con la tecnología actual y que se obtienen los resultados de la misma en un tiempo razonable para tratarse de evaluaciones tan grandes.

2.4. Desarrollo sobre BookSim

Para realizar los experimentos llevados a cabo a lo largo del presente trabajo se han desarrollado sobre el simulador dos topologías, con sus correspondientes algoritmos de enrutamiento y varios patrones de tráfico adicionales.

Utilizar `BookSim` y realizar alguna prueba tal como se descarga es extremadamente sencillo. Sin embargo, aunque programar sobre él no es imposible, la ausencia de documentación publicada y la mínima cantidad de comentarios presentes en el código hacen ardua esta tarea. Es bien sabido que la mejor forma de entender cómo trabaja un simulador es analizar y comprender su código fuente. Sin embargo, la interfaz de los módulos debería estar documentada correctamente para facilitar su reemplazo y modificación.

Este tipo de herramientas de uso académico suelen estar desarrolladas pensando en el objetivo y en la idea a evaluar en cada momento. Por consiguiente, no suelen estar diseñadas como un producto *software* mantenible y expandible a lo largo de los años. Y de hecho, ni siquiera suelen plantearse lineas de desarrollo ni metodologías estándar desde el punto de vista de la ingeniería

de *software*. Además, como es habitual que este tipo de herramientas las desarrollen alumnos, quedan patentes en ellas diferentes estilos de programación lo que no ayuda a comprender el funcionamiento de la herramienta.

En el caso de la primera topología que se ha evaluado en este trabajo, estaba ya implementada en el simulador. Sin embargo, la implementación existente de la red contaba con una extraña división de la misma en coordenadas X e Y y una división similar de los nodos de cómputo conectados a un *router*. Ampliar esta implementación de la red para realizar las pruebas necesarias para el presente trabajo se ha considerado una decisión incorrecta y se ha optado por empezar de cero y recodificar la red, lo que ha supuesto un considerable retraso. Tras haber mantenido un contacto directo con el desarrollador que realizó dicha implementación y que actualmente es una de las personas responsables del simulador, se ha concluido que ha sido una decisión acertada. Por dotar a esta conclusión de peso, es conveniente comentar que él mismo a lo largo del contacto mantenido ha reconocido que su implementación no era la más adecuada y que la que se ha realizado en este trabajo es válida.

En el caso de la segunda topología, ya se había superado parte de la curva de aprendizaje y el desarrollo de la topología ha sido más rápido. Aunque igualmente, a lo largo de la realización de la misma se han aprendido a usar ciertos elementos del simulador, como la elección de los paquetes que dejan traza total mediante parámetros, que se ha empleado después en la primera topología implementada puesto que se ha valorado su utilidad.

3 Caso 1: Flattened Butterfly

En este capítulo se realiza un estudio de las redes conocidas como FBs. Tras analizar sus propiedades topológicas y su encaminamiento, se analiza en detalle la problemática que presentan las redes asimétricas basadas en esta topología. La evaluación de la solución al problema planteado arroja resultados satisfactorios. Durante los experimentos realizados se ha detectado un problema de congestión que es mostrado pero que no ha podido ser estudiado con detalle en el marco de este trabajo.

La red FB surge de la mano de la transición hacia redes de alto grado como las implementadas por primera vez en sistemas como el Cray BlackWidow [SAKD06]. Fue presentada [KDA07] como una evolución de la `Butterfly` [DT03b] y como una topología eficiente en coste para redes de alto grado. Posteriormente ha sido propuesta para ser usada como NOC [KBD07].

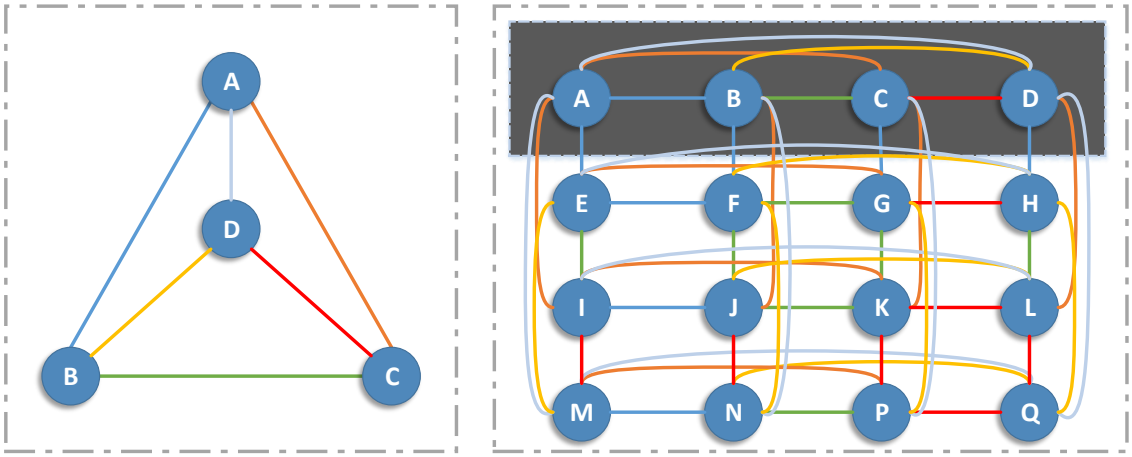
3.1. Topología flattened butterfly

Topológicamente hablando es una red construida sobre un grafo de Hamming, éstos grafos son isomorfos al producto cartesiano de grafos completos. Como en un grafo completo todos los vértices están conectados, en un grafo de Hamming y por ende en la topología FB, cada vértice (o router) está conectado a cualquier otro que solo difiera de este en una componente. En la siguiente ilustración se muestra la representación habitual de un grafo completo K_4, una segunda representación de K_4 que permite visualizar mejor esta topología y junto a ésta, el producto cartesiano de grafos completos $K_4 \times K_4$.

Una red construida con esta topología es una red directa, puesto que todos los *routers* son nodos finales de la red, es decir, están conectados a nodos de cómputo. El diámetro o máxima distancia entre dos nodos (*routers*) es igual al número de dimensiones de la misma.

Desde un punto de vista más tecnológico, una red basada en esta topología está definida por los siguientes parámetros:

- **Número de dimensiones (n)**: la dimensión en la que se conectan los nodos de cómputo a los *routers* se considera la dimensión 0.
- **Tamaño de las dimensiones (K_d)**: número de *routers* presentes en la dimensión d.
- **Concentración (c)**: número de nodos de cómputo conectados a cada *router*.



(a) K_4 (b) K_4 sombreado y $K_4 \times K_4$

Ilustración 3.1: Dos representaciones de un grafo completo y su producto cartersiano.

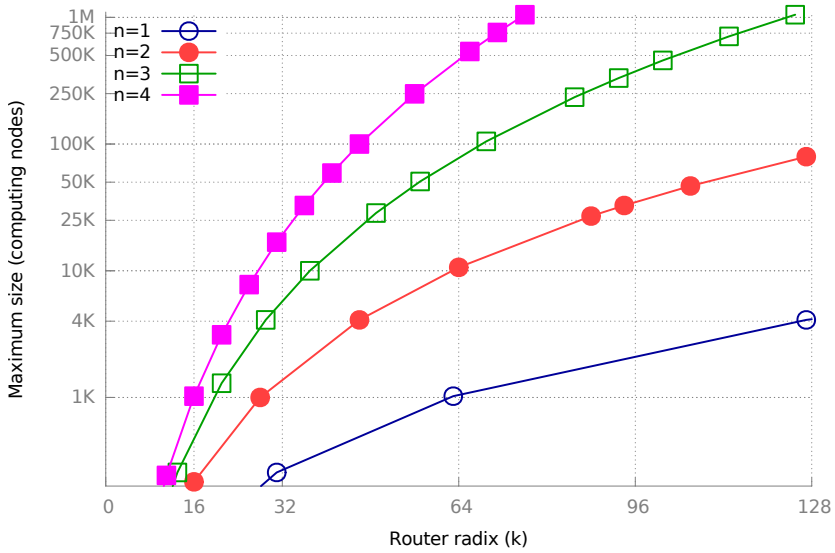


Ilustración 3.2: Escalabilidad de la red FB expresada como el número máximo de nodos de cómputo (N) en función del número de dimensiones (n) y del grado de los *routers*.

El número de nodos (N) que pueden ser conectados en una red con esta topología se muestra en la ilustración anterior como una función del número de dimensiones n y del grado (número de puertos) de los *routers* empleados en su implementación.

Tras tener una visión clara de la topología sobre la que se trabaja en este capítulo, a continuación se realiza un análisis teórico del *throughput* que se puede obtener en una red basada en ella.

3.1.1. Análisis teórico del *throughput*

Lo primero que pasa a estudiarse en esta subsección son los patrones de tráfico benignos y adversos. Estos son respectivamente, el patrón de tráfico uniforme aleatorio y el denominado `MultiDimNeighbor` (MDN). El patrón de tráfico uniforme aleatorio es un tráfico que balancea la carga entre los enlaces de la red. Por el contrario, el patrón de tráfico de peor caso se produce cuando cada nodo de cómputo conectado a un *router* (R_1 en la ilustración) envía tráfico a cualquier otro nodo conectado al *router* (R_6 en la ilustración) que se encuentra en la posición siguiente al *router* origen en todas las dimensiones. Con este patrón de tráfico y bajo encaminamiento mínimo, todos los nodos de cómputo contectados a un *router* intentan enviar tráfico por el mismo enlace.

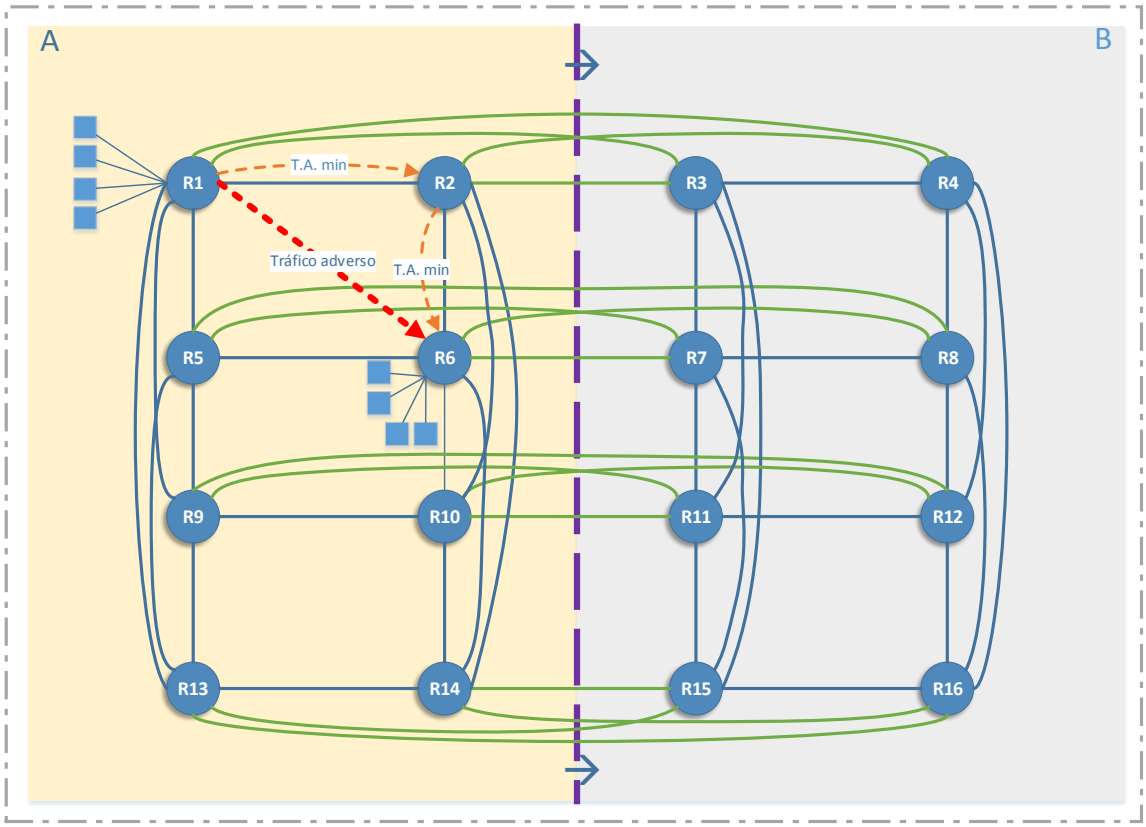


Ilustración 3.3: Representación del tráfico adverso y ejemplo usado en los cálculos teóricos.

En la ilustración anterior se muestra un ejemplo de origen y destino que seguiría un paquete bajo el patrón de tráfico adverso y la ruta mínima elegida por la cual se enviaría el paquete. En la ilustración también está representado el corte que deja a ambos lados el mismo número de nodos, la dirección en la que se analiza el tráfico que atraviesa esta bisección y los enlaces que la atraviesan se han coloreado en verde. En la zona sombreada "A" hay 8 *routers*, a cada *router* se conectan 4 nodos de cómputo, por lo que en esa zona hay 32 nodos de cómputo en total.

En primer lugar, teniendo en cuenta encaminamiento mínimo y bajo tráfico uniforme, la mitad de esos nodos (16) envían tráfico a nodos de cómputo de la zona "B". Puesto que el corte mostrado

es atravesado por 16 enlaces, los nodos de cómputo van a poder inyectar a su máxima carga. En segundo lugar, teniendo en cuenta el mismo encaminamiento bajo tráfico adverso, en la ilustración se puede ver que los 4 nodos de cómputo conectados a un *router* intentan enviar tráfico por un solo enlace, por lo que éstos van a poder inyectar carga hasta un límite de $1/4 = 0{,}25$.

En tercera instancia, teniendo en cuenta encaminamiento Valiant y bajo tráfico uniforme, de los 32 equipos en "A", 16 envían tráfico a nodos de la zona "B". De éstos 16, 8 hacen misrouting hacia un *router* en la zona "B" y posteriormente se encaminan de forma mínima dentro de la zona "B", mientras que otros 8 hacen misrouting hacia un *router* en la zona "A" y posteriormente se encaminan de forma mínima hacia "B", por lo que en ambos casos han atravesado el corte realizado. De los otros 16 nodos de cómputo que quieren enviar tráfico a nodos en "A", la mitad hacen misrouting en "B", por lo que también atraviesan el corte y de igual forma hay 8 nodos de cómputo de la zona "B" que hacen misrouting en "A" pero que posteriormente se encaminan mínimo hacia "B". Teniendo en cuenta los datos anteriores, en total hay 32 nodos de cómputo que quieren enviar tráfico por enlaces que atraviesan el corte propuesto pero solamente hay 16 enlaces, por lo que los nodos de cómputo podrán inyectar tráfico a una carga máxima de $0{,}5$. Otra posible aproximación es ver este análisis como dos encaminamientos mínimos, por lo que se va a conseguir la mitad de rendimiento que en el caso mínimo. Para tráfico adverso bajo el mismo encaminamiento, se puede ver que de la primera columna de la zona "A" 8 nodos hacen misrouting en un *router* de la zona "B", de la segunda columna 8 lo hacen en "A" y posteriormente se encaminan mínimo hacia "B" y otros 8 hacen misrouting en "B". Continuando con el análisis puede verse que, de la tercera columna 8 hacen misrouting en "A" y posteriormente se encaminan mínimo hacia "B" y contando todos los nodos de cómputo mencionados que encaminan paquetes a través de enlaces que atraviesan el corte, se obtiene que la máxima carga a la que van a poder inyectar va a ser de $0{,}5$. Otra manera de ver este análisis es que bajo encaminamiento Valiant, el tráfico se uniformiza, por lo que se va a obtener el mismo resultado que con un patrón de tráfico uniforme y encaminamiento Valiant.

Los resultados teóricos obtenidos en esta subsección se muestran como límites teóricos en las gráficas obtenidas de las simulaciones iniciales presentadas en la siguiente subsección.

3.1.2. Evaluación mediante simulación

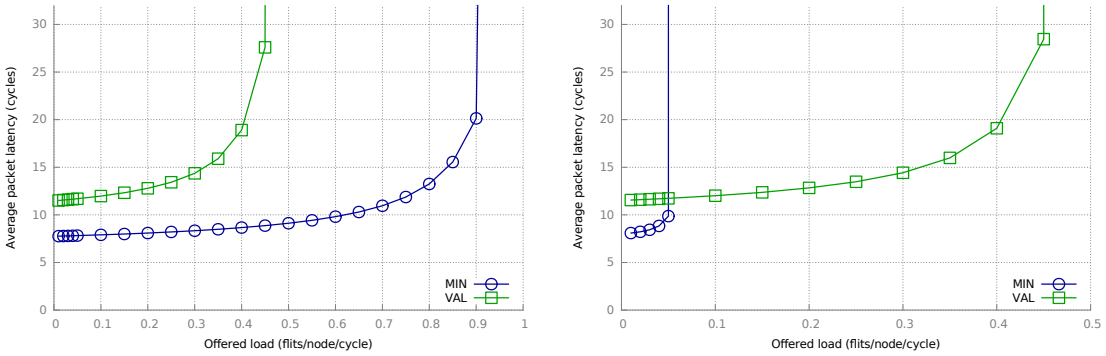
Para evaluar la implementación realizada de la topología y de los algoritmos de encaminamiento se realizan una serie de simulaciones iniciales para contrastar los resultados frente a los fijados de forma teórica previamente.

Los parámetros generales usados se presentan en la tabla 2.1 y en la mostrada a continuación se exponen los parámetros que se han usado de forma particular para esta prueba.

Una vez procesados los datos obtenidos de estas simulaciones iniciales se obtienen las gráficas de latencia de paquete y *throughput* mostradas a continuación. Además de las curvas de resultados, se dibujan los límites teóricos. De las gráfcias mostradas se puede concluir que la implementación realizada es correcta puesto que se ajustan a la teoría calculada previamente.

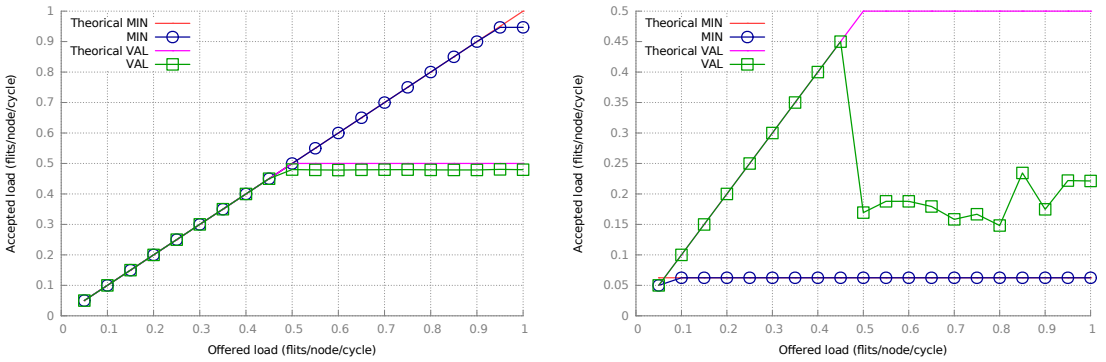
Parámetro	Valor	Unidades / Explicación
topology	flattenedbutterfly	nombre de la topología
n	2	dimensiones
k	{16,16}	routers en cada dimensión
c	16	nodos por router
traffic	{uniform, multidimneighbor}	patrón de tráfico

Tabla 3.1: Parámetros particulares usados para validar la implementación realizada.



(a) Uniform (b) Adverso

Ilustración 3.4: Latencia de paquete de la red FB bajo diferentes patrones de tráfico.



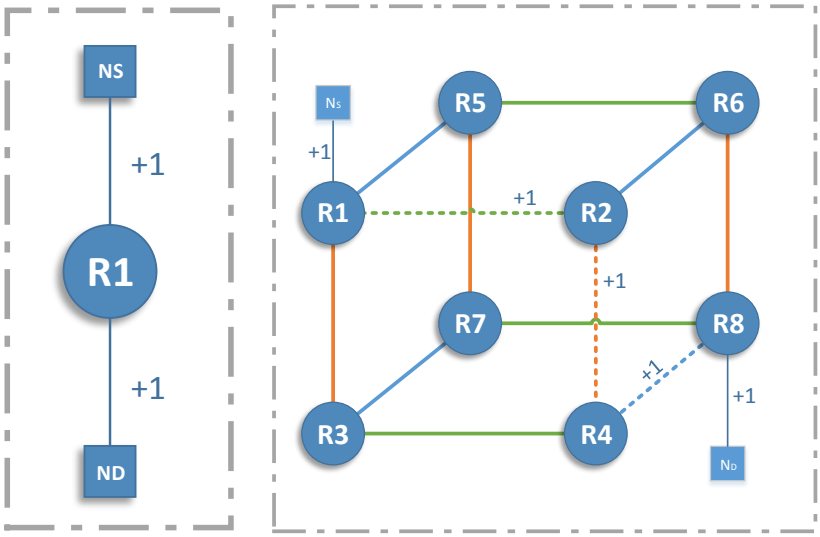
(a) Uniform (b) Adverso

Ilustración 3.5: *Throughput* de la red FB bajo diferentes patrones de tráfico.

Tras haber tratado la topología de una red FB, a continuación se presenta la manera de determinar los caminos entre un origen y un destino, es decir, el encaminamiento.

3.2. Encaminamiento

El encaminamiento de un paquete en una red FB requiere un salto del nodo de cómputo origen (N_S) al *router* al que está conectado (R_S), cero o más saltos entre *routers*, y un salto final del último *router* (R_D) atravesado por el paquete al nodo de cómputo destino (N_D).



(a) 2 - 0 (b) 5 - n=3

Ilustración 3.6: Representación del encaminamiento en una FB mostrando el número de saltos del paquete y el número de saltos entre *routers*.

3.2.1. Encaminamiento *oblivious*

En cuanto a encaminamiento mínimo, bajo *Dimension Order Routing* (DOR), una red construida sobre un grafo de Hamming no presenta ciclos. Por lo que un encaminamiento mínimo dimensión a dimensión está libre de *deadlock* sobre esta red. Este encaminamiento se muestra trivial si se numeran los *routers* en orden creciente por dimensión y respectivamente se numeran los nodos de cómputo conectados a éstos de forma creciente. Usando esta numeración, se puede extraer el *router* (R_D) al que se conecta un nodo N_D mediante la siguiente expresión, representado por $|R|$ el número total de *routers* presentes en la red.

$$R_D = \frac{N_D}{|R|} \tag{3.1}$$

Tras obtener el *router* destino, se obtiene su posición dentro de cada dimensión para usarla a modo de coordenadas. Entonces, se puede ver que un paquete va a dar tantos saltos entre *routers* como diferentes índices haya en las coordenadas de sus *routers* origen y destino.

Encaminar de forma no mínima en la red FB proporciona una diversidad de caminos adicional y permite balancear la carga para patrones de tráfico arbitrarios. En este caso, se puede hacer

libre de *deadlock* usando DOR empleando diferentes canales virtuales para cada mitad del camino [DS87], por lo que para este encaminamiento se requiere un mínimo de dos canales virtuales. Para este tipo de encaminamiento se ha usado el algoritmo de Valiant [VB81] que randomiza cualquier patrón de tráfico y lo asemeja a un patrón aleatorio uniforme.

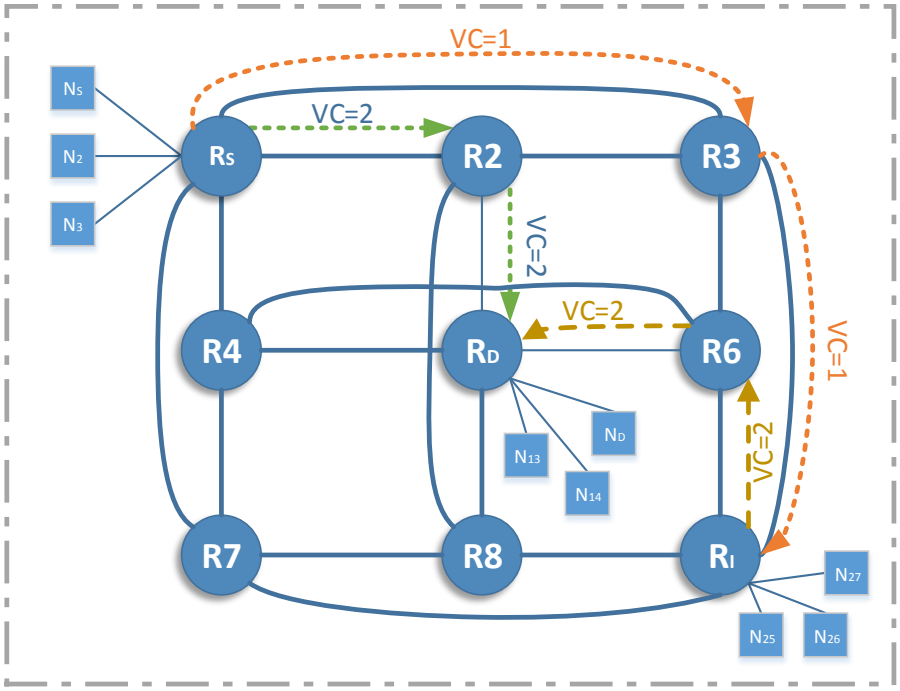


Ilustración 3.7: Representación del encaminamiento *oblivious* en una FB mostrando una ruta mínima y ambas fases de la ruta no-minima.

3.2.2. Encaminamiento adaptativo

Para tener en cuenta el estado de la red a la hora de encaminar los paquetes, se usan algoritmos de encaminamiento denominados adaptativos. Para probar un encaminamiento de este tipo sobre la red FB se ha optado por *Universal Globally Adaptive Load-Balancing* (UGAL) [Sin05]. Este algoritmo decide paquete a paquete si ha de seguir un enrutamiento MIN o Valiant (VAL) para minimizar el retardo estimado para cada paquete. El retardo de una ruta es estimado mediante el producto del número de saltos (H) y la longitud de la cola asociada al puerto (Q). UGAL encamina un paquete de forma mínima si la siguiente inecuación se satisface:

$$H_{min} \cdot Q_{min} \leq H_{val} \cdot Q_{val} + T \tag{3.2}$$

donde T es una constante usada para filtrar cargas temporalmente no balanceadas (vistas como fluctuaciones de la longitud de las colas) y permite balancear el rendimiento entre patrones de tráfico benignos y adversos. La constante *threshold* (T) no forma parte del algoritmo original de UGAL y se ha añadido a la implementación realizada del mismo con el objetivo de llegar a un compromiso entre la “rapidez” con la que el algoritmo responde ante un patrón de tráfico adverso y el rendimiento.

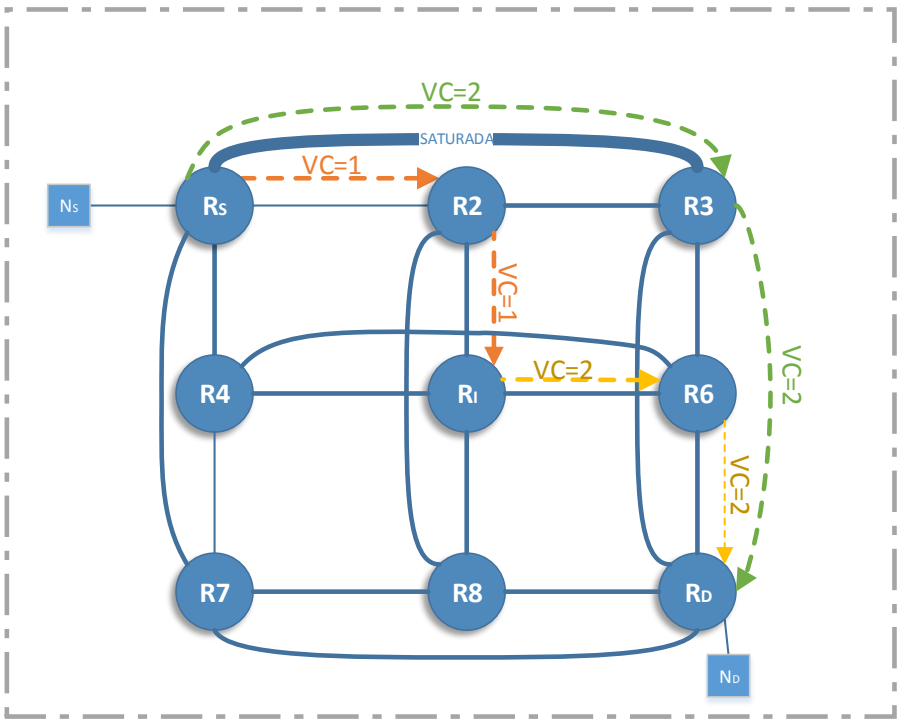
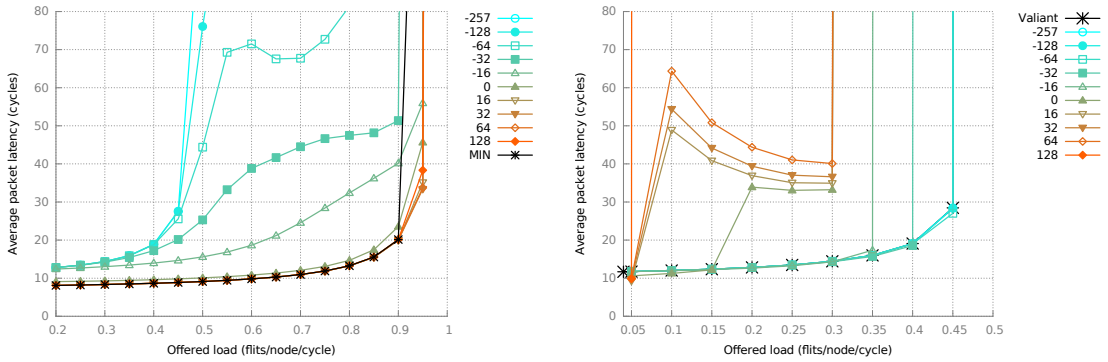


Ilustración 3.8: Representación del encaminamiento adaptativo (mostrando la posible ruta mínima y ambas partes de la no-minima) en una red FB con un canal saturado.

En esta sección se pretende obtener el valor adecuado para dicho umbral. Para ello, se ha lanzado un barrido de simulaciones con los valores de los parámetros generales mostrados en la tabla 2.1, algunos particulares a la topología mostrados en la tabla 3.1 y el conjunto de valores $\{-257, -128, -64, -32, -16, 0, 16, 32, 64, 128\}$ para el umbral (parámetro `threshold`). Los valores de T que tiene sentido evaluar se encuentran comprendidos en el siguiente intervalo $[-1 \cdot (2 \cdot n \cdot vc\_buff\_size + 1), n \cdot vc\_buff\_size]$, cualquier valor fuera del mismo no aporta información adicional. El valor mínimo de este intervalo fuerza a UGAL a hacer misrouting para todos los paquetes y, por el contrario, el valor máximo a realizar un encaminamiento mínimo. Es evidente que el valor del umbral que minimiza la latencia y maximiza el *throughput* va a ser contrario para patrones de tráfico benignos y adversos.

Tras analizar las gráficas resultantes del barrido por la constante T mostradas a continuación, se decide descartar los valores -257, -128, -64 y -32 debido a su resultado bajo el patrón de tráfico uniforme y de igual forma los valores 16, 32, 64 y 128 por el conseguido en adverso.

Si se analizase la implementación de una red para una máquina real se realizarían mas pruebas para determinar el valor exacto a usar, lanzando otro barrido entre -16 y 0, pero los resultados obtenidos son suficientes para el trabajo realizado a lo largo de este capítulo. A partir de este momento, el resto de simulaciones realizadas con la red FB usarán el valor de umbral determinado anteriormente para el algoritmo de encaminamiento UGAL.



(a) Uniforme (b) Adverso

Ilustración 3.9: Evaluación del *threshold* para UGAL-FB en diferentes patrones de tráfico.

Tras analizar el encaminamiento en la topología estudiada en este capítulo, a continuación se estudian las redes asimétricas basadas en la misma.

3.3. Redes asimétricas

En esta sección se analizan las redes FB asimétricas, es decir, las que presentan una topología con dimensiones de diferentes tamaños. Un ejemplo de una red FB asimétrica es la red implementada en un "*electrical group*" del `Cray XC30` [AFRK12]. A continuación se expone el problema que se presenta en las mismas, se estudia una posible solución, se evalúa la propuesta y se muestran unas conclusiones sobre ésta.

3.3.1. Problema en flattened butterflys asimétricas

Teniendo en cuenta que se desea que todos los nodos de cómputo puedan inyectar la totalidad del tráfico generado bajo tráfico uniforme aleatorio, el número de inyectores (nodos de cómputo) por *router* (c) que se puede tener en una red FB asimétrica está limitado por el mínimo de los tamaños (K) de las dimensiones:

$$c \leq min\{K_d \mid d \in \{1, ..., n\}\} \tag{3.3}$$

El problema que se presenta en estas redes se muestra de forma gráfica en la ilustración siguiente. Como se puede observar, se trata de una red de dos dimensiones ($n = 2$) con $K_1 = 4$ y $K_2 = 2$. Existen dos posibles cortes que dejan el mismo número de nodos de cómputo a cada lado aunque en el esquema mostrado solamente se ha representado la más restrictiva. Para resaltar los enlaces que atraviesan esta bisección, éstos se han representado en color verde.

Bajo tráfico uniforme, la mitad de los nodos de cómputo de la zona sombreada "A" van a enviar paquetes a "B" y lo tienen que hacer a través de los 4 enlaces que atraviesan el corte representado.

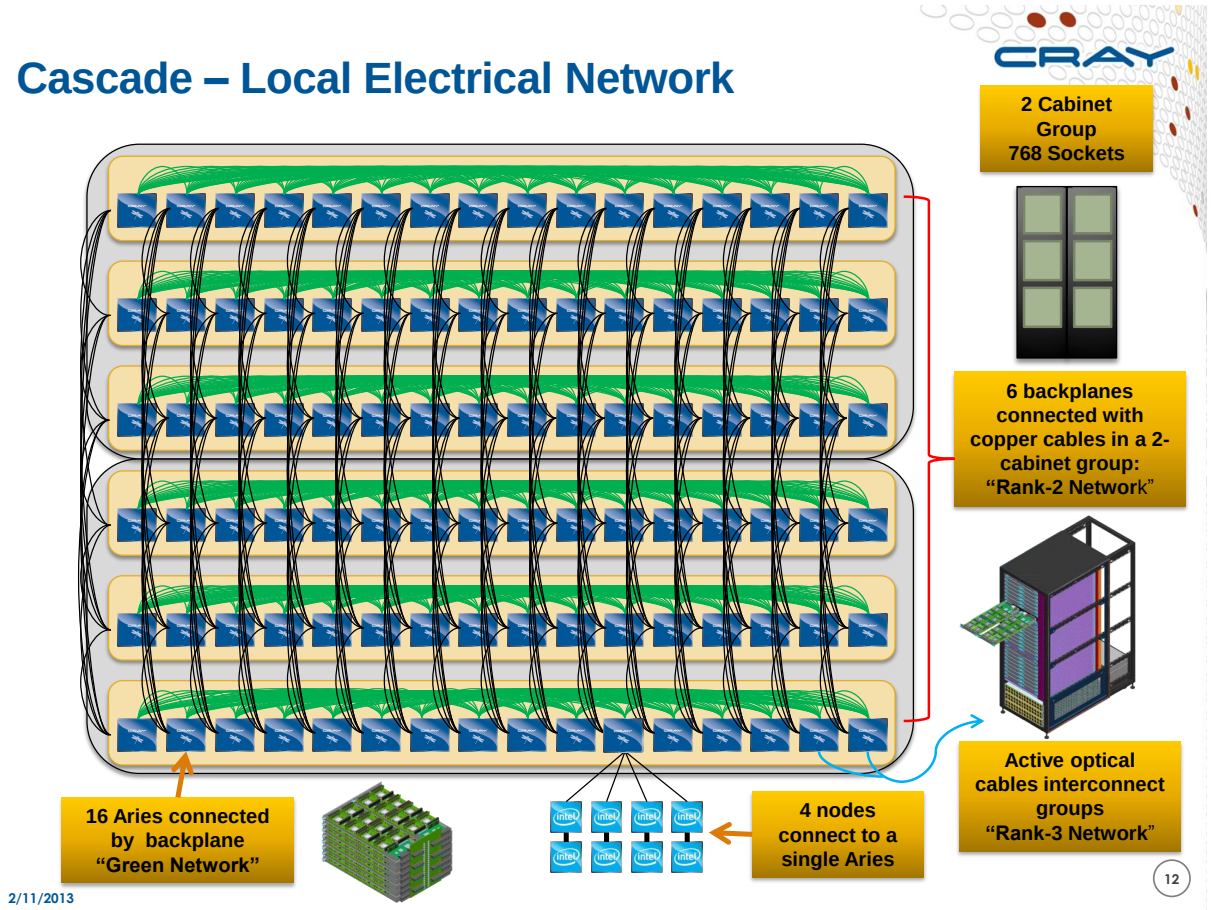


Ilustración 3.10: Representación de la red de un grupo eléctrico de un sistema Cray XC30 [Wic13].

Para garantizar que todos los nodos puedan inyectar a su máxima carga, sólo 4 pueden enviar paquetes de "A" a "B", esto implica que en la zona "A" como máximo puede haber 8 nodos de cómputo. Entonces, puesto que en esa zona sombreada hay 4 *routers*, el número de nodos de cómputo por *router* (c) debe ser igual a 2 (según la ecuación 3.3 $c = min\{4, 2\}$).

Esta limitación supone un problema si se necesitan emplear redes de este tipo por cualquier motivo, como por ejemplo, el problema que se presenta en el `Cray XC30`. Lo que denominan "grupo eléctrico" es la unión de dos *racks* en los que el espacio físico les permite colocar 6 *chassis* de 16 *blades* con 4 nodos de cómputo cada uno (véase ilustración 3.10). Trasladando estos datos, se trata de una red FB de dos dimensiones de tamaño 6 y 16 cada una y $c = 4$ inyectores por *router*.

El problema analizado en esta sección presenta cierta analogía con el que se presenta en un toro en el que los *routers* tienen el mismo grado (número de enlaces) en cada dimensión pero diferente número de *routers* por dimensión [CMV+10]. Esto provoca que la distancia media por dimensión varíe y la dimensión más larga, al dar más saltos por la misma, se convierte en el cuello de botella de la red. La solución propuesta para este caso es hacer un *twist* en los enlaces periféricos, con lo que manteniendo el mismo grado en los *routers*, se iguala el número de saltos por cada dimensión.

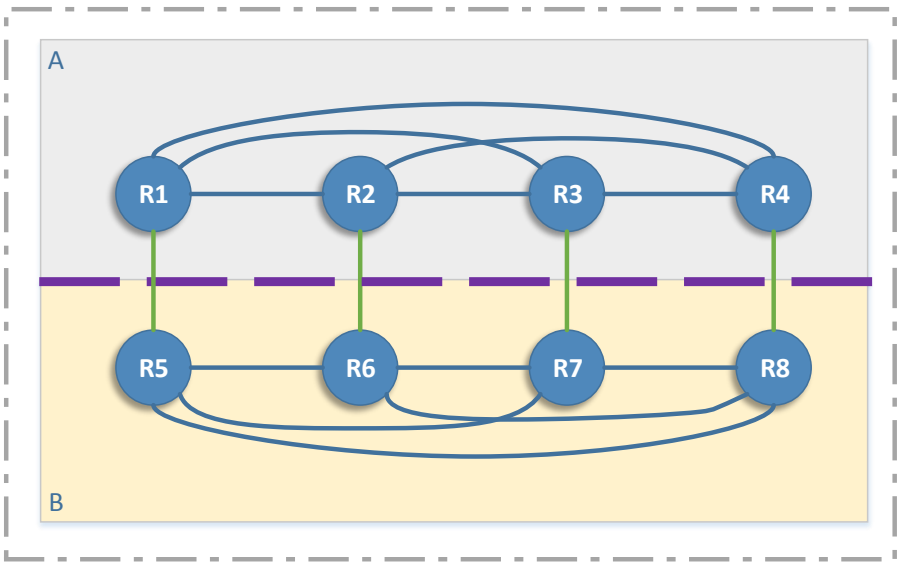


Ilustración 3.11: Representación del problema que se plantea en redes FB asimétricas.

3.3.2. Solución para balancear flattened butterflys asimétricas

En el caso de una topología FB asimétrica, el número de saltos en cada dimensión es el mismo (1), y para igualar el grado, se estudia el uso de varios enlaces entre cada par de routers. El uso de múltiples enlaces físicos para un mismo enlace lógico se conoce como *trunking*.

El grado de un *router* para una dimensión, teniendo en cuenta que los enlaces son bidireccionales y que el conexionado es un grafo completo, se puede obtener calculando el número de subconjuntos de 2 elementos entre el conjunto de *routers* presentes en esa dimensión. Puesto que la solución estudiada consiste en igualar el grado en todas las dimensiones, la red propuesta tiene que satisfacer la siguiente inecuación:

$$t_d * \binom{K_d}{2} = t_{d+1} * \binom{K_{d+1}}{2} \; \forall \; d \in \{1, ..., n-1\} \tag{3.4}$$

donde t representa el *trunking*. Simplificando la expresión anterior y tras haber añadido el concepto de *trunking* a la topología, una red estará bien dimensionada si se satisfacen las siguientes expresiones

$$t_d * (K_d - 1) = t_{d+1} * (K_{d+1} - 1) \; \forall \; d \in \{1, ..., n-1\} \tag{3.5}$$

$$min\{t_d | \; d \in \{1, ..., n\}\} = 1 \tag{3.6}$$

$$c = t_d * (k_d - 1) + 1 | \; d \in \{1, ..., n\} \tag{3.7}$$

En caso de incumplir la ecuación 3.6 y usar un *trunking* superior a 1 en todas las dimensiones, la red seguiría aceptando toda la carga ofrecida por los nodos de cómputo pero estaría sobre-

dimensionada. Esto se puede ver fácilmente realizando un análisis similar al mostrado gráficamente y comentado en la subsección 3.3.1.

En la ilustración siguiente se muestra el *trunking* necesario (según la ecuación 3.5) para dimensionar correctamente la red mostrada en la ilustración 3.11.

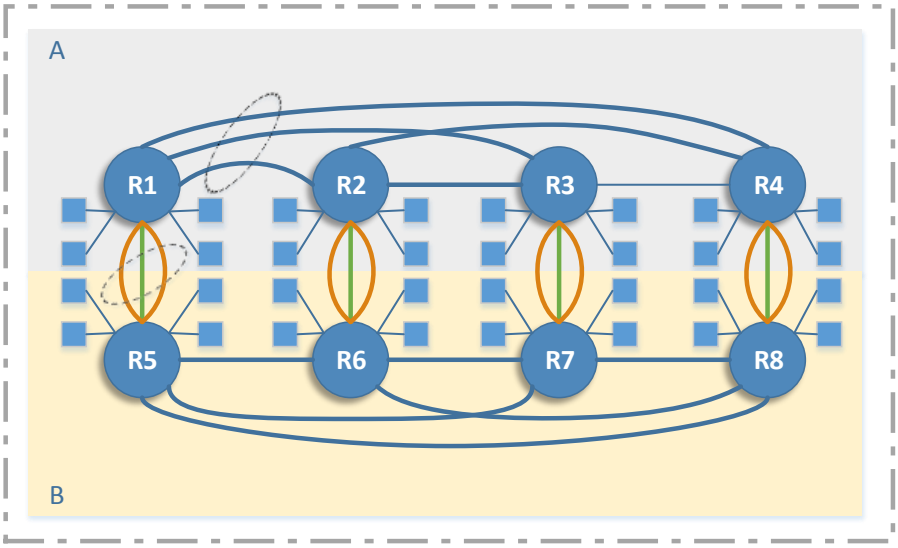


Ilustración 3.12: Representación de la solución aplicada al problema planteado por las redes FB asimétricas.

3.3.3. Implementación de trunking

Se entiende por *trunking* al uso de varios enlaces físicos para representar un enlace lógico, en este caso concreto, se ha implementado un *trunking* al estilo del que implementan las redes `ethernet` conocido como *Link Aggregation Group* (LAG). Es decir, una pareja de *routers* se conecta mediante varios enlaces físicos, tantos como defina el valor de *trunking*.

Este cambio en la topología supone aumentar la diversidad de caminos por los que un paquete puede viajar de un origen a un destino. Si se sigue empleando DOR, este cambio no supone la existencia de *deadlocks*, aunque obliga a modificar los algoritmos de encaminamiento para que tengan en cuenta los nuevos caminos disponibles. Al existir varios enlaces entre un *router* origen (R_S) y uno destino (R_D), se ha de seleccionar por cual se envía cada paquete que quiere viajar de uno a otro. Esta elección se puede realizar de acuerdo a diferentes políticas, en este caso se ha empleado la conocida como *join the shortest queue* (JSQ), mediante la cual cada paquete se envía por el enlace menos cargado.

Teniendo en cuenta el modelo de *router* implementado en el simulador usado, el cual cuenta con colas a la entrada por canal virtual [DT03b, JBM+13], se puede estimar la carga del canal usando el contador de créditos de cada canal virtual de salida. Estos contadores reflejan la ocupación de

la cola de entrada situada al final del canal, el uso de los mismos para estimar el canal de salida a emplear se refleja en el siguiente diagrama.

Ilustración 3.13: Representación de la decisión tomada por la política JSQ.

3.3.4. Evaluación y resultados

La solución estudiada en esta sección se ha evaluado mediante sendas simulaciones con la topología que presenta un "grupo eléctrico" de un sistema `Cray XC30` y con una topología con 3 dimensiones. Como solamente se pretende observar el comportamiento de la topología, se han usado parámetros de simulación que permiten obviar otros factores limitantes en una red como el comportamiento de los *routers* (véase tabla 2.1 de parámetros generales).

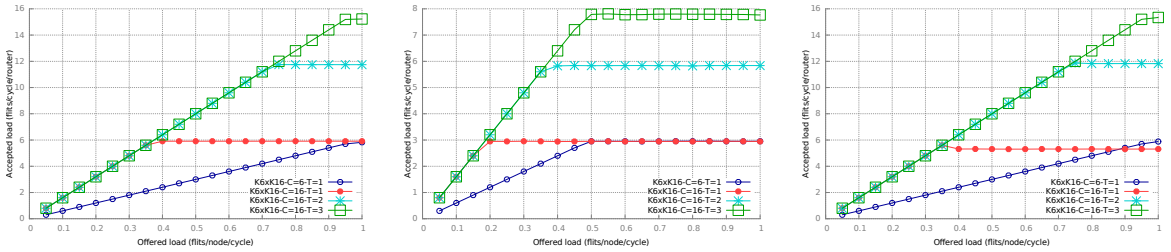
Para evaluar el caso de la red del 'grupo eléctrico" del `Cray XC30` se ha simulado la red sin *trunking*, como un valor intermedio del mismo y con el factor de *trunking* necesario para balancear correctamente la red según las ecuaciones 3.5, 3.6 y 3.7. Los parámetros particulares a estas simulaciones se muestran a continuación:

Parámetro	**Valor**	**Unidades / Explicación**
topology	flattenedbutterfly	nombre de la topología
n	2	dimensiones
k	$\{6,16\}$	routers en cada dimensión
c	16	nodos por router
t	$\{1,1\}$, $\{2,1\}$ y $\{3,1\}$	*trunking* empleado en cada dimensión

Tabla 3.2: Parámetros particulares usados para evaluar el *trunking* en una red FB.

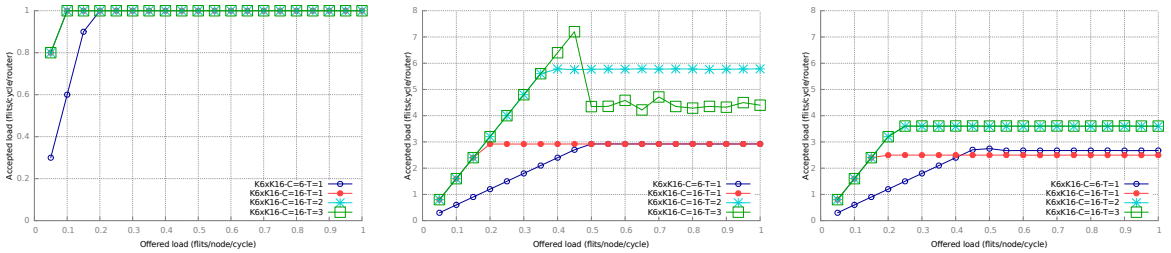
Los resultados obtenidos de la carga aceptada agregada por *router*[1] se representan a continuación, separados por patrón de tráfico y algoritmo de encaminamiento. El eje de ordenadas se ha acotado al valor máximo teórico que se puede conseguir en cada caso, teniendo en cuenta sólo factores propios de la topología para imponer estos límites.

[1] Puesto que se presentan datos de redes con diferentes niveles de concentración ($c = 6$ y $c = 16$), una forma de visualizar que aunque la carga inyectada por nodo de cómputo sea 1, la red con $c = 16$ está aceptando más carga es representar la carga agregada que acepta el *router*.



(a) MIN (b) VAL (c) UGAL

Ilustración 3.14: *Throughput* por *router* obtenido en redes FB asimétricas con *trunking* bajo diferentes algoritmos de encaminamiento y patrón de tráfico uniforme.



(a) MIN (b) VAL (c) UGAL

Ilustración 3.15: *Throughput* por *router* obtenido en redes FB asimétricas con *trunking* bajo diferentes algoritmos de encaminamiento y patrón de tráfico adverso.

En base a las gráficas se puede ver que bajo tráfico uniforme, si la red se dimensiona correctamente, el *throughput* obtenido es el máximo en cada caso. A priori y al igual que ocurre anteriormente, bajo tráfico adverso se podría pensar que con un factor de *trunking* superior a 1, el rendimiento alcanzado frente a un *trunking* de 1 sería superior. Esto es así para los encaminamientos VAL y UGAL, sin embargo, para un encaminamiento MIN no. En el caso de UGAL sería interesante volver a determinar el valor para el umbral a emplear para que realice más misrouting y con esto el resultado se acerque más al obtenido por VAL que por MIN.

Bajo tráfico adverso y encaminamiento mínimo los resultados con *trunking* no mejoran respecto al caso no balanceado puesto que aunque en una de las dimensiones se emplee el *trunking* necesario para balancear la red y el tráfico haga uso de esos múltiples enlaces, en la otra dimensión sigue existiendo solamente un enlace, que se convierte en el cuello de botella en este caso. La explicación comentada anteriormente se muestra gráficamente en la ilustración 3.16.

Un resultado curioso puede observarse en la ilustración 3.15(b), en la cual, el *throughput* obtenido bajo un dimensionamiento correcto es inferior al caso intermedio y presenta un grave problema de congestión. Este problema de congestión también se puede ver en la ilustración 3.5(b), por lo que ha despertado interés y se tratará brevemente en la siguiente sección.

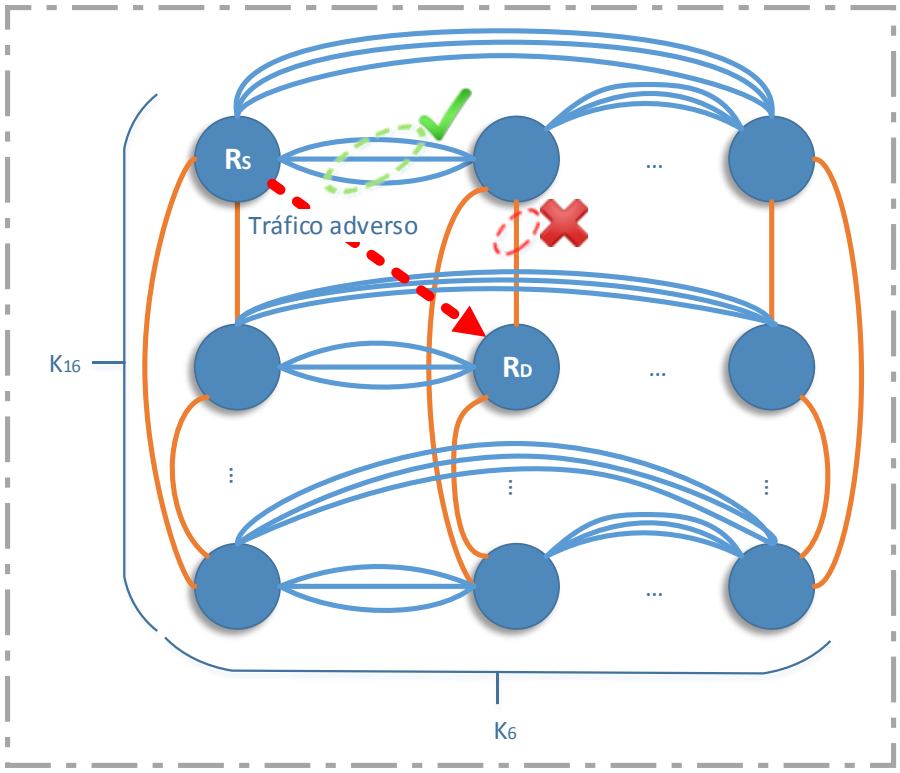


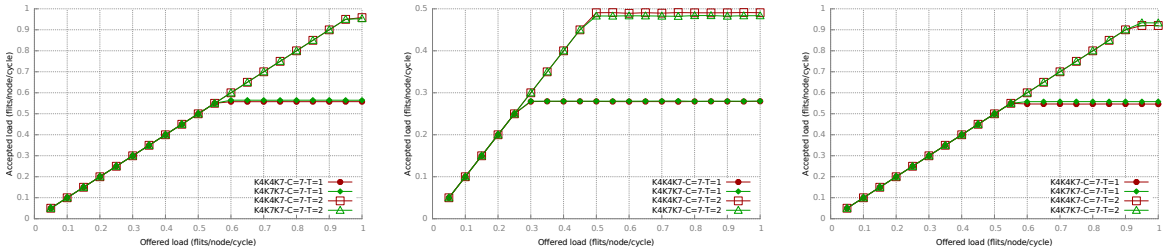
Ilustración 3.16: Representación del problema en FBs con *trunking* en tráfico adverso bajo MIN.

En el caso tridimensional se ha evaluado también si el número de dimensiones en las que se emplea *trunking* afecta al resultado, para esta prueba se han usado los parámetros generales mostrados en la tabla 2.1 y de forma particular los mostrados a continuación.

Parámetro	**Valor**	**Unidades / Explicación**
topology	flattenedbutterfly	nombre de la topología
n	3	dimensiones
k	{4,4,7} y {4,7,7}	routers en cada dimensión
c	7	nodos por router
t	{1,1,1}, {2,2,1} y {2,1,1}	*trunking* empleado en cada dimensión

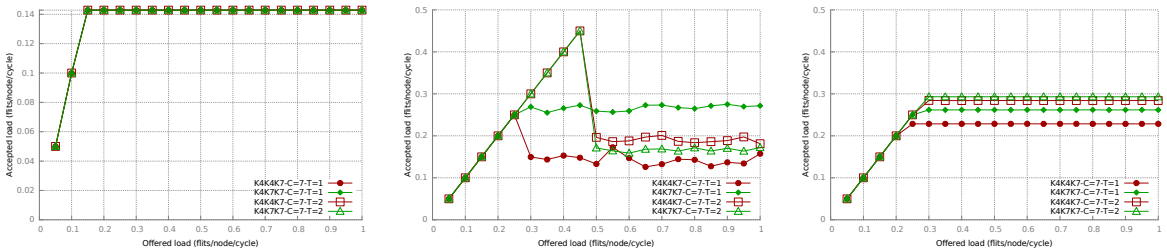
Tabla 3.3: Parámetros particulares usados para evaluar el *trunking* con tres dimensiones.

Los resultados de *throughput* aceptado por nodo de cómputo obtenidos en estas simulaciones se representan a continuación, separados por patrón de tráfico y algoritmo de encaminamiento. Al igual que en las anteriores, el eje de ordenadas se ha acotado al valor máximo teórico que se puede conseguir en cada caso.



(a) MIN (b) VAL (c) UGAL

Ilustración 3.17: *Throughput* obtenido en redes FB asimétricas tridimensionales con *trunking* bajo diferentes algoritmos de encaminamiento y patrón de tráfico uniforme.



(a) MIN (b) VAL (c) UGAL

Ilustración 3.18: *Throughput* obtenido en redes FB asimétricas tridimensionales con *trunking* bajo diferentes algoritmos de encaminamiento y patrón de tráfico adverso.

Los datos obtenidos no aportan conclusiones adicionales en cuento al uso del *trunking* de las extraidas del caso bidimensional, sin embargo, muestran que el número de dimensiones en las que se emplee *trunking* no afecta a los resultados obtenidos.

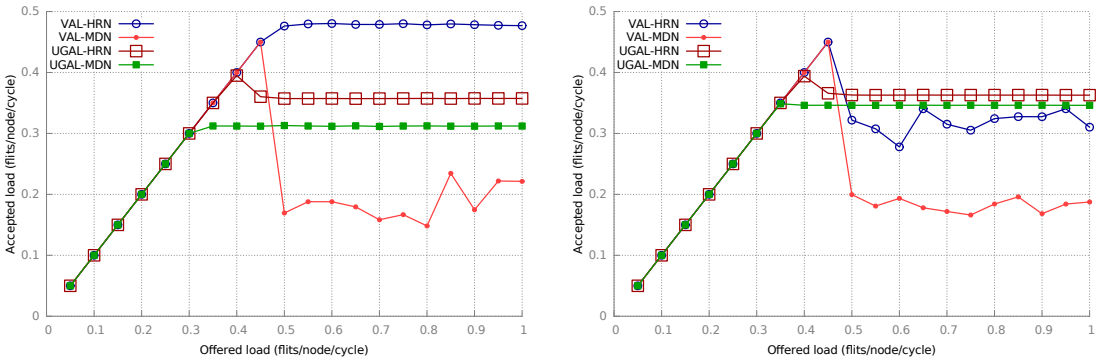
3.4. Problemas de congestión

Como se puede observar en las ilustraciones 3.5(b), 3.15(b), 3.18(b) el *throughput* obtenido bajo encaminamiento VAL y un patrón de tráfico adverso padece de un grave problema de congestión en redes correctamente balanceadas. La congestión de la red se muestra en las gráficas citadas anteriormente como una fuerte caída en el *throughput* obtenido en la red por encima de cierta carga inyectada.

Dado que el patrón de tráfico adverso implementado previamente es el de peor caso, se ha probado implementando otro patrón de tráfico adverso que se puede dar de forma más natural en una máquina real, al mismo se le ha denominado `HamiltonianRingNeighbor` (HRN) y consiste en que todos los nodos de cómputo de un *router* (R_i) intentan enviar tráfico a nodos de cómputo conectados al R_{i+1}. Este patrón de tráfico realiza un ciclo Hamiltoniano sobre todos los *routers* de la red. Este patrón de tráfico tiene el mismo *throughput* teórico que el implementado anteriormente,

aunque el número de saltos ya no es el mismo en todas las rutas, por lo que la latencia media de los paquetes va a ser menor que en el otro patrón de tráfico adverso.

Para evaluar el comportamiento del nuevo patrón de tráfico se lanzan de nuevo simulaciones con los mismos parámetros mostrados en las tablas 2.1 y 3.1 aunque añadiendo este patrón de tráfico y simulando 3 dimensiones. A continuación se muestran las gráficas que representan los resultados obtenidos para ambas redes simuladas, se muestra solamente el encaminamiento VAL que es el que padece estos problemas de congestión.



(a) N2 (b) N3

Ilustración 3.19: Evaluación del problema de congestión en redes de 2 y 3 dimensiones.

Tras un breve estudio del problema presentado, no se ha concluido una explicación al motivo por el que en redes cuadradas bidimensionales un tráfico padece el problema y el otro no. Asimismo, en redes asimétricas el problema se presenta para ambos tráficos al igual que en redes tridimensionales. Se puede intuir que aumentar la longitud de las rutas favorece la aparición de esta problemática, pero un estudio en detalle se deja como trabajo futuro.

A continuación se comentan las conclusiones extraidas del trabajo mostrado a lo largo de las secciones anteriores.

3.5. Conclusiones

En primer lugar, se ha analizado de forma teórica el *throughput* que es posible obtener en una red que implemente esta topología correctamente balanceada y se ha implementado dicha red en un simulador de forma correcta al devolver los mismos resultados.

En segundo lugar, se han determinado las condiciones bajo las que una red FB asimétrica (desbalanceada) recupera su correcto dimensionamiento con *trunking*. Posteriormente se ha implementado y validado el uso de topologías con *trunking* de forma satisfactoria.

En tercer lugar y como conclusión principal se puede extraer que el uso de *trunking* es un método válido para mitigar las consecuencias negativas que se producen al construir una topología FB asimétrica.

4 Caso 2: Dragonfly

En este capítulo se realiza un estudio de las redes basadas en la topología conocida como DF. Tras analizar sus propiedades topológicas y su encaminamiento, se analiza y evalua con resultados satisfactorios el uso del *trunking* en la misma. Además, se relaciona con el capítulo anterior debido a la ausencia del problema de congestión que se mostraba anteriormente si se fija una concentración menor de nodos de cómputo por *router*.

La red DF, al igual que la FB surge con la transición hacia redes de alto grado. Este tipo de redes requieren cables "largos" y en base a la teoría de que el coste de una red está dominado por el número de cables y en particular por los "largos", el número de los mismos debe ser minimizado para realizar una red eficiente en coste. La topología DF se presenta [KDSA08] como una topología que usa un grupo de *routers* de alto grado como un "*router* virtual" que incrementa el radio efectivo de la red. Reduciendo en número de cables globales, una DF reduce el coste en un $20\,\%$ respecto a la FB [KDSA08].

4.1. Topología Dragonfly

Topológicamente hablando es una red directa dividida jerárquicamente en dos niveles. La topología se compone de b grupos compuestos por a *routers* cada uno y a los que se conectan p nodos de cómputo por *router*. Los *routers* pertenecientes a un grupo (primer nivel) se conectan mediante enlaces "locales" cortos, eléctricos y baratos. Los diferentes grupos (segundo nivel) se conectan mediante enlaces "globales" largos, ópticos y caros. Adicionalmente a estos parámetros, recientemente se ha propuesto [CVB14] añadir los siguientes para definir de forma precisa la topología:

- topología local: patrón de conectividad de los *routers* dentro de un grupo,
- topología global: patrón de conectividad entre los diferentes grupos, y
- distribución de los enlaces globales: router al cual se conecta cada enlace global.

El diámetro de una red con esta topología es dependiente del diámetro de la topología global y local empleadas. En el caso estudiado, en el que se emplean grafos completos para ambas topologías, el diámetro es 3.

Una implementación conocida de una red DF es la red empleada en el `Cray XC30` [FBR+12], en la cual se emplea un grafo completo como topología global y un grafo de Hamming bidimensional (red FB de dos dimensiones) para la local.

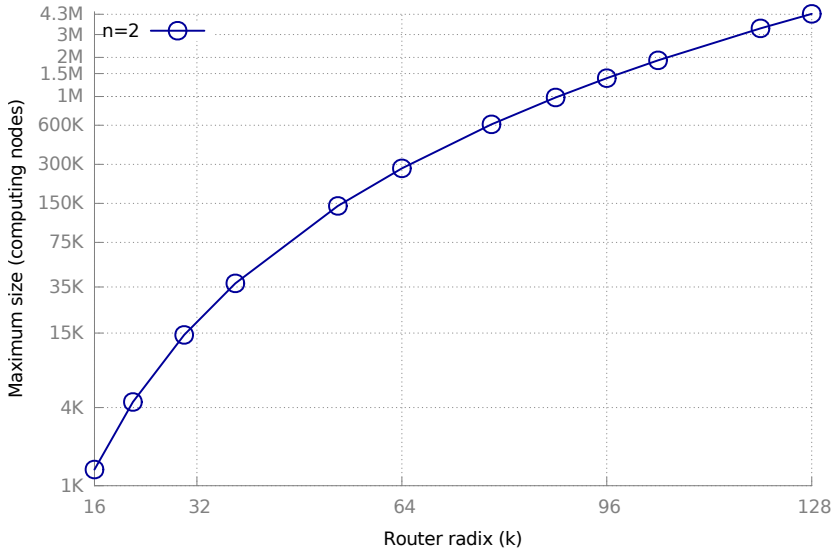


Ilustración 4.1: Escalabilidad de la red DF expresada como el número máximo de nodos de cómputo (N) en función del grado de los *routers* empleados en su construcción.

El número de nodos (N) que pueden ser conectados en una red con esta topología se muestra en la ilustración anterior frente al grado (número de puertos) de los *routers* empleados.

Tras tener una visión clara de la topología sobre la que se trabaja en este capítulo, a continuación se realiza un análisis teórico del *throughput* que se puede obtener en una red basada en ella.

4.1.1. Análisis teórico del *throughput*

Lo primero que pasa a estudiarse en esta subsección son los patrones de tráfico benignos y adversos. Estos son respectivamente, el patrón de tráfico uniforme aleatorio y el conocido como "adversarial +1". El patrón de tráfico uniforme aleatorio es un tráfico que balancea la carga entre los enlaces de la red. Por el contrario, el patrón de tráfico de peor caso se produce cuando cada nodo de cómputo perteneciente al grupo G_i envía tráfico a un nodo de cómputo aleatorio conectado a un *router* perteneciente al grupo G_{i+1}. Con este patrón de tráfico y bajo encaminamiento mínimo, todos los nodos de cómputo en cada grupo G_i intentan enviar tráfico al grupo G_{i+1} por el mismo enlace.

En la ilustración anterior se muestra un ejemplo de origen y destino que seguiría un paquete bajo el patrón de tráfico adverso y la ruta mínima elegida por el cual se enviaría el paquete. La ilustración también se va a tomar como ejemplo para los cálculos mostrados a lo largo de esta subsección.

En primer lugar, teniendo en cuenta encaminamiento mínimo y bajo tráfico uniforme, los enlaces dentro del grupo no suponen un cuello de botella, entonces se considera el tráfico que tiene como origen un nodo de cómputo perteneciente a un grupo (por ejemplo G_1 en la ilustración 4.2) y destino un nodo de cómputo perteneciente a otro grupo (G_3). La probabilidad de que el nodo origen

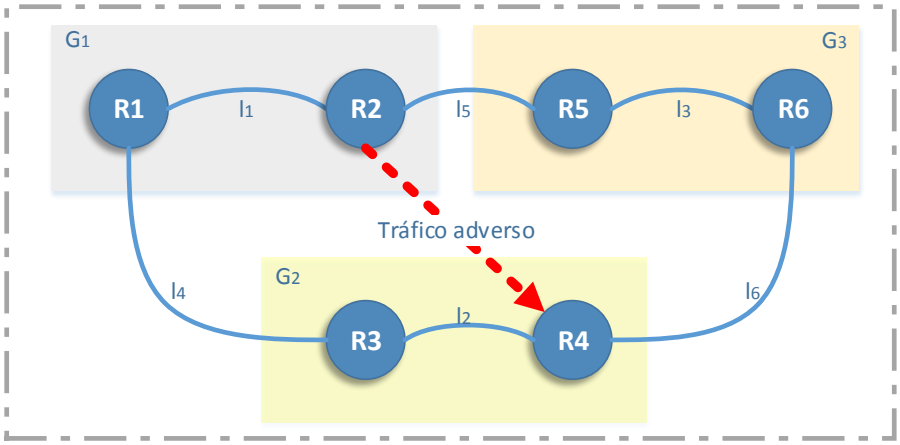


Ilustración 4.2: Representación del tráfico adverso en DF y esquema usado de ejemplo durante los cálculos teóricos.

perteneciente a G_1 envíe tráfico a G_3 es $1/3$. Puesto que en cada grupo hay 2 nodos de cómputo, el tráfico entre una pareja de grupos (G_1 y G_3 en este ejemplo) es $2 * \frac{1}{3}$, que al ser menor que 1 y existiendo un enlace entre cada pareja de grupos queda garantizado que bajo tráfico uniforme y encaminamiento mínimo, todos los nodos de cómputo pueden inyectar la máxima carga posible. Por otro lado, bajo tráfico adverso producido por ejemplo entre G_2 y G_3, todos los nodos de cómputo pertenecientes a G_2 quieren enviar tráfico a nodos de cómputo pertenecientes a G_3 y lo tienen que hacer atravesando el enlace l_6, por lo que sólo podría inyectar a plena carga uno de los nodos del grupo G_2, o lo que es lo mismo, bajo tráfico adverso y encaminamiento mínimo en el ejemplo mostrado el *throughput* obtenido seria de $0{,}5$, obtenido de forma general como $\frac{1}{ac}$).

En tercer lugar, teniendo en cuenta encaminamiento Valiant, para obtener el *throughput* máximo que se podría obtener bajo un patrón de tráfico uniforme, se muestra el tráfico que atravesaría el enlace l_5 de la ilustración anterior. El tráfico que atravesaría dicho enlace de forma no-minima sería $\frac{1}{b-1} * \frac{1}{b} * (b-2) * (ac)$ y la parte del tráfico que lo atravesaría de forma mínima es $(\frac{ac}{b-1})$, si se suman, tras sumar ambos tráficos y teniendo en cuenta que existe un enlace entre ambos grupos, se puede determinar que el *throughput* teórico obtenido se calcula mediante $\frac{1}{\frac{ac}{b-1} * \frac{b-2}{b} + 1}$. Y para terminar, bajo un patrón adverso, el tráfico que atraviesa el enlace l_5 es el tráfico de G_2 que ha elegido a G_1 como intermedio $(\frac{ac}{b-1})$ y el tráfico con origen G_1 que ha elegido a G_2 como intermedio que responde a la misma expresión, el *throughput* teórico sería $\frac{1}{2\frac{ac}{b-1}}$.

Los resultados teóricos obtenidos en esta subsección se muestran como límites teóricos en las gráficas obtenidas de las simulaciones iniciales presentadas en la siguiente subsección.

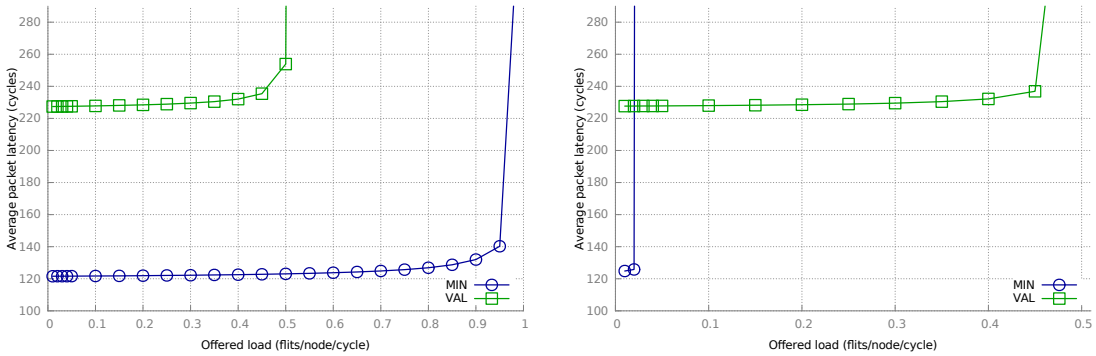
4.1.2. Evaluación mediante simulación

Para evaluar la implementación realizada de la topología y de los algoritmos de encaminamiento se realizan una serie de simulaciones iniciales para contrastar los resultados frente a los fijados de forma teórica. Los parámetros usados son la combinación de los de la tabla 2.1 y estos:

Parámetro	Valor	Unidades / Explicación
topology	dragonfly	nombre de la topología
k	$\{a,b\}$	dimensionamiento de la DF
a	5	routers por grupo
b	11	grupos
c	2	nodos por router
h	2	enlaces globales por grupo
num_vcs	2 en MIN y 3 en VAL y UGAL	numero de canales virtuales por canal físico
traffic	{uniform, dfglobaladversal}	patrón de tráfico
threshold	-16	umbral para UGAL

Tabla 4.1: Parámetros particulares usados en las simulaciones inciales de la red DF.

Tras procesar los datos obtenidos de estas simulaciones iniciales se obtienen las gráficas de latencia de paquete y *throughput* mostradas a continuación, de las que se puede concluir que la implementación realizada es correcta puesto que se ajustan a la teoría mostrada.



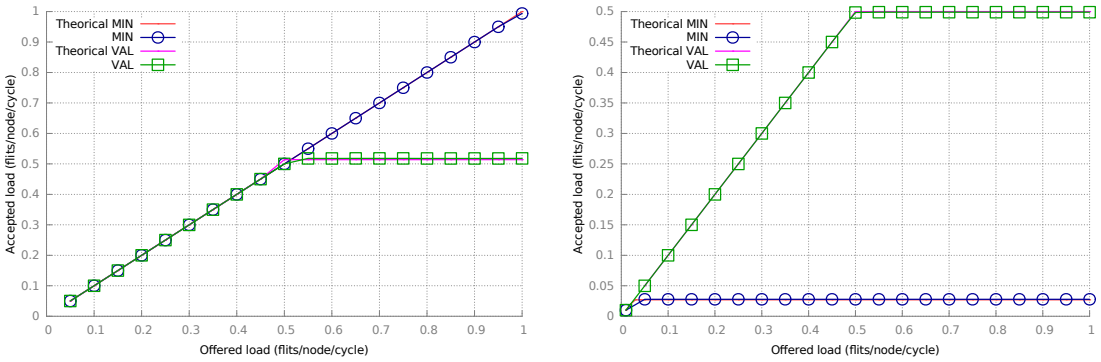
(a) Uniform

(b) Adverso

Ilustración 4.3: Latencia de la red DF bajo un patrón de tráfico benigno y adverso.

4.2. Encaminamiento

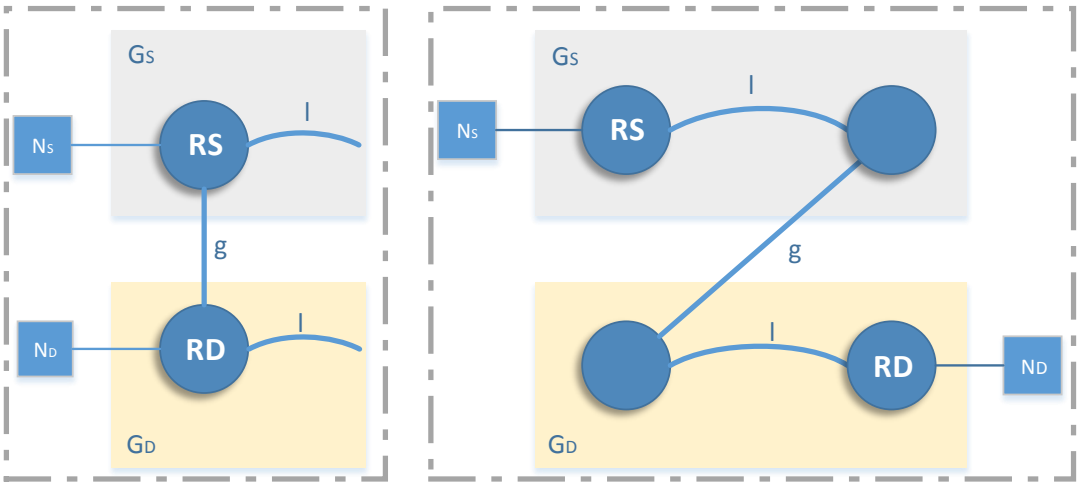
Después de haber tratado la topología de una red DF, a continuación se presenta la manera de determinar los caminos entre un origen y un destino, es decir, el encaminamiento. El encaminamiento de un paquete en una red DF requiere un salto del nodo de cómputo origen (N_S) al *router* al que está conectado (R_S), cero o un salto a un *router* del mismo grupo (G_S) (conocidos como saltos locales), un salto a un *router* de otro grupo (conocidos como saltos globales), cero o un salto



(a) Uniform (b) Adverso

Ilustración 4.4: *Throughput* de la red DF bajo un patrón de tráfico benigno y adverso.

locales en el grupo destino (G_D), y un salto final del último *router* (R_D) atravesado por el paquete al nodo de cómputo destino (N_D).



(a) global (b) local - global - local

Ilustración 4.5: Representación de dos tipos de ruta que puede seguir un paquete en una red DF..

4.2.1. Encaminamiento *oblivious*

El encaminamiento mínimo de un paquete en una red DF, teniendo en cuenta la nomenclatura mostrada previamente, se puede expresar mediante un algoritmo de tres pasos:

Paso 1: Si $G_S \neq G_D$ y R_S no tiene un enlace global a G_D, encaminar localmente (dentro de G_S) de R_S a R_A, siendo R_A un *router* perteneciente a G_S con un enlace global a G_D.

Paso 2: Si $G_S \neq G_D$, atravesar el enlace global entre R_S y R_B, siendo R_B un *router* perteneciente a G_D.

Paso 3: Si $R_B \neq R_D$, encaminar localmente (dentro de G_D) de R_B a R_D.

Encaminar de forma no mínima permite balancear la carga bajo un patrón de tráfico adverso y para ello se puede hacer uso del algoritmo de Valiant [VB81] y encaminar inicialmente los paquetes a un *router* intermedio (R_I). Este encaminamiento no mínimo se puede ver como dos encaminamientos mínimos: uno de R_S a R_I y otro de R_I a R_D. Empleando este algoritmo de encaminamiento se randomiza cualquier patrón de tráfico y lo asemeja a un patrón aleatorio uniforme.

Para que el encaminamiento esté libre de *deadlock* se puede hacer uso de múltiples canales virtuales [DS87], dos para encaminamiento mínimo y tres para no mínimo. En la ilustración XX queda representado el uso de éstos sobre los encaminamientos presentados.

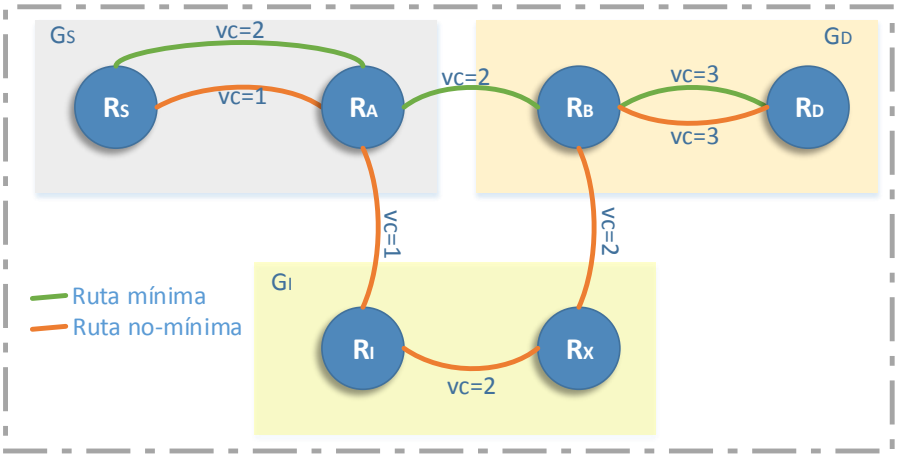


Ilustración 4.6: Representación de rutas mínimas y no-mínimas en la red DF y sus canales virtuales.

4.2.2. Encaminamiento adaptativo

Para tener en cuenta el estado de la red a la hora de encaminar los paquetes, se usan algoritmos de encaminamiento denominados adaptativos. Para probar un encaminamiento de este tipo sobre la red DF se ha optado por UGAL [Sin05]. Este algoritmo decide paquete a paquete si ha de seguir un enrutamiento MIN o VAL para minimizar el retardo estimado para cada paquete. El retardo de una ruta es estimado mediante el producto del número de saltos (H) y la longitud de la cola asociada al puerto (Q). UGAL encamina un paquete de forma mínima si la siguiente inecuación se satisface:

$$H_{min} \cdot Q_{min} \leq H_{val} \cdot Q_{val} + T \tag{4.1}$$

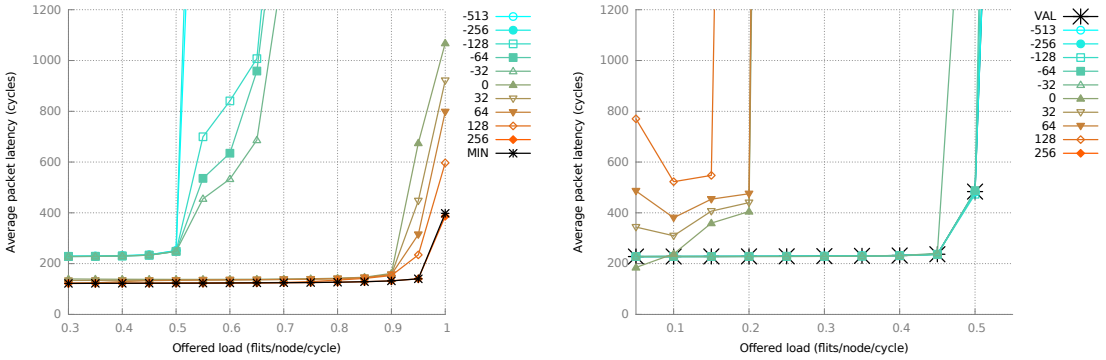
donde T es una constante usada para filtrar cargas temporalmente no balanceadas (vistas como fluctuaciones de la longitud de las colas) y permite balancear el rendimiento entre patrones de tráfico benignos y adversos. La constante *threshold* (T) no forma parte del algoritmo original de UGAL y se ha añadido a la implementación realizada del mismo con el objetivo de llegar a un compromiso entre la "rapidez" con la que el algoritmo responde ante un patrón de tráfico adverso y el rendimiento.

En la DF, según [JKD09] los enlaces globales limitan el ancho de banda de la red y dominan la latencia de red, por lo que la decisión de UGAL (ecuación 4.1) se puede simplificar usando solamente el número de saltos globales: uno para encaminamiento mínimo y dos para encaminamiento no mínimo, quedando la inecuación a satisfacer para que UGAL encamine mínimo tal como se muestra a continuación:

$$Q_{min} \leq 2 \cdot Q_{val} + T \tag{4.2}$$

En esta sección se pretende obtener el valor adecuado para dicho umbral. Para ello, se ha lanzado un barrido de simulaciones con los valores de los parámetros generales mostrados en la tabla 2.1 y de forma particular para esta prueba se han usado también la definición del `threshold` al siguiente conjunto $\{-513, -256, 0, 4, 8, 16, 32, 64, 128, 256\}$. Los valores de T que tiene sentido evaluar se encuentran comprendidos en el siguiente intervalo $[-1 \cdot (2 \cdot vc\_buff\_size + 1), 1 \cdot vc\_buff\_size]$, cualquier valor fuera del mismo no aporta información adicional. El valor mínimo de este intervalo fuerza a UGAL a hacer misrouting para todos los paquetes y, por el contrario, el valor máximo a realizar un encaminamiento mínimo.

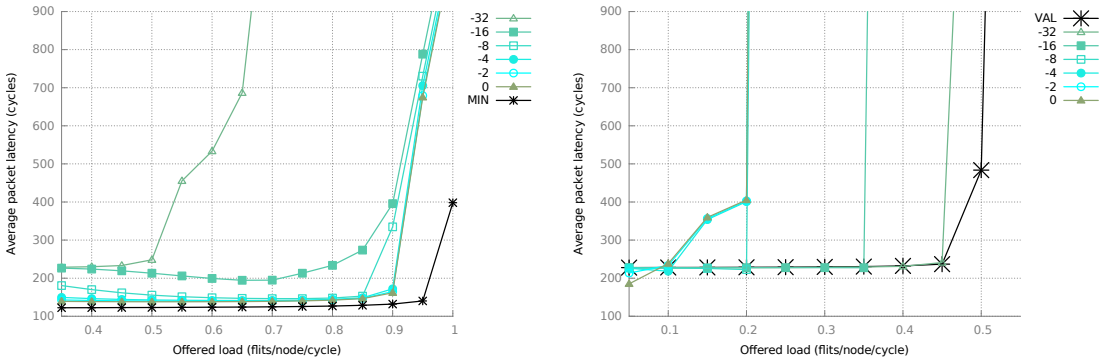
Es evidente que el valor del umbral que minimiza la latencia y maximiza el *throughput* va a ser contrario para patrones de tráfico benignos y adversos. Tras analizar las gráficas resultantes del barrido por la constante T mostradas a continuación, se ha concluido que el detalle de las mismas no era el suficiente y se han lanzado nuevas simulaciones con valores de T comprendidos en el rango $[-32, 0]$.



(a) Uniform (b) DFGlobalAdversal

Ilustración 4.7: Evaluación del *threshold* para UGAL-DF en diferentes patrones de tráfico.

En base a las gráficas resultantes del segundo barrido realizado (véase ilustración 4.8), se puede ver que -32 queda descartado por su rendimiento en tráfico uniforme dado que satura con un valor de carga inyectada bajo y el resto de valores están próximos. Revisando el resultado que se obtiene



(a) Uniform (b) DFGlobalAdversal

Ilustración 4.8: Evaluación del *threshold* para UGAL-DF en diferentes patrones de tráfico con un rango de T más acotado.

bajo tráfico adverso con el resto de valores elegidos para T, se puede ver que el valor que llega a un compromiso es -16.

A partir de este momento el resto de simulaciones realizadas con la red DF usarán el valor de umbral determinado anteriormente para el algoritmo de encaminamiento UGAL, es por esto que dicho valor se muestra como un parámetro general a usar en las simulaciones en la tabla 4.1.

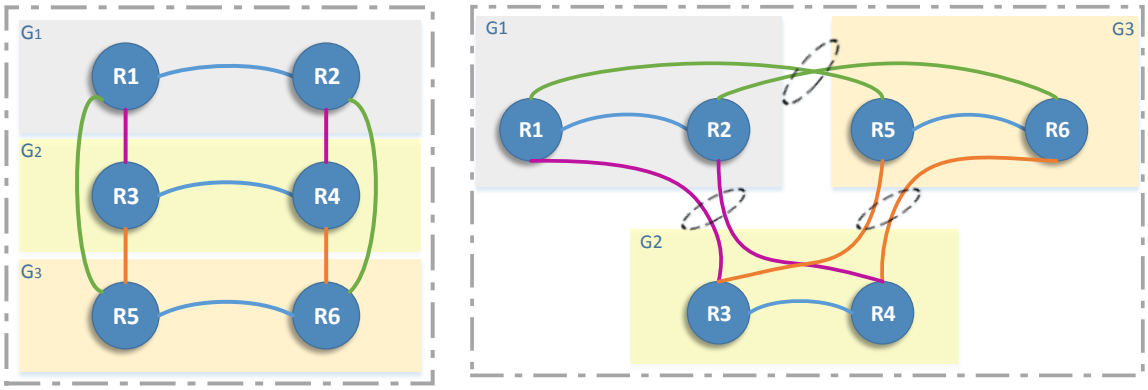
Tras analizar el rendimiento obtenido con UGAL, se puede ver que no es el esperado. Como se comenta en [KDSA08], la información que usa el algoritmo para encaminar es local y esto no es óptimo, en el citado artículo se proponen encaminamientos alternativos que ofrecen mejores resultados y posteriormente se ha presentado un análisis más exhaustivo en [JKD09].

Tras estudiar el encaminamiento en la topología estudiada en este capítulo, a continuación se trata el uso de múltiples enlaces de forma paralela entre parejas de elementos, ya sean grupos o *routers*.

4.3. Trunking en la Dragonfly

En esta sección se considera el *trunking* en la topología DF. En el artículo que presenta originalmente la topología se da a entender que la red puede tener *trunking* pero no indaga en el tema. El factor de *trunking* en una topología se refiere al numero de enlaces paralelos que se emplean para incrementar el ancho de banda agregado, incrementando también el número de puertos usados en el *router*. Si se plantea un esquema en el que los grupos se dibujan como filas y los *routers* de un grupo como columnas, puede verse que un grafo de Hamming responde a la definición de una topología DF con un nivel de *trunking* igual al número de *routers* por grupo (véase ilustración siguiente). Una justificación formal puede encontrarse en [CVB14].

En la topología DF se pueden dar dos tipos de *trunking*, denominados “local” y “global”. Respectivamente, uno se refiere a enlaces paralelos entre pares de routers pertenecientes a un mismo grupo,



(a) Tradicional (b) Grafo de Hamming

Ilustración 4.9: DF con *trunking* global 2 representada de dos formas para facilitar su comprensión.

conocido típicamente como LAG y el otro es el número de enlaces globales entre cada pareja de grupos.

Teniendo en cuenta que la mayoría de las rutas bajo encaminamiento mínimo son "local-global-local", una red DF estará balanceada si cuenta con el doble de enlaces locales que de globales, por lo que se puede determinar que el número de enlaces globales por router es $h = \frac{a-1}{2}$. Sustituyendo el h calculado previamente en el cálculo del número de grupos que responde a la expresión $b = 1 + ah$ se obtiene $b = \frac{1}{2}(a^2 - a + 2)$. Sin tener en cuenta el *trunking*, en una red DF con menos grupos ($b < \frac{1}{2} * (a^2 - a + 2)$, habrá menor número de enlaces globales, los cuales se convertirán en el cuello de botella. Y por el contrario, en una red con un número de grupos mayor, los enlaces locales serán el cuello de botella.

Para re-balancear una red con más o menos grupos de los calculados según la expresión anterior y un a dado, se debe usar *trunking* global en caso de contar con menos grupos y *trunking* local en el caso contrario. Ambos casos han sido evaluados en posteriores subsecciones.

La implementación del *trunking* local se ha realizado igual que en la topología *FB* (véase sección 3.3.3). La implementación del *trunking* global no requiere de mecanismos adicionales y simplemente se han conectado los grupos siguiendo la distribución "*Extended Circulan-based Arrangement*" propuesta recientemente [CVB14].

4.3.1. Experimento 1: *Trunking* global

Como ya se ha comentado, este tipo de *trunking* permite re-balancear una red con menos grupos de los que fijan las expresiones iniciales. Esto permite diseñar pensando en una futura ampliación del sistema (y su correspondiente red) y construir inicialmente sistemas más pequeños.

Al aumentar el número de enlaces globales debido al *trunking*, la probabilidad de que las rutas sean "local-global-local" se reduce, por lo que la condición de balanceo usada ya no es exacta.

Recientemente han sido estudiadas las condiciones de balanceo para una topología DF empleando *trunking* global y se resumen en cumplir la siguiente expresión con un $\alpha \in [\frac{1}{2}, 1]$

$$ah = t(b-1) = a(a-1)\alpha \tag{4.3}$$

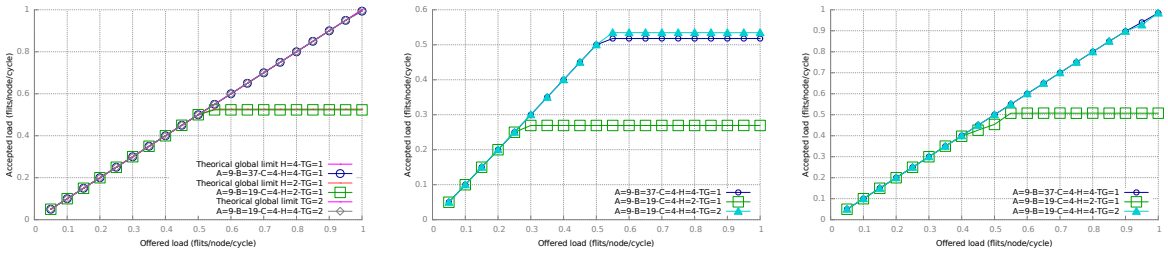
siendo t el valor del *trunking* global.

En base a que en la red hay b grupos que contienen a *routers* cada uno y que se emplean un número de enlaces t entre cada par de grupos, el número de enlaces globales por router en el caso de emplear *trunking* global se puede calcular mediante

$$h = \frac{t(b-1)}{a} \tag{4.4}$$

Partiendo una red inicial con $a = 9$ *routers* en cada grupo y $b = 37$ grupos sin *trunking*, se necesita un $h = 4$ para conectar los grupos. Al establecer un *trunking* global $t = 2$, según la ecuación 4.3 se obtiene un número de $b = 19$ grupos.

A continuación se muestran los resultados de las simulaciones realizadas divididos por algoritmo de encaminamiento y patrón de tráfico inyectado. En dichas gráficas se muestra la carga aceptada por los nodos de cómputo en la red propuesta como base, en una red con menor número de grupos sin *trunking* en la que el número de enlaces globales va a ser el cuello de botella y en otra con el mismo número de grupos empleando *trunking* global para re-balancear la misma. Asimismo, se representan los límites teóricos de cada caso según las exprexiones obtenidas en la subsección 4.1.1.

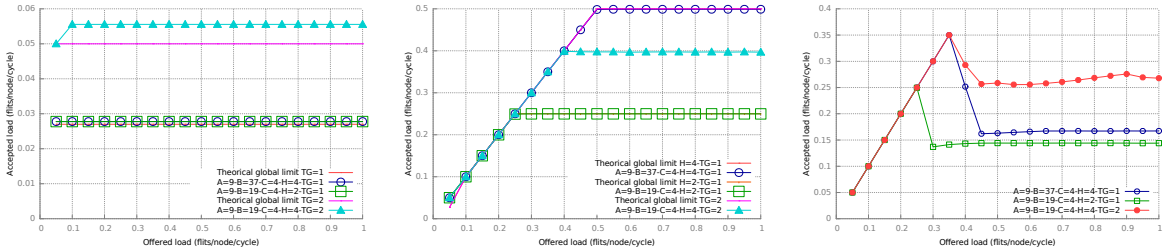


(a) MIN (b) VAL (c) UGAL

Ilustración 4.10: *Throughput* de la red DF evaluando *trunking* global bajo tráfico uniforme y dividido por algoritmos de encaminamiento.

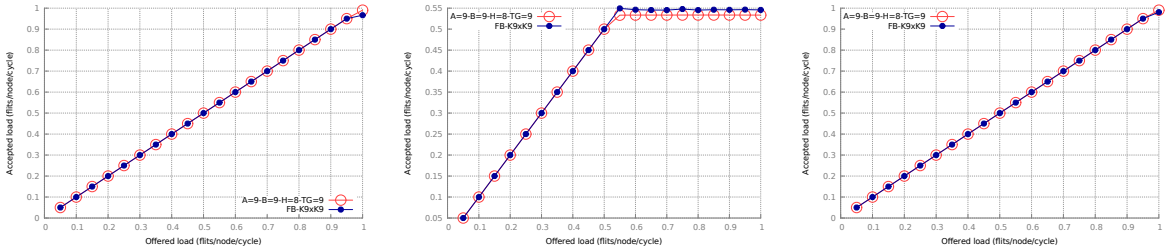
Analizando las curvas de resultados presentadas y comparando los valores obtenidos con los valores teóricos representados en las mismas, destaca la gráfica 4.11(b). Se plantea como hipótesis que los enlaces locales limitan el *throughput* obtenido, sin embargo no se aborda la idea en el alcance del presente trabajo.

Como se ha comentado previamente, el caso extremo de *trunking* global es aplicarlo en un factor igual al número de *routers* por grupo ($t = a$). Bajo estos parámetros, la topología mostrada es isomorfa a un grafo de Hamming o lo que es lo mismo, a la topología FB. A continuación se



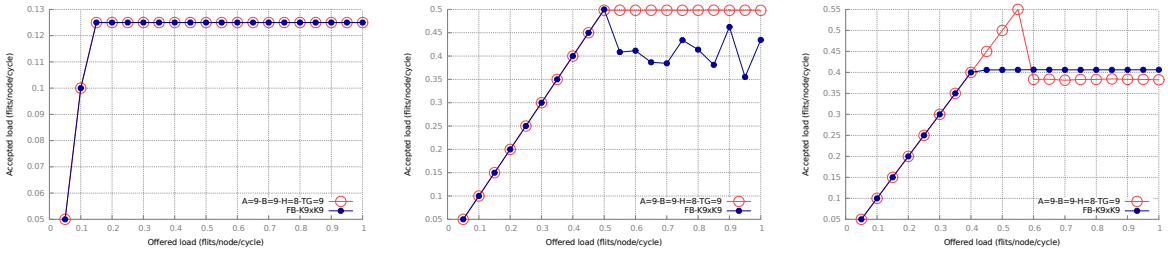
(a) MIN (b) VAL (c) UGAL

Ilustración 4.11: *Throughput* de la red DF evaluando *trunking* global bajo tráfico adverso y dividido por algoritmos de encaminamiento.



(a) MIN (b) VAL (c) UGAL

Ilustración 4.12: *Throughput* obtenido evaluando una DF con *trunking* global extremo bajo diferentes algoritmos de encaminamiento y patrón de tráfico uniforme.



(a) MIN (b) VAL (c) UGAL

Ilustración 4.13: *Throughput* obtenido evaluando una DF con *trunking* global extremo bajo diferentes algoritmos de encaminamiento y patrón de tráfico adverso.

muestran gráficamente los resultados obtenidos de una comparación de ambas redes bajos los parámetros comentados.

Los resultados obtenidos, tal como se esperaba, muestran un comportamiento muy similar de ambas redes. Si bien es cierto, que el cálculo de c para una DF con *trunking* global no es exacto, puesto que se podría llegar a emplear un c mayor a h. Aunque, tal como se muestra en la ilustración 4.13(b) aumentar el tamaño de la red fijando un $c = 9$ (en este caso) supondría padecer los mismos problemas de congestión tratados en la sección 3.4.

4.3.2. Experimento 2: Trunking local

Como se ha comentado, este tipo de *trunking* permite re-balancear una red con más grupos de los que fijan las expresiones iniciales. Sin embargo, este *trunking* no aporta ventajas adicionales en cuando a versatilidad de diseño.

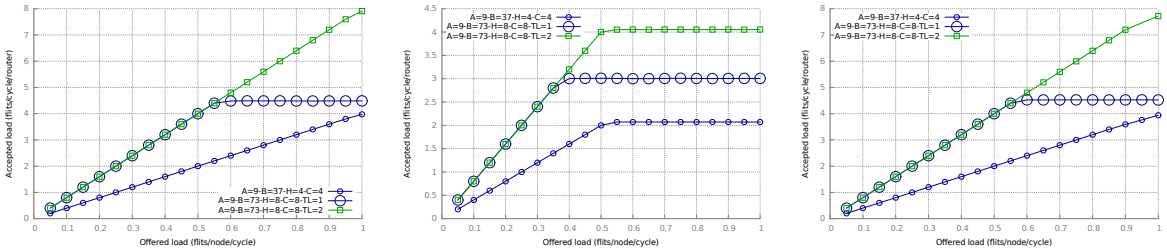
La condición de balanceo de la red no cambia respecto a la topología orinal, sin embargo hay que introducir el factor de *trunking* local t en las ecuaciones que definen la red, concretamente, los parámetros de red elegidos han de satisfacer que

$$t(a-1) = 2h \tag{4.5}$$

siendo t el número de enlaces entre cada par de *routers* pertenecientes al mismo grupo.

Partiendo una red inicial con $a = 9$ *routers* en cada grupo y $b = 37$ grupos sin *trunking*, se trataría de una red balanceada con un $c = 4$. Si se estableciese *trunking* local $t = 2$, en base a la ecuación anterior se obtiene un $h = 8$ con el que se puede disponer de $b = 73$ grupos.

A continuación se muestran los resultados de las simulaciones realizadas divididos por algoritmo de encaminamiento y patrón de tráfico inyectado. En dichas gráficas se muestra la carga aceptada por los *routers* (véase explicación en la página 23) en la red propuesta como base, en una red con mayor número de grupos sin *trunking* en la que el número de enlaces locales va a ser el cuello de botella y en otra con el mismo número de grupos empleando *trunking* local para re-balancear la misma.

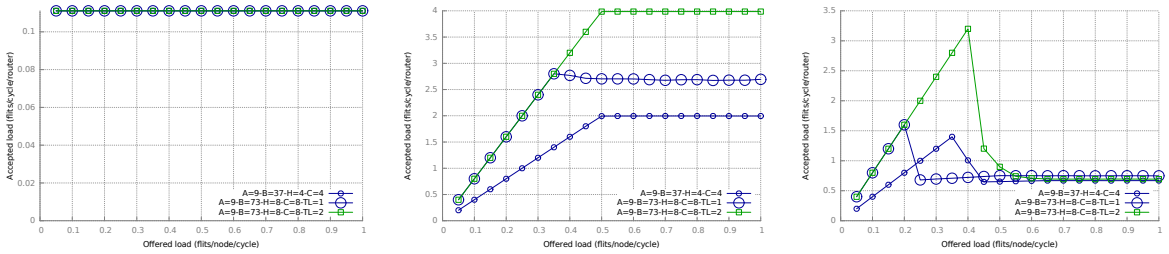


(a) MIN (b) VAL (c) UGAL

Ilustración 4.14: *Throughput* obtenido evaluando una DF con *trunking* local bajo diferentes algoritmos de encaminamiento y patrón de tráfico uniforme.

En base a los resultados obtenidos, el uso del *trunking* local para re-balancear una red limitada por los enlaces locales es válido. De forma práctica, este tipo de *trunking* permite dimensionar un sistema y en un momento inicial construirlo con la mitad de concentración y la mitad de enlaces locales pero no suele ser una forma habitual de construir un sistema.

A continuación se comentan las conclusiones extraídas del trabajo realizado con esta topología mostrado a lo largo de las secciones anteriores.



(a) MIN (b) VAL (c) UGAL

Ilustración 4.15: *Throughput* obtenido evaluando una DF con *trunking* local bajo diferentes algoritmos de encaminamiento y patrón de tráfico uniforme.

4.4. Conclusiones

En primer lugar, se ha analizado de forma teórica el *throughput* que es posible obtener en una red que implemente esta topología correctamente balanceada y se ha implementado dicha red en un simulador de forma correcta al devolver los mismos resultados.

En segundo lugar, se ha podido comprobar que no llegar a una situación de saturación (fijando un $c = h$ en situaciones de *trunking* global máximo) favorece la ausencia del problema de congestión presentado en la sección 3.4.

En tercer lugar, se han estudiado los tipos de *trunking* aplicables a la topología DF y se puede concluir que la aplicación de éstos a la misma, permite re-balancear situaciones limitadas por los correspondientes enlaces.

En cuarto lugar y visto desde un punto de vista más práctico, el *trunking* global es una manera de construir inicialmente un sistema más reducido en base a un diseño grande. Estos diseños además, mientras tienen menos grupos y más enlaces globales, presentan una mayor diversidad de caminos y una menor latencia promedio.

5 Conclusiones y trabajo futuro

5.1. Conclusiones

El presente Trabajo Fin de Máster tenía como objetivo principal la evaluación del simulador `BookSim` y el análisis de su utilidad como herramienta de simulación de las futuras redes a proponer para el prototipo del proyecto `Mont-Blanc`. El objetivo del trabajo se ha llevado a cabo y se ha concluido que el simulador permite realizar las tareas necesarias para la satisfacción de los requisitos impuestos en el marco de la investigación conjunta entre el BSC y la UC.

Durante la evaluación del simulador se ha llevado a cabo la correcta implementación de varias redes basadas en topologías directas que hacen uso de *routers* de alto grado en su construcción. A lo largo de su evaluación mediante la herramienta a analizar en este trabajo se han extraído unas conclusiones adicionales que se presentan a continuación.

Se ha determinado que el *trunking* mitiga las consecuencias negativas de construir una red FB asimétrica que inicialmente está desbalanceada. Adicionalmente, durante el análisis realizado se han expresado formalmente las condiciones bajo las que esta solución es aplicable y se ha validado su uso mediante sendas simulaciones.

Se ha concluido que aunque hay dos tipos de *trunking* aplicables en una topología DF desbalanceada que permiten corregir esta situación, desde el punto de vista práctico solamente es interesante el *trunking* global. Además, este tipo de `trunking` aplicado en una fase inicial de la construcción de una red planteada en varias fases presenta ventajas como una mayor diversidad de caminos y una menor latencia en promedio.

5.2. Trabajos futuros

Aunque se han cumplido los objetivos fijados para este trabajo, debido a la aparición de comportamientos inesperados durante la realización del mismo y a la magnitud de diversas opciones que se desvían del objetivo principal, se proponen las siguientes lineas de investigación futura:

- Indagar en el problema de congestión que se presenta con encaminamiento no-mínimo superado el punto de saturación en las topologias basadas en grafos de Hamming cuando están correctamente dimensionadas.
- La implementación de otros algoritmos de encaminamiento para la red FB como MIN AD.

- La codificación de algoritmos de encaminamiento adaptativos que hagan uso de información remota en la red DF.
- La evaluación de diferentes políticas de selección del canal a emplear en redes con *trunking* y el impacto que éstas tienen en el rendimiento de la red.
- Obtener datos del uso de recursos por parte del simulador para redes DF.

Glosario

Términos

Local Area Network Red que conecta terminales que se encuentran dentro de un área limitada como una casa, un laboratorio o un edificio de oficinas. 1

Metropolitan Area Network Red que proporciona cobertura en un área geográficamente extensa. 1

misrouting Enviar un paquete por un canal no productivo, entendiendo estos como los que pertenecen a una ruta mínima.. 14, 18, 24, 35

Wide Area Network Red que abarca varias ubicaciones físicas. 1

Acrónimos

BSC `Barcelona Supercomputing Center`. 2, 43

CC-NUMA *Cache Coherent Non-Uniform Memory Access*. 1

CMP *Chip Multiprocessor*. 1

DF `Dragonfly`. III, V, IX–XI, 4, 29–41, 43, 44

DOR *Dimension Order Routing*. 16, 17, 22

FB `Flattened Butterly`. III, V, IX, XI, 4, 7, 8, 11, 12, 15–27, 29, 38, 43

flits *flow control digits*. 6

FLOPS *FLoating-point Operations Per Second*. 1

FSS *full-system simulator*. 4

GPU *Graphics Processing Unit*. 1

HPC *High-Performance Computing*. III, V, 1, 2

HRN `HamiltonianRingNeighbor`. 26

JSQ *join the shortest queue*. 22

LAG *Link Aggregation Group*. 22, 37

LAN *Local Area Network*. 1

MAN *Metropolitan Area Network*. 1

MDN `MultiDimNeighbor`. 13

MIN mínimo. IX, 17, 24–26, 34, 38–41

NOC *Network On-Chip*. 1, 4, 6, 11

TIC Tecnologías de la información y las comunicaciones. 1

UC `Universidad de Cantabria`. 2, 4, 43

UGAL *Universal Globally Adaptive Load-Balancing*. 17, 18, 24, 26, 32, 34–36, 38–41

VAL Valiant. 17, 24, 26, 27, 34, 38–41

WAN *Wide Area Network*. 1

Referencias

[AFRK12] B. Alverson, E. Froese, D. Roweth, and L. Kaplan. Cray xc series network. Technical report, Cray Inc, 2012. Wp-Aries01-112.

[AKPJ09] N. Agarwal, T. Krishna, Li-Shiuan Peh, and N.K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 33–42, Abril 2009.

[APM+12] Pablo Abad, Pablo Prieto, Lucia G. Menezo, Adrín Colaso, Valentin Puente, and Jose-Angel Gregorio. Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, NOCS '12, pages 99–106. IEEE Computer Society, 2012.

[BBC+08] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, Katherine Yelick, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Stephen Keckler, Dean Klein, Peter Kogge, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems peter kogge, editor & study lead, 2008.

[BYF+] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. Gpgpu-sim. http://www.gpgpu-sim.org. Disponible a 8 de Octubre de 2014.

[BYF+09] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. Analyzing cuda workloads using a detailed gpu simulator. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2009)*, pages 163–174, Abril 2009.

[CMV+10] J.M. Cámara, M. Moretó, E. Vallejo, R. Beivide, J. Miguel-Alonso, C. Martinez, and J. Navaridas. Twisted torus topologies for enhanced interconnection networks. *Parallel and Distributed Systems, IEEE Transactions on*, 21(12):1765–1778, Diciembre 2010.

[CR85] INC. Cray Research. The CRAY-2 computer system. http://archive.computerhistory.org/resources/text/Cray/Cray.Cray2.1985.102646185.pdf, 1985. Disponible a fecha de 8 de Octubre de 2014.

[CVB14] C. Camarero, E. Vallejo, and R. Beivide. Topological characterization of hamming and dragonfly netowkrs and its implications on routing. *Transactions on Architecture and Code Optimization*, 2014. Aceptado para publicación en Septiembre de 2014.

[DS87] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.*, 36(5):547–553, Mayo 1987.

[DT03a] W. Dally and B. Towles. Booksim interconnection network simulator. http://cva.stanford.edu/books/ppin/booksim-1.0.tar.gz, 2003. Consultado por última vez el 8 de Octubre de 2014.

[DT03b] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[FBR+12] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray cascade: A scalable HPC system based on a dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 103:1–103:9, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[FPP] F. Fazzino, M. Palesi, and D. Patti. Noxim: Network-on-chip simulator,. http://sourceforge.net/projects/noxim/. Disponible a fecha 8 de Octubre de 2014.

[GVB+13] M. Garcia, E. Vallejo, R. Beivide, M. Valero, and G. Rodriguez. Ofar-cm: Efficient dragonfly networks with simple congestion management. In *High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on*, pages 55–62, Agosto 2013.

[JBM+13] Nan Jiang, D.U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D.E. Shaw, J. Kim, and W.J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96, Abril 2013.

[JKD09] Nan Jiang, John Kim, and William J. Dally. Indirect adaptive routing on large scale interconnection networks. *SIGARCH Comput. Archit. News*, 37(3):220–231, Junio 2009.

[JMB+13] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally. *BookSim 2.0 User's Guide*, Mayo 2013. Disponible a fecha de 8 de Octubre de 2013.

[KBD07] John Kim, James Balfour, and William Dally. Flattened butterfly topology for on-chip networks. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 172–182. IEEE Computer Society, 2007.

[KDA07] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: A cost-efficient topology for high-radix networks. *SIGARCH Comput. Archit. News*, 35(2):126–137, Junio 2007.

[KDSA08] J. Kim, W.J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pages 77–88, Junio 2008.

[KOPS10] S. Kamil, L. Oliker, A. Pinar, and J. Shalf. Communication requirements and interconnect optimization for high-end scientific applications. *IEEE Trans. Parallel Distrib. Syst.*, 21(2):188–202, Febrero 2010.

[Meu09] H. Meuer. The future for HPC. http://www.scientific-computing.com/features/feature.php?feature_id=244, Junio/Julio 2009. Disponible a fecha de 9 de Octubre de 2014.

[Mon14] Mont-Blanc. La Universidad de Cantabria participa en el proyecto europeo Mont-Blanc. http://www.montblanc-project.eu/press-corner/news/la-universidad-de-cantabria-participa-en-el-proyecto-europeo-mont-blanc, Septiembre 2014. Disponible a fecha 8 de Octubre de 2014.

[MR09] Cyriel Minkenberg and Germán Rodriguez. Trace-driven co-simulation of high-performance computing systems using OMNeT++. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 65:1–65:8, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Nau12] Philippe Naudin. Bug 783143 - gnu time reports incorrect ram usage. https://bugzilla.redhat.com/show_bug.cgi?id=783143, Enero 2012. Disponible a 30 de Septiembre de 2014.

[Nie13] David C. Niemi. Re: time-1.7 counts rusage wrong on linux. http://lists.gnu.org/archive/html/bug-gnu-utils/2013-02/msg00020.html, Febrero 2013. Disponible a 30 de Septiembre de 2014.

[NMAPR11] Javier Navaridas, Jose Miguel-Alonso, Jose A. Pascual, and Francisco J. Ridruejo. Simulating and evaluating interconnection networks with INSEE. *Simulation Modelling Practice and Theory*, 19(1):494 – 515, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.

[Pis11] Petr Pisar. Bug 703865 - gnu time reports incorrect maximum rss. https://bugzilla.redhat.com/show_bug.cgi?id=703865, Mayo 2011. Disponible a 30 de Septiembre de 2014.

[SAKD06] S. Scott, D. Abts, J. Kim, and W.J. Dally. The blackwidow high-radix clos network. In *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pages 16–28, 2006.

[Sin05] Arjun Singh. *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Stanford University, Marzo 2005.

[STN+14] K. Swaminathan, D. Thakyal, S.G. Nambiar, G. Lakshminarayanan, and Seok-Bum Ko. Enhanced noxim simulator for performance evaluation of network on chip topologies. In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pages 1–5, Marzo 2014.

[TOPa] TOP 500. ASCI Red | TOP 500 Supercomputer Sites. http://www.top500.org/system/168753. Disponible a fecha de 9 de Octubre de 2014.

[TOPb] TOP 500. IBM Roadrunner | TOP 500 Supercomputer Sites. http://www.top500.org/system/176026. Disponible a fecha de 9 de Octubre de 2014.

[VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, pages 263–277. ACM, 1981.

[Wic13] N. Wichmann. Cray xc30 overview. http://www.nersc.gov/assets/Uploads/NERSC.XC30.overview.pdf, Noviembre 2013. Disponible a fecha de 5 de Octubre de 2014.