
*Facultad de Ciencias*

# IMPACTO DE LOS MECANISMOS DE EVITACIÓN DE BLOQUEO EN EL RENDIMIENTO DE REDES DE INTERCONEXIÓN HYPERX
## (IMPACT OF DEADLOCK AVOIDANCE TECHNIQUES IN THE PERFORMANCE OF HYPERX INTERCONNECTION NETWORKS)

**Trabajo de Fin de Máster**
**para acceder al**

**MÁSTER EN INGENIERÍA INFORMÁTICA**

**Autor: Alejandro Cano Cos**

**Directores: Cristóbal Camarero Coterillo y**
**Julio Ramón Beivide Palacio**

**Julio - 2022**

# Resumen

La evolución de los computadores en los últimos años ha traído de la mano cambios profundos en las redes de comunicaciones. Hoy en día, existen computadores compuestos por millones de procesadores capaces de realizar operaciones a una velocidad de hasta $10^{18}$ operaciones de coma flotante por segundo. Con ello, el modelo de interconexión entre los computadores paralelos debe adaptarse para satisfacer las necesidades de rendimiento de todo el sistema, sin suponer una limitación.

No obstante, sin importar los avances en la informática, existe un fenómeno común a cualquier sistema donde exista una compartición de recursos entre diferentes agentes. Este es denominado *deadlock*, y su aparición en tiempo de ejecución deja inhabilitado el sistema por completo. Una de las primeras veces que el *deadlock* es descrito data de 1968, por Edsger Dijkstra [1] donde con *deadly embrace* se describe lo que actualmente se denomina *deadlock*. Esto es un problema vigente a día de hoy, y más aún en redes de interconexión. Una red de interconexión por naturaleza es una estructura compartida por todos los elementos del sistema que están conectados, y donde los diferentes flujos de tráfico que transitan la red hacen uso colectivo de los recursos de esta. Por ello, si el *deadlock* no es contemplado en el diseño, la aparición de uno inhabilita el funcionamiento global de la red.

En este trabajo, se aborda el estudio de una topología de red moderna denominada HyperX, que tiene las capacidades suficientes para ser usada en los computadores actuales mas potentes. Adicionalmente, se han descrito un conjunto de funciones de encaminamiento para esta topología que utilizan diferentes tipos de técnicas para la evitación del *deadlock*. Entre los encaminamientos, se encuentra DAL, un encaminamiento completamente adaptativo que utiliza un canal de escape como mecanismo de evitación de *deadlock*, y Omni-WAR, que también es completamente adaptativo, pero que atraviesa los canales virtuales en orden ascendente para evitar el *deadlock*. Como objetivo, este trabajo pretende introducir la topología HyperX formalmente, su implementación en el simulador de redes Booksim2, y la evaluación de las funciones de encaminamiento propuestas mediante un conjunto de métricas. Se tomarán métricas del rendimiento en escenarios adversos y benignos.

Como conclusión, se ha deducido que los encaminamientos completamente adaptativos no mínimos son los que mejor funcionan en la HyperX, y por lo tanto la evitación del *deadlock* mediante restricción de caminos no es una buena técnica. Además, entre el uso de canal de escape y escalera, el uso de canales de escape precisa de menos recursos que la escalera. También, se ha observado que en algunas mediciones para determinados tráficos ambos mecanismos de evasión de deadlock conllevan una pérdida de fairness.

**Palabras clave:** *deadlock*, red de interconexión, HyperX, función de encaminamiento, Booksim2, DAL, Omni-WAR.

# Abstract

The evolution of computers in recent years have brought deep changes in communication networks. Nowadays, there are computers built-in with millions of CPUs able to make operations at a speed of $10^{18}$ float operations per second. Thus, the interconnection paradigm between parallel computers should fit the performance necessities of the whole system, without being a limitation.

Nevertheless, there is a common phenomenom to every system where resources are shared by different agents. Its called deadlock, and its apparition in real time blocks the system. One of the first times it was depicted was in 1968, by Edsger Dijkstra [1] where deadlock is dubbed as deadly embrace. Deadlock is a current problem, even more in interconnection networks, where resources are shared among every node connected, and the flow of packets trasversing the network makes collective use of them. Therefore, if deadlock is not taken into account since the design step, the aparition of one, nullifies the global performance of the network.

In this work, the HyperX topology, which has enough capabilities to be used in most powerful computers avaliable nowadays, is studied. In addition, a set of routing functions, which make use of different techniques of deadlock avoidance, are depicted and evaluated. Among the routings, there is DAL, a fully-adaptive routing which uses a escape path as deadlock avoidance mechanism. On the other hand, Omni-WAR is a fully adaptive routing too, but it uses distance ordered classes to avoid deadlock. The goals for this work are to introduce the HyperX topology formally, to implement it into a network simulator program called Booksim2, and evaluate the performance of all the routing functions mentioned in the work with a set of defined metrics, in adverse and benign scenarios.

To sum up, it's discerned that fully-adaptive routing is the best option to choose for HyperX, so deadlock avoidance methods should not ban routes when aiming for best performance. Furthermore, between escape channels and ordered classes, escape channels need less resources than ordered classes. Besides, it is observed in some measures that the use of one of both techniques can lead to more unfairness between terminals.

**Keywords:** deadlock, interconnection networks, HyperX, routing function, Booksim2, DAL, Omni-WAR.

# Índice general

# Índice de figuras



# Índice de tablas

# 1 Introducción

## 1.1 Estado del arte

Existen multitud de sistemas informáticos en el mundo, desde teléfonos o computadores personales compuestos generalmente por unos pocos micro-procesadores, hasta centros de Computación de Alto Rendimiento donde se hacen los despliegues de los mayores computadores del mundo. Además, existen sistemas los cuales están conectados a grandes distancias, a través de Internet, y otros que están conectados a distancia ínfima, como los cores de un procesador. Cada uno de los sistemas mencionados funciona con necesidades diferentes, y en cualquiera de ellos múltiples procesadores tienen que poder comunicarse a través de una infraestructura diseñada y dimensionada para ello. Esta infraestructura, es la red de interconexión.

Recientemente, se ha superado la barrera *Exaescale* con computadores que trabajan a una velocidad mayor a 1 Exaflop/s. *Frontier* [2] es el primer computador en superar esta barrera alcanzando hasta 1.102 Exaflop/s, haciendo uso de 8.730.112 cores. Para poder permitir la comunicación mutua a esta gran cantidad de procesadores sin crear cuellos de botella en el sistema, se ha desarrollado el concepto de redes de grado alto. En el pasado, los sistemas tenían pocos procesadores y una red pequeña con pocas conexiones por nodo eran suficientes para que el sistema alcanzara un rendimiento aceptable y balanceado. Por ejemplo, un modelo de conexión muy famoso es el *anillo* [3], donde cada nodo de la red está conectado con otros dos nodos, formando entre todos ellos una estructura conexa. Sin embargo, el problema de este tipo de redes es que el rendimiento no escala bien si subimos el número de nodos [4], lo que hizo que se replanteasen las arquitecturas de red en diferentes sistemas. Por ello, en los computadores actuales se requieren otros modelos de interconexión o topologías que estén capacitados para dar servicio a todos los nodos. En concreto, la red de interconexión de *Frontier* es una *Slingshot-11* [5] donde su topología se construye a partir de grafos completos, los cuales han tomado bastante protagonismo en el contexto de redes de alto grado.

Es interesante estudiar las propiedades de algunas de las topologías que surgen a partir de la construcción de grafos completos, como es la *HyperX* [6], también conocida como *Flattened Butterfly* [7] o Grafo de *Hamming* [8]. Estas redes constituyen una de las mejores soluciones conocidas para sistemas masivamente paralelos, y no se descarta que en el futuro tanto supercomputadores como centros de datos hagan un uso extensivo de ellas [9].

## 1.2 Objetivos del trabajo

Los objetivos de este TFM son los siguientes:

- Estudiar las características de la topología HyperX e implementarla en el simulador *Booksim2*, teniendo en cuenta las limitaciones de este.
- Estudiar y medir el impacto sobre el rendimiento y uso de recursos de diferentes tipos de técnicas de evitación del deadlock sobre una topología HyperX.
- Implementar las funciones de encaminamiento presentadas en [6], [10] para poder comparar su rendimiento y uso de recursos.

El código desarrollado se encuentra subido en un repositorio de Git [1].

## 1.3 Red de interconexión

Una *red de interconexión* es una estructura física cuya finalidad es la transferencia de mensajes entre nodos procesadores. Cada nodo puede inyectar mensajes en la red, consumir mensajes, o ambas a la vez. Comúnmente a los nodos se les denomina servidores dentro del contexto de redes de sistema. Para la definición de una red de interconexión se necesitan tres parámetros fundamentales: topología de red, control de flujo y función de encaminamiento [11].

### 1.3.1 Topología de red

Una *topología* de red define la distribución de los aspectos tangibles de esta, y las conexiones entre los elementos que la componen. Está formada por:

- Routers: elemento de una red encargado de asignar un puerto de salida a un paquete recibido desde un puerto de entrada. Un router está compuesto por:
  - Puertos: Un puerto es la conexión por la cual un router envía y recibe mensajes a través de un enlace. Un puerto esta formado por uno o varios bufferes físicos donde temporalmente se almacenan los paquetes que salen o entran.
  - Canal virtual: Recurso virtualizado que se utiliza para la generación de subredes lógicas dentro de una red de interconexión física. Los recursos físicos tienen la capacidad de simular los recursos lógicos. Un recurso, en general, puede ser un enlace o el buffer de un puerto. Sin embargo, cuando se hable de canales virtuales en este trabajo se referirá a subdivisiones del buffer de un puerto. Generalmente, todos los puertos de todos los routers de la red tienen el mismo número de bufferes y canales virtuales por puerto. Además, a un buffer (virtualizado o no) se accede en orden FIFO por lo que formalmente también se le conoce como *cola*.

[1] Link al repositorio con el código desarrollado: https://github.com/alexcano98/booksim2.git (visitado el 29/06/2022).

  - Unidades hardware encargadas de implementar la lógica de asignaciones de puertos y canales virtuales dentro del router, como: el crossbar, los allocators o la unidad de routing.

- Enlaces: elemento de la red que conecta un router a otro router o un router a un servidor a través de sus puertos y permite la transferencia de paquetes entre ambos elementos. Usualmente es un cable lo que hace de enlace y está conectado a un puerto de entrada y otro de salida.

- Servidores: elemento de la red encargado de la inyección (poner nuevos paquetes en la red) y consumo (retirar paquetes de la red) de mensajes. Cada mensaje tiene un servidor origen y un servidor destino.

Una topología especifica las relaciones entre estos tres elementos. Una forma de modelar una topología es mediante un grafo, donde: $G = (V, E)$, $V :=$ Routers, donde Routers el conjunto de routers de la red y $E :=$ Enlaces, donde Enlaces es el conjunto de enlaces de la red. Para incluir al conjunto de servidores dentro de la definición formal de una topología, es común añadir una variable $c$ denotando la concentración o el numero de servidores por router. Gracias a haber formalizado una topología en forma de un grafo se pueden utilizar métodos teoría de grafos para definir topologías:

- Método de Cayley: Un grafo de Cayley se forma mediante un grupo $G$ y un subconjunto $S \subseteq G$ donde:
  - Los vértices son los elementos del grupo $V := G$.
  - Las aristas $E$ se construyen: $(a, b) \in E \iff a^{-1}b \in S$.

- Método del producto cartesiano de grafos: El producto cartesiano de dos grafos $G_a \square G_b$ está definido como:

$$G_a \square G_b = (V_a \times V_b, E_{ab}),$$

$$E_{ab} := \{((a_1, b_1), (a_2, b_2)) | (a_1 = a_2 \wedge (b_1, b_2) \in E_b) \vee (b_1 = b_2 \wedge (a_1, a_2) \in E_a)\}.$$

  En grafos de Cayley, el resultado de aplicar el producto cartesiano de grafos, es equivalente a el grafo generado por el producto directo de grupos [12] entre los grupos que generan los grafos iniciales.

- Existen topologías como la *Dragonfly* construidas de forma jerárquica y otros métodos ad-hoc.

El grafo que representa la topología HyperX [6], [7] se puede construir a través del producto cartesiano de grafos completos. Un grafo completo es cualquier grafo $G = (V, E)$ que cumple lo siguiente:

$$\forall a, b \in V, (a, b) \in E.$$

Es decir, todos los nodos están conectados con todos. Una HyperX 1D es equivalente a un grafo completo y se puede definir como este. Un ejemplo gráfico de un grafo completo con $k = 4$ vértices se ve en la Figura 1.1.

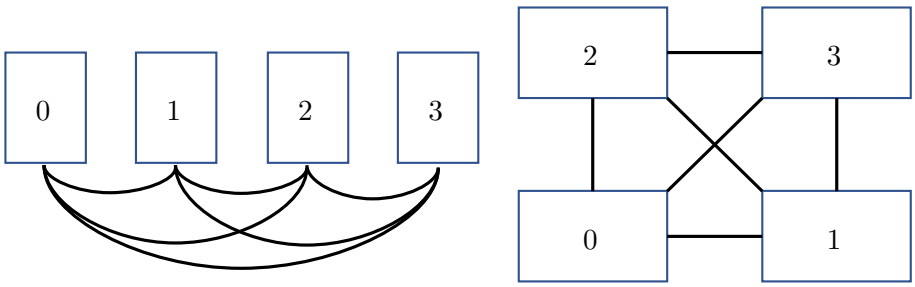


**Figura 1.1:** Dos representaciones de HyperX 1D (grafo completo) con $k = 4$ vértices.

La construcción de una HyperX de $n$ dimensiones, con $n > 1$, se hace a través del producto cartesiano de $n$ grafos completos, donde el parámetro $k$ representa el número de vértices del grafo completo original. Cuando $n > 1$ al parámetro $k$ se le suele llamar *lado* del grafo. Se puede ver un ejemplo de una red HyperX 2D cuyo grafo completo original es $k = 4$ en la Figura 1.2. Como se puede observar, los nodos de la red quedan etiquetados según el producto cartesiano y se trabajará sobre estas coordenadas durante todo el documento. De forma equivalente, un grafo completo puede definirse mediante el método de Cayley con $S = G - \{0\}$ y se puede utilizar el producto directo de grupos para definir una HyperX generalizada. A continuación, se pasan a enumerar algunas de las características topológicas que tiene una HyperX.

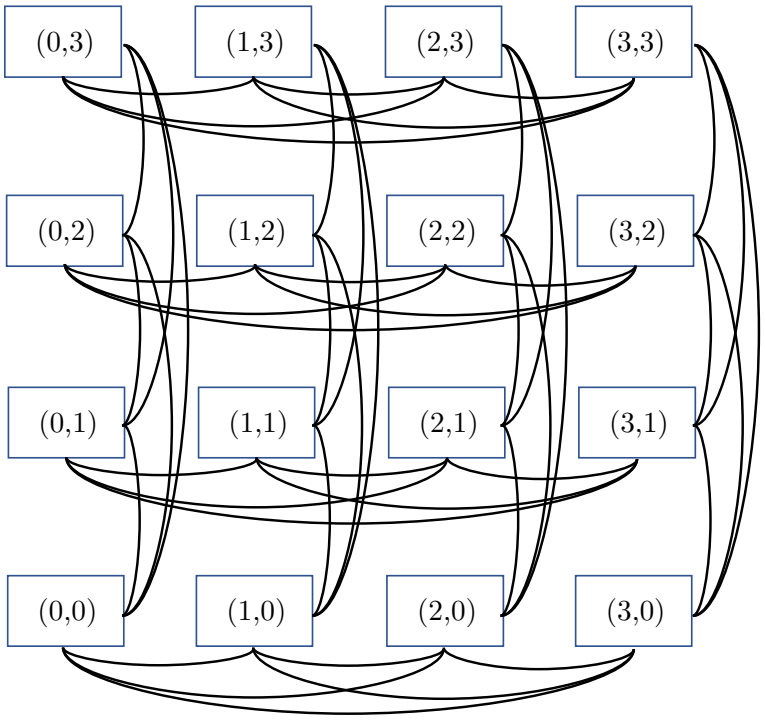


**Figura 1.2:** Ejemplo de HyperX 2D de lado $k = 4$.

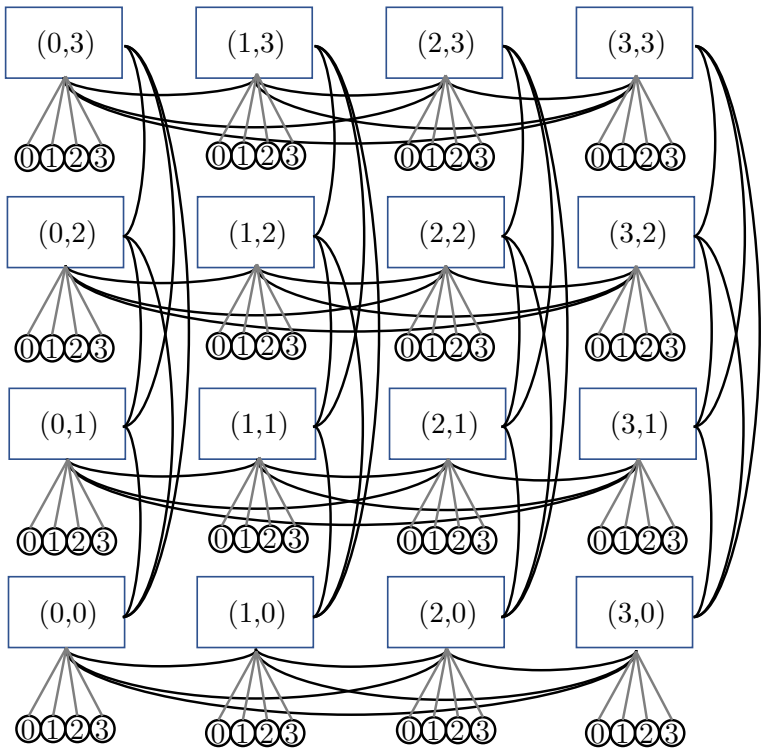


**Figura 1.3:** Ejemplo de HyperX 2D de lado $k = 4$ y concentración $= 4$.

En el grafo de una HyperX de lado $k$ y $n$ dimensiones, el número de vértices es $k^n$ y el número de enlaces es $n(k-1)k^n$.

El *grado* de un vértice es el número de aristas conectadas a él en un grafo. Se dice que un grafo es *regular* si cumple que todos los vértices tienen el mismo grado. Esto puede aplicarse también al grafo de una topología, por ello se puede decir que la HyperX es una *topología regular*. Para una topología también se introduce el concepto de *radix* de un router. El *radix* de un router es el número de enlaces que tiene conectados, contando enlaces entre routers y enlaces de router a servidor. En una HyperX de lado $k$, con dimensiones $n$ y con $c$ servidores por router, el grado de un vértice cualquiera es $n(k-1)$ y el radix de un router cualquiera es $n(k-1)+c$.

El *diámetro* de una red se define como la distancia máxima de un camino mínimo a la que están dos nodos en el grafo de la topología. La distancia se mide en número de saltos sobre el grafo. En una HyperX el diámetro es igual al número de dimensiones $n$.

Además, el número de caminos mínimos sobre el grafo de una HyperX con $n$ dimensiones para un nodo fuente $a$ y un destino $b$ es:

$$\left(\sum_{i=0}^{n-1}[a_i \neq b_i]\right)!,$$

donde $a_i$ y $b_i$ es la componente $i$ de sus coordenadas cartesianas y $[a_i = b_i]$ representa la operación definida por el *Iverson bracket* [13]. Por ejemplo, en la Figura 1.2 entre los nodos $(0,1)$ y $(2,3)$, se tiene que el número de caminos es: $[0 \neq 2] + [1 \neq 3] = 2$. Se puede ir a través

del nodo $(0, 3)$, o a través del $(2, 1)$.

El *ancho de la bisección* de una topología es el mínimo ancho de banda disponible entre dos particiones de la topología del mismo tamaño. Se puede medir como el mínimo número de enlaces que pasan de una partición a otra. Esto sirve para detectar cuellos de botella en una topología. En el caso de una HyperX de lado $k$ y dimensión $n$, el ancho de la bisección es $\frac{k^{n+1}}{4}$ enlaces:

$$\frac{k}{2}\left(\frac{k}{2}k^{n-1}\right) = \frac{k^{n+1}}{4},$$

donde $\frac{k}{2}k^{n-1}$ es el número de routers en una partición y $\frac{k}{2}$ el número de enlaces que cruzan la bisección por router. Una vez calculado el cuello de botella, para dimensionar la concentración $c$ de la red se calcula el número de enlaces en la bisección por nodos en la partición:

$$\frac{\frac{k^{n+1}}{4}}{\frac{k^n}{2}} = \frac{2k^{n+1}}{4k^n} = \frac{k}{2}.$$

Luego la bisección soporta hasta $\frac{k}{2}$ servidores por nodo inyectando a través de ella. Además, para aprovechar aún más la red se puede suponer que la mitad del tráfico de los servidores es lo que pasa a través de la bisección y la otra mitad se queda en la partición. Por ello, la concentración para una HyperX se puede deducir como: $c = \frac{k}{2} + \frac{k}{2} = k$. Con la concentración $c = k$ se considera que la red está bien dimensionada y es la que se asumirá en todo el trabajo [7]. Un ejemplo gráfico de una HyperX bien dimensionada se ve en la Figura 1.3, donde los servidores por cada router están etiquetados con un escalar en el rango $[0, c - 1]$.

### 1.3.2 Control de flujo

El *control de flujo* es el mecanismo de la red encargado de asignar recursos a los mensajes según avanzan en esta. Los recursos más usuales son los enlaces y los canales virtuales [11, Capítulo 13]. A una red que no contenga control de flujo se le denomina red con pérdidas, debido a que existe la posibilidad de que se desechen paquetes antes de llegar al destino. Por ejemplo, si un paquete llega a un puerto de entrada de un router y el canal virtual al que va dentro del puerto está lleno (es decir, el buffer designado no tiene espacio para el paquete), el router procederá a tirar el paquete. Cuando se utiliza una red con pérdidas los nodos inyectores deberán ser conscientes de la posibilidad de que el paquete se pierda y se tengan que llevar a cabo retransmisiones. Esto degrada mucho el aprovechamiento de la red y está en desuso en el contexto de redes de alto rendimiento. Por ello, no se consideran en este trabajo.

Dentro de las redes que utilizan mecanismos de control de flujo, existe un nuevo tipo de red denominada red sin pérdidas, donde se utilizan créditos. Un *crédito* es un trozo de información que indica si existe espacio suficiente para dar uso a un recurso por un agente determinado. Los routers con puertos de entrada y canales virtuales están encargados de emitir créditos para cada buffer, y los routers con puertos de salida irán guardándolos incrementalmente. Si el router envía un paquete a un canal virtual específico, el número de créditos de ese canal se decrementa, no pudiéndose enviar a un canal virtual si tiene 0 créditos. Existen diferentes formas de llevar a cabo el control de flujo:

- *Flit-based flow control*: también conocido como *Wormhole* [14], divide el paquete en trozos llamados *flits*, que es la unidad de control de flujo. Un crédito de un recurso representará que este se puede asignar a un flit que pertenece a un paquete. Por ello, un paquete puede ocupar varios recursos al mismo tiempo.
- *Packet-based flow control*: la unidad de control de flujo es un paquete y un crédito de un recurso indicará que se puede hacer una asignación para un paquete entero. Esto, aplicado a los canales virtuales, indica que un paquete ocupa un buffer al mismo tiempo en toda la red excepto mientras se está produciendo una transmisión entre routers. Algunos ejemplos específicos de controles de flujo basados en paquetes son:
  - *Store-and-forward*: El paquete no se transmite al siguiente nodo hasta no haberlo recibido y guardado completamente.
  - *Virtual cut-through*: No es necesario almacenar todo el paquete antes de empezar una nueva transmisión.

En este trabajo se va a trabajar siempre con redes sin pérdidas. Para ello, se va utilizar Packet-based flow control, en concreto Virtual cut-through como mecanismo de control de flujo.

### 1.3.3 Función de encaminamiento

Una *función de encaminamiento* dentro de un router se encarga de determinar el puerto de salida y canal virtual que tomará un paquete dentro de un puerto de entrada. La misma función de encaminamiento se ejecuta en todos los routers de la red. Una función de encaminamiento $R$ tiene la forma,

$$R : V \times V \rightarrow \mathcal{P}(Q),$$

donde $V$ es el conjunto de nodos de la red, $Q$ es el conjunto de todas las colas de la red y $\mathcal{P}(Q)$ es el conjunto de todos los subconjuntos de todas las colas de la red. La función $R$ recibe como argumentos dos nodos: el nodo actual y el destino, con ello devuelve un conjunto de colas de salida que llevan al siguiente nodo en el camino. La asignación de una cola a un paquete se le denomina *salto*. Es trabajo de otra unidad del router escoger una de las colas que se han retornado.

Existen variantes de la forma de la función de encaminamiento. Por ejemplo, si distingue entre los canales virtuales dentro de un mismo puerto de entrada, la función tiene la forma,

$$R : Q \times V \rightarrow \mathcal{P}(Q).$$

Para una cola $q_i \in Q$ perteneciente al router $i$ actual y nodo destino $n_i \in V$, $E$ devuelve un conjunto de colas pertenecientes al siguiente salto $\mathcal{P}(Q)$ [11, Capítulo 11].

Por último, se puede definir la forma de una función de peso $p$:

$$p : Q \rightarrow \mathbb{N},$$

donde a cada canal virtual salida $q$ se le asigna una prioridad, es decir, un número natural. Una función de peso $p$ está diseñada para evaluar rutas respecto al estado actual de la red.

Para la evaluación de una cola, es común utilizar su *ocupación*, que indica el número de paquetes que están almacenados en la cola en el instante de tiempo de la medición. Con las rutas evaluadas, un router podría coger siempre la cola mejor valorada del conjunto.

La aplicación sucesiva de una función de encaminamiento sobre un paquete determina el *camino* que sigue un mensaje, desde que se inyecta en un servidor, hasta que se consume en otro. Se pueden clasificar las funciones de encaminamiento según los caminos que generan:

- Encaminamiento mínimo: la función de encaminamiento debe generar caminos que den el mínimo número de saltos desde un nodo fuente a un destino.
- Encaminamiento no mínimo: los caminos generados por la función de encaminamiento pueden dar más saltos de lo necesario debido a que hay congestión en la red u otros motivos, como se detallará mas adelante.

Además, las funciones de encaminamiento se pueden clasificar dependiendo de la diversidad de caminos que generan o de su grado de libertad para asignar rutas:

- Determinista: la función de encaminamiento siempre genera el mismo camino para un par fuente-destino.
- Oblivious: la función de encaminamiento selecciona un camino de los que tiene posibles sin considerar el estado de la red. Los deterministas son un subconjunto de los oblivious.
- Adaptativo: un encaminamiento adaptativo acopla los caminos que atraviesa un paquete al estado de la red. Normalmente se utiliza una función de peso que evalua los caminos posibles y elige uno de ellos en base a la ocupación, numero de saltos a destino, etc. Pueden haber diferentes tipos de adaptatividad:
  - Adaptativo en fuente: el camino a seguir se decide en la inyección y a partir de ahí la función de encaminamiento es determinista.
  - Adaptativo medio: un subconjunto de todos los posibles caminos está restringido, normalmente, por cuestiones de deadlock como se detalla en la Sección 1.4.
  - Adaptativo completo: el encaminamiento genera caminos sin restricción.

## 1.4 Deadlock

Un *deadlock* es un fenómeno que ocurre en muchos sistemas, donde un conjunto de agentes no pueden avanzar en sus tareas porque todos ellos están ocupando un recurso diferente, y a su vez esperando la liberación de otro, (ocupado por uno de los agentes del sistema) para usarlo antes de liberar el propio. Específicamente, cuando en una red se forma un deadlock, un conjunto de paquetes están dispuestos en forma de ciclo dentro de la red y cada paquete pretende acceder a su recurso inmediato siguiente. Sin embargo, este recurso está bloqueado por otro paquete (o conjunto de paquetes) que, de igual forma, están esperando para acceder al siguiente recurso antes de liberar el que tienen bloqueado. Existen diferentes formas de clasificar *deadlocks* y técnicas para paliar sus efectos [11, Capítulo 14].

Existen diversas maneras de definir un ciclo en un grafo. Se puede decir que un grafo contiene un *ciclo* si este contiene un subgrafo isomorfo a un grafo cíclico. Un *grafo cíclico* es

un grafo regular conectado de grado dos. Usualmente se utiliza la notación $C_i$ para referirse a un grafo cíclico con $i$ vértices. En la Figura 1.4 puede verse un ejemplo de un grafo completo con un grafo cíclico contenido en él.

### 1.4.1 Tipos de deadlock

Se puede clasificar el deadlock en la red dependiendo la causa por la que se produce [15]:

- Deadlock por la función de encaminamiento: la función de encaminamiento puede generar ciclos dentro la red y que se produzca un deadlock.

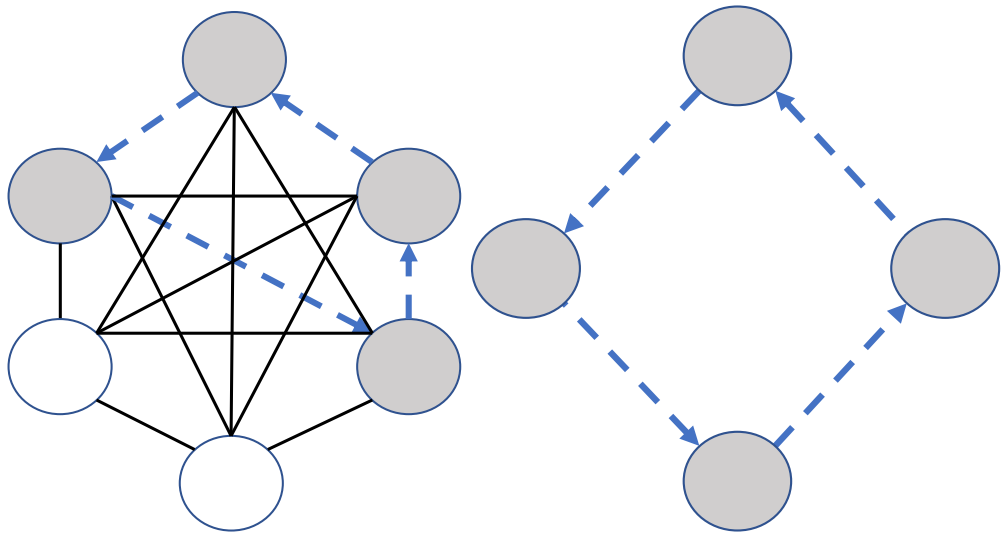

**Figura 1.4:** A la izquierda, ciclo en un grafo completo con vértices $k = 6$ y a la derecha, el subgrafo cíclico $C_4$ isomorfo al ciclo del grafo completo.

- Deadlock por protocolo: una aplicación que ejecuten los servidores e intercambie mensajes a través de la red, se puede quedar en deadlock en un conjunto de servidores debido a una dependencia de mensajes. Esto es independiente de que haya o no un deadlock entre routers de la red.

- Deadlock por cambio de configuración: un cambio en la configuración en la red puede provocar que se produzcan situaciones imprevistas en las transiciones y se produzca un deadlock.

### 1.4.2 Tratamiento del deadlock

En el diseño de una red de interconexión es fundamental tener en cuenta el tratamiento que se va a utilizar para subsanar los efectos del deadlock. Se pueden mitigar sus efectos recuperándose de él cuando surja, o se puede evitar su aparición utilizando algunas restricciones en la definición de una red de interconexión. Este trabajo se va a centrar en la evitación del deadlock utilizando restricciones en la función de encaminamiento.

Uno de los mecanismos para tratar el deadlock se basa en evitar que este ocurra. Para una red construida con unos parámetros definidos *red* = {*topología, control de flujo, encaminamiento*} se dice que *red* cumple la condición de ser *deadlock-free* (libre de deadlock) si por definición esta red no permite la formación de un deadlock. Esta condición es sensible a cualquiera de los 3 parámetros de *red*. Para saber si una red es *deadlock-free* se introduce el concepto de grafo de dependencias de colas, el cual es construido de la siguiente forma:

$$G_q = (Q, E_q),$$

donde $Q$ es el conjunto de colas de la red, $E_q = \{(q_i, q_j) | q_j \in E(q_i, n) \wedge n \in V\}$, $E$ es una función de encaminamiento y $V$ el conjunto de nodos de la red. Una red es *deadlock-free* si cumple que su grafo de dependencias entre colas es acíclico ($DAG$) o dinámicamente acíclico [16], [17], es decir, que en cualquier potencial ciclo siempre se garantiza un camino acíclico sobre el grafo $G_q$ [2]. Las técnicas más utilizadas para evitar ciclos dentro de un grafo de dependencias son:

- Restricción de rutas: La función de encaminamiento restringe algunos giros para evitar la formación de la dependencia cíclica. Por ejemplo: *DOR* [11, Capítulo 8], *Turn model* [18], etc.
- Añadir canales virtuales: Añadir canales virtuales para romper un ciclo en la red. Por ejemplo: *dateline* [19], escalera [11, Capítulo 14], etc.
- Utilización de canal de escape: Utiliza un subconjunto de los canales virtuales disponibles en la red con función de encaminamiento acíclica que siempre están disponibles para romper el deadlock [17].

Otro mecanismo para tratar el deadlock se basa en la detección y recuperación. Existen técnicas de *recuperación regresiva*, donde se desechan los paquetes que han causado el deadlock y técnicas de *recuperación progresiva*, donde no se desechan paquetes y se utiliza un buffer de soporte para temporalmente sacar los paquetes que producen el deadlock y dejar pasar a los demás. Sin embargo, en general estas técnicas apenas se usan y no serán estudiadas en este trabajo [11, Capítulo 14].

## 1.5 Patrón de tráfico

Un patrón de tráfico representa la distribución de los mensajes dentro de una red [11, Capítulo 3]. Normalmente se representa con una matriz $L = (\lambda_{i,j})$, donde cada elemento $\lambda_{i,j}$ representa la fracción de tráfico enviado del nodo $i$ al nodo $j$. Formalmente, podemos definir la función $T$ que representa a un patrón de tráfico de la siguiente forma:

$$T : (\mathbb{Z}/k\mathbb{Z})^{(n+1)} \rightarrow (\mathbb{Z}/k\mathbb{Z})^{(n+1)},$$

[2] Para definir $G_q$ se han utilizado únicamente las dependencias directas generadas por la función de encaminamiento. También se pueden generar dependencias indirectas, las cuales no se van a definir en este trabajo porque no se va a considerar control de flujo a nivel de flit o *wormhole*.

donde $k$ es el lado de la HyperX, $n$ es el número de dimensiones y $\mathbb{Z}/k\mathbb{Z}$ son los enteros módulo $k$. En el caso de una red HyperX, cada servidor está identificado por un vector de coordenadas de tamaño $n+1$ donde $n$ es el número de dimensiones de la red. Por ejemplo, en una red HyperX 2D tenemos los vectores de servidores: $(0,0,0)$ y $(1,0,0)$ los cuales están conectados al router $(0,0)$. La primera coordenada extra es la del servidor dentro del router y se suele denominar $w$. Se pueden ver ejemplos de funciones de patrones de tráfico en las diferentes tablas de la Sección 5.

En este trabajo se va a centrar la atención en patrones de tráfico sintético modelados mediante funciones matemáticas, que pretenden simular las comunicaciones de una aplicación real. Dentro de estos podemos diferenciar el *tráfico benigno*, que es un tráfico que aprovecha por si solo todos los recursos de la red por igual sin generar cuellos de botella, y el *tráfico adverso*, que es un tipo de tráfico que genera puntos calientes donde se concentra mucho tráfico y hace que baje el rendimiento de toda la red. Estresa ciertas partes de la red y deja ociosas otras. Una red mal diseñada puede ser que funcione bien con tráfico uniforme, por ello, como realmente se debe medir el rendimiento global de una red es combinando el tráfico uniforme con el tráfico adverso y así sacar diferentes tipos de métricas del comportamiento de la red. Además, como apoyo para la definición de algunos tráficos se va a utilizar una función extra, aunque no sea función en terminología matemática, que tiene la siguiente forma:

$$\text{rand} : \mathbb{N} \to \mathbb{N}.$$

La función rand devuelve para un $n \in \mathbb{N}$ un número aleatorio en el intervalo $[0, n-1]$. Esta función se encuentra ya implementada en el simulador que se ha utilizado en este trabajo.

## 1.6 Métricas de rendimiento

El rendimiento de la red se estima mediante diferentes tipos de métricas. Dependiendo de los objetivos de la red, se pueden tomar diferentes tipos. En este trabajo las que se han considerado son las siguientes:

- Throughput promedio: el es la tasa media a la que todos los nodos consumen paquetes en una red. Se mide con el número de paquetes consumidos por instante de tiempo. En el caso de este trabajo, se van a utilizar $\frac{\text{Paquetes}}{\text{Ciclo}}$ como unidad de medida, y los ciclos se miden con un reloj general a todo el sistema.

- Throughput mínimo: es la tasa mínima a la que un nodo consume paquetes en una red. En aplicaciones que se ejecutan en sistemas homogéneos de forma distribuida, es importante que todos los nodos del sistema consuman carga por igual de la red porque si alguno recibe menos que los demás, puede generar cuellos de botella en el sistema. Sirve para estimar el *fairness* de una red, es decir, la medida por la cual el tráfico es aceptado por todos los nodos de forma equitativa. Cuanto mayor sea la diferencia entre el Throughput promedio y el medio, peor será el fairness de la red.

- Saltos promedio: número de saltos promedio que dan todos los paquetes que se inyectan en la red. Provee información sobre como se adapta un encaminamiento a un tráfico.

# 2 Herramientas de simulación y desarrollo

La simulación es una técnica muy utilizada para la evaluación de sistemas físicos, químicos, informáticos, etc. los cuales pueden modelarse a través de algoritmos y funciones matemáticas. Es una práctica común a muchas disciplinas y permite obtener resultados con mayor o menor detalle de la estructura simulada.

El acceso a un sistema simulado se caracteriza por tener un bajo coste en comparación con el gasto requerido para la implantación y puesta en marcha de ese sistema. Además, existen muchos detalles y aspectos a tener en cuenta que son prácticamente imposibles reconocer sin poder tener acceso a un entorno simulado. Por ello, simular es un paso necesario antes de invertir en una implementación física. Suele ser común en entornos de investigación.

Sin embargo, un modelo puede carecer del realismo necesario para poder validar una implementación real. Cada vez existen simuladores más fieles a la realidad, pero esta nunca puede ser exactamente replicada. Por ello, las herramientas de simulación son muy útiles para el descarte de ideas pero no para conseguir una validación absoluta. Algunas de las limitaciones del simulador utilizado en este trabajo se presentan en la Sección 2.1.1.

Por ello, se puede decir que mediante la simulación se consigue ahorro evitando gasto, considerándose una de las fases más importantes antes de realizar una inversión.

## 2.1 Booksim2

*Booksim2* es un simulador de redes de interconexión presentado en [20] que extiende del anterior simulador *Booksim*. Booksim2 se caracteriza por lo siguiente:

- Basado en ciclos: Utiliza un reloj global para hacer todas las mediciones. La simulación tiene dos partes: una de lectura de todos los datos del entorno simulado y otra de escritura para que todos los componentes actualicen sus datos. De esta manera, cualquier elemento del sistema que necesite obtener un dato para hacer alguna operación, podrá obtenerlo en la primera fase.

- Simula a nivel de flit: El uso de recursos es simulado a nivel de flit, detallando el paso de estos por todos los componentes de la red.

Además, es un simulador de redes muy famoso, con un gran número de citas y por ello se ha escogido para el trabajo. Tiene un enfoque a NoCs (Networks on Chip), prestando atención a detalles de microarquitectura, latencias, etc. pero también se ha utilizado para simulación en trabajos actuales que no están específicamente orientados a NoCs como en [21]-[24] y el aquí presente. Para lanzar simulaciones, el ejecutable con el simulador recibe como argumento un fichero con todos los parámetros requeridos para la simulación. Algunos de ellos se encuentran en la Tabla 2.1, al final del capítulo.

### 2.1.1 Limitaciones del simulador

Con el uso del simulador, se identifican diferentes limitaciones que afectan a la experiencia de usuario. A modo de ver del autor del presente trabajo, el uso y desarrollo sobre este simulador es de una complejidad muy alta y se echan en falta algunos elementos en este.

Para poder comprender lo que hace el simulador hay que leerse el código en profundidad, lo cual es algo complejo y supone claramente una limitación. Las descripciones sobre el simulador son incorrectas y escasas tanto en la documentación oficial como en los comentarios en el código. Se utilizan estructuras de datos, objetos y variables que no están explicadas y no tienen nombres representativos. Por ello, realizar desarrollo en este simulador es una tarea complicada.

Además, el simulador no tiene implementado control de flujo Virtual cut-through y solo trabaja con wormhole. Sin embargo, en el caso de utilizar paquetes de tamaño un flit se puede decir que ambos controles de flujo son equivalentes, pero no se pueden utilizar diferentes tamaños de paquetes si se pretende utilizar Virtual cut-through. Por ello, para este trabajo se ha utilizado Virtual cut-through con paquetes de tamaño un flit.

El uso de *lookahead* en una red es una técnica para la reducción de la latencia de red de un paquete a su paso por un router. Esta, se basa en hacer el cálculo de la función de encaminamiento del siguiente router una vez se ha hecho la asignación de una cola al paquete en el router actual [11, Capítulo 16]. En la Sección 2.1.2, se ve como afecta el lookahead en el paso de un paquete por un router. En Booksim2, el uso del lookahead puede llevar a resultados poco realistas debido a que no existe una propagación en la red de la información de la ocupación de las colas del siguiente router. Esto está reconocido en [20].

La topología HyperX se encuentra en el simulador bajo el nombre de *flat-fly*, pero está incompleta y solo se puede simular hasta dos dimensiones, con valores de $c$ cuadrados. Sin embargo, como se ha visto en la Sección 1.3.1, la topología se puede definir para cualquier $k$, $n$ y $c = k$.

El simulador implementa tres modelos de routers, pero dos de ellos están completamente desfasados. El más actual es un input-queue router [11], [25], que además está limitado por no tener alguna funcionalidad usual como el recómputo de la función de encaminamiento para un paquete que en la etapa de asignación del canal virtual del siguiente salto no ha tenido ninguna asignación. La descripción de estas etapas se ven en la Sección 2.1.2.

### 2.1.2 Detalles del router

El router *input-queue* implementado en Booksim2 se compone de las siguientes etapas:

1. Cómputo de la función de encaminamiento: Aqui el router computa la función de encaminamiento y devuelve un conjunto de canales virtuales posibles.

2. Asignación de canal virtual del siguiente salto: El router asigna un canal virtual del conjunto retornado por la fase de routing.

3. Asignación de recursos del crossbar: El router asigna al paquete una entrada y una salida al crossbar.

4. Paso del crossbar: El paquete sale de la cola del puerto input y pasa a la cola de un puerto de salida.

5. Paso al siguiente router: El paquete sale del router actual y llega al siguiente.

Los créditos de las colas del siguiente salto se consumen en la etapa 3. Los canales virtuales de los puertos de salida se modelan como una única cola FIFO para cada puerto, donde se junta el tráfico de todos los canales virtuales y va saliendo uno a uno.

El paso de un paquete por el router, sin ninguna modificación, constaría al menos de 4 ciclos, aunque se puede reducir a 2. Utilizando especulación se pueden hacer en paralelo las etapas 2 y 3, y activando el lookahead el paquete no pasa por la etapa 1 y esta se computa en la fase 4 del router anterior, cuando el paquete tiene asignado a una cola de salida. Para que el router no suponga un cuello de botella en la red se va a utilizar speedup interno y speedups en el crossbar[1].

## 2.2 Desarrollo realizado

Debido a las restricciones y limitaciones en la implementación original de la topología HyperX se ha implementado de cero esta topología. Además, se han desarrollado diferentes encaminamientos y patrones de tráfico específicos para ella. El trabajo se ha ido desarrollando de forma iterativa y utilizando el sistema de control de versiones Git [26] donde se han ido guardando los cambios sobre el código. El repositorio se encuentra disponible en el siguiente enlace[2].

En general, para la validación de la implementación se ha comprobado de forma exhaustiva que las métricas resultantes de las simulaciones se acerquen a los valores teóricos calculados para cada simulación. Además, en la medida de lo posible, se han contrastado estos resultados con resultados en la implementación original. Y por último, en muchos casos se han comparado los resultados obtenidos en Booksim2 con los obtenidos en otros simuladores y trabajos. Esto hace bastante extensa y completa la parte de validación realizada. No se va a adjuntar un conjunto de pruebas específicas de validación de la topología porque no es el enfoque de este trabajo, pero en la Sección 5 se puede comprobar que los resultados obtenidos en las configuraciones se asemejan a los valores teóricos descritos.

La nueva implementación de la topología HyperX ahora acepta cualquier configuración de $n$, $k$ y $c$. Para el modelado de esta en el simulador, se han hecho diferentes formalizaciones de los componentes de la topología. Para empezar, se ha dado una numeración a los routers de la red:

$$\mathrm{Router_{id}(a)} = \sum_{i=0}^{n-1} k^i a_i,$$

donde $a$ representa las coordenadas cartesianas de un router y $a_i$ es la componente $i$, que toma valores entre $0 \leq a_i \leq k-1$. Siempre se cumple que $0 \leq \mathrm{Router_{id}(a)} \leq k^n - 1$. Además, también, se ha dado una numeración general a los enlaces entre routers:

[1] Este concepto no se ha introducido en la memoria porque no se considera esencial para entender el trabajo.
[2] Link al repositorio con el código desarrollado: https://github.com/alexcano98/booksim2.git (visitado el 29/06/2022).

$$\text{Enlace}_{\text{id}}(\text{a}, \text{b}, \text{d}) = n(k-1)\text{Router}_{\text{id}}(a) + b(k-1) + d,$$

donde $a$ representa las coordenadas cartesianas de un router, $b$ es la dimensión por la que sale el enlace, con valores entre $0 \leq b \leq n-1$ y $d+1$ es el desplazamiento en la dimensión seleccionada, con valores entre $0 \leq d \leq k-2$. Siempre se cumple que $0 \leq \text{Enlace}_{\text{id}}(\text{a}, \text{b}, \text{d}) \leq n(k-1)k^n - 1$.

Y también se ha dado una numeración a los enlaces de inyección/consumo, que van por otra parte:

$$\text{Consumo}_{\text{id}}(\text{a}, \text{w}) = c\,\text{Router}_{\text{id}}(a) + w,$$

donde $a$ representa las coordenadas cartesianas de un router, $w$ la coordenada del servidor $0 \leq s \leq c-1$ y $c$ es la concentración.

Los funciones de encaminamiento implementadas para la topología se encuentran formalizadas en los Capítulos 3 y 4. Asimismo, los tráficos se encuentran en el Capítulo 5.

## 2.3 Metodología

Se han desarrollado varios scripts en el lenguaje *Python* para el lanzamiento de pruebas de forma automatizada en un sistema de gestión de colas *SLURM*[27]. Para ello, se ha utilizado como argumento un fichero *JSON* [28] con todas las configuraciones de red que simular. Para lanzar las simulaciones se ha hecho uso de un clúster formado por cinco nodos, donde cada nodo está compuesto por dos sockets Intel Xeon Silver 4114. Cada socket contiene a su vez 10 cores físicos, haciendo un total de 20 cores por nodo y 100 en toda la máquina. No se utiliza *multi-threading*. Además, la máquina dispone de 30GB de RAM por nodo.

Para lanzar una simulación en Booksim2 se pasa como argumento al simulador un fichero de configuración, indicando los parámetros específicos de la simulación. Un esquema de algunos de los parámetros mas importantes se puede ver en la Tabla 2.1, donde se especifica el nombre del parámetro, el tipo de componente al que afecta, el valor o rango de valores con los que se ha trabajado para hacer las simulaciones y una descripción sobre el parámetro.

| Nombre | Tipo | Valor | Descripción |
|---|---|---|---|
| $k$ | Topología | 4-32 | Lado de la HyperX. |
| $n$ | Topología | 1-3 | Número de dimensiones. |
| $c$ | Topología | 4-32 | Concentración o número de servidores por router. |
| num_vcs | Router | 2-12 | Número de canales virtuales por puerto del router. |
| input_speedup | Router | 2-12 | Speedup de entrada al crossbar. |
| output_speedup | Router | 2-12 | Speedup de salida al crossbar. |
| injection_rate | Router | 1.0 | Tasa de inyección. |
| routing_delay | Router | 1 | Ciclos que tarda el router en el cálculo de la función de encaminamiento. |
| speculative | Router | 1 (Activado) | Especulación en la selección del canal virtual, se hacen las etapas 2 y 3 a la vez. |
| alloc_iters | Router | 2-3 | Iteraciones para hacer asignaciones en el allocator dentro de las etapas de asignación de canal virtual y asignación de puertos del crossbar. |
| vc_busy_when_full | Router | 1 (Activado) | Cuando un canal virtual de salida está lleno y no tiene créditos no se le puede asignar otro paquete. Crítico en cuestiones de deadlock utilizando un encaminamiento con canal de escape. |
| vc_prioritize_empty | Router | 1 (Activado) | Prioriza canales virtuales vacios. |
| traffic | Simulación | Tablas: 5.2, 5.4, 5.6 | Patrón de tráfico a usar. |
| routing_function | Simulación | Tablas: 5.1, 5.3, 5.5 | Función de encaminamiento. |

**Tabla 2.1:** Parámetros para la topología HyperX usados en las simulaciones.

# 3 Encaminamiento mínimo

En este capítulo se describen las funciones de encaminamiento mínimas con mayor relevancia para la topología estudiada. Para cada función de encaminamiento, se especificará su clasificación dentro de las funciones de encaminamiento, cómo evita el deadlock, los recursos que necesita y la definición formal de la función de encaminamiento para una HyperX 2D de lado $k$. Se describen en 2D para simplificar la exposición. Las definiciones generalizan a más dimensiones y es como están implementadas en el código desarrollado para este trabajo. Respecto a la formalización, se pasa a definir la función $s$ de ayuda, que asigna una numeración a los puertos de salida de cada nodo para ir de un nodo $a$ a un nodo $b$ adyacente:

$$s(a, b, i) = i(k-1) + ((b_i - a_i - 1) \bmod k),$$

donde $a$ es el nodo actual, $b$ es el nodo destino, $k$ es una constante que representa el lado de la red HyperX e $i$ es la dimensión por la que se quiere salir. Las variables $a_i$ y $b_i$ representan las proyecciones de $a$ y $b$ en la dimensión $i$: $0 \leq i \leq n-1$. Además, para cualquier nodo $a$ y dimensión $i$ en la red, se cumple que: $0 \leq a_i \leq k-1$. En resumen, la función $s$ devuelve el puerto de salida de $a$ correspondiente para acercarse a $b$ en la dimensión $i$. Un ejemplo del enumerado en dos routers diferentes en una red HyperX 2D se ve en la Figura 3.1. Cuando los vectores de dos nodos tienen el mismo valor en la componente $i$, se dice que están alineados en la dimensión $i$ y no hace falta moverse en esa dimensión para llegar al destino.

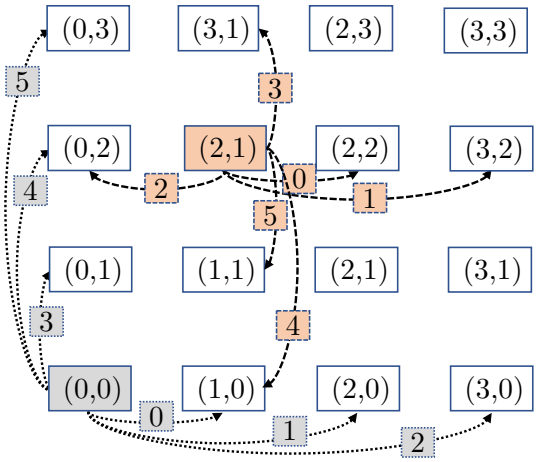


**Figura 3.1:** Ejemplo de etiquetado de los puertos de salida de dos routers diferentes en una HyperX 2D.

Para definir las funciones de encaminamiento se van a seguir las estructuras presentadas en la Sección 1.3.3. Para distinguir entre los elementos dentro de $\mathcal{P}(Q)$ se va a definir una notación para designar algunos de ellos:

- Se define $Q_{(a,i)}$ como el conjunto de canales virtuales del puerto $i$ en un nodo $a$. Con esto, $Q_{(a,s(a,b,0))}$ es el conjunto de canales virtuales del puerto $s(a,b,0)$ por el cual nos acerquemos a $b$ en la dimensión 0 desde $a$.

- Se define $Q_{(a,\text{consumo})}$ como el conjunto de canales virtuales que consumen los paquetes, es decir, que van a los servidores desde ese router.

- Se define $Q_{(a,\text{inyección})}$ como el conjunto de canales virtuales que inyectan paquetes a la red, es decir, generado desde los servidores de ese router.

Además, para la definición de algunos encaminamientos se va a hacer uso de una notación inspirada en el Iverson bracket [13], donde:

$$[Q \text{ si } P] = \begin{cases} Q & \text{si } P \text{ es Verdadero,} \\ \emptyset & \text{en caso contrario.} \end{cases}$$

## 3.1 Dimension-Order-Routing

DOR [11, Capítulo 8] es un encaminamiento determinista que atraviesa las dimensiones de la red en orden para evitar el deadlock. Por ello, este encaminamiento evita el deadlock mediante la restricción de rutas. El paquete no cambia de dimensión hasta que no se alinea con el destino en la dimensión ordenada más baja. La definición de la función DOR para una HyperX 2D moviéndose primero en la dimensión $x$ y luego en la $y$ (como en la Figura 3.2) sería la siguiente:

$$\text{DOR}(a,b) = \begin{cases} Q_{(a,s(a,b,0))} & \text{si } a_0 \neq b_0, \\ Q_{(a,s(a,b,1))} & \text{si } a_0 = b_0 \wedge a_1 \neq b_1, \\ Q_{(b,\text{consumo})} & \text{si } a = b. \end{cases}$$

En la Figura 3.2 se ven diferentes estilos de flecha representando los caminos que seguirían los paquetes desde un nodo fuente a un destino. El camino negro de puntos en la figura empieza en las coordenadas del router $(2,2)$, salta al $(0,2)$ y llega al router destino con coordenadas $(0,3)$. Se ha alineado con el destino primero en la coordenada $x$ y luego en la $y$. Además, se pueden contar dos saltos extra, desde el servidor inyector al primer router: $(w_i,2,2) \rightarrow (2,2)$ y desde el router destino al servidor destino: $(0,3) \rightarrow (w_j,0,3)$. Serían un total de cuatro saltos, pero normalmente se descartan los saltos servidor $\rightarrow$ router y router $\rightarrow$ servidor y solo se cuentan los router $\rightarrow$ router por lo que quedan un total de dos saltos.

También, se pueden recorrer las dimensiones en orden inverso:

$$\text{DOR\_INV}(a,b) = \begin{cases} Q_{(a,s(a,b,1))} & \text{si } a_1 \neq b_1, \\ Q_{(a,s(a,b,0))} & \text{si } a_1 = b_1 \wedge a_0 \neq b_0, \\ Q_{(b,\text{consumo})} & \text{si } a = b. \end{cases}$$

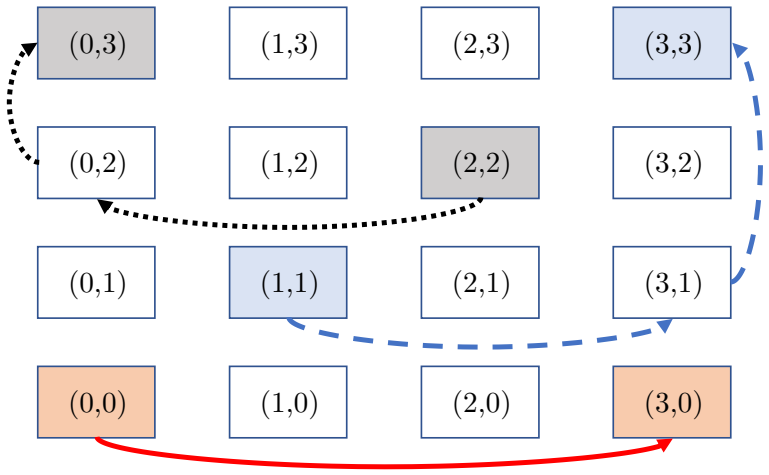


**Figura 3.2:** Ejemplos de diferentes caminos DOR representados con colores diferentes en una red HyperX.

## 3.2 Turn model mínimo

El encaminamiento Turn model mínimo es un encaminamiento adaptativo medio que restringe las rutas estrictamente necesarias para evitar el deadlock. Es más adaptativo que el encaminamiento DOR, pero permite menos rutas que un encaminamiento fully-adaptive. El Turn model es introducido en [18] donde el algoritmo de encaminamiento mínimo *p-cube* estaba diseñado para un hipercubo, pero se puede generalizar para una HyperX de lado mayor a dos [29]:

$$\begin{aligned}&\text{TURN\_MODEL}(a,b) = [\text{DOR}(a,b) \text{ si } a_0 < b_0] \cup [\text{DOR\_INV}(a,b) \text{ si } a_1 < b_1] \\ &\cup [\text{DOR}(a,b) \text{ si } a_0 > b_0 \wedge a_1 \geq b_1] \cup [\text{DOR\_INV}(a,b) \text{ si } a_1 > b_1 \wedge a_0 \geq b_0] \\ &\cup \left[Q_{(b,\text{consumo})} \text{ si } a = b\right]\end{aligned}$$

El Turn Model, siendo un encaminamiento con más adaptatividad que un DOR, tiene el problema de que hay nodos que para un destino solo pueden generar caminos deterministas y otros completamente adaptativos. Esta asimetría hace que el uso de la red esté desbalanceado. Se puede ver un ejemplo en la Figura 3.3, donde para ir del $(1,1)$ al $(3,3)$ hay dos caminos disponibles, pero para ir del $(2,2)$ al $(0,3)$ solo hay uno disponible.

## 3.3 O1Turn

El O1Turn es un encaminamiento adaptativo fuente introducido en [30]. En O1Turn, cuando un paquete atraviesa la cola de inyección se decide a través de una función de peso si este debe atravesar las dimensiones en orden o en orden inverso. Para evitar el deadlock, se asigna la mitad de los canales virtuales para ir DOR (xy) y la otra mitad para ir DOR inverso (yx). Ambas particiones de canales virtuales se recorren independientemente sin ciclos, y un paquete no puede cambiarse de partición una vez sea introducido en una de las dos, luego

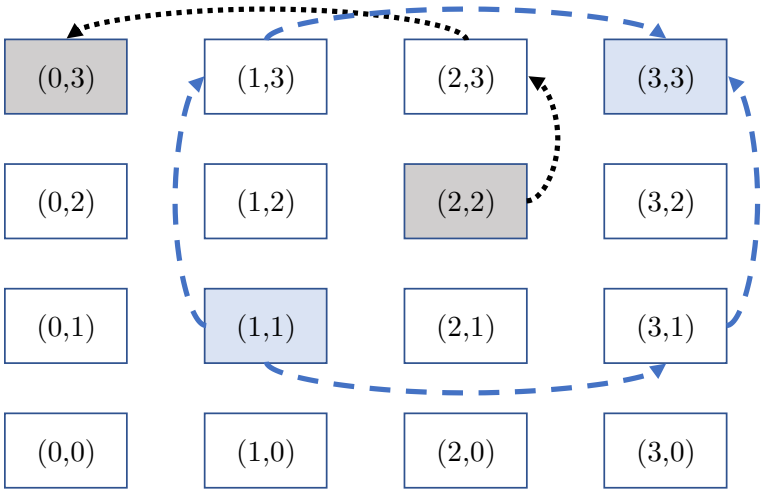


**Figura 3.3:** Ejemplos de la adaptatividad del Turn model dependiendo del par fuente-destino en una red HyperX.

no existe deadlock en la red. Para la formalización del encaminamiento, la primera mitad de los canales virtuales está dentro del conjunto $Q_A$ y a la segunda mitad en $Q_B$. Por ello, $Q := Q_A \cup Q_B$. El encaminamiento se formaliza de la siguiente manera:

$$\text{O1TURN}(q_a, b) = \begin{cases} \text{DOR}(a,b) \cap Q_A & \text{si } q_a \in Q_{(a,\text{inyección})} \wedge f(a,b) = 0, \\ \text{DOR\_INV}(a,b) \cap Q_B & \text{si } q_a \in Q_{(a,\text{inyección})} \wedge f(a,b) = 1, \\ \text{DOR}(a,b) \cap Q_A & \text{si } q_a \in Q_A, \\ \text{DOR\_INV}(a,b) \cap Q_B & \text{si } q_a \in Q_B, \\ Q_{(b,\text{consumo})} & \text{si } a = b, \end{cases}$$

donde $f(a,b)$ indica la salida con menor ocupación en los bufferes entre DOR$(a,b)$ y DOR_INV$(a,b)$. En caso de coincidir ambas ocupaciones, devuelve una aleatoria.

## 3.4 Mínimo con canal de escape

Mínimo con canal de escape es un encaminamiento completamente adaptativo. Utiliza un subconjunto de los canales virtuales para utilizarles de forma completamente adaptativa y reserva un subconjunto disjunto con este para romper el deadlock [16], [17]. A este último subconjunto se le denomina canal de escape y se utilizarán con encaminamiento DOR. El canal de escape convierte el grafo de dependencias de colas en dinámicamente acíclico, lo cual, libera de deadlock el encaminamiento.

Para hacer una distinción entre los canales virtuales de escape y completamente adaptativos, se define $Q$ como el conjunto total de canales virtuales, $Q_c$ como el conjunto de canales que no son de escape y $Q_1$ como el conjunto de canales de escape.

$$\text{ESCAPE\_MIN}(a,b) = (\text{DOR}(a,b) \cap Q_1) \cup \left[Q_{(a,s(a,b,0))} \cap Q_c \text{ si } a_0 \neq b_0\right] \cup \left[Q_{(a,s(a,b,1))} \cap Q_c \text{ si } a_1 \neq b_1\right] \cup \left[Q_{(b,\text{consumo})} \text{ si } a = b\right],$$

donde $(\text{DOR}(a,b) \cap Q_1)$ representa la ruta de escape y $(Q_{(a,s(a,b,i))} \cap Q_c)$ representa la rutas completamente adaptativas.

Es importante no permitir que un paquete se quede bloqueado en la cabeza de una cola esperando exclusivamente la liberación de un canal virtual adaptativo. El comienzo de un deadlock se forma siempre con un paquete bloqueado en la cabeza de un canal virtual [17, Lema 1], por ello, dando siempre la opción de irse por el canal de escape, eventualmente se liberará y el paquete avanzará sin deadlock en la red.

## 3.5 Mínimo con escalera

Mínimo con escalera es un encaminamiento completamente adaptativo. El método de la escalera da una ordenación a los canales virtuales de cada puerto, y los paquetes los atraviesan de forma ascendente. Esto rompe las dependencias en el grafo de dependencias de colas [11, Capítulo 14], [31] y esta técnica también es conocida como *distance ordered classes*. Para la implementación de este encaminamiento, se necesita que el número de canales virtuales por puerto sea mayor o igual al camino mínimo más largo (o al número de saltos máximo que se vaya a dar en la red) y dar una numeración a los canales virtuales. Se define $h$ como el entero que representa el número de saltos actuales del paquete y $Q_h$ el conjunto de canales virtuales numerados como $h$:

$$\text{ESCALERA\_MIN}(a,b) = \left[Q_{(a,s(a,b,0))} \cap Q_h \text{ si } a_0 \neq b_0\right] \cup \left[Q_{(a,s(a,b,1))} \cap Q_h \text{ si } a_1 \neq b_1\right] \cup \left[Q_{(b,\text{consumo})} \text{ si } a = b\right].$$

# 4 Encaminamiento no mínimo

En este capítulo se describen las funciones de encaminamiento no mínimas con mayor relevancia para la topología estudiada. Para cada función de encaminamiento, se especificará su clasificación dentro de las funciones de encaminamiento, cómo evita el deadlock, los recursos que necesita y la definición formal de la función de encaminamiento para una HyperX 2D de lado $k$. Las formalizaciones se llevarán a cabo con la misma notación descrita para el Capitulo 3. De nuevo, se expone el caso 2D para simplificar la lectura.

Para estos encaminamientos, se introduce el término *miss-hop* que representa un salto a un nodo que no está alineado con el destino en una dimensión. A excepción de Valiant, las siguientes funciones de encaminamiento seleccionan un router en cada dimensión no alineado con el destino por si se quiere ir no mínimo en esa dimensión antes de moverse al nodo alineado.

## 4.1 Valiant

Valiant es el encaminamiento no mínimo más extendido [32]. Cada paquete inyectado en la red es asignado con un router escogido aleatoriamente para que el paquete tome la siguiente dirección: fuente → intermedio → destino. Con esto, se logra repartir el tráfico uniformemente por toda la red independientemente del patrón del mismo, pero también se sacrifica el rendimiento debido a que cada paquete da el doble de saltos por la red. Normalmente, Valiant es un buen encaminamiento ante tráfico adverso y sirve de referencia para comparar con otros encaminamientos no mínimos. Sin embargo, para tráficos benignos se queda lejos de llegar a un resultado óptimo.

Para evitar el deadlock, la primera mitad de los canales virtuales se usa para ir del nodo fuente al intermedio con encaminamiento DOR. La segunda mitad, del intermedio al destino, se hace también con encaminamiento DOR. Por ello, es un encaminamiento estático una vez se conoce el nodo intermedio. Se puede definir Valiant como un encaminamiento con 2 fases:

1. Nodo fuente a nodo intermedio: utiliza el conjunto de colas $Q_A$
2. Nodo intermedio a nodo destino: utiliza el conjunto de colas $Q_B$

Cuanto llegamos al nodo intermedio, cambiamos de fase = 0 a fase = 1

$$\text{VALIANT}(a,b) = \begin{cases} \text{DOR}(a,\text{int}) \cap Q_A & \text{si fase} = 0, \\ \text{DOR}(a,b) \cap Q_B & \text{si fase} = 1, \\ Q_{(b,\text{consumo})} & \text{si } a = b, \end{cases}$$

donde int es el nodo intermedio seleccionado aleatoriamente. Además, también se ha implementado un Valiant que utiliza escalera en las dos fases para evitar el deadlock:

$$\text{VALIANT\_ESCALERA}(a,b) = \left[Q_{(a,s(a,\text{int},0))} \cap Q_h \text{ si } a_0 \neq \text{int}_0 \wedge \text{fase} = 0\right]$$
$$\cup \left[Q_{(a,s(a,\text{int},1))} \cap Q_h \text{ si } a_1 \neq \text{int}_1 \wedge \text{fase} = 0\right] \cup \left[Q_{(a,s(a,b,0))} \cap Q_h \text{ si } a_0 \neq b_0 \wedge \text{fase} = 1\right]$$
$$\cup \left[Q_{(a,s(a,b,1))} \cap Q_h \text{ si } a_1 \neq b_1 \wedge \text{fase} = 1\right] \cup \left[Q_{(b,\text{consumo})} \text{ si } a = b\right].$$

## 4.2 Turn model no mínimo

Turn model no mínimo es un encaminamiento adaptativo medio. Restringe las rutas estrictamente necesarias para evitar el deadlock [18]. A diferencia de Valiant, se escoge por cada dimensión no alineada con el destino un nodo en esa dimensión que no esté alineado con el nodo destino, y se tomará una decisión mínima (ir al nodo alineado) o no mínima (ir al nodo no alineado), según sea conveniente.

$$\text{TURN\_MODEL\_MISS}(a,b) = \text{TURN\_MODEL}(a,b) \cup \left[Q_{(a,s(a,\text{int},0))} \text{ si } a_0 < b_0 \wedge \text{miss}_0 = 0\right]$$
$$\cup \left[Q_{(a,s(a,\text{int},1))} \text{ si } a_1 < b_1 \wedge \text{miss}_1 = 0\right] \cup \left[Q_{(a,s(a,\text{int},0))} \text{ si } a_0 > b_0 \wedge a_1 \geq b_1 \wedge \text{miss}_0 = 0\right]$$
$$\cup \left[Q_{(a,s(a,\text{int},1))} \text{ si } a_1 > b_1 \wedge a_0 \geq b_0 \wedge \text{miss}_1 = 0\right],$$

donde $\text{miss}_\text{i}$ es una variable que indica si se ha realizado un previo miss-hop en la dimensión $i$. Si esta es igual a 0, significa que no ha habido miss-hop en esa dimensión. La variable $int$ representa al vector del router al que se hace miss-hop e $int_i$ la componente $i$ del vector. La componente $int_0$ está acotada y definida para los siguientes valores:

$$\text{int}_0 = \begin{cases} \text{rand}(a_0+1,\, k-1) & \text{si } a_0 < b_0, \\ \text{rand}(b_0+1,\, a_0-1) & \text{si } a_0 > b_0 \wedge a_1 \geq b_1. \end{cases}$$

De igual manera, $int_1$ se define de la siguiente forma:

$$\text{int}_1 = \begin{cases} \text{rand}(a_1+1,\, k-1) & \text{si } a_1 < b_1, \\ \text{rand}(b_1+1,\, a_1-1) & \text{si } a_1 > b_1 \wedge a_0 \geq b_0. \end{cases}$$

La función $\text{rand}(n,m)$ devuelve un número aleatorio entre $n \leq \text{rand}(n,m) \leq m$. Se puede deducir que las parejas fuente-destino tienen un número desigual de nodos intermedios disponibles, e incluso algunas no tienen ninguno.

Basado en esto, existe el mecanismo *BRINR* [33] que hace un balanceo de los miss-hops disponibles en un Turn model no mínimo. Sin embargo, el Algoritmo 2 del artículo no es correcto por las siguientes razones:

- En la línea 4 el invariante no está bien definido. No queda claro si $i < \left\lfloor \frac{N-1}{2} \right\rfloor$ ó $i < \left\lfloor \frac{N-1}{2} \right\rfloor + 1$.
- En la línea 11, se produce un acceso con un índice fuera del rango definido. Se define la variable $N$ con valor igual al número de servidores, e $i = 0$. En la primera iteración del bucle iniciado en la línea 4, se acceden a las variables $R_0$ y $R_N$, rango donde existen $N+1$ routers. Se ha tratado de dar diferentes inicializaciones a las variables, pero el algoritmo sigue sin estar definido correctamente.

## 4.3 DAL

DAL es un encaminamiento completamente adaptativo que usa un canal de escape DOR como mecanismo de evitación de deadlock, orginalmente propuesto en [6]. Al asumir que se utiliza control de flujo *virtual cut-through*, no es necesario incluir en el grafo de dependencias de colas las dependencias indirectas. De igual forma que en la definición del Turn model no mínimo 4.2, DAL solo permite un salto no mínimo en cada dimensión no alineada con el destino. La formalización de DAL en una HyperX 2D es:

$$\begin{aligned}\text{DAL}(a,b) = \text{ESCAPE\_MIN}(a,b) \cup \left[Q_{(a,s(a,\text{int},0))} \cap Q_c \text{ si } a_0 \neq b_0 \wedge \text{miss}_0 = 0\right] \\ \cup \left[Q_{(a,s(a,\text{int},1))} \cap Q_c \text{ si } a_1 \neq b_1 \wedge \text{miss}_1 = 0\right],\end{aligned}$$

donde $\text{miss}_\text{i}$ es una variable que indica si se ha realizado un previo miss-hop en la dimensión $i$. Si esta es igual a 0, significa que no ha habido un miss-hop previo en esa dimensión. La variable $int_i$ representa la componente $i$ del router al que se hace miss-hop. En este caso, no hay ninguna restricción respecto a los nodos a los que se puede hacer miss-hop por dimensión, todos pueden a todos.

DAL es un encaminamiento que tiene muchas colas de salida disponibles al aplicar la función de encaminamiento, y para hacer una buena selección dentro del conjunto de colas de salida, es necesario evaluar con una función de peso una a una. En la definición original del encaminamiento no se da una función de peso explícita, simplemente se especifica que el encaminamiento va primero por las colas mínimas si no están llenas, si están llenas se utilizan las colas miss, y si está todo lleno se usa la cola de escape. Debido a no alcanzar los valores teóricos de rendimiento que debería alcanzar DAL con el algoritmo de selección de rutas propuesto inicialmente, en este trabajo se ha optado por dar una función $p$ ad-hoc para evaluar rutas, que sí que alcanza los objetivos explicados más en detalle en la Sección 5:

$$p_{\text{dal}}(q_i) = (\text{OCC\_BIAS} - \text{occupacion}(i))(n - \text{dist\_to\_dest} + \text{HOP\_BIAS}).$$

En [10] se menciona que, para el uso de funciones de encaminamiento completamente adaptativas y no mínimas que usan canales de escape como mecanismo de evitación de deadlock, es necesario el uso de *atomic queue allocation*[1][11], pero esto es solo en el contexto de estar usando control de flujo *wormhole* o *flit-based flow control*, dato que los autores no dejan claro. En este trabajo se ha fijado el control de flujo *virtual cut-through*. Además, para la implementación de DAL, no hay que añadir etiquetas nuevas en la cabecera de los paquetes ni el router tiene que tener ningún tipo de característica micro-arquitectural compleja.

[1] Este concepto no se ha introducido en la memoria porque no se considera esencial para entender el trabajo y se introduce para evitar confusiones con la bibliografía mencionada. En pocas palabras, el *atomic queue allocation* impone una condición a la red en la cual no se permite a un canal virtual contener más de un paquete al mismo tiempo. Esto limita mucho el uso que se le da a los canales virtuales de la red y afecta negativamente al rendimiento en caso de no tratarlo.

## 4.4 Omni-WAR

Omni-WAR es un encaminamiento completamente adaptativo que utiliza escalera como mecanismo de evitación del deadlock [10]. Se define que los canales virtuales necesarios para la implementación de la escalera en Omni-WAR son $2n$, donde $n$ es el número de dimensiones. Al igual que en DAL, en cada dimensión se da la posibilidad de dar 2 saltos: uno para hacer un misshop a cualquier nodo no alineado de la dimensión, y otro salto para alinearse con el destino. Por ello, el encaminamiento necesita múltiplos de $2n$ en los canales virtuales y usar 1 o más de 1 canal virtual por salto. Por ejemplo, en una red 3D lo mínimo son 6 canales para tener 1 canal por salto, con 12 canales se tienen 2 canales por salto, 18 canales se tienen 3 por salto, etc.

Sin embargo, en el trabajo original se considera que lo más justo es hacer las pruebas del encaminamiento en una red 3D con 8 canales virtuales, 6 de ellos necesarios por definición, y 2 sobrantes que son utilizados según los autores para evitar el *head-of-line-blocking* [11], fenómeno por el cual no se permite que paquetes que no se encuentran en la cabeza de una cola accedan a recursos ociosos, debido a que el paquete en la cabeza de la cola quiere acceder a un recurso que está ocupado y no puede avanzar. Por ello, la descripción dada no deja claro el uso real de los canales virtuales en la función de encaminamiento. Un ejemplo de un mecanismo común que podría hacer uso de esos 2 canales virtuales sobrantes es el caso de usar *overlaping resource classes*[2][11, Capítulo 14], sin embargo, la condición esencial para el uso de este mecanismo es equivalente a la de los canales de escape[3], y los autores aseguran no usar canales de escape en su encaminamiento.

Tras la incertidumbre arrojada, se ha analizado el código fuente del simulador donde se realizan las pruebas en el trabajo original[4], y se ha encontrado un encaminamiento para la HyperX denominado *SkippingDimensionsRoutingAlgorithm* que se parece al Omni-WAR descrito en el paper pero no se encuentra algo que se ajuste al completo ni una configuración explícita indicada. Por ello, se ha decidido implementar el encaminamiento aplicando la restricción de utilizar múltiplos de $2n$ canales virtuales. El encaminamiento queda de la siguiente forma:

$$\begin{aligned}&\text{OMNI\_WAR}(a,b) = \text{ESCALERA\_MIN}(a,b) \cup \left[Q_{(a,s(a,\text{int},0))} \cap Q_h \text{ si } a_0 \neq b_0 \wedge \text{miss}_0 = 0\right] \\ &\cup \left[Q_{(a,s(a,\text{int},1))} \cap Q_h \text{ si } a_1 \neq b_1 \wedge \text{miss}_1 = 0\right].\end{aligned}$$

Al igual que en DAL, el conjunto de colas de salida es muy amplio y es necesaria una función de peso para la evaluación de las diferentes colas disponibles. En el artículo se hace mención a un *saltos restantes* $\times$ *ocupación*, pero es muy simple y no llega al rendimiento exigido, los autores dejan esa función de peso de como algo orientativo. En el código del simulador ya mencionado, el encaminamiento *SkippingDimensionsRoutingAlgorithm* tiene nueve

[2] Este concepto no se ha introducido en la memoria porque no se considera esencial para entender el trabajo y se introduce para evitar confusiones con la bibliografía mencionada.

[3] En el caso de usar 8 canales (6 + 2), se utilizan 1 + 2 canales por salto. El primer canal es completamente adaptativo, y pertenece a la relación de orden establecida en la escalera, donde el canal actual es mayor al último canal ordenado visitado. Los otros 2 canales sobrantes son utilizados como canales completamente adaptativos no ordenados, y por tanto, sin ninguna restricción. Por ello, el canal de la escalera actúa como un canal de escape en el encaminamiento.

[4] El código del simulador SuperSim puede encontrarse en: https://github.com/ssnetsim/supersim (visitado el 29/06/2022).

parámetros sin documentar que modelan la función de peso, lo cual es algo muy complejo de ajustar. Por ello, se ha optado por hacer una aportación y definir una función de peso nueva que trate de ajustarse a la siguiente premisa: para alcanzar el *throughput* máximo permitido por la red la ocupación de todas las colas debe de ser similar. Con esta idea se llega a la siguiente definición:

$$p_{\text{omni}}(q) = \begin{cases} \text{free}_{\text{q}} & \text{si } (\text{occ}_{\text{min}} - \text{occ}_{\text{media}} - 3c \geq 0) \wedge q \in Q_n, \\ \text{free}_{\text{q}} + 3\text{c} & \text{sino si } (\text{occ}_{\text{media}} + 3c \geq \text{occ}_{\text{min}}) \wedge q \in Q_m, \\ \text{free}_{\text{q}} + 3\text{c} & \text{sino si } q \in Q_m, \\ 0 & \text{en otro caso,} \end{cases}$$

donde $Q_m$ son colas mínimas a destino y $Q_n$ son no mínimas a destino, $c$ es el número de canales virtuales por salto en la escalera, $\text{occ}_{\text{min}}$ es la ocupación de las colas mínimas y $\text{occ}_{\text{media}}$ es la ocupación media excluyendo las mínimas. $\text{free}_{\text{q}}$ es el número de créditos de la cola $q$.

# 5 Evaluación

En este capítulo se muestran los resultados obtenidos en la evaluación de los encaminamientos descritos para la HyperX con una, dos y tres dimensiones. Para cada configuración, se han seleccionado tres tráficos: uno benigno y dos adversos, siendo de los dos tráficos adversos uno mas liviano que el otro. Además, se muestran los recursos utilizados y métricas del rendimiento por cada función de encaminamiento.

Los experimentos se agrupan de la siguiente forma. Para cada tráfico probado, se adjunta una tabla con el throughput obtenido en los encaminamientos seleccionados para la configuración, respecto al número de canales virtuales utilizados. Y en cada configuración, se adjunta una figura con el número de saltos promedio para cada encaminamiento y tráfico.

Las funciones de tráfico descritas en la Sección 1.5 reciben como input las coordenadas cartesianas del servidor fuente y devuelven las coordenadas cartesianas del servidor destino. Se incluye la coordenada del servidor, denotada $w$. Los diferentes tráficos para todas las dimensiones están descritos en las Tablas 5.2, 5.4, 5.6 con su nombre, función matemática del tráfico (todas las operaciones se hacen en módulo $k$) y características del tráfico sobre la red. Entre las características, se encuentra el throughput medio mínimo teórico, que indica el throughput medio que se debiera alcanzar usando encaminamiento DOR, y el número de saltos medio mínimo teórico, que indica el número de saltos medios que se dan usando encaminamiento mínimo. Además, se introducen las funciones:

- rand($n$): Devuelve un número entero aleatorio entre 0 y n-1. Estrictamente no es una función matemática, pero es razonablemente sencilla de implementar mediante un algoritmo.
- par($n$): Devuelve 0 si $n$ es un número par, 1 en caso contrario.
- $\neg n = (-n-1) \bmod k$. Es el complemento de $n$ respecto a $k$. Notar que si $k$ es par todos los $n$ tienen pareja diferente y con $k$ impar habría que utilizar otra función. En este trabajo siempre se utilizará un $k$ que sea par.

Las funciones de encaminamiento utilizadas se encuentran en las Tablas 5.1, 5.3 y 5.5. Las funciones están clasificadas como encaminamientos mínimos y no mínimos.

## 5.1 HyperX 1D

En este apartado se van a mostrar las métricas recogidas de las pruebas realizadas en una HyperX de 1 dimensión (grafo completo), con $k = 32$ y $c = 32$. Se van a evaluar las funciones de encaminamiento presentes en la Tabla 5.1 con los tráficos de la Tabla 5.2, donde el vector del servidor fuente está representado con las coordenadas ($w$, $x$) y el vector del servidor destino con las coordenadas ($w'$, $x'$). Exceptuando el Turn model no mínimo, los encaminamientos no mínimos presentados necesitan dos canales virtuales para poder funcionar. En el caso de Valiant, se escogerá un nodo aleatorio para ir por el primer canal virtual al nodo

intermedio, y por el segundo canal virtual del intermedio al destino. Respecto a DAL, utiliza un canal adaptativo no mínimo y otro de escape mínimo, y Omni-WAR utiliza el primer canal virtual para dar el primer salto, y si este ha sido no mínimo, utiliza el segundo para ir a destino. En la Figura 5.1 se ve el número de saltos promedio para cada tráfico y función de encaminamiento obtenidos en las simulaciones.

| Encaminamiento mínimo | Encaminamiento no mínimo |
|---|---|
| Mínimo [1] | Valiant<br>Omni-WAR<br>DAL<br>Turn Model no mínimo |

**Tabla 5.1:** Tabla de funciones de encaminamiento para HyperX 1D.

| Tráfico | Función de tráfico | $\overline{\text{Saltos}}$ | $\overline{\text{Throughput}}$ |
|---|---|---|---|
| Uniforme | $w' = \text{rand}(k),\ x' = \text{rand}(k)$ | $\frac{31}{32}$ | $\approx 1$ |
| Tornado | $w' = w,\ x' = x + \lfloor \frac{k-1}{2} \rfloor$ | 1 | $\frac{1}{32}$ |
| Tornado con paridad | $w' = w, x' = x + \lfloor \frac{k-1}{2} \rfloor + \text{par}(w)$ | 1 | $\frac{1}{16}$ |

**Tabla 5.2:** Tabla de tráficos para la HyperX 1D $k = 32$.

### 5.1.1 Tráfico Uniforme

El tráfico Uniforme se considera un tráfico benigno donde cada servidor inyecta paquetes a otro servidor elegido de forma aleatoria para cada paquete. En teoría, el tráfico se distribuye de igual forma por toda la red. Por esto, los encaminamientos mínimos se comportan mejor que los no mínimos aunque los encaminamientos adaptativos no mínimos deberían de poder acercase a ellos. Aquí el encaminamiento Valiant no obtiene un buen rendimiento debido a que la carga ya está distribuida por la red de forma natural. Se pueden ver los resultados en la Figura 5.2.

En la Figura 5.2 se muestra que el encaminamiento Mínimo con dos canales virtuales llega al rendimiento esperado. El throughput de DAL es igual a Mínimo en todas las configuraciones, y llega a un rendimiento máximo con dos canales virtuales. El throughput de Omni-WAR con dos canales virtuales se queda un 25% por debajo del mínimo, y con cuatro canales virtuales un 2% por debajo. Por ello, son recomendables cuatro canales virtuales para este encaminamiento. El problema de Omni-WAR con dos canales virtuales se atribuye a un problema de head-of-line-blocking, el cual, aparece inherentemente en las redes de comunicaciones y en el contexto de este trabajo es solventado añadiendo canales virtuales. Los encaminamientos

[1] Equivalente al DOR en una dimensión.

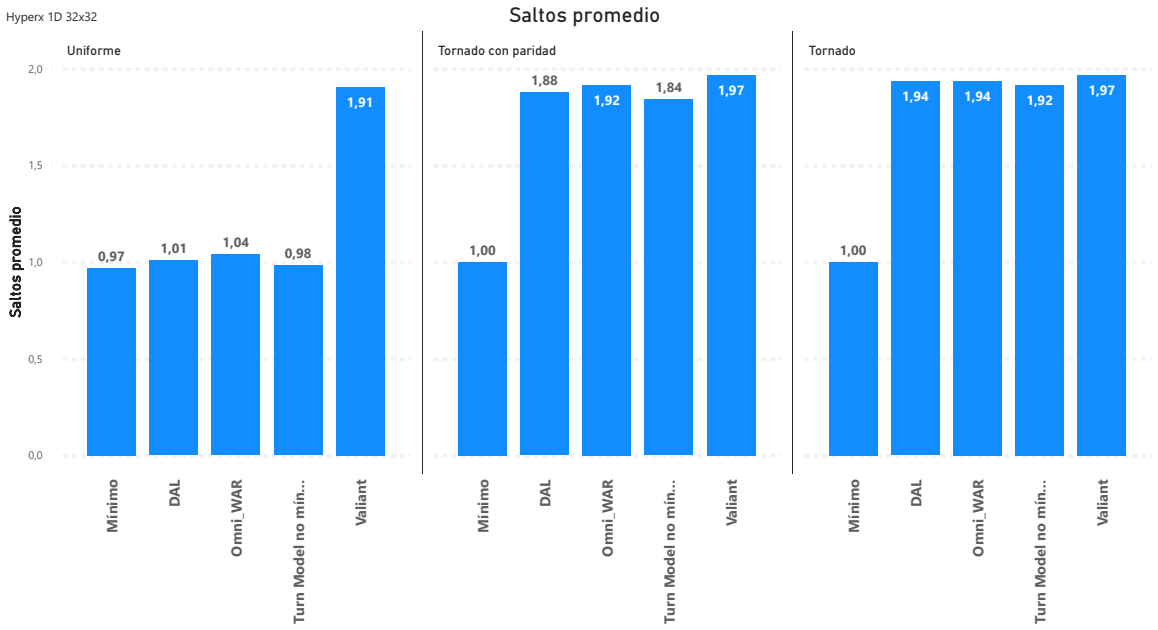


**Figura 5.1:** Resultados saltos promedio 1D.

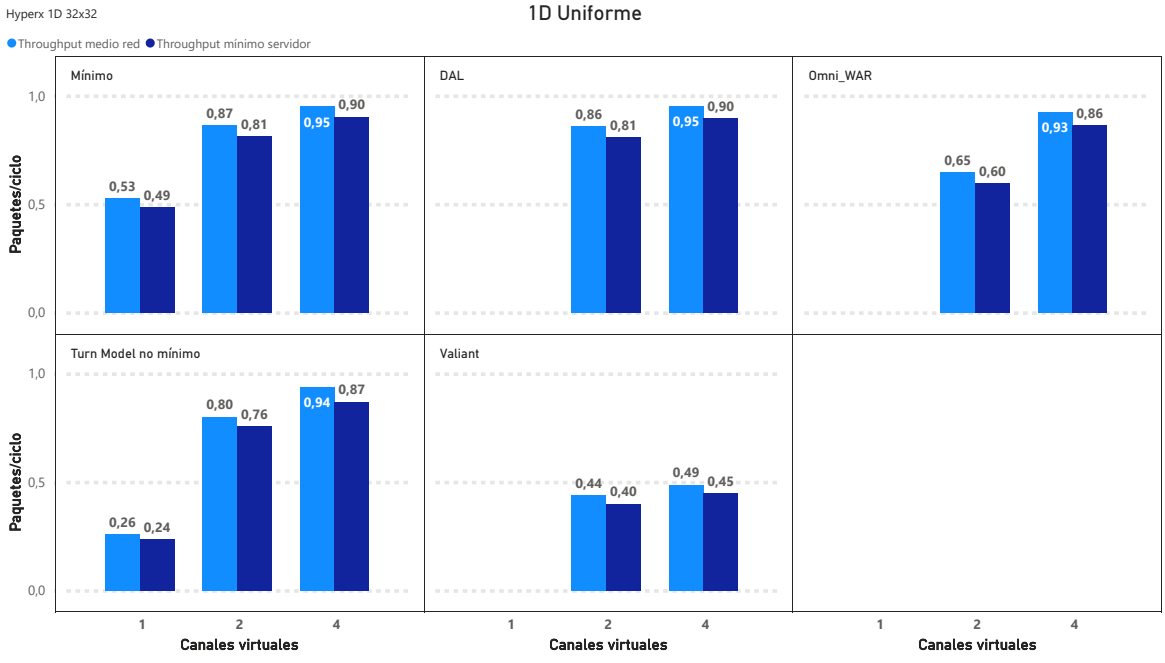


**Figura 5.2:** Resultados carga tráfico Uniforme 1D.

adaptativos no mínimos tienen un promedio de saltos similares al encaminamiento mínimo, como debe de ser. Esto se ve en la Figura 5.1.

### 5.1.2 Tráfico Tornado

El tráfico Tornado [34] es un tipo de tráfico adverso originalmente definido para una topología de *toro* [11, Capítulo 5] donde cada servidor envía paquetes a un vecino a distancia Tornado $\frac{k-1}{2}$. Este tráfico es adaptado a una HyperX de forma que los servidores de un mismo nodo inyectan paquetes a los servidores de un nodo adyacente. Simula algo parecido a una permutación de paquetes entre servidores de nodos diferentes. En los encaminamientos mínimos, todos los paquetes van sobre el mismo enlace provocando una sobre-suscripción de ese enlace, y mientras, dejan ociosos los enlaces que no llevan al nodo con los servidores destino. En este tráfico, Valiant da el rendimiento máximo debido a que utiliza todos los enlaces por igual. El objetivo de un encaminamiento adaptativo no mínimo sería acercase a este. Se pueden ver los resultados en la Figura 5.3.

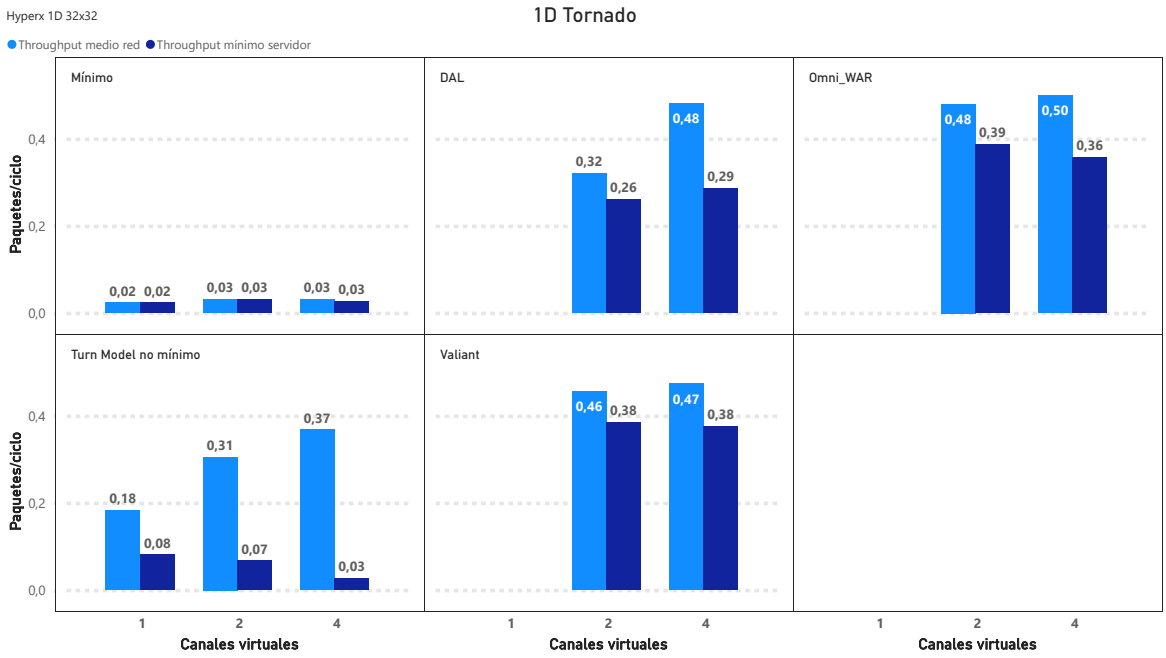


**Figura 5.3:** Resultados carga tráfico Tornado 1D.

En la Figura 5.3 se muestra que el encaminamiento Mínimo no funciona bien por los motivos previamente descritos. Sin embargo, los encaminamientos adaptativos no mínimos se asemejan a Valiant en throughput. En concreto, DAL con dos canales virtuales se queda a un 30% de Valiant, y con cuatro canales virtuales funciona un 2% mejor que Valiant. No llega al rendimiento esperado con dos canales virtuales debido a que solo puede hacer uso de un canal para ir completamente adaptativo y el otro canal virtual es el de escape. Este

último solo permite ir mínimo[2] a través de él y esto perjudica al rendimiento ante un tráfico adverso. Por ello, en esta situación se considera que DAL necesita cuatro canales virtuales para poder funcionar a un rendimiento aceptable. En el caso de Omni-WAR, con dos canales virtuales tiene un rendimiento un 4% por encima de Valiant, y con cuatro canales queda un 6% por encima. En ambos casos se considera que en Omni-WAR alcanza el rendimiento máximo, necesitando solo dos canales virtuales. Respecto al fairness, el throughput mínimo de DAL baja un 40% respecto el medio de la red, y Omni-WAR un 20%. Ambos tienen un fairness peor al de Valiant, y es muy pronunciada la caida en los dos. El número de saltos promedio en ambos encaminamientos se asemejan a Valiant, como se ve en la Figura 5.1. En el caso del Turn Model no mínimo, se aprecia que es un encaminamiento donde hay una gran diferencia entre el throughput medio y el mínimo. Esto es debido a que hay servidores en la red que no tienen la opción de ir no mínimo, mientras que otros sí. La única ventaja de este encaminamiento es que es un encaminamiento no mínimo que funciona sin necesidad de utilizar canales virtuales.

### 5.1.3 Tráfico Tornado con paridad

El tráfico Tornado con paridad es un tráfico adverso más benigno que el Tornado. Los servidores inyectores salen por dos enlaces diferentes del router fuente. En este tráfico, de nuevo, Valiant da el rendimiento máximo teórico y el objetivo de un encaminamiento adaptativo no mínimo seria acercase a este. Se pueden ver los resultados en la Figura 5.4.

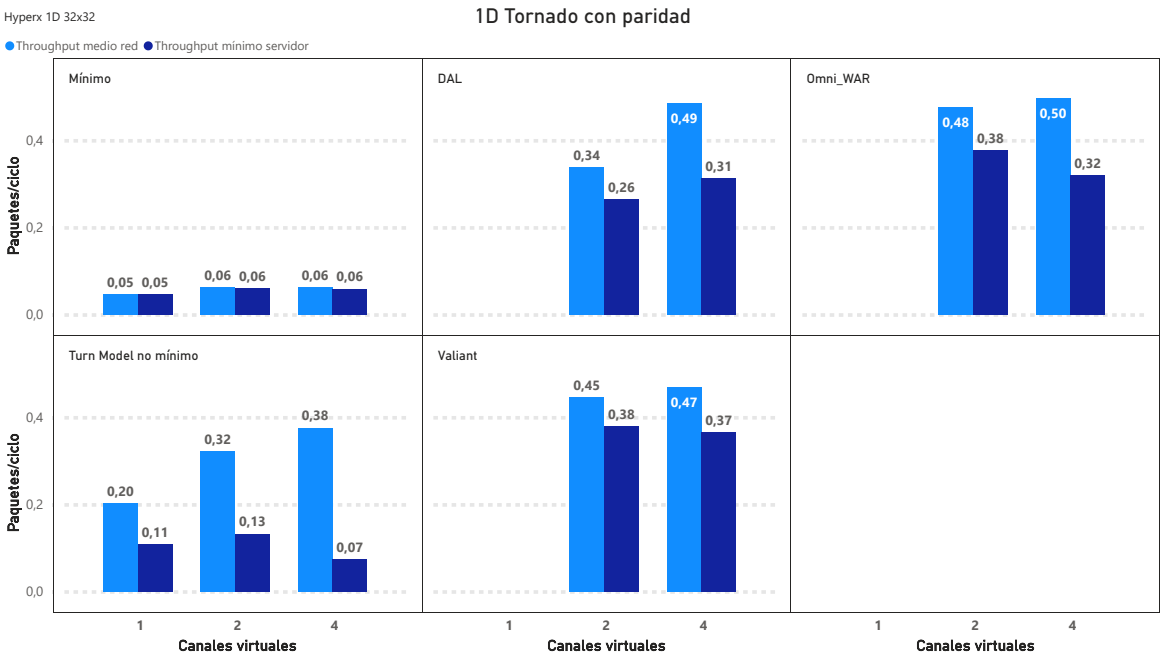


**Figura 5.4:** Resultados carga tráfico Tornado con paridad 1D.

[2]Esto es en el caso de un grafo completo, si no el canal de escape tiene un encaminamiento DOR.

En la Figura 5.4 se muestra que el encaminamiento mínimo da un rendimiento malo pero mejor que en el tráfico Tornado, como es de esperar. Respecto a Omni-WAR y DAL exceputando el fairness que están igualados, ambos siguen el mismo comportamiento que con el tráfico Tornado.

Analizando los rendimientos de DAL y Omni-WAR para la HyperX 1D, ninguno de los encaminamientos funciona bien con dos canales virtuales en todos los escenarios. Omni-WAR no llega al rendimiento para el tráfico Uniforme, y DAL no llega en los tráficos adversos. En el caso de utilizar cuatro canales virtuales los dos encaminamientos llegan al máximo en todos los tráficos, y están empatados. Sobre el unfairness, se puede apreciar que DAL tiene algo más de unfairness que Omni-WAR para el tráfico Tornado.

## 5.2 HyperX 2D

En este apartado se van a mostrar las métricas recogidas de las pruebas realizadas en una HyperX 2D, con $k = 10$ y $c = 10$. Se van a evaluar las funciones de encaminamiento presentes en la Tabla 5.3 con los tráficos presentes en la Tabla 5.4, donde el vector del servidor fuente está representado con las coordenadas $(w, x, y)$ y el vector del servidor destino con las coordenadas $(w', x', y')$.

Para evitar presentar resultados redundantes y simplificar la lectura, en la Figura 5.5 donde se presenta el número de saltos promedio por encaminamiento obtenidos en la experimentación, no se van a mostrar los saltos de todos los encaminamientos mínimos adaptativos, y solo se va a mostrar el encaminamiento Mínimo escalera.

| Encaminamiento mínimo | Encaminamiento no mínimo |
|---|---|
| DOR | Valiant |
| Escalera mínimo | Omni-WAR |
| Escape mínimo | DAL |
| O1Turn | |

**Tabla 5.3:** Tabla de funciones de encaminamiento para HyperX 2D.

| Tráfico | Función de tráfico | $\overline{\text{Saltos}}$ | $\overline{\text{Throughput}}$ |
|---|---|---|---|
| Uniforme | $w' = \text{rand}(k),\ x' = \text{rand}(k),\ y' = \text{rand}(k)$ | $\frac{18}{10}$ | $\approx 1$ |
| Swap2 | $w' = w,\ x' = x + \lfloor \frac{k-1}{2} \rfloor \text{par}(w),\ y' = y + \lfloor \frac{k-1}{2} \rfloor \text{par}(w+1)$ | 1 | $\frac{1}{5}$ |
| DCR 2D | $w' = \neg x,\ x' = \neg w,\ y' = \neg y$ | $\frac{19}{10}$ | $\frac{1}{10}$ |

**Tabla 5.4:** Tabla de tráficos para la HyperX 2D.

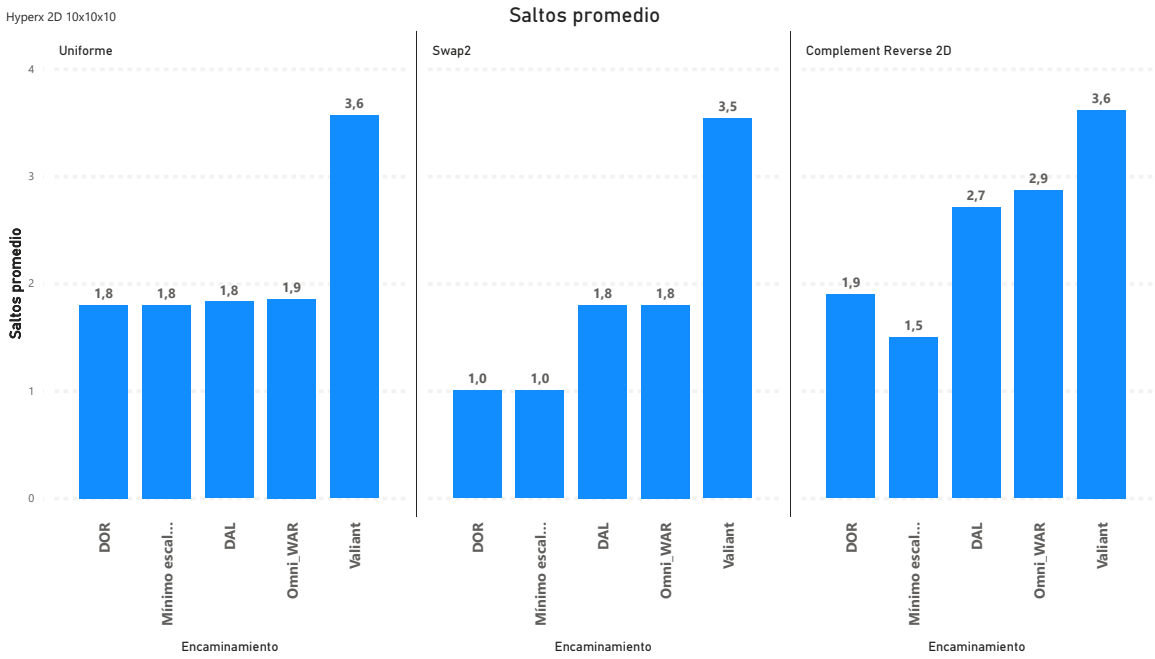


**Figura 5.5:** Resultados saltos promedio 2D.

### 5.2.1 Tráfico Uniforme

El tráfico Uniforme se considera un tráfico benigno donde cada servidor inyecta paquetes a otro servidor elegido de forma aleatoria para cada paquete. En teoría, el tráfico se distribuye de igual forma por toda la red. Por esto, los encaminamientos mínimos se comportan mejor que los no mínimos aunque los encaminamientos adaptativos no mínimos deberían de poder acercase a ellos. Aquí el encaminamiento Valiant no obtiene un buen rendimiento debido a que la carga ya está distribuida por la red de forma natural. Se pueden ver los resultados en la Figura 5.6.

En la Figura 5.6 se muestra que los encaminamientos mínimos son los que mejor se adaptan al tráfico. DAL funciona igual que el encaminamiento DOR en todas las configuraciones. El rendimiento de Omni-WAR también se asemeja al de los encaminamientos mínimos, pero con cuatro y ocho canales virtuales está un 1% por debajo que el DOR. Omni-WAR al menos utiliza cuatro canales virtuales por definición, mientras que DAL con dos canales virtuales ya puede funcionar. Los encaminamientos adaptativos no mínimos se asemejan a los encaminamientos mínimos en número promedio de saltos, como debe de ser. Esto se ve en la Figura 5.5. Por otro lado, el encaminamiento Turn Model mínimo no funciona igual que el resto de los encaminamientos mínimos porque hay inyectores que pueden enviar paquetes con mas adaptividad que otros, que desemboca en un uso desbalanceado de los recursos, afectando negativamente al throughput de la red.

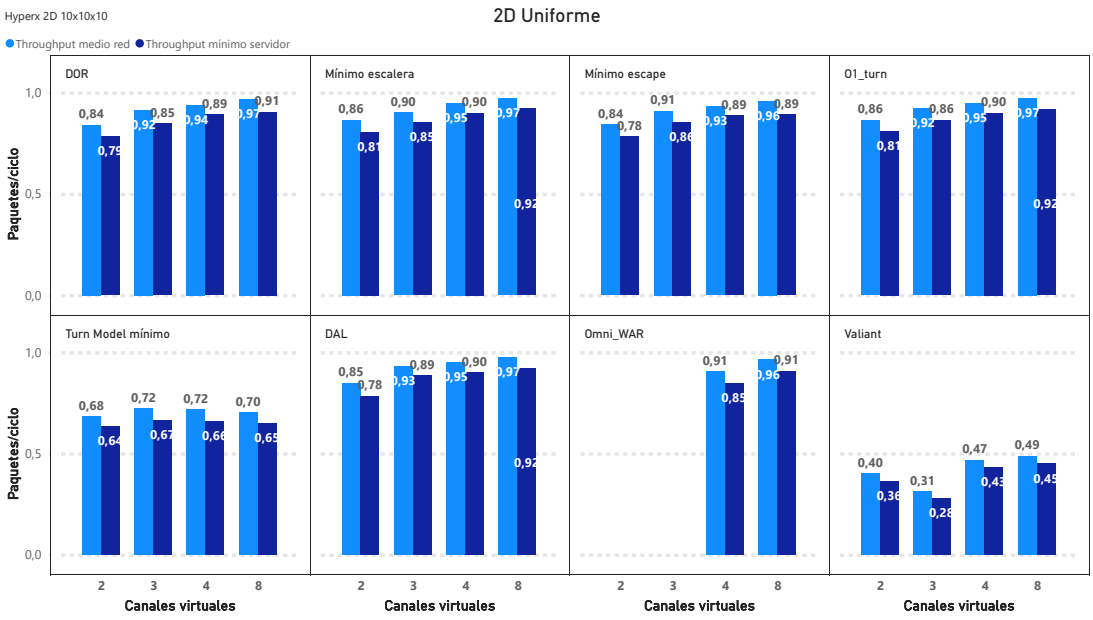


**Figura 5.6:** Resultados carga tráfico Uniforme 2D.

### 5.2.2 Tráfico Swap2

El tráfico Swap2 [6] es un tráfico adverso al que ni Valiant ni los encaminamientos mínimos se adaptan bien. Es un caso en el que Valiant no marca la cota máxima de rendimiento. Sin embargo, los encaminamientos adaptativos no mínimos no sobre-suscriben las bisecciones de la red y alcanzan el rendimiento máximo.

En la Figura 5.7 los encaminamientos mínimos no alcanzan mucho rendimiento, mientras que los no mínimos adaptativos explotan la red al máximo, sobrepasando a Valiant. DAL llega al rendimiento máximo a partir de tres canales virtuales. Omni-WAR con cuatro canales virtuales está un 6% por debajo de DAL, y con ocho alcanza el mismo rendimiento. Ambos encaminamientos alcanzan más de un 100% de rendimiento sobre Valiant. El número promedio de saltos en los encaminamientos adaptativos no mínimos se encuentra entre Valiant y los encaminamientos mínimos, como debe de ser. Valiant hace missrouting a un nodo no alineado con el destino y esto hace que de más saltos y no igualar a los adaptativos no mínimos ni en throughput ni en número de saltos promedio. Los encaminamientos mínimos, van siempre por el enlace mínimo, lo que les hace dar menos saltos dejando ociosos algunos enlaces de la bisección. Las medidas de los saltos promedios se encuentra en la Figura 5.5.

### 5.2.3 Tráfico Direct Complement Reverse 2D

El tráfico Direct Complement Reverse 2D es el tráfico mas adverso para esta configuración de los presentados en el trabajo. Es una adaptación a 2D del tráfico Direct Complement Reverse 3D. Aquí los encaminamientos mínimos funcionan mal y Valiant marca el rendimiento máximo que se puede esperar.

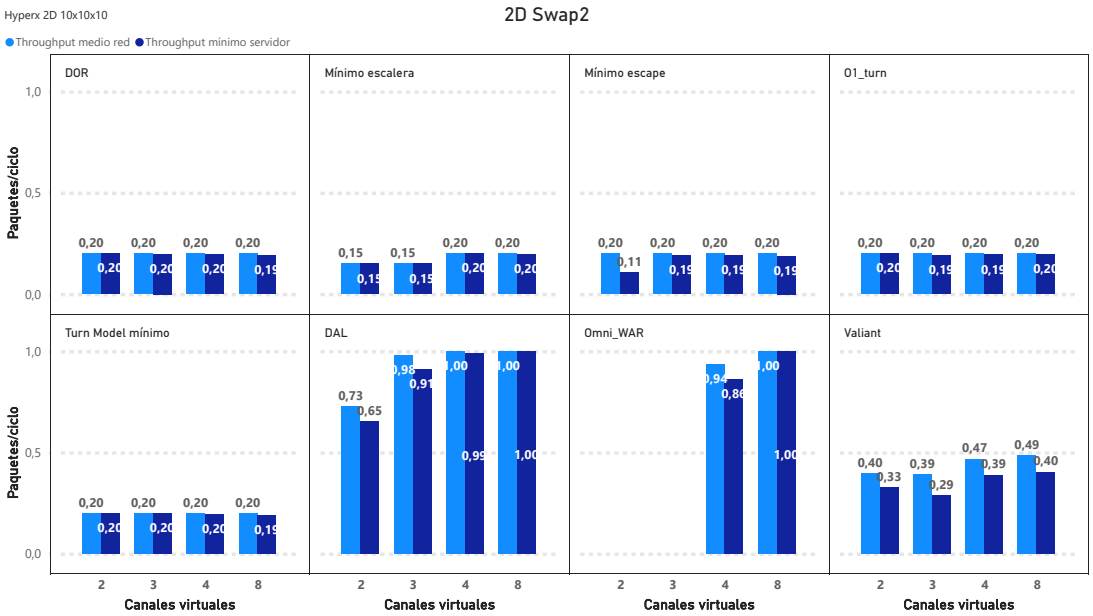


**Figura 5.7:** Resultados carga tráfico swap2 2D.

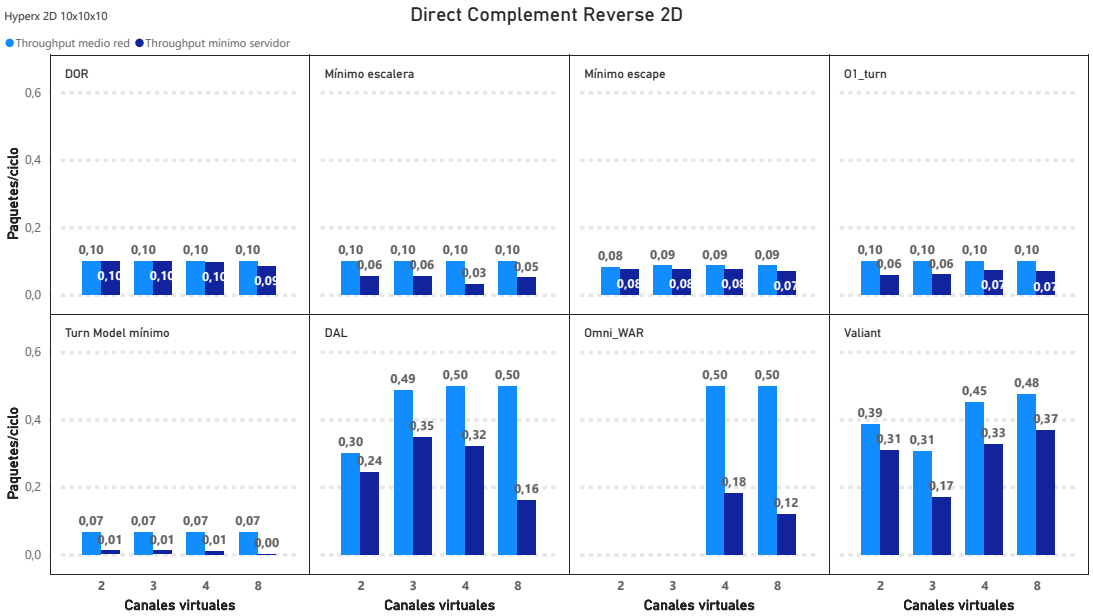


**Figura 5.8:** Resultados carga tráfico DCR 2D.

En la Figura 5.8 los encaminamientos mínimos alcanzan muy poco rendimiento respecto a los no mínimos adaptativos y Valiant. DAL con dos canales virtuales se queda a un 24% de Valiant, pero con tres canales virtuales ya alcanza el rendimiento máximo, superando por un 4% a Valiant con ocho canales virtuales [3]. Omni-WAR necesita cuatro canales virtuales por definición, y con ellos alcanza el rendimiento máximo, igual a DAL. Se observa en DAL y Omni-WAR que el throughput mínimo en la red es muy bajo en comparación con el promedio de todos los nodos, y este efecto aumenta según se añaden mas canales virtuales. En DAL, el throughput mínimo repecto al medio de la red baja un 68%, y en Omni-WAR un 76%. Ocurre un efecto similar con Mínimo escalera. Es probable que sea debido a que en este tráfico la distancia no es la misma para los paquetes que inyecta cada servidor. Algunos servidores inyectan solo paquetes a un salto del destino y otros inyectan paquetes que están a distancia dos. Esto puede dar lugar a que ciertos nodos tengan más throughput que otros lo cual afecta al fairness de la red. Las medidas de los saltos promedios se encuentra en la Figura 5.5. En el caso de Valiant, el fairness solo decae un 23%, lo cual parece un valor más razonable. Se ve que el promedio de saltos de Mínimo escalera es menor que el DOR, lo cual demuestra que los servidores que inyectan paquetes a distancia de un salto, están inyectando por encima de la media.

Analizando los rendimientos de DAL y Omni-WAR para la HyperX 2D, parece que DAL obtiene rendimientos óptimos con dos o tres canales virtuales, mientras que Omni-WAR necesita cuatro e incluso ocho para alcanzar el mismo rendimiento. Respecto al unfairness, DAL se comporta mejor que Omni-WAR en los tres tráficos, aunque el resultado de fairness obtenido en Direct Complement Reverse 2D no haya sido satisfactorio en ninguno de los dos tráficos.

## 5.3 HyperX 3D

En este apartado se van a mostrar las métricas recogidas de las pruebas realizadas en una HyperX 3D con $k = 10$ y $c = 10$. Se van a evaluar las funciones de encaminamiento presentes en la Tabla 5.5 con los tráficos presentes en la Tabla 5.6, donde el vector del servidor fuente está representado con las coordenadas $(w, x, y, z)$ y el vector del servidor destino $(w', x', y', z')$.

| Encaminamiento mínimo | Encaminamiento no mínimo |
|---|---|
| DOR<br>Escalera mínimo | Valiant<br>Valiant escalera<br>Omni-WAR<br>DAL |

**Tabla 5.5:** Tabla de funciones de encaminamiento para HyperX 3D.

[3] Se observa que Valiant con tres canales virtuales tiene una caida en el rendimiento debido a utilizar un número de canales impar, lo cual, desbalancea los recursos de la primera fase respecto a la segunda. Por ello, se pasa a comparar con el Valiant con mayores recursos y número de canales pares.

| Tráfico | Función de tráfico | $\overline{\text{Saltos}}$ | $\overline{\text{Throughput}}$ |
|---|---|---|---|
| Uniforme | $w' = \text{rand}(k),\ x' = \text{rand}(k),\ y' = \text{rand}(k),\ z' = \text{rand}(k)$ | $\frac{2538}{1000}$ | $\approx 1$ |
| Tornado_3dim | $w' = w,\ x' = x + \lfloor \frac{k-1}{2} \rfloor,\ y' = y + \lfloor \frac{k-1}{2} \rfloor,\ z' = z + \lfloor \frac{k-1}{2} \rfloor$ | 3 | $\frac{1}{10}$ |
| DCR 3D | $w' = w,\ x' = \neg z,\ y' = \neg y,\ z' = \neg x$ | $\frac{28}{10}$ | $\frac{1}{100}$ |

**Tabla 5.6:** Tabla de tráficos para la HyperX 3D.

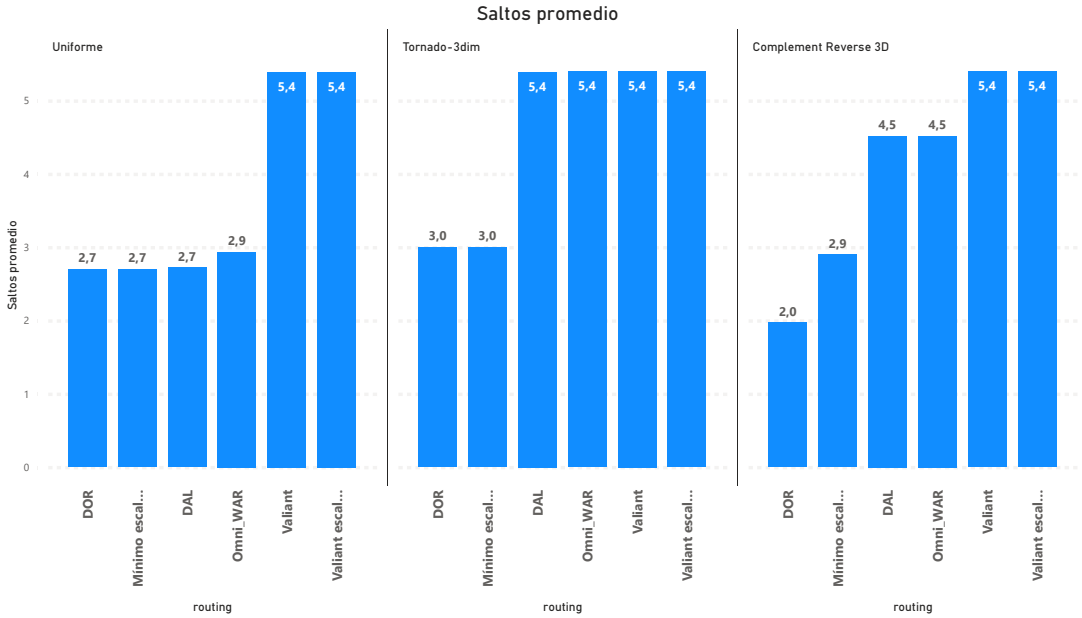


**Figura 5.9:** Resultados saltos promedio 3D.

### 5.3.1 Tráfico Uniforme

El tráfico Uniforme se considera un tráfico benigno donde cada servidor inyecta paquetes a otro servidor elegido de forma aleatoria para cada paquete. En teoría, el tráfico se distribuye de igual forma por toda la red. Por esto, los encaminamientos mínimos se comportan mejor que los no mínimos aunque los encaminamientos adaptativos no mínimos deberían de poder acercase a ellos. Aquí las métricas de los encaminamientos Valiant no son representativas debido a que la carga ya está distribuida por la red de forma natural. Se pueden ver los resultados en la Figura 5.10.

En la Figura 5.10 los encaminamientos mínimos alcanzan el throughput esperado y los adaptativos no mínimos se asemejan a los mínimos. DAL mejora el rendimiento un 2% respecto a Mínimo escalera con cuatro canales virtuales, alcanzando un rendimiento máximo. Omni-WAR con seis canales virtuales funciona un 5% peor que Mínimo escalera y necesita doce canales virtuales para igualar a DAL. Sobre el unfairness, se ve que en DAL el through-

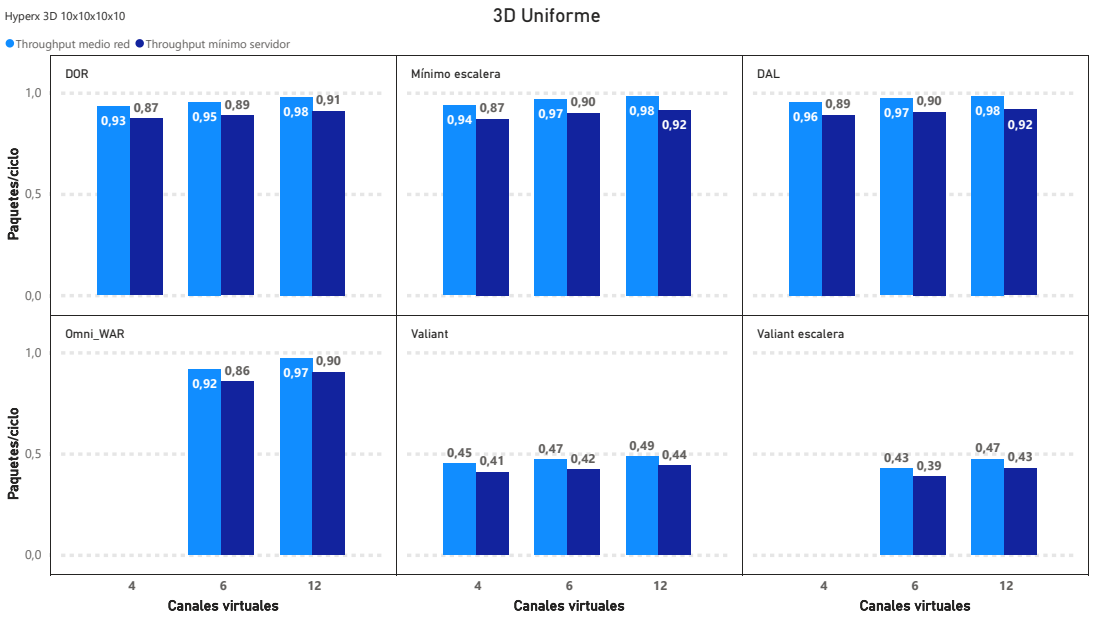


**Figura 5.10:** Resultados carga tráfico Uniforme 3D.

put mínimo cae un 6% respecto al promedio de la red, y en cambio Omni-WAR cae un 7%. Los encaminamientos adaptativos no mínimos se acercan también a los encaminamientos mínimos en número promedio de saltos, como debe de ser. Esto se ve en la Figura 5.9.

### 5.3.2 Tráfico Tornado 3-dimensiones

El tráfico Tornado 3-dimensiones es un tráfico adverso. Es una modificación del tráfico Tornado original, donde cada servidor envía a un destino que se encuentra a un desplazamiento Tornado por cada dimensión. Como en el tráfico original, los encaminamientos mínimos funcionan mal y los encaminamientos Valiant deberían de marcar el rendimiento máximo que se puede esperar.

En la Figura 5.11 se aprecia que los encaminamientos mínimos no aprovechan la capacidad de la red mientras que la mayoría de los no mínimos sí. Se aprecia que el encaminamiento Valiant, basado en DOR, no está alcanzando un rendimiento aceptable, al contrario de Valiant escalera. Se ha buscado en la literatura las razones específicas por las cuales este tráfico afecta tanto a Valiant, pero no se ha encontrado una causa precisa. Sobre los demás encaminamientos no mínimos adaptativos, DAL con cuatro canales virtuales alcanza el rendimiento máximo, con un 19% por encima de Valiant escalera. En el caso de Omni-WAR, este necesita seis canales virtuales por definición, y se coloca a la par de DAL. Sobre el unfairness, se ve que en DAL el throughput mínimo cae un 16% respecto al promedio de la red, y en cambio Omni-WAR cae un 26%. En el caso de Omni-WAR, esta caída se considera demasiado pronunciada para ser un tráfico donde el número de saltos de todos los flujos está equilibrado. En términos de número de saltos promedio, ambos encaminamientos dan el mismo número de saltos que Valiant escalera. Esto se puede ver en la Figura 5.9.

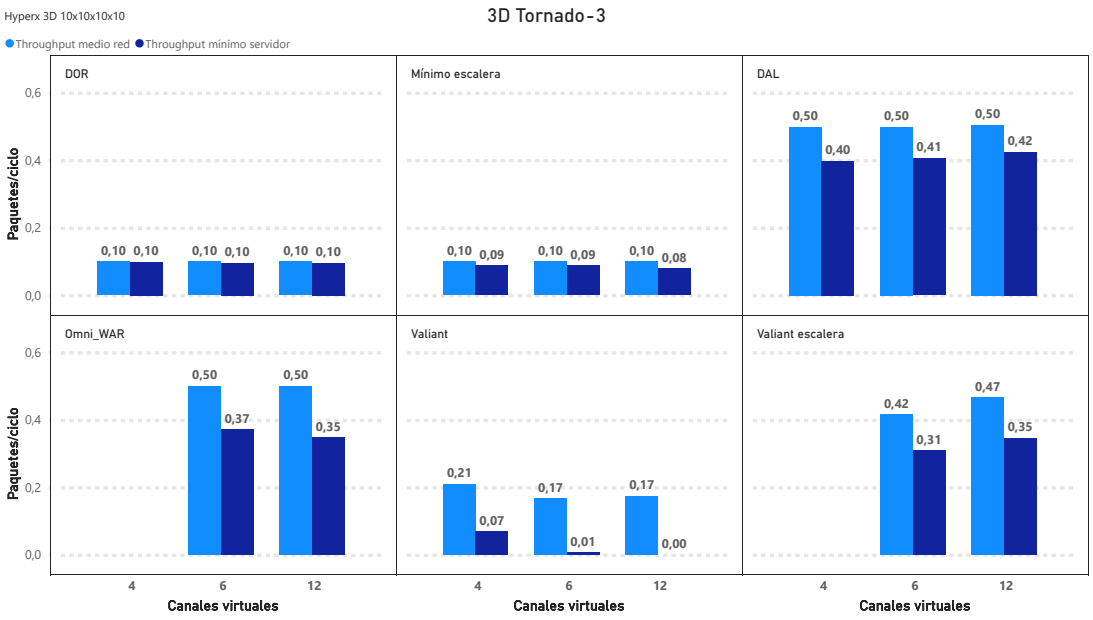


**Figura 5.11:** Resultados carga tráfico Tornado-3 3D.

### 5.3.3 Tráfico Direct Complement Reverse 3D

El tráfico Direct Complement Reverse 3D está considerado como el peor tráfico para una HyperX [10]. Un routing DOR en el tráfico Direct Complement Reverse para una HyperX de lado $k$ tiene un throughput de $\frac{1}{k^2}$. En este sentido, es el tráfico más adverso estudiado en este trabajo. Los encaminamientos Valiant deberían marcar el rendimiento máximo, a igualar por los encaminamientos no mínimos adaptativos.

En la Figura 5.12 los encaminamientos mínimos aprovechan muy poco los recursos de la red mientras que los no mínimos si que llegan al throughput medio teórico máximo. DAL funciona bien con cuatro canales virtuales, situandose un 16% por encima de Valiant. Sin embargo, con seis canales virtuales se degrada un rendimiento un 20%, y luego con doce canales virtuales vuelve a recuperar. Esta forma de valle entre las simulaciones puede que haya sido causada por el unfairness existente en este tráfico. En el caso de Omni-WAR, se necesitan seis canales virtuales por definición, y se coloca un 13% por encima de Valiant, alcazando un rendimiento máximo. Se pueden apreciar diferencias muy significativas entre el throughput promedio de la red y el throughput mínimo, principalmente en DAL. Respecto al unfairness, ambos encaminamientos decaen mucho. En el caso de DAL, el throughput mínimo cae un 79% respecto al promedio y en Omni-WAR un 62%. Hay flujos de paquetes que realizan diferente número de saltos por lo que esto puede influir en que determinados servidores acepten más paquetes que otros y conlleve a una red inestable. El fenómeno descrito es similar a lo que ocurre en el Direct Complement Reverse 2D. En el caso del encaminamiento DOR, el número de saltos teóricos no se ajusta con el real debido al unfairness existente en la red que hace que algunos servidores no reciban paquetes en toda la simulación.

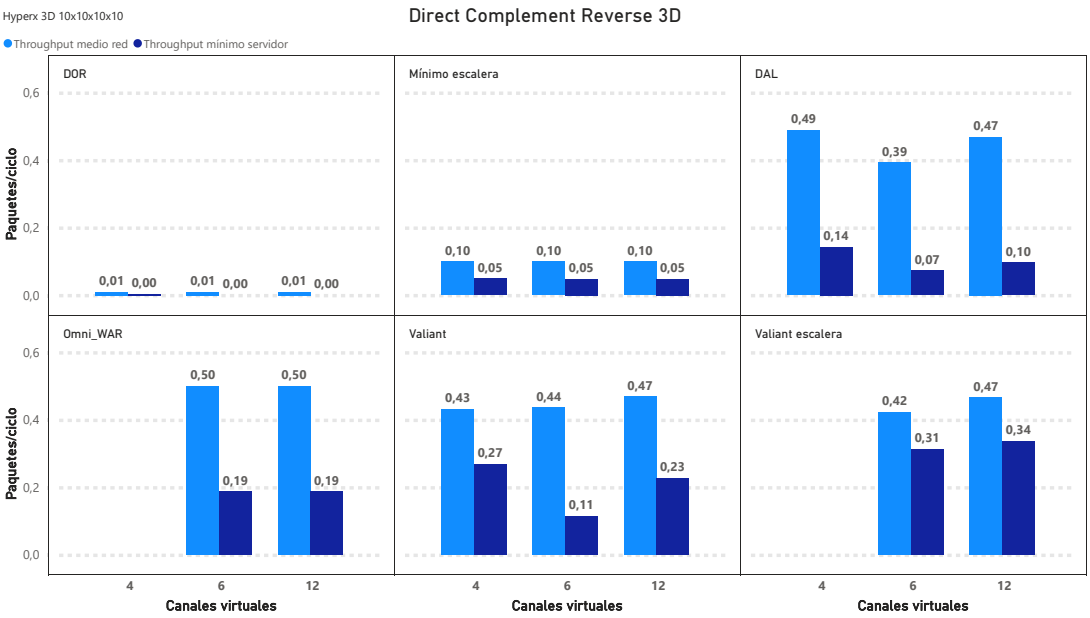


**Figura 5.12:** Resultados carga tráfico DCR 3D.

Analizando los rendimientos de DAL y Omni-WAR para la HyperX 3D, parece que DAL obtiene rendimientos óptimos con cuatro canales virtuales para todos los tráficos, mientras que Omni-WAR necesita seis porque con menos canales no está definido, y en algunos casos se requieren doce para igualar a DAL. Se puede observar que ambos encaminamientos padecen problemas de unfairness para el tráfico Direct Complement Reverse 3D, un poco mas pronunciado en DAL. Sin embargo, ninguno de los dos encaminamientos obtiene un rendimiento adecuado para este tráfico. En el caso de tráfico Tornado-3 y Uniforme, DAL tiene siempre un mejor fairness y throughput mínimo mayor al de Omni-WAR.

# 6 Conclusiones

En este trabajo se han descrito diferentes características de la topología HyperX como: el ancho de la bisección, el diámetro o la construcción del grafo de la red. Además, se ha implementado la topología en Booksim2 sin restricciones en el dimensionamiento de la red.

Tras las pruebas realizadas, se deduce que los encaminamientos completamente adaptativos no mínimos explotan los recursos de la red al máximo. En cambio, encaminamientos que no son completamente adaptativos porque aplican restricciones en los caminos para evitar el deadlock como: DOR, Turn Model, etc., no alcanzan rendimientos satisfactorios en muchas de las pruebas. Por ello, para aprovechar mejor la capacidad de la red en cualquier escenario, es recomendable utilizar como mecanismo de evitación de deadlock la escalera o el canal de escape, que se centran en añadir canales virtuales para evitar el deadlock, sin restringir rutas. DAL y Omni-WAR son dos encaminamientos modernos que usan uno de estos dos mecanismos en su implementación, y se considera que ambos aprovechan los recursos de la red al máximo en las pruebas realizadas. Sin embargo, los resultados obtenidos con el uso de estos encaminamientos muestran unfairness para algunos de los tráficos probados, y no se han encontrado trabajos que pongan el foco en este problema. Además, puede ser que la configuración del router utilizado, o de la red simulada, pueda dar lugar a amplificar este efecto. Esto puede ser objeto de estudio de otros trabajos. Según los datos analizados, DAL es el mejor encaminamiento probado en este trabajo ya que utiliza menos canales virtuales que Omni-WAR, obteniendo el mismo rendimiento de manera más económica.

Por otra parte, en los desarrollos de DAL y Omni-WAR se ha propuesto una función de peso diferente para cada encaminamiento. Proponer y ajustar una función de peso es un proceso difícil de confrontar debido a que éstas se acoplan al tamaño y dimensionamiento de la red. Por ello, se han probado las funciones para diferentes dimensionamientos de la red, y se ha tratado de hacer un ajuste general añadiendo parámetros y constantes a las funciones. Concluyendo, recientemente se ha publicado una propuesta [24] que toma otro acercamiento al problema, definiendo una función de peso que se ajusta en tiempo de ejecución a la red y puede ser objeto de interés para otros trabajos.

Una vez finalizado el trabajo, se considera que los tres objetivos propuestos en la Sección 1.2 se han cubierto, (1) se ha estudiado la topología HyperX e implementado en un simulador, (2) se han estudiado y evaluado en profundidad diferentes técnicas de evitación del deadlock y (3) se ha dado una implementación de los encaminamientos modernos propuestos y se ha analizado el rendimiento de estos.

Como trabajo futuro, se propone hacer pruebas con tráfico de aplicaciones reales, con métricas semejantes a las que se describen en [35]. Los tráficos y métricas utilizadas en este trabajo sirven para descartar diferentes encaminamientos, siendo pruebas necesarias en una primera fase de experimentación.

# Bibliografía


[1] E. W. Dijkstra, "Cooperating Sequential Processes", en 1968. dirección: https://www.cs.utexas.edu/users/EWD/ewd01xx/EWD123.PDF.

[2] E. Strohmaier, J. Dongarra, H. Simon, M. Meuer. "TOP500 - June 2022". inglés. (1 de jun. de 2022), dirección: https://www.top500.org/lists/top500/2022/06/ (visitado 26-06-2022).

[3] B. Meador, "A survey of computer network topology and analysis examples", *Washington University*, págs. 2-3, 2008. dirección: https://www.cse.wustl.edu/~jain/cse567-08/ftp/topology/index.html.

[4] J. Kim y H. Kim, "Router Microarchitecture and Scalability of Ring Topology in On-Chip Networks", en *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, ép. NoCArc '09, New York, New York: Association for Computing Machinery, 2009, págs. 5-10, ISBN: 9781605587745. DOI: 10.1145/1645213.1645217.

[5] Hewlett Packard Enterprise. "HPE Slingshot Interconnect". inglés. (2022), dirección: https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html# (visitado 26-06-2022).

[6] J. H. Ahn, N. Binkert, A. Davis, M. McLaren y R. S. Schreiber, "HyperX: topology, routing, and packaging of efficient large-scale networks", en *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, págs. 1-11. DOI: 10.1145/1654059.1654101.

[7] J. Kim, W. J. Dally y D. Abts, "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks", en *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ép. ISCA '07, San Diego, California, USA: Association for Computing Machinery, 2007, págs. 126-137, ISBN: 9781595937063. DOI: 10.1145/1250662.1250679.

[8] R. Lidl y G. Pilz, *Applied abstract algebra*. Springer Verlag, 1980.

[9] J. Domke, S. Matsuoka, I. R. Ivanov y col., "HyperX Topology: First at-Scale Implementation and Comparison to the Fat-Tree", en *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ép. SC '19, Denver, Colorado: Association for Computing Machinery, 2019, ISBN: 9781450362290. DOI: 10.1145/3295500.3356140.

[10] N. McDonald, M. Isaev, A. Flores, A. Davis y J. Kim, "Practical and Efficient Incremental Adaptive Routing for HyperX Networks", en *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ép. SC '19, Denver, Colorado: Association for Computing Machinery, 2019, ISBN: 9781450362290. DOI: 10.1145/3295500.3356151.

[11] W. J. Dally y B. P. Towles, *Principles and Practices of Interconnection Networks (The Morgan Kaufmann Series in Computer Architecture and Design)*, 1ª ed. Morgan Kaufmann, 2004.

[12] J. Rotman, *An Introduction to the Theory of Groups (Graduate Texts in Mathematics, 148)*, 4th. Springer, 1994.

[13] *Iverson bracket*, inglés, en 27 de mayo de 2022. dirección: `https://en.wikipedia.org/wiki/Iverson_bracket` (visitado 29-06-2022).

[14] W. Dally, "Virtual-channel flow control", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, nº 2, págs. 194-205, 1992. DOI: 10.1109/71.127260.

[15] T. Pinkston, "Deadlock Characterization and Resolution in Interconnection Networks", en *Deadlock Resolution in Computer-Integrated Systems*. Amsterdam University Press, 2018, págs. 445-491.

[16] G. Pifarré, L. Gravano, S. Felperin y J. Sanz, "Fully Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes, and Other Networks: Algorithms and Simulations", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 5, págs. 247-263, abr. de 1994. DOI: 10.1109/71.277792.

[17] J. Duato y T. Pinkston, "A general theory for deadlock-free adaptive routing using a mixed set of resources", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 12, págs. 1219-1235, ene. de 2002. DOI: 10.1109/71.970556.

[18] C. Glass y L. Ni, "The Turn Model for Adaptive Routing", en *[1992] Proceedings the 19th Annual International Symposium on Computer Architecture*, 1992, págs. 278-287. DOI: 10.1109/ISCA.1992.753324.

[19] J. Upadhyay, V. Varavithya y P. Mohapatra, "Routing Algorithms for Torus Networks", feb. de 2001.

[20] N. Jiang, D. U. Becker, G. Michelogiannakis y col., "A detailed and flexible cycle-accurate Network-on-Chip simulator", en *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, págs. 86-96. DOI: 10.1109/ISPASS.2013.6557149.

[21] M. Jeon, K.-N. Joo, T. Kim, S. Kim y C.-H. Youn, "FleX: A Flex Interconnected HPC System With Stochastic Load Balancing Scheme", *IEEE Access*, vol. 10, págs. 37 164-37 180, 2022. DOI: 10.1109/ACCESS.2022.3163542.

[22] G. Maglione-Mathey, J. Escudero-Sahuquillo, P. J. Garcia y F. J. Quiles, "Reducing the Impact of Interjob Interference in Dragonfly Networks Using Virtual Partitions", *IEEE Micro*, vol. 42, nº 3, págs. 50-56, 2022. DOI: 10.1109/MM.2022.3151258.

[23] Y. Ro, S. Jin, J. Huh y J. Kim, "Ghost Routing to Enable Oblivious Computation on Memory-centric Networks", en *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, págs. 930-943. DOI: 10.1109/ISCA52012.2021.00077.

[24] H. Kasan, G. Kim, Y. Yi y J. Kim, "Dynamic Global Adaptive Routing in High-Radix Networks", en *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ép. ISCA '22, New York, New York: Association for Computing Machinery, 2022, págs. 771-783, ISBN: 9781450386104. DOI: 10.1145/3470496.3527389.

[25] N. McKeown, A. Mekkittikul, V. Anantharam y J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", en *IEEE TRANSACTIONS ON COMMUNICATIONS*, 1996, págs. 296-302.

[26] S. Chacon y B. Straub, *Pro Git*, 2ª ed. Apress, 2014.

[27] SchedMD, *Slurm Workload Manager - Documentation.* dirección: `https://slurm.schedmd.com/documentation.html`.

[28] Bray, T., Ed., *The JavaScript Object Notation (JSON) Data Interchange Format , RFC 7159.* DOI: `10.17487/RFC7159`.

[29] Bhuyan y Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network", *IEEE Transactions on Computers*, vol. C-33, nº 4, págs. 323-333, 1984. DOI: `10.1109/TC.1984.1676437`.

[30] D. Seo, A. Ali, W. Lim, N. Rafique y M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks", en *ISCA 2005*, 2005.

[31] K. Gunther, "Prevention of Deadlocks in Packet-Switched Data Transport Systems", *IEEE Transactions on Communications*, vol. 29, nº 4, págs. 512-524, 1981. DOI: `10.1109/TCOM.1981.1095021`.

[32] L. G. Valiant y G. J. Brebner, "Universal Schemes for Parallel Communication", en *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, ép. STOC '81, Milwaukee, Wisconsin, USA: Association for Computing Machinery, 1981, págs. 263-277, ISBN: 9781450373920. DOI: `10.1145/800076.802479`.

[33] G. Kwauk, S. Kang, H. Kasan, H. Son y J. Kim, "BoomGate: Deadlock Avoidance in Non-Minimal Routing for High-Radix Networks", en *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, págs. 696-708. DOI: `10.1109/HPCA51647.2021.00064`.

[34] B. Towles y W. J. Dally, "Worst-Case Traffic for Oblivious Routing Functions", en *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, ép. SPAA '02, Winnipeg, Manitoba, Canada: Association for Computing Machinery, 2002, págs. 1-8, ISBN: 1581135297. DOI: `10.1145/564870.564872`.

[35] S. Chunduri, T. Groves, P. Mendygral y col., "GPCNeT: Designing a Benchmark Suite for Inducing and Measuring Contention in HPC Networks", en *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ép. SC '19, Denver, Colorado: Association for Computing Machinery, 2019, ISBN: 9781450362290. DOI: `10.1145/3295500.3356215`.