eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Análisis e implementación de redes de interconexión de diámetro bajo.

Autor

Jon Ander López Portugal

2019

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Análisis e implementación de redes de interconexión de diámetro bajo.

Autor

Jon Ander López Portugal

Director

Jose A. Pascual Saiz

Resumen

El objetivo de este trabajo de fin de grado es el estudio de las topologías de red Dragonfly, Slimfly y Megafly, así como de su implementación dentro del entorno de simulación Scheduling Simulation Environment (SSE).

Estas topologías han surgido en la última década y tienen como principal característica el diámetro bajo y la alta escalabilidad. Actualmente, supercomputadores que implementan Dragonfly se encuentran en puestos destacados del top 500, mientras que Megafly, propuesta por Intel, ésta pensada para dar el salto a la computación exaescalar.

El simulador SSE simula el proceso de scheduling en supercomputadores y permite conseguir un gran número de métricas, tanto del proceso de scheduling en sí, como de las características topológicas de las redes. Una vez implementadas las topologías, SSE permitirá simular un entorno real en el cual múltiples aplicaciones paralelas se ejecutan concurrentemente sobre la misma red y medir ciertas propiedades, como distancias medias y diámetros, de las particiones (conjuntos de nodos) asignadas a cada aplicación.

Índice general

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

Desde el inicio de la computación una de las principales limitaciones a la hora de ejecutar aplicaciones ha sido la capacidad de cómputo de los procesadores. Para superar esta limitación surgió la supercomputación que se centra en el desarrollo de potentes sistemas para la ejecución de aplicaciones paralelas las cuales se ejecutan de manera distribuida. Hoy en día estos sistemas están compuestos por miles o cientos de miles de nodos interconectados entre sí mediante una red de interconexión. Esta red es la encargada del intercambio de información entre las diferentes tareas que componen dichas aplicaciones y, debido a ello, juega un papel fundamental en el rendimiento que obtendrán. Desde el inicio de la supercomputación, se han propuesto y utilizado múltiples tipos de redes de interconexión cuyo objetivo ha sido penalizar lo menos posible el rendimiento de las aplicaciones paralelas que las utilizan.

Con el objetivo de que la red de interconexión no se convierta en un cuello de botella para el intercambio de información entre las tareas de las aplicaciones, se han desarrollado numerosas formas de interconectar los nodos de cómputo, las cuales se conocen como topologías de red. Cada topología de red tiene diferentes características/propiedades y para su diseño hay que tener en cuenta numerosos factores, tanto económicos como de rendimiento, entre los cuales podemos destacar tres: (1) el coste, (2) la distancia máxima entre los nodos (diámetro) y (3) la diversidad de caminos entre cada par de nodos.

Uno de los factores que más impacto tiene en la construcción de una topología es el coste económico que tendrá el despliegue físico de lo elementos necesarios entre los que se pueden destacar routers, nodos de cómputo con interfaces de red, cableado, etc. El ca-

bleado, es decir los enlaces de comunicaciones, es precisamente uno de los elementos que normalmente se ha intentando reducir, sin embargo, esta reducción puede penalizar el rendimiento debido a que, por ejemplo, el número de caminos entre nodos se verá también reducido. Otro de los factores que puede afectar al rendimiento es la distancia máxima entre cada par de nodos de cómputo debido a que puede afectar negativamente a la latencia media de la red incrementando considerablemente el tiempo necesario para mandar un mensaje entre dos nodos cualesquiera. El tercero de los factores es la diversidad de caminos entre los nodos debido a la posibilidad de que se generen cuellos de botella al saturarse ciertas conexiones (enlaces), es decir, ciertos caminos que se utilizaban más que otros. Por supuesto, existen técnicas proactivas y preventivas para evitarlos y/o reducir su impacto en el rendimiento de las aplicaciones entre los que se pueden destacar la optimización del encaminamiento entre nodos y/o la reducción de la distancia media de los nodos de cómputo.

En vista de los factores que afectan al coste y rendimiento de una red de interconexión, gran parte de la investigación reciente en el campo de la redes de interconexión se ha centrado en buscar la manera de reducir las limitaciones que tenían las topologías tradicionales. En el año 2008 se propuso de manera teórica la topología Dragonfly [Kim et al., 2008], cuyo principal objetivo era la reducción de la distancia media entre nodos, del diámetro y del número de enlaces (cables) necesario para su despliegue. La importancia de estas características para la construcción de supercomputadores se ve reflejada en el uso de dicha topología comercialmente por parte de Cray Inc. [Cray, 2012] a partir del año 2012. En particular, Cray ha incorporado Dragonfly como red de interconexión en las familias XC40 y XC50 que se encuentran incluidos dentro de varios supercomputadores, como son el Trinity y el Piz Daint y que actualmente ocupan de las primeras posiciones (sexta y séptima) en la lista Top500 [top500, 2018].

Desde el desarrollo de la topología Dragonfly se han realizado numerosas propuestas de nuevas topologías de red con el objetivo de mejorar las propiedades teóricas de ésta. Entre ellas podemos destacar Slimfly [Besta and Hoefler, 2014], topología que fue propuesta en 2014 cuyo objetivo es maximizar el número de conexiones entre nodos para reducir el diámetro de la red. Esta reducción tiene como efecto la reducción de la latencia media (mayor rendimiento) y la reducción del coste económico con respecto a la topología Dragonfly. Gracias a la topología Dragonfly ha sido posible la creación de supercomputadores *petascale*, es decir, capaces de realizar mas de 10^{15} operaciones de coma flotante por segundo (FLOPS).

Sin embargo, el siguiente reto para la supercomputación es la computación *exaescale*

(10^{16} FLOPS), la cual requerirá de la interconexión de miles o cientos de miles de nodos heterogéneos de cómputo basados en CPUs tradicionales, o en aceleradores como pueden ser GP-GPUS o FPGAS. Debido a esto, en 2018, con el objetivo de crear una topología adecuada para interconectar dichos nodos, Intel dio a conocer una nueva propuesta en la que se proponía una nueva topología, Megafly [Flajslik et al., 2018]. Esta topología está basada en la topología Dragonfly y, en teoría, mejora las propiedades topológicas de ésta como son el throughput y la escalabilidad. En particular, utilizando el mismo tipo de routers se consigue doblar el número de nodos de cómputo.

El objetivo de este trabajo de fin de grado (TFG) es el estudio y comprensión de las tres topologías anteriormente mencionadas (Dragonfly, Slimfly y Megafly) así como su implementación dentro del entorno de simulación Scheduling Simulation Environment (SSE). SSE simula el proceso de scheduling en supercomputadores y permite conseguir un gran número de métricas, tanto del proceso de scheduling en sí, como de las características topológicas de las redes. Una vez implementadas las topologías, SSE permitirá simular un entorno real en el cual múltiples aplicaciones paralelas se ejecutan concurrentemente sobre la misma red y medir ciertas propiedades, como distancia medias y diámetros, de las particiones (conjuntos de nodos) asignadas a cada aplicación.

Para la realización de la búsqueda de dichas particiones, SSE implementa algunos algoritmos clásicos y que no dependen de la topología como por ejemplo *first fit* o una búsqueda aleatoria. Sin embargo estos algoritmos no tienen en cuenta la topología de red utilizada por lo cual no explotaran la localidad que ofrecen las topologías descritas anteriormente. Por ello se ha decidido crear tres algoritmos de búsqueda, uno especifico para cada una de las tres topologías, que tenga en cuenta la disposición de los nodos con el objetivo de reducir la distancia media y el diámetro de los nodos asignados a cada aplicación.

2. CAPÍTULO

Documento de objetivos del proyecto

El objetivo principal del presente proyecto es el estudio e implementación de tres topologías de red en el simulador de scheduling SSE. Antes de iniciarse el proyecto, el simulador ya disponía de las siguientes topologías clásicas utilizadas para interconectar nodos de cálculo en supercomputadores: mallas, toros y arboles. Los nuevos módulos que se incorporarán al simulador implementarán las siguientes topologías de red: Dragonfly [Kim et al., 2008], Slimfly [Besta and Hoefler, 2014] y Megafly [Flajslik et al., 2018]. Así mismo se diseñarán e incorporarán políticas de búsqueda de nodos específicas para cada una de las topologías que tengan en cuenta la estructura física con el objetivo de reducir la distancia media y el diámetro de las particiones asignadas a las aplicaciones.

La primera tarea para la realización del proyecto será el estudio del funcionamiento del simulador para determinar cuáles son los pasos a seguir en la implementación de las nuevas topologías. Dentro de esta tarea una parte de vital importancia será el análisis de los módulos y algoritmos ya implementados con el objetivo de reutilizar código ya existente, por ejemplo, topologías y/o algoritmos de búsqueda similares.

La segunda tarea se corresponde con el estudio de las topologías de red a implementar. Para la realización de estas tareas se utilizarán los artículos de investigación en los que se propusieron y el objetivo será entender los pasos necesarios para su construcción y las propiedades topológicas que las caracterizan.

Una vez realizado el estudio conjunto del simulador y de las topologías, dará comienzo la fase de implementación en el simulador SSE. En esta fase se implementará la estructura de las topologías y las funciones asociadas que requiera el simulador como algoritmos de

asignación o cálculo de métricas específicas. Tras la implementación de cada topología se realizarán pruebas para determinar que las estructuras de datos que simulan la red física se han creado de manera correcta.

Después, se deben adecuar las métricas de cálculo del simulador para cada nueva topología teniendo en consideración las características únicas de las mismas. También, se deben crear los algoritmos específicos de selección de nodos adecuados a cada tipo de red.

Por último, se van a realizar varias comprobaciones finales, la primera de ellas, consistirá en la creación de un entorno de pruebas; la segunda, será una comparativa de los resultados obtenidos en el simulador con los resultados teóricos, y la tercera, una comparativa de distancia media entre los distintos tipos de búsqueda de nodos.

A continuación se presentan explícitamente los pasos a seguir para la realización del presente proyecto:

1. Planificación de los tareas necesarias para la realización del proyecto.
2. Análisis del diseño y funcionamiento del simulador Scheduling Simulation Environment (SSE).
3. Estudio de las topologías, así como de las funciones auxiliares necesarias para su inclusión en SSE.
4. Implementación de las topologías en SSE.
5. Implementación y adaptación de las métricas y de los algoritmos de búsqueda de nodos.
6. Diseño de los experimentos y análisis de los resultados.
7. Análisis del proyecto realizado y discusión sobre el posible trabajo futuro.

Para finalizar este capítulo hay que destacar que los puntos 3, 4 y 5 se realizarán de manera independiente para cada una de las 3 topologías en el siguiente orden: Dragonfly, Slimfly y Megafly.

3. CAPÍTULO

Topologías de red clásicas

En este capítulo se presenta un análisis de los tipos de topologías de red más habituales que se han utilizado para construir grandes sistemas de cómputo paralelo, así como algunas de las propiedades/métricas más utilizadas a la hora de evaluar el rendimiento. En particular, se describirán las topologías de tipo malla/toro así como las topologías de tipo árbol (fat tree), las cuales, pese a haber sido desarrolladas en la década de los 90, siguen siendo ampliamente utilizadas y proporcionan una gran rendimiento. Ejemplos de sistemas que las utilizan son las familias de supercomputadores de IBM Blue Gene/Q de IBM que implementa un toro de 5 dimensiones y Summit que utiliza un fat tree y que actualmente ocupa la primera posición en la lista Top500.

3.1. Topologías de red

La topología de red determina cómo se realizan las conexiones entre los diferentes elementos que componen un sistema paralelo, ya sean nodos de cómputo o routers. La estructura de la red se puede representar como un grafo y por lo tanto analizar desde el punto de vista de la teoría de grafos. En este caso los nodo de procesamiento y/o los routers representan los vértices del grafo, y los enlaces físicos (cables) los arcos.

Existen principalmente dos categorías de redes de interconexión o topologías: directas e indirectas:

- En las topologías **directas** cada nodo tiene una conexión directa a un subconjunto

del resto de nodos del sistema, los llamados nodos vecinos. En este caso, lo más habitual es que los nodos incluyan tanto la capacidad de cómputo como de comunicación (router). Un ejemplo de este tipo de topologías son las mallas/toros.

- En las topologías **indirectas**, en cambio, los nodos de cómputo están conectados a elementos de comunicación (routers), los cuales a su vez están conectados con otros switches. Los ejemplos más representativos de este tipo de topologías son los árboles en los cuales únicamente las hojas del árbol (nivel 0) se corresponden con nodos de computo y el resto de los elementos de otros niveles son switches encargados de las comunicaciones.

3.2. Propiedades/métricas de las topologías de red

A continuación se detallan algunas de las propiedades/métricas más utilizadas para medir el rendimiento de las topologías de red.

- La **distancia media** (d) representa la media de la suma de las distancias entre todos los pares de nodos de cómputo de la red. Para calcular la distancia entre dos nodos se contabiliza el número mínimo de saltos que hay que dar para llegar desde el nodo origen al nodo destino.

- El **diámetro** (D) se define como la distancia máxima que existe entre todos los pares de nodos de la red considerando la distancia mínima de cada par. El diámetro de la red está muy relacionado con la latencia máxima de la red. Estos dos parámetros, nos dan una primera aproximación a la latencia de las comunicaciones de la red cuando la comunicación entre los nodos es aleatoria.

- El **grado** de la red indica el número de enlaces que tiene un nodo de cómputo. Cuando el grado de todos los nodos es el mismo, entonces, la red se dice que es regular. En caso contrario la red se le denomina irregular. De cara a la construcción física de la red interesa que el grado sea lo más bajo posible con el objetivo de minimizar el número de conexiones lo cual reduce la complejidad de la implementación de la red.

- La **simetría** de una red determina la visión que los nodos tienen del resto. Cuando todos los nodos tienen la misma visión se dice que la red es simétrica. Esta propie-

dad es una característica deseable, ya que simplifica la elección de los caminos para la comunicación entre nodos.

- La **escalabilidad** de una red de comunicación indica lo fácil que es ampliar dicha red. Es una característica deseable en cualquier topología debido a que de este modo aumentar el número de nodos de cómputo es sencillo.

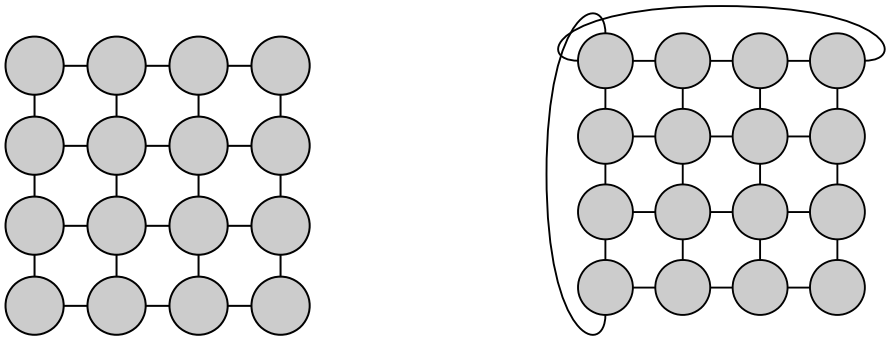
- La **tolerancia a fallos** de una red es una medida que nos indica la capacidad de la red para seguir funcionando cuando ocurren fallos, por ejemplo cuando algún componente de la red deja de funcionar. Se trata de una característica imprescindible cuando el número de nodos es elevado porque la probabilidad de fallo se incrementa.

- La **bisección de la red** representa el número mínimo de enlaces que se deben eliminar para dividir la red en dos partes iguales. Este parámetro representa el tráfico máximo (throughput) que puede soportar la red, el cual es conocido como ancho de banda de la bisección.

3.3. Topologías de tipo malla y toro

Las mallas y los toros son topologías directas resultantes de generalizar la cadena y el anillo respectivamente a n dimensiones. Como su propio nombre indica, la cadena se trata de un topología de red de una dimensión en la cual cada nodo está conectado con otros dos (excepto los de las esquinas). El anillo, mejora esta topología de una dimensión creando un enlace entre el nodo inicial y el nodo final para crear un camino alternativo. Esto repercute en una menor distancia media y diámetro así como en una mayor tolerancia a fallos. En la Figura 3.1 se puede ver un ejemplo de malla 2D (3.1a) y de toro 2D (3.1b).

En los últimos años una de las topologías más usada ha sido el toro, debido a las claras ventajas que posee sobre la malla. En particular se han utilizado toros de $n = 3$ dimensiones para construir los sistemas Titan [Alverson et al., 2010] de Cray y los Blue Gene/L y P [Adiga et al., 2005] de IBM. Más recientemente, IBM actualizó la serie Blue Gene con el modelo Q [Chen et al., 2011] que utiliza como red de interconexión un toro de $n = 5$ dimensiones.



(a) Malla 2D **(b)** Toro 2D

Figura 3.1: Ejemplo de dos topologías de tipo cubo de 2 dimensiones.

3.3.1. Propiedades topológicas

En esta sección se va a analizar cuáles son algunas de las propiedades topológicas (ver Sección 3.2) de las mallas y toros. Para ello, solo se van a considerar topologías regulares, es decir, aquellas que disponen del mismo número de nodos en cada dimensión.

En ambos casos, la cantidad de nodos de la red es $N = K^n$ donde n es el número de dimensiones y K es el número de nodos por dimensión. El grado de ambas topologías es $2n$, es decir, existen dos enlaces por cada dimensión lo cual es un valor bajo. Con respecto a la simétrica, hay que destacar que mientras el toro sí que es una topología simétrica, la malla no lo es ya que los nodos de los extremos en cada dimensión tienen menos enlaces que el resto.

Otra propiedad importante de las redes de interconexión es la tolerancia a fallos, la cual es alta en este tipo de topologías debido a que el número de caminos para ir de un nodo origen a un nodo destino es alto. Para dividir la red en dos partes iguales (bisección), es necesario eliminar k enlaces en la malla y $2k$ enlaces en el toro (debido a los enlaces que unen los nodos extremos de cada dimensión).

Para concluir esta sección se presentan dos propiedades muy importantes relacionadas con la distancia. La primera propiedad es la distancia media que se calcula como $n \times (K/3)$ en la malla y como $n \times (K/4)$ en el toro cuando K es par. Por otro lado, la segunda propiedad es el diámetro el cual se calcula para las mallas y toros como $n \times (K-1)$ y $n \times K/2$ respectivamente. Ambas distancias son claramente menores en el caso de las topologías de tipo toro, sobre todo, el diámetro el cual se va a traducir en un mayor rendimiento de este tipo de topologías. De hecho, las topologías de tipo malla únicamente fueron utilizadas en los primeros supercomputadores, pero fueron sustituidas por los toros rápidamente.

Sin embargo, el hecho de que el diámetro de la red dependa del número de nodos por dimensión hace que escalar estas redes no sea eficiente, debido a que el diámetro de redes de interconexión usando estas topologías será alto. Lo cual repercutirá en la latencia media de las comunicaciones.

3.4. Topologías de tipo árbol

Otro tipo de topologías muy utilizadas en el ámbito de la supercomputación son los árboles. Existen topologías tipo árbol que pueden estar estructuradas de diferentes formas, sin embargo, la estructura más utilizada es aquella en la que los nodos de cómputo se colocan en las hojas del árbol (nivel 0) y los routers en el resto de niveles del árbol. Debido a esto, como se ha explicado en la Sección 3.1 los árboles son topologías indirectas. En la figura 3.2 está representado un árbol de tres niveles en el que se pueden ver cuatro nodos de cómputo conectados a cada router del nivel 0.

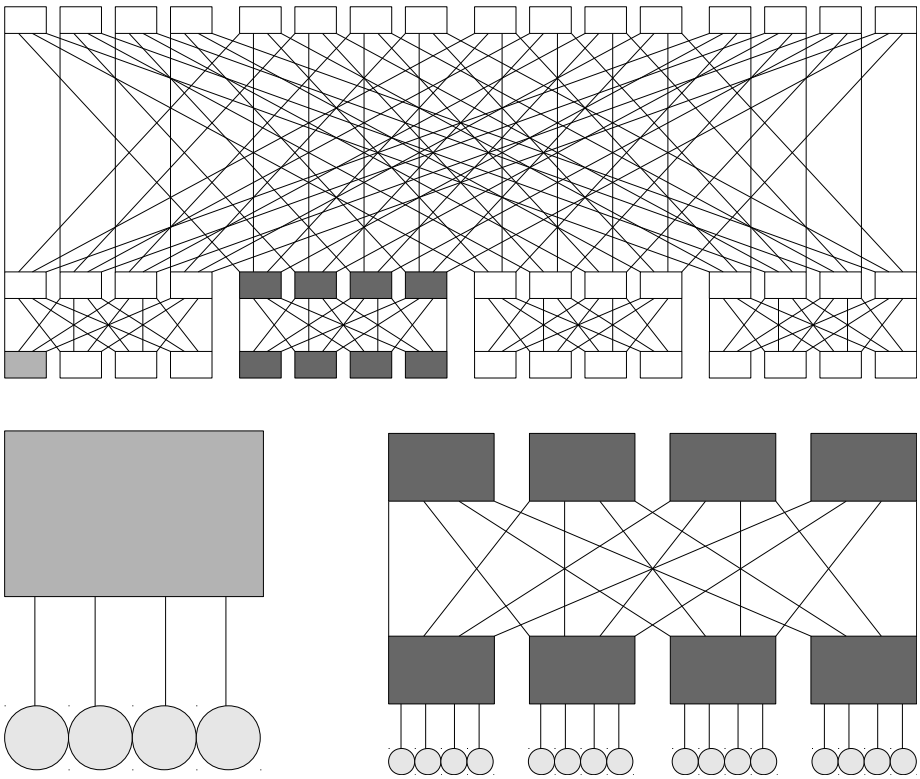


Figura 3.2: Ejemplo de un fat tree de tres niveles.

3.4.1. Propiedades topológicas

En principio, la topologías de tipo árbol no son regulares, ya que los routers del nivel superior y del primer nivel (hojas) tienen menos enlaces que los routers colocados en los niveles intermedios (ver parte superior de la Figura 3.2). Sin embargo, si los procesadores únicamente están en las hojas del árbol, la visión que tiene cada uno de ellos de la red es la misma, por lo que se puede decir que, para los procesadores, las topologías de tipo árbol son regulares.

El grado de este tipo de topologías es k, el número de nodos que salen de cada nodo (como excepción, el grado no representa en este caso el número de enlaces). Así, el número de niveles del árbol resulta ser $log_k P$.

Considerando las comunicaciones entre nodos, este tipo de topologías están muy desequilibradas. Por ejemplo, todas las comunicaciones que se generan desde nodos que se encuentran en la mitad derecha del árbol, y que vayan dirigidas a nodos de cómputo situados en la mitad izquierda, tiene que llegar hasta el nivel superior y luego bajar hasta los nodos destino. Esto puede causar que esa zona de la red se sature, y la convierte en punto de fallo crítico.

Para superar ese problema, las redes en forma de árbol que se utilizan en la actualidad disponen de mayor cantidad de recursos según se avanza hacia el nivel superior, con lo que la capacidad para soportar el tráfico de comunicaciones es mayor en los niveles superiores. A este tipo de árboles se les conoce como árboles densos o *fat trees*. El ejemplo representado en la Figura 3.2 es un ejemplo de árbol denso de 3 niveles. Por lo tanto, la bisección de un árbol denso es de $P/2$.

A continuación, al igual que se ha hecho con las topologías de tipo toro y malla, se presentan dos propiedades relacionadas con la distancia siendo k el grado del árbol. La primera propiedad es la distancia media que se calcula como $\frac{2P}{P-1} \times log_k P - \frac{2}{k-1}$ y la segunda propiedad es el diámetro el cual se calcula como $2 \times log_k P$. Ambas distancias dependen del número de niveles del árbol por lo que según se vaya aumentando el número de nodos que componen el árbol, la distancia media y el diámetro aumentarán.

4. CAPÍTULO

Topologías de red de diámetro bajo

En este capítulo se describen tres de las topologías de red propuestas para la construcción de redes de interconexión en la última década. El diseño de estas redes tiene claramente un objetivo, minimizar el diámetro lo cual causará una reducción de la latencia máxima de la red. Para conseguirlo, el diseño se basa, en contraposición a las topologías clásicas, en el uso de routers de radio alto lo cual permite conectar múltiples nodos de cómputo y de comunicaciones a cada router. Cabe destacar que la primera de ellas, la topología Dragonfly, ha sido utilizada en numerosos sistemas y actualmente 2 de los 10 primeros sistemas de la lista Top500 la utilizan.

4.1. Topología Dragonfly

Dragonfly es una topología de red enfocada a obtener un alto rendimiento y escalabilidad manteniendo un bajo coste de fabricación. Para ello utiliza routers de alto grado con el objetivo de minimizar el cableado necesario para su despliegue. Esta red de interconexión es una topología directa la cual está formada por dos niveles de routers. Dichos routers se encuentran divididos en tres niveles jerárquicos en los que se usan distintos planteamientos para la construcción de la red.

El planteamiento para la construcción de esta topología es el siguiente: en el nivel más bajo tenemos **grupos** de routers que forman una red completamente conectada e independiente, es decir, cada router en un grupo se conecta mediante enlaces locales con el resto de routers del mismo grupo. A su vez, los nodos de cómputo se encuentran conectados a

cada uno de estos routers. Cada grupo de routers es visto por el resto de grupos como un único router virtual. Este nivel se encuentra representado en la figura 4.1.

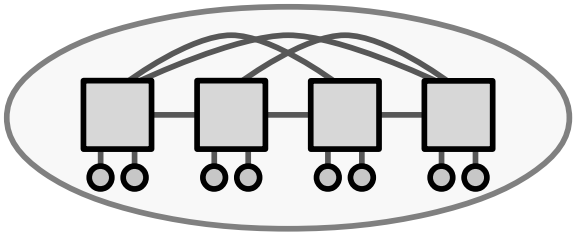


Figura 4.1: Ejemplo de un grupo en la topología Dragonfly.

El segundo nivel se construye conectando cada router de un grupo con otros routers de otros grupos utilizando para ello múltiples conexiones globales, es decir, está compuesto por los distintos grupos (o routers virtuales) interconectados entre sí mediante enlaces globales. Esto provoca que el resto de routers de cada grupo este también comunicado con cualquier otro router de la red vía el router virtual.

Este tipo de interconexión tiene como efecto la minimización del número de conexiones globales, y en consecuencia, la reducción del coste. Además, también tiene efectos positivos sobre la latencia, debido a que solo hay que realizar un salto cuando se tenga que ir de un nodo de procesamiento del grupo i al grupo j. En la Figura 4.2 se ha representado el uso de enlaces globales para la unión de los routers virtuales.

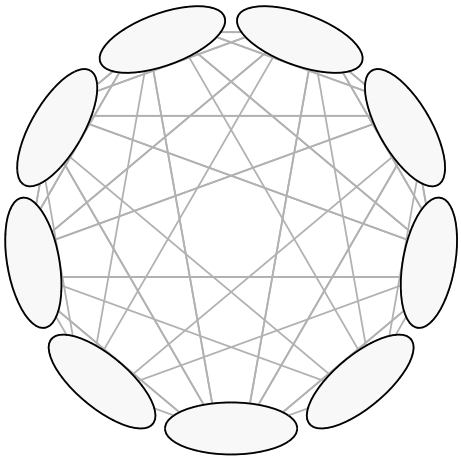


Figura 4.2: Ejemplo de interconexiones de los switch virtuales en la topología Dragonfly.

El tercer nivel de la jerarquía es el propio sistema y está representado en la figura 4.3.

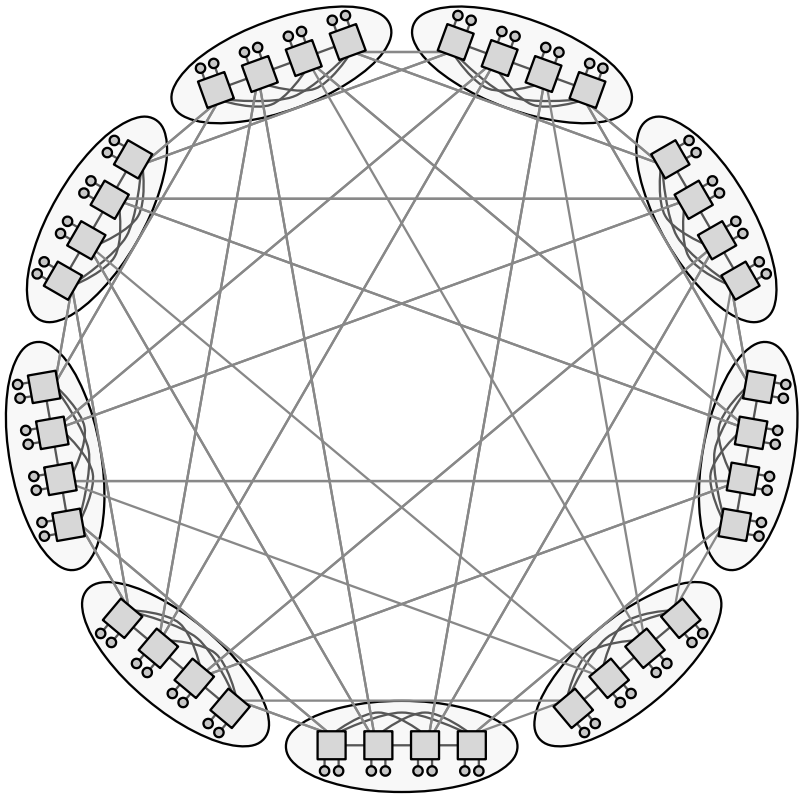


Figura 4.3: Ejemplo de una Dragonfly de 72 nodos divididos en 9 grupos.

La definición de la red requiere únicamente de tres parámetros a, p y h, y a partir de ellos se calculan el resto de las características de la topología. A continuación se describen los parámetros que definen una topología de red Dragonfly con la notación utilizada en el artículo original [Kim et al., 2008]:

a : Número de routers por grupo.

p : Número de nodos de cómputo unidos a cada router.

h : Número de enlaces globales por cada router.

k : Grado del router (número de puertos de interconexión).

g : Número de grupos.

N : Número de nodos de cómputo en la red.

Nótese que el número de nodos (N) de este tipo de topologías no puede ser elegido libremente debido a que depende de los parámetros presentados anteriormente. A continuación de describen las fórmulas para calcular las características: k, g y N.

$$N = p * a * (a * h + 1)$$

$$g = a * h + 1$$

$$K = p + h + (a - 1)$$

La elección de los parámetros a, p y h se puede realizar de manera arbitraria, lo que puede generar redes desbalanceadas que utilicen de forma excesiva los enlaces globales. Con el objetivo de evitarlo se recomienda el uso de valores que cumplan las siguientes inecuaciones:

$$a \geq 2h, 2p \geq 2h$$

En particular, cuando $a = 2p = 2h$ se considera que la topología está balanceada. Esto se debe a lo siguiente: aunque la red tenga tres grados de libertad (a, p y h) para que el uso de los enlaces locales y globales esté correctamente balanceado es necesario establecer una dependencia entre ellos. Esto se consigue haciendo que el número de enlaces locales de cada router sea el doble que el de los enlaces globales. En caso contrario, alguno de los dos tipos de enlaces será más restrictivo que el otro y limitará el rendimiento global de la red.

El diámetro de esta topología, usando la distancia mínima entre nodos, es 5 y, a diferencia de la topologías vistas en la sección anterior, no depende del tamaño de la red. Dicho camino comprende un salto local desde el nodo de cómputo de origen al router origen, un salto local entre el router origen y otro router del mismo grupo conectado al grupo destino, un salto global hacia el grupo destino, un salto local hacia el router destino y un salto local final al nodo de cómputo destino. Por lo tanto, la distancia máxima entre routers es 3.

Para construir una Dragonfly de 72 nodos balanceada como la que aparece en la imagen 4.3 con cuatro routers por grupo, se debe tener en cuenta las fórmulas anteriormente mencionadas para establecer la dependencia entre los tres parámetros (grados de libertad). Por lo tanto, cada grupo está compuesto por 4 routers y cada router por dos nodos y dos enlaces globales, es decir: $a = 4, p = 2, h = 2$, cumpliéndose la fórmula: a = 2p = 2h.

4.2. Topología Slimfly

Slimfly [Besta and Hoefler, 2014] es una topología de red enfocada a obtener el máximo rendimiento manteniendo un bajo coste de fabricación. Para ello, esta topología intenta maximizar la utilización de recursos mientras minimiza la cantidad de saltos entre nodos, consiguiendo un diámetro de 4 entre cualquier nodo de la red (2 entre los routers). Slimfly fue desarrollada con el objetivo de mejorar las características de la topología Dragonfly.

Slimfly es una topología directa compuesta por dos niveles de routers, orientada al uso de routers de alto rendimiento. Para la creación de la red se utiliza una familia de grafos bipartitos propuesta en [McKay et al., 1998] a la que denominan *SF MMS*, la cual está compuesta por grupos de nodos que se entrelazan entre las dos partes del grafo como se representa en la Figura 4.4.

Las uniones entre los routers se establecen en dos niveles. En el primer nivel se encuentran las uniones entre los routers que pertenecen al mismo grupo 4.4a. En el segundo nivel se encuentran las uniones entre los diferentes routers de las dos distintas partes que conforman el grafo bipartito siguiendo la construcción de un grafo SF MMS 4.4b.

A continuación se describe la notación que se utiliza para describir la topología Slimfly:

N : Número de nodos en toda la red

p : Número de nodos conectados a un router

k' : Número de interconexiones con otros routers

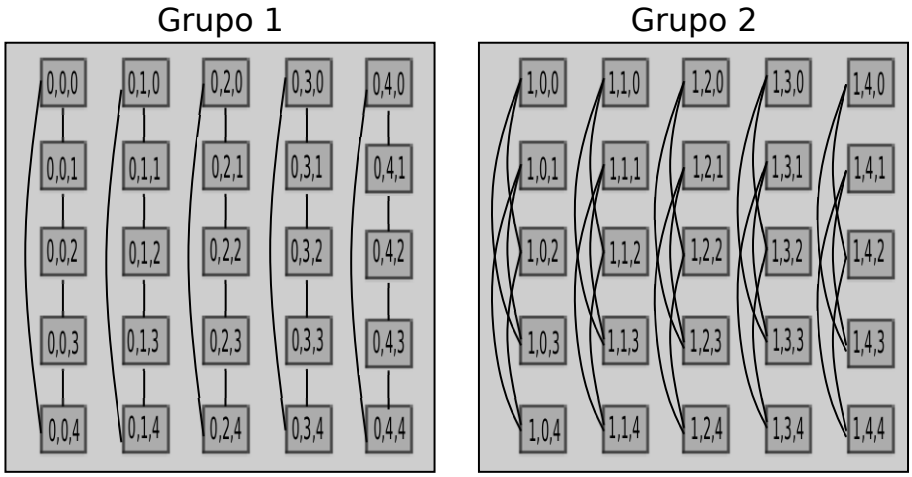
k : Grado de los routers $(k = k' + p)$

N_r : Cantidad de routers en la red

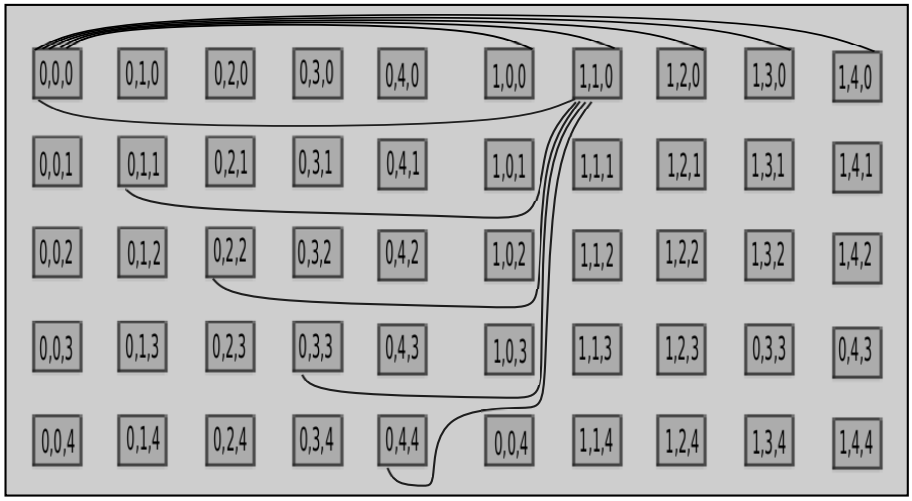
D : Diámetro de la red

La construcción de una topología Slimfly consiste en 4 fases, en las cuales se realizan las conexiones entre routers para conseguir una red de diámetro 2. Nótese que este diámetro se consigue si se consideran únicamente los switches, en la práctica, cuando se consideran los nodos de cómputo, el diámetro será 4. A continuación se detallan los pasos necesarios:

1. **Inicio de la construcción de un grafo SF MMS**: Para ello, hay que encontrar una potencia prima que cumpla lo siguiente: $q = 4w + \delta$ donde $\delta \in \{-1, 0, 1\}$ y $w \in N$.



(a) Uniones en el mismo grupo.



(b) Uniones entre grupos.

Figura 4.4: Uniones en la topología Slimfly.

Para ese valor de q se genera un grafo MMS donde el número de routers de la red será $N_r = 2q^2$ y el número de interconexiones con otros grupos es $k' = \frac{3q-\delta}{2}$.

2. **Construcción del campo de Galois**: Se debe encontrar un elemento ξ (generador) que pertenezca al campo de Galois y que lo genere. En particular se puede escribir cualquier elemento distinto de cero como ξ^i. En general no existe ningún método para encontrar el elemento ξ, por lo tanto, para hallarlo, hay que realizar una búsqueda (en campos pequeños) para cada caso concreto. Para encontrar este elemento, se ha utilizado el código python disponible en la siguiente dirección [Singh, 2017].

3. **Construcción de los de los sets X y X' generadores**: En este paso se utiliza el valor ξ obtenido en el paso anterior para construir los dos sets generadores. Para

$\delta = 1$ tenemos que $X = \{1, \xi^2, \ldots, \xi^{q-3}\}$ y que $X' = \{\xi, \xi^3, \ldots \xi^{q-2}\}$. Se usaran los dos para interconectar los routers.

4. **Conexión entre los routers**: Los routers en este tipo de red están interconectados de la siguiente manera:

   - El router (0,x,y) está conectado al router $(0, x, y')$ si y sólo si $y - y' \in X$
   - El router (1,m,c) está conectado al router $(0, m, c')$ si y sólo si $c - c' \in X'$
   - El router (0,x,y) está conectado al router $(1, m, c)$ si y sólo si $y = mx + c$

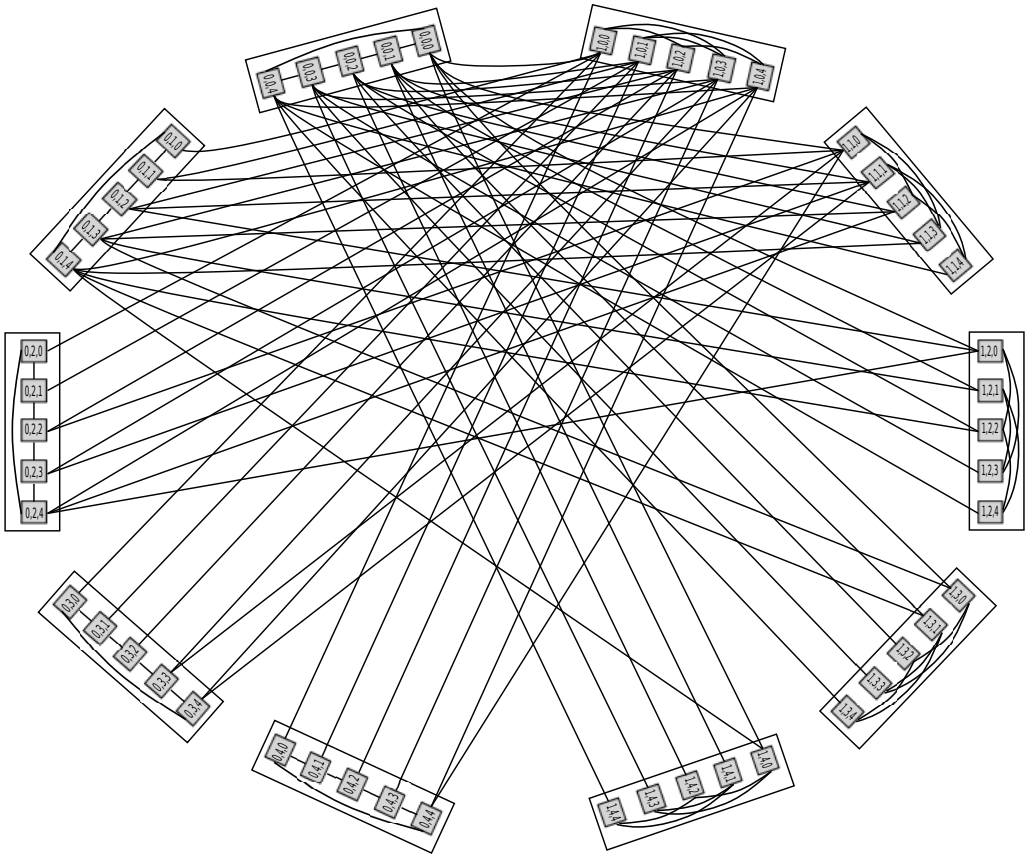


Figura 4.5: Ejemplo de una Slimfly de 50 nodos divididos en 10 grupos.

Intuitivamente, los grafos SF MMS tienen una estructura interna muy simétrica que consiste en dos subgrafos con la misma cantidad de nodos. El primer subgrafo está compuesto por los routers $(0, x, y)$ y el segundo subgrafo por los routers $(1, m, c)$.

Por lo tanto, debido a cómo se generan las conexiones entre los routers, se consigue una red en la que las distancias entre cualquiera de los routers es 2, siendo 4 cuando se cuenta

con la interconexión hacia los nodos de cómputo. Para evitar la saturación de los routers, la cantidad de nodos de cómputo conectados a cada router debería la siguiente:

$$p = k'/2$$

Por consiguiente, aproximadamente el 67 % de las conexiones en el router están destinadas a realizar conexiones con otros routers y el 33 % a conectar nodos de cómputo. Para construir una topología Slimfly de 50 nodos de cómputo como la que aparece en la Figura 4.5, se ha utilizado $q = 5$.

4.3. Topología Megafly

En vista de la aceptación y éxito comercial que ha tenido la topología Dragonfly, debido sobre todo a la reducción del coste de despliegue, por el bajo número de enlaces globales y al rendimiento que proporciona, investigadores de Intel han propuesto una variación llamada Megafly [Flajslik et al., 2018].

En la topología Megafly se ha aumentado el número de conexiones dentro de los grupos usando un grafo bipartito en vez de la flatenned Butterfly usada en Dragonfly [Kim et al., 2007]. En la Figura 4.6. se han representado las conexiones entre routers dentro de un mismo grupo.

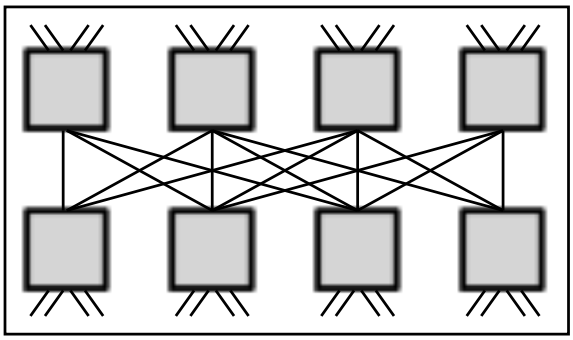


Figura 4.6: Ejemplo de las conexiones dentro de un grupo en la topología Megafly.

El principal beneficio de esta topología con respecto a su predecesora es la cantidad de nodos de cómputo que puede soportar. Por ejemplo, utilizando routers de 36 puertos, esta topología puede soportar cuatro veces más nodos de cómputo que la Dragonfly tradicional.

Otro de sus beneficios más interesantes, es que al utilizar un grafo bipartido en las conexiones dentro del grupo, se pueden reutilizar los componentes previamente utilizados en otras topologías, siendo los árboles densos (fat tree) la más común.

La topología sigue el mismo planteamiento que Dragonfly, se trata de una topología de red jerárquica en tres niveles. La diferencia principal con respecto a ésta, es en el nivel intragrupal. El mismo lo componen dos tipos de routers, los routers globales o *spine* y los routers nodo o *leaf*. Los primeros sirven únicamente para interconectar con otros grupos, mientras que los segundos sirven exclusivamente para interconectar con los nodos de cómputo. Obviamente, los dos tipos de routers están conectados, en concreto, cada uno de los routers de un tipo está conectado con todos los routers del otro tipo.

La descripción de la topología Megafly se realiza utilizando la siguiente notación:

N : Número de nodos en toda la red

N_g : Número de nodos en cada grupo

p : Número de nodos conectados a un router

r : Grados de los routers

El número de nodos que componen un grupo de la topología Megafly viene determinado por el grado de los routers y se calcula de la manera siguiente:

$$N_g = (r/2)^2$$

De esta fórmula se puede observar que la cantidad de nodos por router leaf será $r/2$ y la cantidad de nodos en toda la red será la siguiente:

$$N = (r/2)^2 * ((r/2)^2 + 1)$$

donde se puede ver que el número de nodos de una topología Megafly es cuatro veces superior al de una Dragonfly, como se ha remarcado anteriormente.

Para construir una Megafly de 272 nodos y 17 grupos como la que aparece en la Figura 4.7, lo único que hay que tener en cuenta es el grado de los routers que van a formar la red. Al utilizar la mitad de sus conexiones para interconectarse con el resto de routers del grupo y la otra mitad para interconectar nodos, no hay grados de libertad en la creación de la red y todo está condicionado por esa selección inicial, en este caso 8.

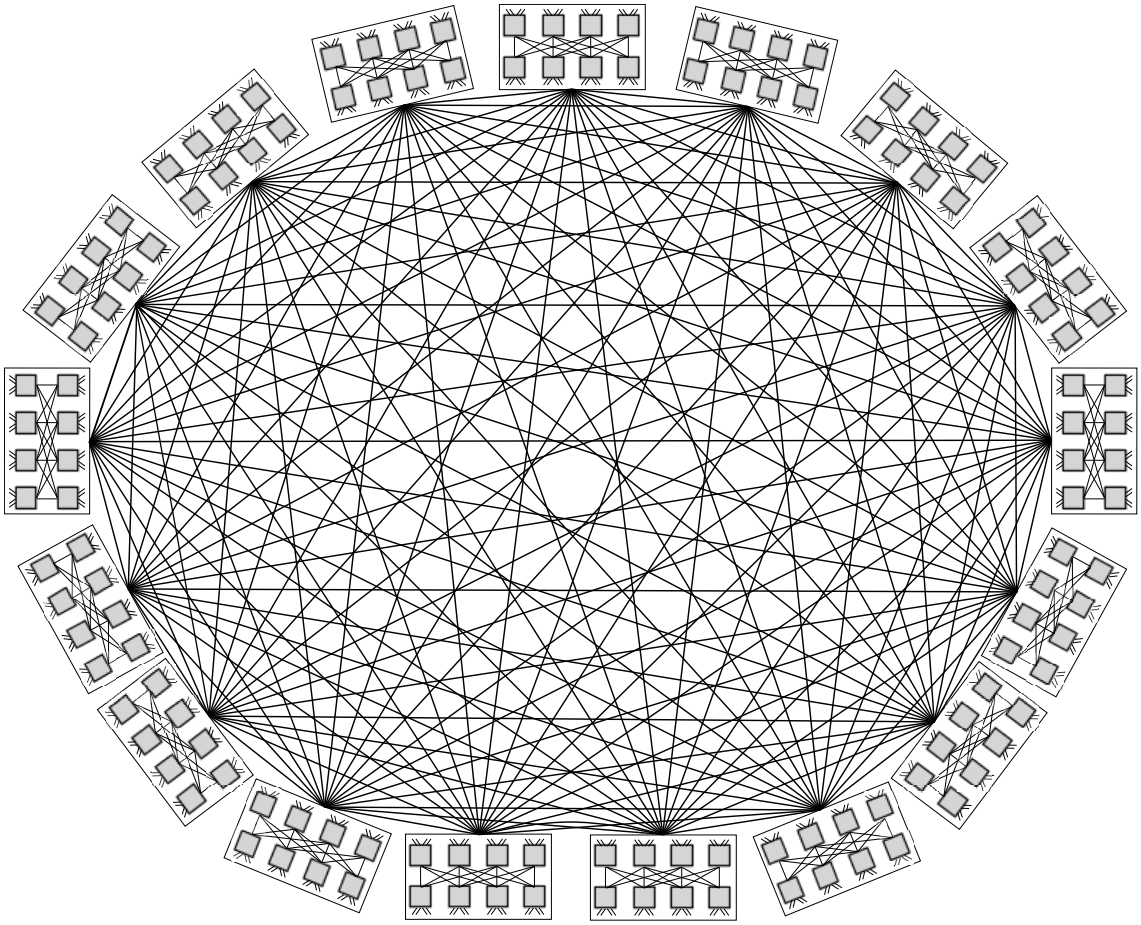


Figura 4.7: Ejemplo de las uniones entre grupos de una Megafly de 272 nodos y 17 grupos.

5. CAPÍTULO

El simulador Scheduling Simulation Environment (SSE)

El simulador con el que se ha trabajado tiene como función principal analizar las diferentes métricas que afectan a las topologías de red usadas en supercomputación con diferentes tipos de scheduling. El simulador está creado utilizando el lenguaje de programación python y el mismo consta de varios módulos los cuales están representados en la Figura 5.1: con funciones diferenciadas que sirven para realizar las siguientes funciones: creación de la topología de red seleccionada, selección de los nodos de cómputo dependiendo de diferentes algoritmos, elección de los tipos de scheduling, cálculo de las métricas y reporte de los resultados obtenidos.

5.1. Módulo de scheduling

Este módulo implementa la funcionalidad fundamental del simulador que es realizar el proceso de scheduling, el cual, se divide en tres fases. Hay que indicar, sin embargo, que debido a que dos de las fases (búsqueda y asignación de nodos y liberación de nodos) dependen de la topología de red utilizada, estas están implementadas en el módulo de topologías de red (Sección 5.2). A continuación se detallan las fases del scheduling:

- Selección de trabajos: En la primera etapa del proceso en la cual se procede a seleccionar un trabajo de la cola en base a una política que se halla seleccionado. Dicha política es la encargada de establecer el orden en que los trabajos serán llevados a ejecución. SSE implementa varias políticas de selección como First Come First

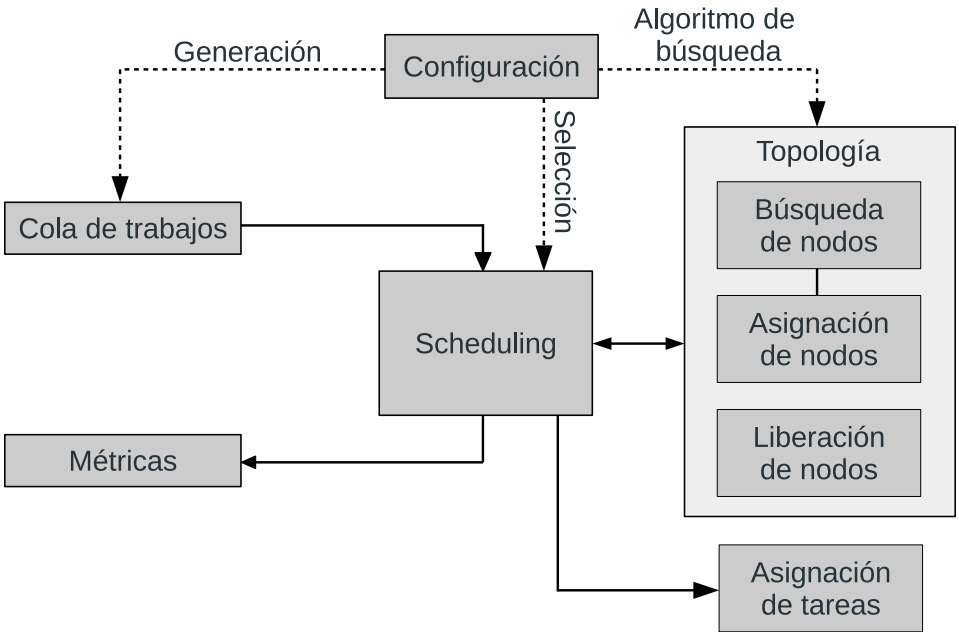


Figura 5.1: Representación de los módulos de SSE.

Serve (FCFS), y dos tipos de Backfilling. Para la realización de este trabajo únicamente utilizaremos FCFS la cual impone un orden estricto en la ejecución de los trabajos según el tiempo de llegada a la cola (ver Figura 5.2).

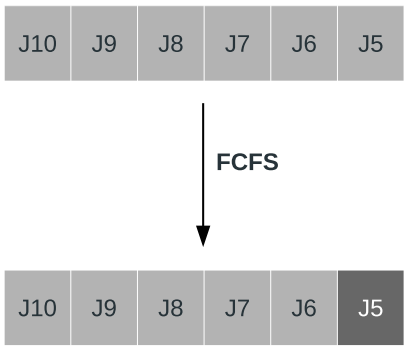


Figura 5.2: Selección de J5 usando FCFS como política de planificación.

- Búsqueda y asignación de nodos: Esta segunda etapa se puede dividir en dos fases: en la primera se procede a buscar un conjunto de nodos solicitados por el trabajo seleccionado por la política de scheduling (ver Figura 5.3). Si se encuentra, dicho conjunto será reservado. En caso contrario, el trabajo tendrá que esperar hasta que el conjunto de nodos requerido esté disponible. En la segunda fase el conjunto de nodos encontrado es asignado al trabajo. Debido a que SSE implementa varias to-

pologías de red, cada topología contará con sus mecanismo propios de búsqueda de nodos.

Figura 5.3: Búsqueda y asignación de 6 nodos pedidos por J5.

- Mapeo tareas: En la última etapa, después de que se haya encontrado un conjunto de nodos apropiado, cada tarea que compone el trabajo (aplicación) será asignada a un nodo de los reservados en la segunda etapa.

5.2. Módulo network

El segundo de los módulos, se encarga de realizar las tareas más generales del simulador. Primeramente, recoge los datos introducidos por el usuario y los almacena para utilizarlos en el simulador. Dependiendo de la topología seleccionada, llamara al módulo específico de la topología para simular la creación de la red.

Una vez realizada la creación, busca nodos de cómputo libres y los asigna utilizando diferentes políticas de asignación de nodos, la cual estará definida por el usuario al iniciar el simulador. Cada vez que se termine una de las simulaciones, se encargará de liberar los nodos para que puedan utilizarse más tarde. Además, se encarga de transferir las estructuras de datos y la información al resto de nodos.

5.3. Módulo Metrics y data

Estos módulos, se encargan de recibir la información enviada por el módulo network y procesarla para realizar el cálculo de las métricas definidas en el simulador. Algunas de las métricas utilizadas en el simulador, son las siguientes:

- **Nodes affected**: Nodos que se ven afectados al interactuar cualquier par de nodos. Se calcula dependiendo el tipo de red. Recorriendo la estructura definida en el tipo de red, adecuándose a la topología, yendo del nodo origen al nodo final pasando por todos los nodos intermedios.

- **Apps affected**: Aplicaciones en uso entre cualquier par de nodos y los nodos intermedios que se han tenido que cruzar para llegar desde el nodo origen al nodo destino. Se calcula tras calcular la métrica anterior, revisando que se está ejecutando en cada nodo de tránsito más inicio y final.

- **Pairwise distance**: Distancia entre un par de nodos cualquiera. Se calcula dependiendo el tipo de red. Utilizando las fórmulas asociadas al tipo concreto de red y las casuísticas asumidas al definir las estructuras de datos.

- **Diameter**: Diámetro de la red. Se calcula obteniendo la distancia máxima entre cualquier par de nodos.

5.4. Módulo de las topologías

Cada topología de red dispone de un módulo asociado para poder crear cada topología con su tratamiento específico. En los módulos específicos están las funciones específicas asociadas. La inclusión de nuevas topologías en el simulador es sencilla ya que dispone de una interfaz común para todas. Únicamente hay que definir e implementar las características propias de cada una: definición de la estructura y algoritmo de búsqueda de nodos.

A continuación se presenta un resumen del estudio realizado sobre las topologías que se encuentran ya implementadas. El objetivo fue entender cómo se representaban y cómo interactuaban con otros componentes del simulador.

5.4.1. Topologias de tipo malla y toro

La construcción de una malla en el simulador es bastante sencilla, al poder representarse como una matriz. Debido a ello, la creación de los nodos y su construcción es directa utilizando funciones ya definidas en el módulo *numpy* de python. En particular hay que pasarle como parámetro el número de nodos que tendrá en cada dimensión. En cuanto a

las uniones de los nodos, al ser una estructura fija y bien definida, no existe la necesidad de crear una matriz de interconexiones ya que se puede calcular de manera sencilla si dos nodos están interconectados conociendo sus coordenadas.

En cuanto a la construcción de un toro, se realiza de manera idéntica a la malla. La estructura vuelve a imitar perfectamente una matriz. Se aplican los mismos principios con los toros que con las mallas para conocer las interconexiones entre nodos y no es necesaria la creación de una matriz de interconexión.

Las funciones que calculan las distancias entre nodos son directas y únicamente en el nodo hay que considerar los enlaces periféricos.

5.4.2. Topologías de tipo árbol

La construcción de un árbol es algo más compleja que la de las topologías anteriores. Se necesita conocer la cantidad de niveles que va a tener el árbol y la cantidad de nodos en los routers en cada nivel. En esta topología no es necesario, al igual que con las mallas y toros, crear una matriz de interconexión de nodos ya que al estar estructurado en niveles las interconexiones son fáciles de conocer.

5.4.3. Búsqueda, selección y liberación de nodos

Los algoritmos de búsqueda pueden ser de dos tipos: aquellos que tienen en cuenta la estructura de la topología a la hora de buscar nodos para asignarlos a una aplicación y aquellos genéricos en los que la estructura no se tiene en cuenta.

En el caso concreto de las topologías que aparecen en este apartado: mallas, toros y árboles, se encuentra implementado para las tres el algoritmo de búsqueda *first fit*. Este algoritmo busca entre los nodos disponibles por orden de identificador y selecciona los primeros libres sin tener en cuenta la estructura de la red.

Tras realizar la búsqueda de los nodos que componen la partición, se asigna a cada uno de ellos el identificador de la aplicación (trabajo) a la que va a pertenecer. De esta manera, esos nodos quedan reservados y no se podrán asignar a otra aplicación. En concreto el valor de cada nodo se cambia de -1 (no en uso) al identificador de la aplicación a la que se asigna. Una vez la aplicación ha terminado, se vuelve a asignar a cada nodo de la partición utilizada el valor -1 (no en uso) para que pueda volver a ser asignado a otro trabajo.

6. CAPÍTULO

Implementación de las topologías de red de diámetro bajo

En este apartado se explica como se ha realizado la implementación de las topologías explicadas en la sección anterior en el simulador SSE y que decisiones se han tomado para definir las estructuras de datos asociadas. Además, se explicarán los algoritmos de búsqueda de nodos implementados: un algoritmo first fit que no tiene en cuenta la topología, un algoritmo de búsqueda de nodos basado en la proximidad de los nodos (grupos) que tienen en cuenta la estructura de cada topología y uno de selección aleatoria.

Todo el código desarrollado para la creación de las topologías y algoritmos de búsqueda se encuentra disponible en gitlab[1].

6.1. Dragonfly

Para la creación de la topología Dragonfly se han reutilizado varias estructuras ya presentes en otras topologías implementadas. Se han realizado pequeñas modificaciones y se han añadido nuevas estructuras.

6.1.1. Creación de la topología

Primero, al tratarse de una topología de red directa, se ha podido determinar que todo router va a compartir dos parámetros: el grupo al que pertenece y en qué posición se

[1] https://gitlab.com/joseapascual/sse

encuentra dentro del grupo.

Por ello, se ha dividido la creación de la topología en dos fases:

1. **Fase 1**: Creación del array de arrays que va a almacenar el estado de los nodos, para el cual se ha utilizado una estructura de tres dimensiones.
2. **Fase 2**: Definición de las uniones entre routers. Para crear el array de arrays que va a contener las uniones, se ha creado una clase *routerdr* que tiene dos características: su posición y sus conexiones. Una vez terminada la asignación de la posición de cada router a la estructura, se realiza el establecimiento de sus conexiones.

Para establecer las conexiones, se recorren todos los routers y en cada uno de ellos, se busca el primer grupo sin enlaces hacia el grupo al que pertenece el router actual. Dentro de ese grupo se busca el primer router que tenga enlaces disponibles todavía. Se establece la conexión registrando las coordenadas del otro router en el array de conexiones del router y se siguen buscando conexiones a otros grupos hasta que se llene su array de conexiones.

Por último, debido a que en cada grupo las conexiones intragrupo se realizan entre todos los routers, no se registran en el array de conexiones, pero se tendrán en cuenta cuando sea necesario.

6.1.2. Métricas

En cuanto a las métricas, solo ha habido que realizar una implementación específica en dos de ellas: nodes affected y el cálculo de la distancia. Para el resto, con unas mínimas variaciones se ha podido reutilizar el código ya implementado.

- **Nodes affected:** Para calcular los nodos afectados dada una partición, se ha decidido recorrer la misma, buscando los nodos intermedios entre cada par de nodos cualquiera. Al saber que cualquier unión entre grupos era única, se ha buscado el nodo intermedio buscando la conexión al grupo objetivo desde los routers del grupo origen. Una vez recorrida toda la partición se ha podido calcular el total de routers afectados.
- **Distancia**: Para el cálculo de distancia se han utilizado varias condiciones: (1) si dos nodos pertenecen al mismo router la distancia será dos, (2) cuando pertenecen

al mismo grupo la distancia será tres y (3) si pertenecen a grupos diferentes, se revisará si el router al que pertenece el nodo origen tiene conexión con el router del nodo destino. Si es así la distancia será tres, en caso contrario, si tiene conexión con algún router del grupo objetivo, será cuatro y si no, cinco.

6.2. Slimfly

La implementación de la topología Slimfly ha sido complicada, sobre todo trasladar el cálculo de las conexiones entre routers al código y ha habido que tomar decisiones para simplificar la implementación de la topología.

6.2.1. Creación de la topología

Viendo que se podía utilizar un método similar al utilizado en la Dragonfly, se ha vuelto a utilizar el planteamiento de separar en dos la construcción: por un lado el estado de los nodos y por otro las conexiones entre routers.

En cuanto al estado de los nodos la creación ha sido prácticamente idéntica, excepto que en vez de dividir en dos el nivel más alto de la estructura por ser un grafo bipartito y crear dos grupos iguales, se ha optado por tenerlo en cuenta al hacer los cálculos y utilizar un único array de grupos. Esto se ha realizado para evitar complicar la creación de la topología con un nivel extra innecesario.

En cuanto al apartado de conexiones entre los routers y siguiendo el método utilizado en Dragonfly, solo se han incluido las conexiones entre routers de distintos grupos utilizando la misma estructura, que se llama *routersf*. La creación de las conexiones ha sido sencilla debido a que existe una fórmula para conocer si dos routers están conectados.

6.2.2. Métricas

En cuanto a las métricas, igual que en la topología Dragonfly, solo ha habido que hacer implementaciones específicas de *nodes affected* y de la *distancia*. El resto de las métricas se han implementado realizando variaciones mínimas a las que ya se encuentran implementadas.

- **Nodes affected:** Para calcular los nodos afectados dada una partición, se recorre los nodos que componen la misma, usando la métrica de distancia para conocer si existen nodos intermedios entre cada par de nodos. Si la distancia es menor que tres, están conectados, si no, se ha buscado el nodo intermedio utilizando las fórmulas expuestas anteriormente.

- **Distancia**: Para el cálculo de la distancia, se han utilizado varias condiciones. En principio, hay que saber si los dos nodos forman parte de la misma parte del grafo, pero no del mismo grupo, ya que la distancia en ese caso siempre será cuatro. Si son del mismo grupo, dependiendo de cuál de las dos partes del grafo sean, se calculara de una u otra manera para saber si están unidos o no, siendo su distancia tres o cuatro. Si los nodos están cada uno en una parte del grafo, se ha usado la fórmula de conexión entre routers para saber si estaban interconectados y decidir si la distancia es tres o cuatro.

6.3. Megafly

En este caso, aunque la topología sea una evolución de la Dragonfly y jerárquicamente muy parecida, al tratarse de una topología indirecta se ha tenido que implementar siguiendo un método diferente, tanto para la generación como para el cálculo de las métricas.

6.3.1. Creación de la topología

Para la creación de la topología Megafly, siguiendo el ejemplo de las dos anteriores, la estructura que la representa se ha mantenido igual. Separar en dos la construcción de la misma.

En cuanto al estado de los nodos la creación ha sido prácticamente idéntica a dragonfly y la creación de las interconexiones de los routers ha sido completamente igual.

6.3.2. Métricas

En cuanto a las métricas, igual que en la topología Dragonfly, se han implementado *nodes affected* y *distancia*, mientras que el resto se han implementado realizando variaciones mínimas a las que ya se encuentran implementadas.

- **Nodes affected**: Para calcular los nodos afectados dada una partición, se ha decidido recorrer la misma, utilizando la misma estrategia que en la topología Dragonfly. La única diferencia con respecto a ella, era que los routers intermedios, no iban a contener nodos de cómputo por ser una topología indirecta, y por lo tanto, se han creado dos estructuras diferentes para almacenar esa casuística y poder después calcular el total de nodos afectados.

- **Distancia**: El cálculo de distancia en Megafly ha sido bastante sencillo de implementar: si pertenecían al mismo router la distancia era dos, si pertenecían al mismo grupo cuatro y sino, cinco.

6.4. Búsqueda, selección y liberación de los nodos

En el presente apartado se va a hablar de los diferentes algoritmos de búsqueda implementados para las topologías de diámetro bajo. Aunque las topologías presenten diferencias por sus características propias, en todas ellas se puede aplicar un patrón que permita tomar una aproximación parecida para crear los algoritmos de búsqueda.

En cuanto a la selección y liberación de los nodos, al tratarse de funciones no dependientes de la topología se explicaran en un apartado común.

6.4.1. Algoritmo first fit

El algoritmo first fit basa su función en buscar el primer nodo que se encuentre libre y repetir esta acción hasta encontrar la cantidad de nodos libres solicitados. La implementación del algoritmo first fit ha sido bastante sencilla, al poder reutilizar el código existente en el simulador. La única diferencia entre la implementación de las tres topologías ha tenido lugar en el cálculo del número de grupos totales.

Para seleccionar los nodos que van a componer la partición, se ha buscado por orden de identificador el primer nodo disponible, es decir, el primero que tuviera un valor -1 (no en uso), hasta encontrar el número de nodos requerido por la aplicación (partición).

6.4.2. Algoritmo de búsqueda aleatoria

El algoritmo de búsqueda aleatoria de nodos implementa una búsqueda de nodos libres aleatoria hasta encontrar la cantidad de nodos solicitados.

La implementación del algoritmo se ha basado en la implementación del algoritmo first fit realizando únicamente cambios en la forma de seleccionar el primer nodo disponible. Para ello, se ha utilizado un generador de números aleatorios delimitado por el tamaño máximo de nodos totales, comprobando en cada iteración si el identificador de nodo generado estaba libre.

6.4.3. Búsqueda de nodos en base a la proximidad (grupos)

El algoritmo creado para la selección de nodos en las topologías de diámetro bajo, utiliza la principal característica jerárquica de los tres, los grupos, e intenta maximizar la proximidad de la selección de nodos, Para ello busca grupos que tienen la cantidad de nodos libres solicitados y si eso no es posible, buscando los grupos subsiguientes con mayor cantidad de interconexiones hacia ese primer grupo seleccionado.

Para realizar la búsqueda, se han diferenciado tres posibilidades: (1) que la cantidad de nodos que necesita la partición entre en un único router, (2) que entre en un único grupo o (3) que no entre en un grupo.

En el primer caso, se ha buscado el primer router disponible que tuviera suficientes nodos libres como el tamaño de la partición, de forma similar a como hace first fit. En el segundo caso, se ha buscado el primer grupo con suficientes nodos libres en los routers que lo forman. En el tercer caso, al no poder buscar un único grupo disponible, se ha tenido que tener en cuenta cada topología implementada para optimizar la proximidad entre grupos y buscar los grupos disponibles más adecuados donde asignar los nodos.

En el caso de Dragonfly y Megafly, al no haber distinciones entre enlaces de nodos, se ha ido buscando los grupos completamente libres, para llenarlos y repetir el proceso hasta asignar la partición completa.

En el caso de Slimfly, los diferentes grupos que conforman la topología tienen diferentes cantidades de enlaces de conexión. Para realizar la búsqueda de la partición se han ido cogiendo los grupos con todos los nodos libres que tuvieran mayor cantidad de enlaces.

6.4.4. Asignación y liberación de los nodos

Para realizar la asignación y liberación de los nodos se ha utilizado el mecanismo genérico que provee el simulador por lo que únicamente ha sido necesario guardar el identificador de los trabajos en los nodos al ser reservados y asignar un -1 cuando el trabajo finaliza su ejecución y los nodos son liberados.

7. CAPÍTULO

Resultados experimentales

Tras la implementación de las topologías de diámetro reducido en el simulador se ha decidido realizar un entorno de pruebas para poder verificar que las topologías han sido correctamente implementadas y que no se han cometido fallos en el diseño de las mismas.

Las pruebas a realizar se han dividido en dos partes: primero, se ha analizado la distancia entre los nodos para comprobar que el resultado era coherente a lo que de manera teórica se esperaba y en el segundo se han realizado una serie de experimentos para evaluar las políticas de búsquedas de nodos.

7.1. Distancia entre nodos

Se han analizado las distancias entre nodos de las tres topologías implementadas en el proyecto. Además, se explicarán los motivos que influyen para que las distancias varíen en cada topología. Por motivos de espacio y claridad únicamente se representan las distancias desde el primer nodo (nodo 0) al resto de los nodos.

7.1.1. Dragonfly

En el caso de la Dragonfly la distancia de un nodo al resto puede tener 4 valores diferentes: 2, 3, 4 o 5. En la Figura 7.1 se ha representado la distancia desde el nodo 0 al resto de los

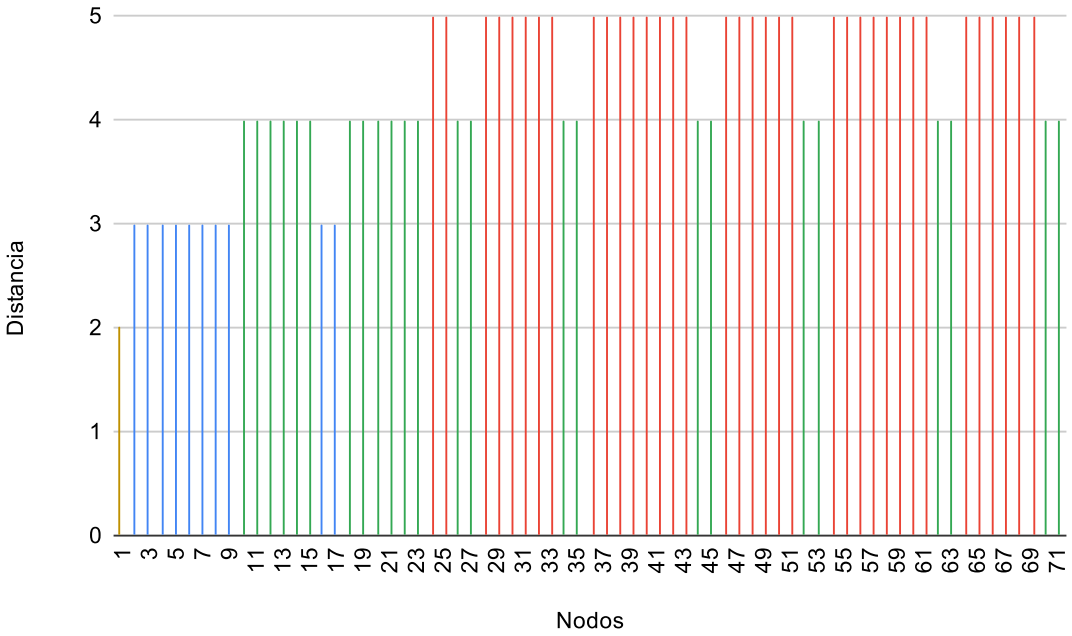


Figura 7.1: Representación de la distancia desde el nodo 0 al resto de nodos en Dragonfly.

nodos en una topología de 72 nodos. A continuación, se explica que distancia existe entre dos nodos en función de su situación en la estructura de la red:

- **Distancia 2**: cuando el nodo destino se encuentra en el mismo router que el nodo 0.
- **Distancia 3**: En este caso hay dos posibilidades: el nodo destino se encuentra en un router con conexión directa con el router en el que se encuentra el nodo 0 o el nodo destino se encuentra en un router del mismo grupo.
- **Distancia 4**: En este caso también, hay dos posibilidades: el nodo destino se encuentra en un grupo con conexión directa con el router en el que se encuentra el nodo 0 pero no en el router conectado o el nodo destino se encuentra conectado a un router que tiene conexión directa con un router del grupo del nodo 0 distinto al router donde esta conectado el nodo 0.
- **Distancia 5**: En esta caso se trata del resto de nodos de la red. Los cuales no están en el router que conecta el grupo con el grupo del nodo 0.

7.1.2. Slimfly

En el caso de la topología Slimfly las distancias pueden ser de dos tipos. Si el router del nodo 0 se encuentra interconectado con el router del nodo destino o no. En el primer caso, la distancia será de 4 saltos y en el otro, será de 5 saltos ya que tendrá que ir al router intermedio que está conectado con el router destino.

En la Figura 7.1 se ha representado la distancia desde el nodo 0 al resto de los nodos en una topología Slimfly de 50 nodos.

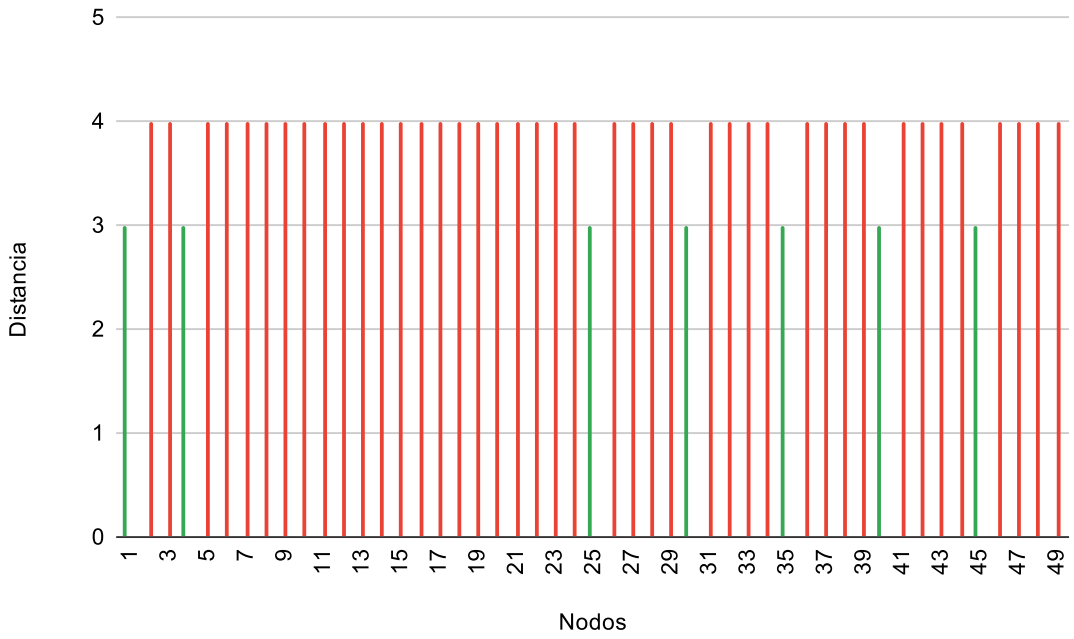


Figura 7.2: Ejemplo de la distancia desde el nodo 0 al resto de nodos en Slimfly

7.1.3. Megafly

En la topología Megafly, las distancias pueden tomar 3 valores:

- Si el nodo se encuentra en el mismo router que el nodo 0 la distancia es 2.
- Si el nodo se encuentra en el mismo grupo, la distancia será de 4 saltos, ya que tendrá que ir a uno de los routers intermedios para volver al router del nodo destino.
- Si el nodo se encuentra en otro grupo, la distancia será de 5 saltos en todos los casos porque siempre tendrá que recorrer el mismo camino: ir a un nodo intermedio de

su grupo desde su router, desde el router intermedio al router conectado del grupo destino y desde ahí ir al router del nodo objetivo.

En la Figura 7.3 se ha representado la distancia desde el nodo 0 al resto de los nodos en una topología Megafly de 20 nodos.

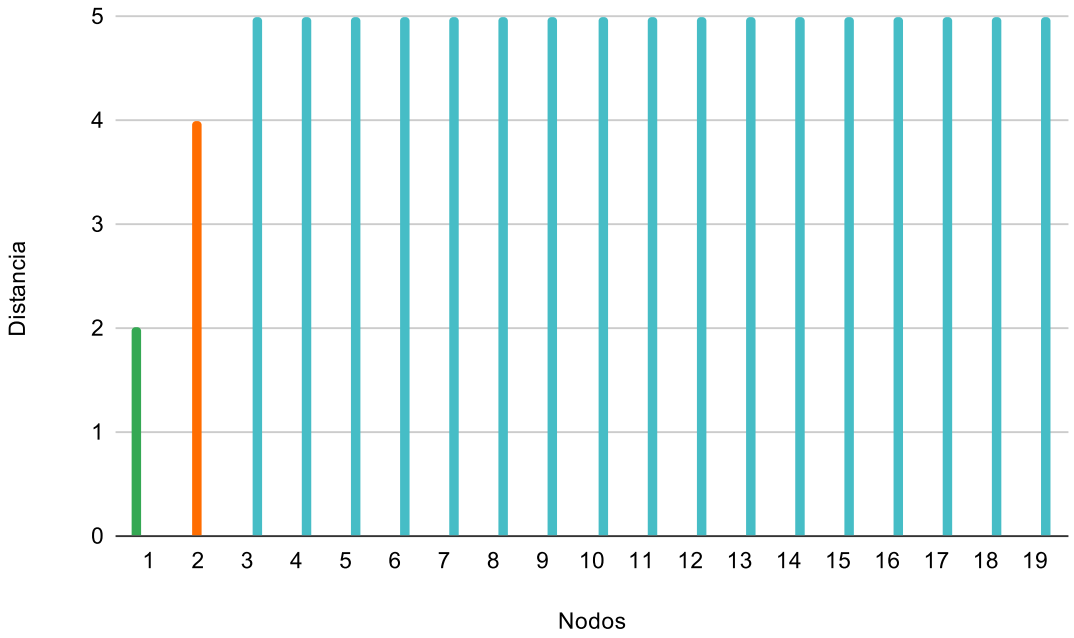


Figura 7.3: Ejemplo de la distancia desde el nodo 0 al resto de nodos en Megafly

7.1.4. Comparativa de las tres topologías

Tras revisar los datos recibidos tras la simulación y trasformarlos en gráficas, se puede establecer un ranking de topologías con los mejores resultados de este apartado (distancia entre el nodo 0 al resto de nodos de la topología) que es el siguiente: Slimfly, Dragonfly y Megafly.

Este resultado es consecuente con lo visto anteriormente, ya que Slimfly se ideó como una forma de recortar el diámetro y las distancias de la red con respecto de Dragonfly y el objetivo de Megafly no ha sido mejorar esta métrica con respecto a Dragonfly, ya que basa el diseño en ella, sino implementar una topología que se asemeje a esta última, preparándola para ser utilizada en la computación exaescalar.

7.2. Evaluación de las topologías

Para realizar la evaluación de las topologías, se han realizado una serie de simulaciones en las hay que asignar un conjunto de nodos (partición) a 100 trabajos que llegan a la cola del planificador. Esta asignación se realizará utilizando las tres políticas de búsqueda implementadas para cada topología.

El objetivo de estas simulaciones es medir la distancia media de las particiones y el diámetro que cada algoritmo es capaz de conseguir sobre cada topología. Estas métricas tienen mucho interés porque evalúan los algoritmos y las topologías en entornos multi-aplicación donde muchas aplicaciones son ejecutadas de manera concurrente y los recursos deben ser compartidos.

Debido que no es posible generar las tres topologías con el mismo número de nodos, se han generado tres de tamaño similar, en torno a 10.000 nodos. A continuación se explican los parámetros utilizados para la generación de cada una de ellas:

- Para construir una topología Dragonfly con un número de nodos $\cong 10K$, los parámetros utilizados han sido los siguientes: número de routers por grupo $q = 14$ y para que este balanceada, el número de enlaces globales y el número de nodos por router han tenido que ser iguales y la mitad que q, es decir $h = p = 7 = q$. Utilizando estos parámetros la red tiene un número de nodos $N = 9702$
- Para la construcción de la red Slimfly, se han utilizado los mismos parámetros que utilizaba el artículo original en el que se proponía la topología [Besta and Hoefler, 2014]. La potencia prima en este caso ha sido 19 y el número de nodos de computo 15. En cuanto al número primitivo asociada a la potencia se ha utilizado el código mencionado anteriormente [Singh, 2017] para calcularlo. Por lo tanto el número total de nodos de la red es $N = 10830$
- Con respecto a la topología Megafly, utilizando las ecuaciones asociadas a la topología se ha buscado una que tuviera un número de nodos $\cong 10K$ y se ha encontrado una que tiene $N = 10100$ utilizando routers de radio $r = 20$.

7.2.1. Distancia media asignada a las particiones en Dragonfly

En la Figura 7.4 se han representado la distancia media de cada partición asignada a cada uno de los 100 trabajos utilizando los tres algoritmos de búsqueda en Dragonfly.

Nótese que al principio de la simulación, cuando la red esta todavía vacía, la distancia media es mas baja para first fit (FF) y el algoritmo basado en grupos (GR) debido a que estos algoritmos asignan nodos consecutivos. Sin embargo, cuando la red se llena y las aplicaciones empiezan a salir (cuando terminan) y entrar vemos que la distancia media de las particiones se incrementa. Este efecto no se produce con el algoritmo que utiliza búsqueda aleatoria (RAND).

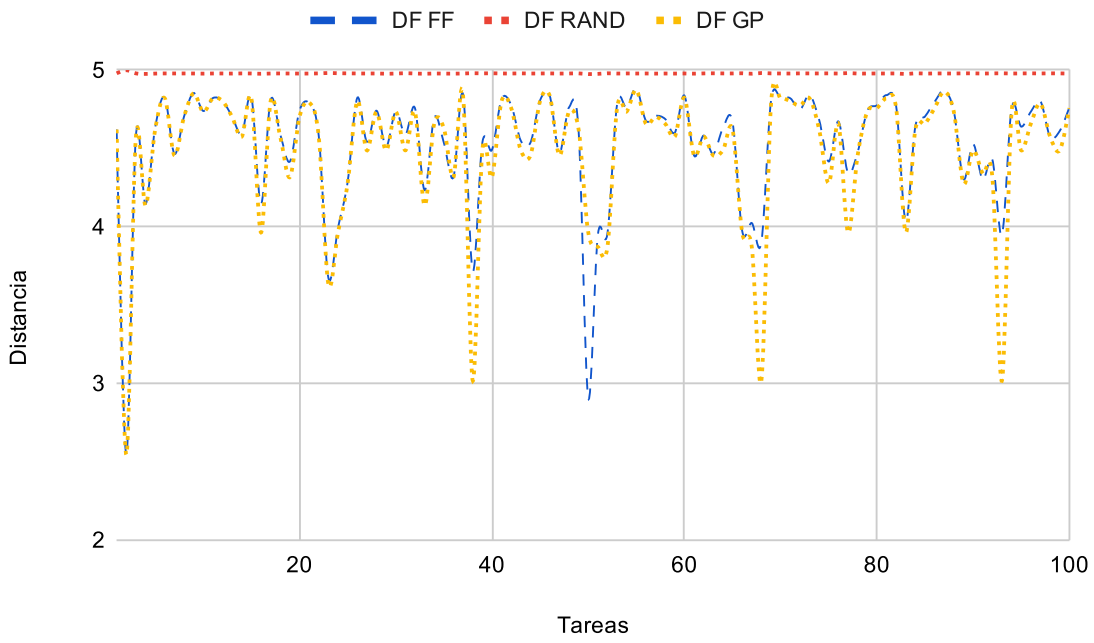


Figura 7.4: Distancia media de la partición asignada a cada tarea en un topología Dragonfly.

Como se puede ver en esta topología la distancia mínima es 2 y medio y la máxima es 5. Como era de esperar el algoritmo RAND consigue particiones con una distancia media que es aproximadamente 5. Esto es debido a que no considera la estructura de la red y elige los nodos de manera aleatoria.

Si nos fijamos en el algoritmo FF, pese a no estar basado en la estructura de la red, obtiene particiones con un tamaño medio similar a GP. Esto es debido a que la búsqueda se produce de manera secuencial por lo que es muy probable que los nodos asignados se encuentren en el mismo switch, lo que hace que la distancia media sea baja.

De todas formas, el mejor algoritmo es GP que está basado en la estructura de la red. Podría parecer que las diferencias son pequeñas, pero hay que tener en cuenta que estamos

representando la media y que particiones mas compactas tienen un efecto importante en el rendimiento de las aplicaciones.

7.2.2. Distancia media asignada a las particiones en Slimfly

En la Figura 7.5 se ha representado la distancia media de cada partición asignada a cada uno de los 100 trabajos utilizando los tres algoritmos de búsqueda en Slimfly.

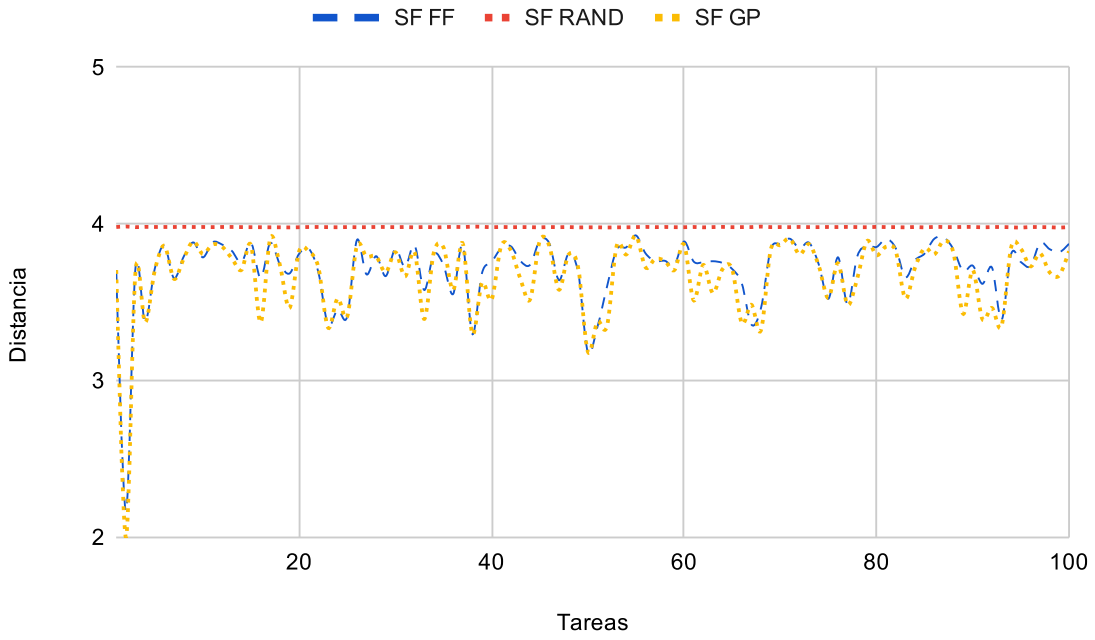


Figura 7.5: Distancia media de la partición asignada a cada tarea en un topología Slimfly.

Como se puede ver en esta topología la distancia mínima es 2 y la máxima es 4. Las conclusiones que podemos extraer de los resultados son similares a las sacadas para Dragonfly. El algoritmo RAND obtiene en todos los casos particiones con distancia media cercana a la máxima de la red. Respecto a FF y GP, se puede ver como GP es capaz de obtener particiones mas compactas en prácticamente todas las asignaciones.

7.2.3. Distancia media asignada a las particiones en Megafly

En la Figura 7.6 se ha representado la distancia media de cada partición asignada a cada uno de los 100 trabajos utilizando los tres algoritmos de búsqueda en Megafly.

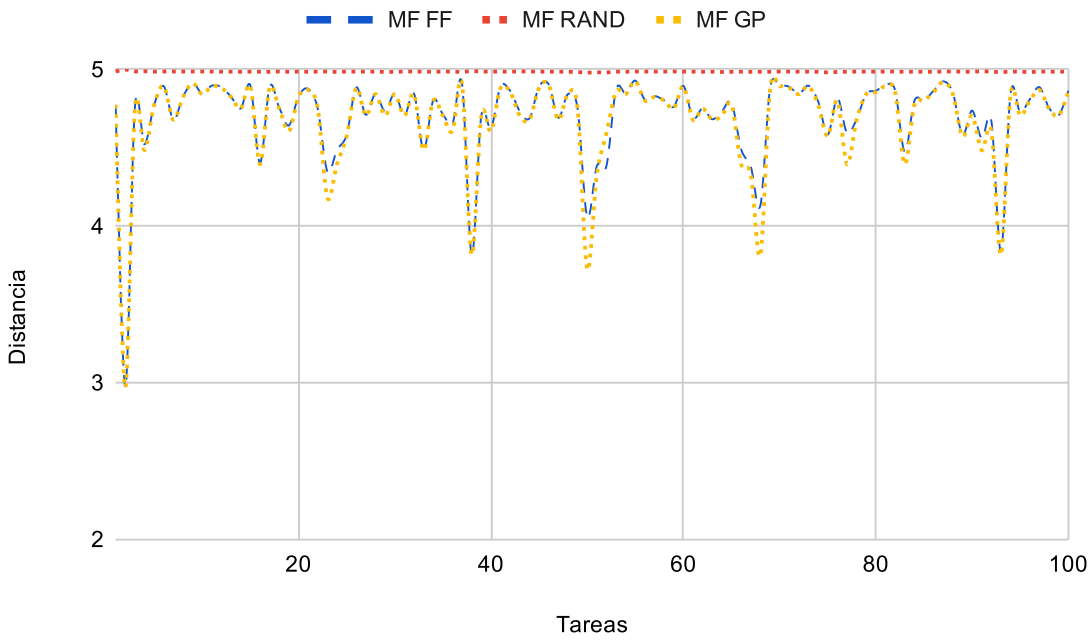


Figura 7.6: Distancia media de la partición asignada a cada tarea en un topología Megafly.

Como se puede ver en esta topología la distancia mínima es 3 y la máxima es 5. Las conclusiones que podemos extraer de los resultados son similares a las sacadas para Dragonfly y Slimfly. El algoritmo RAND obtiene en todos los casos particiones con distancia media cercana a la máxima de la red. Los mejores resultados son obtenidos de nuevo por el algoritmo GP.

7.2.4. Análisis global de los resultados

Tras haber revisado los gráficos y los valores asociados a los mismos, está claro que tener en cuenta la estructura de la red beneficia la búsqueda de particiones en términos de distancia.

En las Tablas 7.1 y 7.2 se han representado la distancia media y el diámetro medio de las 100 particiones respectivamente. Como era de esperar, la topología en la que se consiguen las particiones más compactas es Slimfly, seguida de Dragonfly y Megafly.

En cuanto al diámetro de las particiones se puede ver que, para cualquier algoritmo y topología es siempre cercano a la distancia máxima de la red. Esto puede deberse a muchos factores, como el grado de los switches y el tamaño de las aplicaciones. Este análisis se deja como trabajo futuro.

Búsqueda	Dragonfly	Slimfly	Megafly
FF	4,54	3.72	4.71
Grupos	4.49	3.68	4.69
Random	4.98	3.98	4.99

Tabla 7.1: Distancia media de las 100 particiones asignadas a las aplicaciones (media).

Búsqueda	Dragonfly	Slimfly	Megafly
FF	4.96	3.99	4.98
Grupos	4.98	3.98	4.95
Random	5.00	4.00	5.00

Tabla 7.2: Diámetro medio de las 100 particiones asignadas a las aplicaciones.

Para concluir el análisis, los resultados claramente muestran la necesidad de incorporar conocimiento de la topología en los algoritmos de búsqueda. En la practica, el algoritmo más utilizado es first fit ya que proporciona buenos resultados con poco esfuerzo, es decir, es un algoritmo fácil de implementar y rápido de ejecutar. Sin embargo, hemos visto como incorporando información en el proceso de búsqueda es posible mejorar los resultados de first fit sin incrementar el coste del algoritmo.

8. CAPÍTULO

Gestión y planificación del proyecto

A la hora de llevar un proyecto a buen puerto, se exige que el proyecto cuente con una buena gestión que respalde el trabajo realizado y que pueda dar cuenta de las desviaciones que se producen a la hora de entregarlo. Por consiguiente, la planificación y el plan de riesgos asociado deben ser coherentes y amplios para remediar las desviaciones que siempre se producen en los proyectos.

Para regular la realización del proyecto se ha decidido dividir el trabajo en tres líneas o fases diferenciadas. En este caso, las fases han sido las siguientes: Fase de gestión del proyecto, fase de desarrollo del proyecto y fase de documentación del proyecto. Cada una de estas líneas, está divida en líneas menores que incluyen las tareas a realizar en cada fase del proyecto.

8.1. Descripción de las fases y sus subfases

En este apartado, se explican las fases que han formado el proyecto y las tareas individuales que se han unido para componer cada línea del proyecto.

8.1.1. Gestión del proyecto

La primera de estas líneas, es la parte de gestión del proyecto, el cual estará presente desde el inicio hasta el fin. El mayor cómputo de tiempo en esta fase se ha realizado en

las primeras semanas del proyecto, ya que, se necesita realizar una planificación inicial que pueda dar una visión general del tiempo necesario y como ajustarlo al tiempo real disponible. Por eso mismo, esta fase se ha dividido en las siguientes líneas:

- **Planificación**: En este apartado, se ha intentado identificar los diferentes objetivos que había que cumplir para llevar a término el presente proyecto y que tareas se podían extraer de ellos, para paquetizar todo en tareas asumibles y concisas. Además, se ha realizado un cálculo inicial del coste temporal que van a suponer las tareas y en cuales de ellas el coste puede aumentar, creando para ello un plan de riesgos.
- **Comunicación**: El objetivo de esta tarea es mantener una comunicación periódica con el tutor del proyecto para poder realizar un seguimiento del mismo con él. Para ello, se realizarán reuniones con una periodicidad variable en las que se trataran los temas pendientes de las anteriores reuniones y los nuevos objetivos a corto plazo. De esa manera se pueden establecer tareas asumibles, que equilibren la carga de trabajo y realizar un seguimiento extenso del estado del proyecto.

8.1.2. Desarrollo del proyecto

La parte práctica del proyecto ha sido la que ha contenido la mayoría de las tareas a realizar en el mismo y la que ha tenido el coste temporal más elevado. Esta línea se ha realizado durante casi todo el proyecto. Iniciándose una vez terminada la primera planificación inicial y terminando al acabar el análisis de los datos.

Se ha dividido esta fase del proyecto en múltiples tareas que a su vez se han divido en subtareas para poder ajustarlas a los espacios temporales entre reuniones. Las tareas más generales han sido las siguientes: Estudio previo de las topologías, estudio previo del simulador, creación de las métricas, creación del entorno de pruebas y análisis de los datos.

- **Estudio previo topologías**: Primero de todo, se realizó un estudio previo de las topologías a crear en el simulador. Para ello, se revisó la documentación disponible sobre las mismas, revisando los artículos donde las definían.
- **Toma de contacto con el simulador**: Esta tarea se ha dividido en varias tareas más pequeñas, siendo la primera de ellas el estudio del simulador: que tipo de módulos tenía, como se habían implementado las topologías, etc. Las demás subtareas en este apartado corresponden con la creación en el simulador de cada nueva topología.

- **Creación de las métricas**: Adaptación de las métricas a las nuevas topologías.

- **Creación del entorno de pruebas**: Creación de un entorno de pruebas que permita extraer los datos para su posterior análisis y que permita verificar que la estructura de las topologías es correcta.

- **Análisis de los datos**: Análisis de datos obtenidos y su comparación con los datos teóricos.

8.1.3. Documentación del proyecto

Esta línea del proyecto es similar a las otras líneas ya que muchas tareas son compartidas. Las tareas a destacar son las siguientes: investigación y escritura y revisión.

- **Investigación**: Para poder realizar el proyecto, primero se ha tenido que realizar una investigación de las topologías a implementar. Además, para realizar la memoria del proyecto, se ha realizado un guión con las partes más importantes para poder dividir la escritura en fases bien diferenciadas.

- **Escritura y revisión**: Redacción de la memoria del proyecto y realización de la presentación para la defensa.

8.2. Estimación y desviación de las tareas

Una vez explicadas las líneas que componen el proyecto y descritas las tareas realizadas en cada fase del proyecto, es hora de mostrar mediante la siguiente tabla como ha sido el reparto de horas en el proyecto. Para ello, a cada tarea previamente identificada, se le concedió una estimación de horas necesarias para su realización y en la tabla 8.1 se observa cuál ha sido el tiempo real invertido en la misma.

	Horas planificadas	Horas reales
Gestión del proyecto	50h	45h
Planificación	40h	30h
Comunicación	10h	15h
Desarrollo del proyecto	145h	175h
Estudio previo topologías	20h	15h
Toma de contacto con el simulador	30h	20h
Creación inicial	15h	30h
Creación Dragonfly	5h	10h
Creación Slimfly	5h	15h
Creación Megafly	5h	5h
Métricas	20h	30h
Búsqueda, Allocate y release	0h	40h
Algoritmos de búsqueda Dragonfly	0h	15h
Algoritmos de búsqueda Slimfly	0h	15h
Algoritmos de búsqueda Megafly	0h	10h
Creación del entorno de pruebas	30h	20h
Análisis de los resultados experimentales	30h	20h
Documentación del proyecto	105h	110h
Guionización e investigación	30h	20h
Escritura	75h	90h
Total	300h	330h

Tabla 8.1: Horas proyecto

8.2.1. Desviaciones significativas

Las mayores desviaciones del proyecto han venido, como no podía ser de otra manera, de las tareas más grandes del desarrollo del proyecto, la toma de contacto inicial y la creación de las métricas.

Durante la tarea de creación de las topologías, en la fase de diseño de las estructuras de datos, se vio la posibilidad de reutilizar el diseño con cambios menores, reduciendo las horas reales necesarias para la creación de todas ellas. Sin embargo, durante la creación de una de ellas, Slimfly, el tiempo estimado se quedó corto, ya que la dificultad de comprensión de la construcción de la topología fue elevado, y se necesitó más tiempo para entender su funcionamiento y creación.

En cuanto a la otra desviación significativa que se ha producido en el proyecto, se ha debido a un error al implementar parte de las métricas, ya que no se estaban teniendo en cuenta algunos factores que alteraban los resultados. Por ello, se tuvieron que rehacer las

mismas, dedicándole más tiempo del planificado. Esto sucedió calculando el indicador de los nodos afectados ya que no se estaban incluyendo de manera adecuada los nodos afectados entre cada par de nodos que conformaban la partición seleccionada.

Aparte de esto, la tarea extra de búsqueda de nodos en base a grupos no estaba contemplada al inicio del proyecto, y durante su realización han sucedido los mismos problemas que en el resto del proyecto, siendo la implementación de la topología Slimfly más compleja que las otras.

Por último, al haberse alargado el proyecto, se han realizado más reuniones que las contempladas inicialmente.

8.3. Plan de riesgos

En todos los proyectos, se producen desviaciones e imprevistos y por ello, hay que contar con un plan de riesgos para solucionarlos. Aun así, muchas veces esos imprevistos son por factores que no se habían tenido en consideración al realizar el plan de riesgos inicial. Por lo tanto, hay que contar con un plan de riesgos que tenga una base preventiva y una base proactiva.

Con el fin de evitar la mayoría de problemas derivados del cálculo del coste de las tareas, a las tareas más importantes o amplias, se les ha asignado un tiempo estimado superior al que se podía estimar inicialmente. Esto se debe a que una tarea con un coste temporal significativo siempre va a estar más expuesta a las desviaciones.

Con el fin de evitar problemas derivados del apartado de la documentación y memoria siempre se ha tenido en cuenta las fechas disponibles para la presentación de los trabajos de fin de grado. Pero en el apartado del desarrollo se ha tenido en cuenta la primera de esas fechas como fecha fin, para evitar la dilatación de los tiempos y evitar que se caigan en el olvido los problemas superados y los conocimientos adquiridos.

8.4. Metodología de trabajo

Para llevar el proyecto a su finalización, se ha decidido que la mejor manera de orientar la realización del proyecto era mediante el establecimiento correcto de tareas a realizar entre las sesiones de tutoría, ya que en este caso particular, aparte del proyecto, había una

jornada laboral que cumplir. Por lo tanto, no se podía retrasar la realización del proyecto y no se podían asumir esprines significativos.

Debido a ello, la división del proyecto en tareas asumibles a realizar para fechas concretas ha sido la clave de la metodología de trabajo utilizada. Siguiendo este planteamiento, la realización del proyecto se alargó inicialmente durante 20 semanas, comenzando en el mes de Febrero y finalizando en la segunda semana de Junio.

Debido a retrasos en la escritura del proyecto y menor disponibilidad horaria en Junio, se decidió alargar el proyecto a Septiembre. Para aprovechar esta situación, se decidió añadir un apartado extra al desarrollo del proyecto con el tiempo extra disponible, ésta es: la búsqueda, asignación y liberación de nodos específicos a las topologías.

El tiempo dedicado siempre ha sido entorno a las 15h semanales, exceptuando los momentos vacacionales en los que se ha podido dedicar algo más de tiempo para compensar las pequeñas desviaciones surgidas por cargas de trabajo puntuales, superiores a las estimadas.

8.5. Cronograma del proyecto

Para estructurar el proyecto y su coste horario se ha utilizado una variación del diagrama de gantt, un cronograma de proyecto, para visualizarlo de manera gráfica.

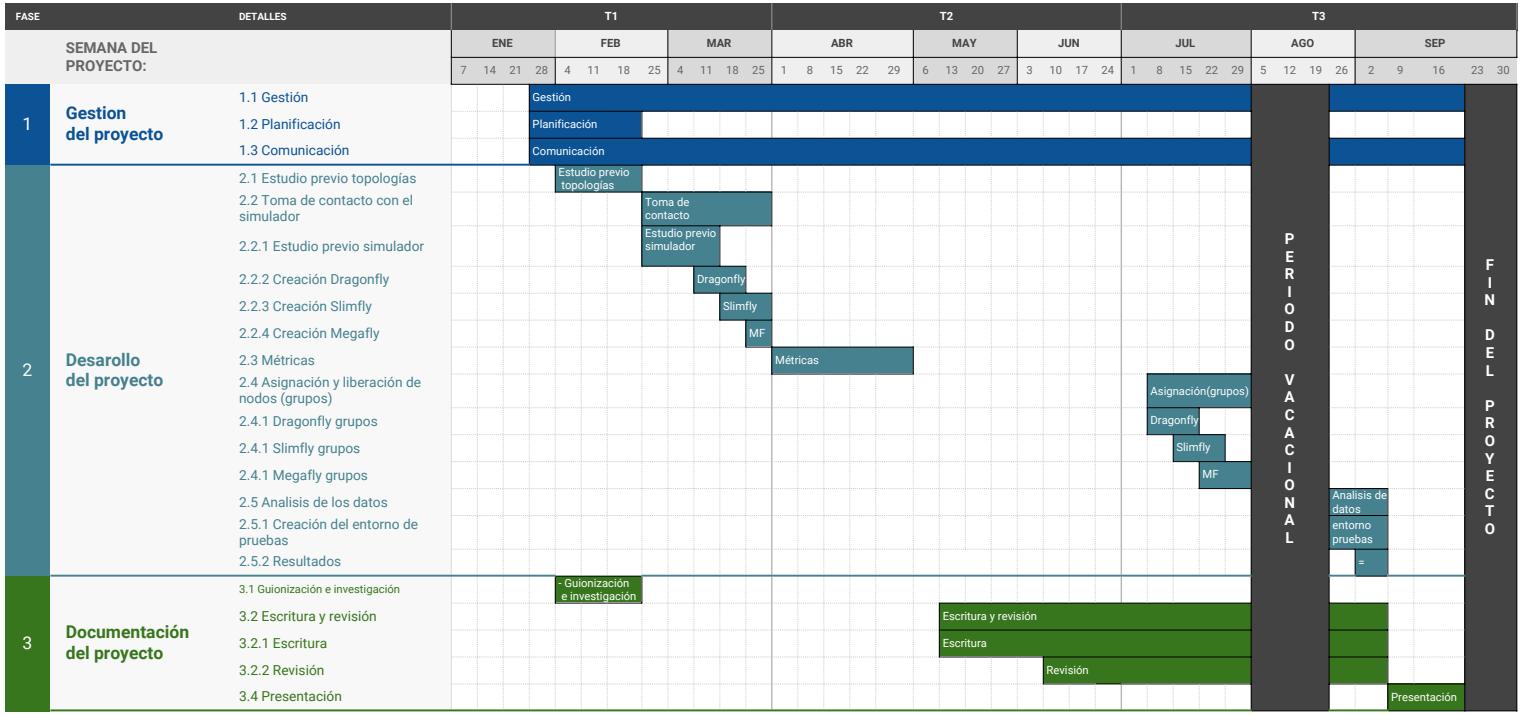


Figura 8.1: Cronograma del proyecto

9. CAPÍTULO

Conclusiones

En lo que respecta al apartado técnico, se han conseguido alcanzar los objetivos definidos al inicio del proyecto. Cabe destacar, que se han obtenido diversos conceptos teóricos y prácticos en la realización del proyecto.

- En cuanto a los conceptos teóricos, se ha profundizado en el estudio de diferentes topologías de red usadas en supercomputación como son Dragonfly, Slimfly y Megafly. Además, al utilizar el simulador SSE, se ha profundizado en el estudio de diferentes formas de realizar búsqueda de nodos en el proceso de scheduling.
- En el apartado práctico, se han realizado las implementaciones de varias topologías de red, sus métricas asociadas y distintos algoritmos de búsqueda de nodos. Por lo tanto, se ha profundizado en el conocimiento del lenguaje Python de programación.

Respecto al apartado de conocimientos, se ha conseguido comprender las dificultades de realizar un proyecto dentro de unos plazos determinados: en particular, la dificultad de gestionar el tiempo y de planificar los procesos/tareas a realizar.

En el ámbito personal, me reconforta mucho haber podido realizar un proyecto de estas características después de haber tenido que retrasar la realización del TFG y por ende la finalización del grado, debido al trabajo y otras decisiones personales. Me ha permitido valorar mi propio trabajo y mi implicación en proyectos de estas dimensiones de otra manera ayudándome a manejar la intranquilidad que me provocan.

10. CAPÍTULO

Trabajo futuro

Para finalizar se van a mencionar algunas posibles líneas para continuar el trabajo:

- Una de las posibles mejoras que se podrían realizar sería intentar crear otros algoritmos de búsqueda de nodos específicos para cada una de las tres topologías que se comporten de manera más óptima que los propuestos.
- Ampliar el estudio realizado considerando topologías de red con diferentes tamaños y configuraciones y conjuntos de aplicaciones diferentes para evaluar su comportamiento.
- Otra de las mejoras, es la creación visual de las topologías: la mejor forma de comprender una topología de red es visualizarla. Debido a la forma en la que se han creado las topologías, no debería ser complejo utilizar algún módulo de python especializado en la visualización de grafos para trasladarlo a las topologías desarrolladas.
- SSE es un simulador que no simula las comunicaciones. Dado que el código que genera las topologías ya está desarrollado no debería ser complicado su traspaso a otros simuladores que simulen la red de manera explícita como puede ser el simulador INSEE.

Bibliografía

[Adiga et al., 2005] Adiga, N. R., Blumrich, M. A., Chen, D., Coteus, P., Gara, A., Giampapa, M. E., Heidelberger, P., Singh, S., Steinmacher-Burow, B. D., Takken, T., Tsao, M., and Vranas, P. (2005). Blue gene/l torus interconnection network. *IBM Journal of Research and Development*, 49(2.3):265–276.

[Alverson et al., 2010] Alverson, R., Roweth, D., and Kaplan, L. (2010). The gemini system interconnect. In *2010 18th IEEE Symposium on High Performance Interconnects*, pages 83–87.

[Besta and Hoefler, 2014] Besta, M. and Hoefler, T. (2014). Slim fly: A cost effective low-diameter network topology. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 348–359.

[Chen et al., 2011] Chen, D., Eisley, N. A., Heidelberger, P., Senger, R. M., Sugawara, Y., Kumar, S., Salapura, V., Satterfield, D. L., Steinmacher-Burow, B., and Parker, J. J. (2011). The ibm blue gene/q interconnection network and message unit. In *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10.

[Cray, 2012] Cray, I. (2012). Cray XC Series Network. https://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf/. [Online; accessed 10-June-2019].

[Flajslik et al., 2018] Flajslik, M., Borch, E., and Parker, M. A. (2018). Megafly: A topology for exascale systems. In Yokota, R., Weiland, M., Keyes, D., and Trinitis, C., editors, *High Performance Computing*, pages 289–310, Cham. Springer International Publishing.

[Kim et al., 2007] Kim, J., Dally, W., and Abts, D. (2007). Flattened butterfly: a cost-efficient topology for high-radix networks. volume 35, pages 126–137.

[Kim et al., 2008] Kim, J., Dally, W. J., Scott, S., and Abts, D. (2008). Technology-driven, highly-scalable dragonfly topology. In *2008 International Symposium on Computer Architecture*, pages 77–88.

[McKay et al., 1998] McKay, B. D., Miller, M., and Širáň, J. (1998). A note on large graphs of diameter two and given maximum degree. *Journal of Combinatorial Theory, Series B*, 74(1):110 – 118.

[Singh, 2017] Singh, S. (2017). Primitive root of a prime number n modulo n. https://www.geeksforgeeks.org/primitive-root-of-a-prime-number-n-modulo-n/. [Online; accessed 6-September-2019].

[top500, 2018] top500 (2018). Top 500 November 2018. https://www.top500.org/. [Online; accessed 10-June-2019].