

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales.

Buenos Aires
Argentina

Tesis de Licenciatura en Ciencias de la Computación

Algoritmos paralelos y arquitecturas

Ruteo minimal, escalable, full-adaptivo, deadlock free, worm-hole para topologías Bruijn

Abril de 1999

Director de tesis

Gustavo D. Pifarré

Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

E.Mail: pifarre@ar.ibm.com

Tesistas

Harari, Oscar Marcos

Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

LU:206/91

E.Mail: oharari@illotempore.com

Goyeneche, Héctor Ariel

Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

LU:357/91

E.Mail: goyene@dc.ba.ar

Índice de contenidos

1. Resumen	6
2. Contenido del trabajo realizado	7
3. Introducción al tema	8
4. Topologías	9
4.1 Características de las topologías	9
4.2 Algunas topologías.....	9
4.2.1 Hipercono.....	9
4.2.2 Butterfly	10
4.2.3 Cube-Connected-Cycles	10
4.2.4 Shuffle-Exchange	10
4.2.5 Buijn	11
4.3 Análisis y características de las topologías presentadas.....	11
4.3.1 Hipercono	11
4.3.2 Butterfly	12
4.3.3 Cube-connected-cycles	12
4.3.4 Shuffle-Exchange y Buijn	12
4.3 Cuadro comparativo de las topologías	13
5. Ruteos	14
5.1. Condiciones a tener en cuenta.....	14
5.1.1 Tipo de inyección	14
5.1.2 Network delay, latencia.....	14
5.1.3 Deadlock.....	14
5.1.4 Livelock	15
5.1.5 Starvation	16
5.2 Características de los ruteos	16
5.2.1 Minimal.....	16
5.2.2 Full-Adaptivo	16
5.2.3 Adaptivo	16
5.2.4 Oblivious.....	17
5.2.5 Escalable.....	17
5.3 Modelos de mensajes.....	17
5.3.1 Packet-switching	17
5.3.2 Worm-hole	17
5.3.3 Pipeline circuit switching: (PCS)	17
5.3.4 Acknowledged pipeline circuit switching (APCS).....	18
5.3.5 Scouting	18
5.4. Comparación entre modelos de mensajes	18
5.4.1 Latencia	18
5.4.3 Fallas.....	19
6. Redes de Buijn Direccionales - Función de Ruteo	22
6.1 Propiedad de la conservación de bits	22
6.2 Cálculo de la longitud de los caminos mínimos.....	23
6.3 Teorema del camino único	25
6.4 Función de Ruteo.....	26
7. Grafo dirigido acíclico (DAG) para Buijn Direccional.	28
7.1. Idea intuitiva	28
7.2. Resultados Conocidos.....	29
7.3. Definiciones	30
7.4. Características de los vector- W s y su relación con el período	31
7.5. Funciones de comparación	32
7.5.1 Definiciones.....	33
7.5.2 Definiciones: AvgW1, AvgW2.	34

7.5.3	Orden total de los nodos	34
7.5.4	Definiciones de los órdenes: Orden1, Orden2, Orden3, Orden4.....	34
7.6.	Comparaciones según el comportamiento	35
7.6.1	Propiedad de los valores contiguos constantes.	35
7.6.2	Propiedad de los valores contiguos crecientes.	36
7.6.3	Propiedad de los valores contiguos decrecientes.	37
7.7.	Análisis del comportamiento de las ocurrencias del vector-W	38
7.7.1	Propiedad de la posición del vector-W creciente.....	38
7.7.2	Propiedad de la cantidad de ocurrencias del vector-W en un camino mínimo.	39
7.7.3	Propiedad del comportamiento de vector-W en caminos mínimos.....	39
7.7.4	Propiedad de la cantidad de movimientos.	41
7.8.	DAG en Bruijn	41
7.9.	Análisis de la cantidad de buffers necesarios según el DAG presentado.	42
7.9.1	Abandono del nivel de buffers 1	42
7.9.2	Abandono del nivel de buffers 2	42
7.9.3	Abandono del nivel de buffers 3	43
7.9.4	Abandono del nivel de buffers 4	43
8	Redes de Bruijn Bidireccional.....	44
8.1	Propiedades preliminares de las Redes de Bruijn Bidireccionales	44
8.1.1	Definición de ventana	44
8.1.2	Existencia de ventanas en caminos mínimos	45
8.1.3	Corolario de la conservación de la ventana	45
8.2	Conjunto de Ventanas	46
8.2.1	Definición de secuencia	46
8.2.2	Definición de configuración	46
8.2.3	Definición de orden en configuraciones.....	46
8.2.4	Definición de posición en configuraciones	47
8.3	Propiedad de las ventanas en orden y posición	47
8.4	Teorema de configuraciones sin orden.....	48
8.5	Teorema de configuraciones sin posición.....	48
8.6	Ventana única	48
8.7	Cálculo de la longitud de un camino	48
8.8	Función de Ruteo.....	51
9.	Grafo dirigido acíclico (DAG) para Bruijn Direccional.	54
9.1.	Introducción	54
9.2.	Definiciones.....	54
9.3.	Cantidad necesaria de movimientos contiguos	54
9.5	Presentación del DAG en Bruijn Direccional Inverso.....	54
9.6	DAG en Bruijn Bidireccional.....	55
10.	Conclusiones.....	57
11.	Extensiones posibles al presente trabajo.....	58
12.	Gráficos.....	59
12.1.	Topologías.....	59
12.2.	Comparación de técnicas de ruteo	61
12.3.	Cuadros de identificadores ordenados	63
13.	Bibliografía	65

Índice de figuras y cuadros

Figura 5.1.3.A: Anillo de 5 nodos.	15
Figura 5.1.3.B: DAG para un anillo de 5 nodos.	15
Figura 5.4.2.4.A: Generación de kill flits ante la presencia de una falla.	20
Figura 6.1.A: Desplazamiento de Bits	22
Figura 6.2.A: Ventana no volcada a la derecha en nodo origen	24
Figura 6.2.B: Ventanas no volcada a la izquierda en nodo destino	24
Figura 6.3.A: Bifurcación en caminos	25
Figura 7.7.A: Ejemplo de caminos mínimos	49
Figura 8.7.B: Parámetros de función de ruteo	50
Figura 8.7.C: Extremos opuestos	50
Figura 8.7.D: Mismo extremo	50
Figura 8.7.E: Ventanas libres	51
Figura 12.1.1: Hipercubos n-dimensionales para n igual a 2, 3 y 4.	59
Figura 12.1.2: Butterfly 3-dimensional.	59
Figura 12.1.3: CCC 3-dimensional.	60
Figura 12.1.5: Bruijn 3-dimensional.	60
Figura 12.2.1: Diagrama de tiempo espacio para ruteo worm-hole.	61
Figura 12.2.2: Diagrama de tiempo espacio para ruteo pipeline circuit switching	61
Figura 12.2.3 Diagrama de tiempo espacio para ruteo acknowledged pipeline circuit switching	62
Figura 12.2.4: Diagrama de tiempo espacio para ruteo scouting.	62
Cuadro 12.3.1: Identificadores pertenecientes a $V(2,5)$ ordenados según $<_1$ y $<_3$	63
Cuadro 12.3.2: Identificadores pertenecientes a $V(2,5)$ ordenados según $<_2$ y $<_4$	64

1. Resumen

Para que los nodos de una computadora masivamente paralela puedan comunicarse, es necesario el intercambio de información. Los algoritmos de ruteo tienen por función llevar a cabo esta comunicación. Cada uno de estos algoritmos posee diferentes características que habilitan comparaciones entre sí.

Un algoritmo de ruteo es minimal cuando sólo puede disponer de caminos de longitud mínima; escalable cuando utiliza una cantidad de buffers constante por nodo independiente de la cantidad de nodos de la computadora masivamente paralela; full-adaptivo cuando se utilizan de todos los caminos posibles; worm-hole cuando el tipo de mensajes ruteados no tienen longitud fija, sino que los mismos se dividen en segmentos de tamaño constante.

Se conocen algoritmos minimales, escalables, deadlock-free para las topologías más estudiadas (meshes, hipercubos, toros). Sin embargo, no es conocido un algoritmo con estas características para las topologías de Bruijn.

En este trabajo se presentan y desarrollan dos importantes resultados originales. Primero, se expone un algoritmo de ruteo minimal, escalable, full-adaptivo, deadlock free y worm-hole para las topologías de Bruijn Direccionales. Segundo, se presenta un algoritmo con idénticas características para las topologías de Bruijn Bidireccionales.

2. Contenido del trabajo realizado

La presente tesis de licenciatura está estructurada en dos secciones.

La primera de éstas es una recopilación del estado del arte del tema que abarca los capítulos 3, 4 y 5. Luego de una breve introducción desarrollada en el capítulo 3, en el capítulo 4 se focaliza en diferentes topologías y sus características. En el capítulo 5 se exponen las diferentes características y propiedades de los algoritmos de ruteo.

La segunda sección constituye la más importante y se extiende desde el capítulo 6 al 9.

Generalmente, cuando se presentan y demuestran algoritmos de ruteo para arquitecturas masivamente paralelas, se establece una función de ruteo y un grafo acíclico dirigido sobre la topología seleccionada. La función de ruteo se encarga de dirigir los mensajes desde el nodo origen al nodo destino, respetando el grafo presentado. El grafo asegura que el algoritmo sea deadlock-free (ver 7.2)

En el capítulo 6 y 7 se presenta un algoritmo original de ruteo escalable, minimal, deadlock-free, full-adaptivo, worm-hole para redes de Bruijn Direccionales utilizando la técnica anteriormente descripta. En el primero de éstos, se trabaja sobre la función de ruteo y en el segundo sobre el grafo acíclico dirigido. Ambos conforman el algoritmo de ruteo.

Concluyendo, se desarrolla un algoritmo original de ruteo escalable, minimal, deadlock-free, full-adaptivo, worm-hole para redes de Bruijn Bidireccionales. De igual manera que para las redes de Bruijn Direccionales, en el capítulo 8 se trabaja sobre la función de ruteo y en el capítulo 9 sobre el grafo acíclico dirigido.

Finalmente, en el capítulo 10 se exponen las conclusiones del trabajo y en el siguiente capítulo se plantean posibles extensiones a esta tesis de licenciatura.

3. Introducción al tema

El amplio desarrollo alcanzado en el área de VLSI permite la construcción de computadoras masivamente paralelas con cientos o miles de procesadores.

Una computadora masivamente paralela o multiprocesador es una red de procesadores, buffers y ejes. Dos procesadores están conectados entre sí por ejes. Estos ejes pueden ser Direccionales o Bidireccionales. La forma de conexión de los procesadores es lo que define la topología de la red.

Cada topología tiene distintas características que pueden influir en la elección de la misma para la resolución de un problema en particular. Principalmente se tendrá en cuenta el grado (*ver 4.1.1*), regularidad (*ver 4.1.2*), diámetro (*ver 4.1.3*) y bisection-width (*ver 4.1.4*).

En estas nuevas arquitecturas resulta crítica la comunicación entre los procesadores para que puedan cooperar entre ellos y realizar tareas en conjunto. Esta comunicación, comúnmente llamada ruteo, varía según las características de la computadora paralela a la cual se aplique.

Los aspectos más importantes en el diseño de algoritmos de ruteo son: evitar situaciones de deadlock (*ver 5.1.3*), livelock (*ver 5.1.4*) y starvation (*ver 5.1.5*); lograr la mayor performance posible en términos de mensajes ruteados por la red y obtener baja latencia de comunicación (*ver 5.4.2*). Asimismo también es importante asegurar que todos los pares de nodos se puedan comunicar aún en presencia de fallas (*ver 5.4.3*) de los componentes de la red. Además, es deseable que un algoritmo sea full-adaptivo (*ver 5.2.2*) lo que garantiza la posible utilización de cualquier camino entre origen y destino.

El ruteo se puede realizar con diferentes modelos de mensajes. Se pueden citar como los más notables packet-switching (*ver 5.3.1*) y worm-hole (*ver 5.3.2*). También existen algunos modelos híbridos tales como pipeline-circuit-switching (*ver 5.3.3*), acknowledged-pipeline-circuit-switching (*ver 5.3.4*) y scouting (*ver 5.3.5*) que generaliza los casos anteriores. No todos los algoritmos de ruteo soportan todos estos modelos de mensaje.

4. Topologías

4.1 Características de las topologías

- 4.1.1 Grado constante:** El grado de un nodo es el número de ejes que acceden a este. Una topología de grado constante es aquella en la que el grado de cada nodo se mantiene constante a medida que crece en cantidad de nodos. Esta es una buena propiedad dado que la complejidad de implementación de un nodo no varía con el tamaño de la red.
- 4.1.2 Vertisimétrico o Regular:** Una topología es vertisimétrica si para todo par de nodos existe una función biyectiva entre los ejes que acceden y parten de él y los nodos que comunican. Esta propiedad resulta interesante para poder tratar a todos los nodos de igual manera, sin tener que considerar excepciones.
- 4.1.3 Diámetro:** Es el camino máximo de todos los caminos mínimos que conectan a dos nodos cualquiera. (Contando los ejes). Esta medida acota superiormente la cantidad de ejes atravesados utilizando solo caminos mínimos.
- 4.1.4 Bisection-width:** Es el número mínimo de ejes a ser removidos para particionar la red en dos mitades con idéntica cantidad de procesadores. Frecuentemente para algunos problemas es necesario transmitir mensajes de una mitad de la red a la otra, por lo tanto el Bisection-width es un factor crítico para determinar la velocidad en que la red puede producir un cálculo.

4.2 Algunas topologías

A continuación, se definirán las topologías más usuales y representativas. Una de éstas (Bruijn) va a ser profundamente estudiada en el resto de este trabajo.

Se puede imaginar cualquier *topología* como un grafo $G = (V, E)$ donde:

- i) V son los nodos del grafo que representan los nodos de las redes
- ii) E son los ejes del grafo que representan los ejes de las redes.

Se llamará $V(m, n)$ a todo vector α , tal que $\alpha = (\alpha_0, \dots, \alpha_{n-1})$ donde $0 \leq \alpha_i \leq m-1$ para todo i , $0 \leq i \leq n-1$. Se llamará valor menos significativo del vector α (o bit menos significativo cuando m es igual a 2) a α_0 . Asimismo el valor más significativo del vector α (o bit más significativo cuando m es igual a 2) es α_{n-1} .

4.2.1 Hipercubo

Un hipercubo n -dimensional, $H(n)$, tiene 2^n nodos llamado N y $n \cdot 2^{(n-1)}$ ejes.

Cada nodo corresponde a un vector distinto α tal que $\alpha \in V(2, n)$.

Dos nodos cualesquiera α y β pertenecientes a $H(n)$ están conectados con un eje $e(\alpha, \beta)$ si

- 1) $\alpha_i \neq \beta_i$ para algún $i : 0 \leq i \leq n-1$

y

- 2) $\alpha_j = \beta_j$ para todo $j : 0 \leq j < i$ y $i < j \leq n-1$

O sea si y solo si α y β difieren en un solo bit.

Como consecuencia de lo anterior, cada nodo de un hipercubo es incidente a $n=\log(N)$ otros nodos, uno por cada posición de bit en la cual pueda diferir. (Ver 13.10)

(Ver figura 12.1.1)

4.2.2 Butterfly

La butterfly n -dimensional tiene $(n+1)2^n$ nodos llamado N y $n2^{(n+1)}$ ejes. Los nodos identificados como pares $\{w, i\}$ donde i es el nivel o dimensión de los nodos ($0 \leq i \leq n$) y $w \in V(2, n)$. Dos nodos $\{w, i\}$ y $\{w', i'\}$ están unidos por un eje si $i' = i + 1$ y también:

- 1) w y w' son idénticos,
- o
- 2) w y w' difieren precisamente en el bit i -ésimo..

Si w y w' son idénticos, el eje es llamado eje derecho. De otra manera el eje es llamado eje cruzado.

(Ver figura 12.1.2)

4.2.3 Cube-Connected-Cycles

Un Cube-connected-cycles (CCC) n -dimensional se construye a partir de un Hipercubo n -dimensional, donde se reemplaza cada nodo del Hipercubo por un ciclo de n nodos. El eje de la i -ésima dimensión que incide a un nodo del Hipercubo es entonces conectado al i -ésimo nodo del correspondiente ciclo en el CCC. El grafo resultante tiene $n2^n$ nodos llamado N cada uno con grado 3. Para modificar la forma de identificar a cada nodo, se puede representar a éstos como un par $\{w, i\}$ donde i ($1 \leq i \leq n$) es la posición de un nodo dentro de este ciclo y $w \in V(n, 2)$ es el identificador del nodo del Hipercubo que corresponde a ese ciclo.

Entonces 2 nodos $\{w, i\}$ y $\{w', i'\}$ están unidos por un eje en un CCC si:

- 1) $w = w'$ y $i - i' = \pm 1 \text{ mod } n$
- o
- 2) $i = i'$ y w difiere desde w' en precisamente el i -ésimo bit.

Los ejes del primer tipo son llamados *ejes cíclicos*, mientras que los del segundo tipo son llamados *ejes hipercubos*.

(Ver figura 12.1.3)

4.2.4 Shuffle-Exchange

Una Shuffle-Exchange n -dimensional tiene 2^n nodos llamados N y $3 \cdot 2^{n-1}$ ejes. Cada nodo corresponde a un único vector $V(2, n)$, y dos nodos α y β están unidos por un eje Direccional si:

- 1) $\alpha_i = \beta_i$ para $0 \leq i < n-1$ y $\alpha_{n-1} \neq \beta_{n-1}$. Es decir, α y β difieren precisamente en el último bit (el más significativo)
- o
- 2) $\alpha_i = \beta_{i-1}$ para $0 < i \leq n-1$ y $\alpha_0 = \beta_{n-1}$. Es decir β es igual α desplazado hacia la izquierda.

Una topología Suffle-Exchange bidireccional permitiría desplazar, en el punto 2, el identificador del nodo α tanto a la derecha como a la izquierda, eliminando los ejes repetidos.

Si α y β difieren en el último bit, el eje es llamado *eje exchange*. De otra manera, el eje es llamado *eje shuffle*.

(Ver figura 12.1.4).

4.2.5 Bruijn

La topología Bruijn, notada como $B(n)$, consiste en 2^n nodos llamados N y 2^{n+1} ejes direccionales. Cada nodo corresponde a un vector $V(2,n)$. Dos nodos $\alpha=(\alpha_0, \dots, \alpha_{n-1})$ y $\beta=(\beta_0, \dots, \beta_{n-1})$ están unidos por un eje direccional notado como $L(\alpha, \beta)$ si:

$$1) \alpha_i = \beta_{i-1} \text{ para } 0 < i \leq n-1$$

y

$$2) 0 \leq \beta_{n-1} \leq 1.-$$

Al conjunto de ejes que pertenecen a $B(n)$ lo notará $LG(B(n))$.

(Ver figura 12.1.5).

La topología Bruijn Bidireccional, notada como $UB(n)$, consiste en 2^n nodos llamados N . Cada nodo corresponde a un vector $V(2,n)$. Dos nodos $\alpha=(\alpha_0, \dots, \alpha_{n-1})$ y $\beta=(\beta_0, \dots, \beta_{n-1})$ están unidos por un eje direccional notado como $L(\alpha, \beta)$ si:

$$1) \alpha_i = \beta_{i-1} \text{ para } 0 < i \leq n-1 \quad \text{y} \quad 0 \leq \beta_{n-1} \leq 1.-$$

o

$$2) \alpha_i = \beta_{i+1} \text{ para } 0 \leq i < n-1 \quad \text{y} \quad 0 \leq \beta_0 \leq 1.-$$

4.3 Análisis y características de las topologías presentadas

4.3.1 Hipercubo

Dado un $H(n)$, se denominan *ejes de dimensión k* a aquellos ejes pertenecientes a $H(n)$ que conectan dos nodos que difieren en la k -ésima posición.

Dado un k , $0 \leq k \leq n-1$, removiendo todo el conjunto de ejes de dimensión k se obtienen dos hipercubos disconexos de dimensión $n-1$. Consecuentemente, un hipercubo de dimensión n , puede ser construido a partir de dos hipercubos $H'(n-1)$ y $H''(n-1)$. Esto se logra agregando ejes entre todos los identificadores de igual valor de $H'(n-1)$ y $H''(n-1)$, y concatenando a cada identificador de $H'(n-1)$ el valor 0 (como bit mas significativo) y 1 a los de $H''(n-1)$.

Por lo expuesto en el párrafo anterior, el hipercubo posee una estructura recursiva. Además, el diámetro es pequeño $\log(N)$, su bisección-width es alto: $(N/2)$ y es vertisimétrico.

Entre sus desventajas, para su uso en arquitecturas paralelas, se puede mencionar que el grado de los nodos crece con su tamaño ya que no es de grado constante.

4.3.2 Butterfly

La butterfly y el hipercubo son muy similares en estructura. En particular, el i -ésimo nodo de un hipercubo n -dimensional corresponde naturalmente a la i -ésima columna de una butterfly n -dimensional, y un eje i -ésimo (u,v) de un hipercubo n -dimensional corresponde a un eje cruzado $(\{u,i-1\},\{v,i\})$ y $(\{v,i-1\},\{u,i\})$ en el nivel i de la butterfly n -dimensional.

En efecto, el hipercubo es un plegado de la butterfly (Se puede obtener un hipercubo de una butterfly fusionando todos los nodos butterfly que están en la misma fila y luego removiendo los ejes duplicados).

Como existe una gran similitud entre el hipercubo y la butterfly, ésta última también tiene buenas propiedades. Primeramente, tiene una estructura recursiva, es vertisimétrica y no es de grado constante. Otra importante propiedad de una butterfly n -dimensional, es que el nivel de nodo 0 en cualquier fila w está ligado al nodo del nivel r en cualquier fila w' por un único camino de longitud r . El camino atraviesa exactamente cada nivel una sola vez, usando los ejes cruzados desde el nivel i al nivel $i+1$ si w y w' difieren en el $(i+1)$ -ésimo bit.

Como el hipercubo, la butterfly también tiene un gran bisection-width. Esta medida es del orden de $(N/\log(N))$.

4.3.3 Cube-connected-cycles

Por la forma de construcción es simple apreciar la similitud de un CCC con la estructura del hipercubo. En particular, cualquier paso de un hipercubo n -dimensional puede ser simulado en $\log(n)$ pasos en un CCC n -dimensional usando cada ciclo de tamaño $\log(n)$ del CCC para simular la acción del correspondiente nodo en el hipercubo.

Las propiedades que presenta esta topología, a diferencia del hipercubo, es que tiene grado constante. Además se puede decir que tiene un diámetro de orden $(\log(N))$ y bisection-width de orden $(N/\log(N))$ y es vertisimétrica.

4.3.4 Shuffle-Exchange y Bruijn

En esta sección, se describirán dos redes que poseen una similitud en sus estructuras. Estas son Shuffle-exchange y Bruijn. Se puede apreciar que cada paso en una topología de Bruijn es equivalente a realizar a lo sumo dos pasos en la topología Shuffle-exchange, un eje shuffle y si es necesario un exchange.

A simple vista no es posible apreciar las propiedades de ambas topologías. El diámetro de Shuffle-exchange es de $2\log(N-1)$. En Bruijn el diámetro también es de orden logarítmico, $\log(N-1)$.

El bisection-width de estas topologías es de orden $(N/\log(N))$

Otra de las características importantes de estas es el que tienen grado constante.

Es bueno mencionar que estas topologías son vertisimétricas o regulares. Esta característica es favorable para las funciones de ruteo, dado que no tienen que hacer consideraciones especiales de acuerdo al nodo donde se encuentran.

4.3 Cuadro comparativo de las topologías

	Hipercubo n-dimensional	Butterfly n-dimensional	CCC n-dimensional	Shuffle-Exchange n-direccional	Bruijn n-direccional
Diámetro	Orden($\log(N)$)	Orden($\log(N)$)	Orden($\log(N)$)	Orden($\log(N-1)$)	Orden($\log(N-1)$)
Vertisimetría	Sí	Sí	Sí	Sí	Sí
Bisection width	Orden($N/2$)	Orden($N/\log(N)$)	Orden($N/\log(N)$)	Orden ($N-\log(N)$)	Orden($N-\log(N)$)
Grado Constante	No	No	Sí	Sí	Sí

5. Ruteos

Dentro de los nodos se encuentran las funciones de ruteo que van a determinar, ante la presencia de un mensaje que proviene de un origen, cuál va a ser el eje de salida para llegar su destino.

Las funciones de ruteo pueden ser comparadas tomando como parámetro la cantidad de recursos críticos que utilizan. Principalmente se consideran los buffers que sirven para almacenar los mensajes.

Cada nodo de la red puede tener asociado un conjunto de buffers ordenados en dos colas distintas, llamadas de inyección y de entrega. Los mensajes deben ser inyectados en la red en la primera cola y éstos deben ser consumidos desde la segunda.

Además de acceder a un destino dado en la menor cantidad de pasos posibles y utilizando todas las alternativas, también se trata de tener baja *latencia* y evitar situaciones de *deadlock*, *livelock* y *starvation*. A continuación se describirá cada uno de estos puntos.

5.1. Condiciones a tener en cuenta

5.1.1 Tipo de inyección

Cuando dos nodos quieren comunicarse, utilizan el pasaje de mensajes. Estos mensajes deben ser inyectados en la red.

Si todos los nodos inyectan al mismo tiempo la inyección es clasificada como *estática*.

Por el contrario, si cada nodo inyecta los mensajes en un tiempo arbitrario, el modelo de inyección es *dinámico*.

5.1.2 Network delay, latencia

Es el tiempo que tarda un mensaje desde que es inyectado en la red hasta que alcanza su destino final. Cada algoritmo de ruteo debería tener baja latencia para entregar los mensajes a su destino.

5.1.3 Deadlock

Garantizar en una técnica de ruteo la no existencia de deadlock es un problema importante. Todos los algoritmos deben contar con algún mecanismo con el fin de evitar esta situación o bien permitirla, detectarla y finalmente recuperarla.

El gran problema con las diferentes técnicas de ruteo es el uso de los buffers que son utilizados para almacenar mensajes. Un deadlock puede ocurrir si y solo si existe un conjunto de recursos (buffers) ocupados por mensajes que no pueden arribar a sus destinos porque todos estos están esperando un recurso que no tienen (un próximo buffer en el siguiente nodo del camino a su destino) y esta espera genera en forma global un ciclo de esperas.

En este punto todos estos mensajes se encuentran bloqueados dado que ninguno puede acceder al próximo nodo en su camino por estar en una situación idéntica a la anterior.

Si uno establece un orden estricto monotónico de los recursos críticos de manera tal que no se produzcan ciclos y la función de ruteo pueda cumplir con éste, se puede garantizar que la función va a

estar libre de deadlock. En caso contrario, existen dos posibles políticas para solucionar este tipo de problemas: los mensajes pueden ser descartados con el consecuente overhead que la recuperación del mismo acarrea o puede ser desruteado.

El orden estricto monotónico de los recursos críticos (buffers) produce un grafo acíclico dirigido (DAG).

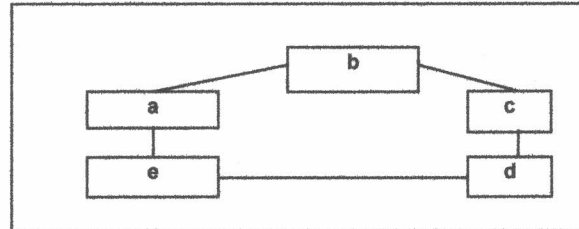


Figura 5.1.3.A: Anillo de 5 nodos.

Considérese un anillo de cinco nodos (figura 5.1.3.A). Se puede generar un DAG para esta topología como es mostrado en la figura 5.1.3.B donde el nodo 'b' contiene 3 buffers y en cada uno de los demás 2.

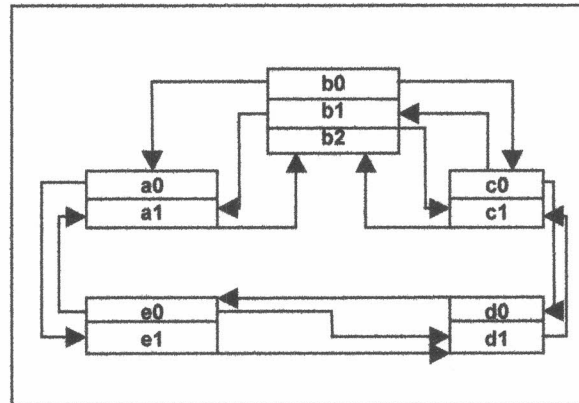


Figura 5.1.3.B: DAG para un anillo de 5 nodos.

5.1.4 Livelock

Este problema ocurre cuando un mensaje puede estar moviéndose infinitamente pero sin poder acceder a su destino. Esto se puede dar, por ejemplo, cuando un mensaje es desruteado de un nodo a otro y luego es desruteado del segundo al primero en forma permanente.

Obviamente en un modelo en el que se utilizan solo caminos mínimos de comunicación, la situación de livelock no puede ocurrir dado que en cada paso los mensajes se aproximan a su destino.

En otros casos, diferentes políticas han sido propuestas en orden de evitar este problema. Por ejemplo, se establece un conjunto de prioridades que están totalmente ordenadas. Al ser inyectado un mensaje, se le otorga una prioridad mínima. Cada vez que el mensaje se rutea de un nodo a otro se aumenta la prioridad. Luego, cada nodo ruteará teniendo en cuenta las prioridades mayores. (Ver 13.3)

Este tipo de políticas han sido criticadas porque tienen al menos dos inconvenientes. El primero es el hecho de que las colas de prioridades deben ser implementadas para mantener los paquetes ordenados resultando mucho más costosas que una simple cola FIFO. El segundo, es la necesidad de transmitir más información en los mensajes con el objetivo de guardar las prioridades.

5.1.5 Starvation

La situación de starvation ocurre cuando un mensaje no puede ser inyectado en la red. Esta situación no es factible si el modelo de inyección es estático y si se establece una política justa de utilización de los buffers (Donde se asegura la inyección de todos los datos almacenados en los buffers).

Si esto ocurre, el problema es evitado definiendo una política de control de la inyección. El objetivo es asegurar que cada nodo esté en condiciones de inyectar un paquete en la red. Algunas políticas propuestas son las siguientes:

a) Competición de inyección:

Todos los nodos tienen una cola donde se guardan los mensajes a ser inyectados en la red. Esta cola es considerada de la misma forma que las colas de los ejes entrantes al nodo compitiendo con éstas. Como el administrador de colas considera a todas con idéntica prioridad, este método evita starvation. (Ver 13.3)

b) Recirculación de buffers privados:

Esta política permite que cada nodo reserve alguna fracción de la cola de inyección. Cada paquete debe tener una marca para determinar si éste fue inyectado en la fracción privada o pública. Cuando un paquete es inyectado en una fracción privada del nodo fuente es enviado al destino final. El receptor usa el buffer utilizado por el que entregó el paquete para enviar un mensaje *null* hacia el emisor. Cuando un nodo necesita enviar un mensaje, este mira por un buffer libre en su propia cola. Si encuentra uno, el paquete es inyectado allí. De otra manera, el nodo debe esperar por una respuesta *null* enviada hacia él y luego envía el paquete utilizando este buffer. Esta política ha sido muy discutida porque genera un número importante de paquetes *null*. (Ver 13.3)

5.2 Características de los ruteos

5.2.1 Minimal

Un algoritmo se considera minimal si todo camino que utiliza es mínimo. La longitud de los caminos puede ser medida en ejes o en nodos.

5.2.2 Full-Adaptivo

Esta característica describe la capacidad de la función de ruteo para usar todos los caminos posibles para acceder a un nodo destino. En los puntos de bifurcación de caminos, la congestión de la red y la presencia de fallas pueden ser tenidas en cuenta.

Combinando las dos características anteriores se llega a conjugar un algoritmo full-adaptivo minimal, que se refiere a aquel que puede elegir entre todos los caminos mínimos posibles de dos nodos cualquiera.

5.2.3 Adaptivo

Esta característica describe la capacidad de la función de ruteo para usar algunos de los caminos posibles para acceder a un nodo destino.

5.2.4 Oblivious

Si el algoritmo considera solo un camino entre dos nodos de la red, se define como oblivious. Esta técnica es aplicada para reducir el conjunto de recursos necesarios en cada nodo.

5.2.5 Escalable

Un algoritmo es escalable si utiliza una cantidad de buffers constante por nodo independiente de la cantidad de nodos de la topología.

5.3 Modelos de mensajes

Usualmente dos tipos de técnicas son definidas: *Packet-switching* y *Worm-hole*. Además se han desarrollado otros casos que pueden ser considerados como técnicas híbridas. Estas son *Pipeline circuit switching* (PCS), *Acknowledged pipeline circuit switching* (APCS) y *Scouting*. (Ver 13.5)

Los últimos cuatro casos pueden ser incluidos dentro de la categoría de *mensajes pipeline*, donde los mensajes son divididos en flits de igual tamaño. El primero, denominado cabeza, es el que va a establecer el camino entre el origen y destino para que luego, o mientras tanto, los demás flits de datos puedan atravesar la red por el camino ya reservado. Por lo tanto, un mensaje puede ocupar buffers en múltiples nodos y estar desparramado a través de múltiples ejes. Este tipo de mecanismo es particularmente susceptible a fallas de los ejes.

5.3.1 Packet-switching

Los mensajes tienen una longitud constante y pequeña y se denominan paquetes. En este tipo de ruteo los paquetes son movidos desde un nodo a otro, almacenándose íntegramente en cada nodo hasta que es transmitido al próximo.

5.3.2 Worm-hole

En esta técnica los mensajes no tienen una longitud fija, sino que cada uno es dividido en una secuencia de *flits*. Estos son de longitud constante. El primer *flit* lleva consigo la dirección destino ya que es el encargado de determinar el camino entre el origen y el destino dentro de la red. Cuando un buffer es accedido por la cabeza del mensaje es ocupado y no será liberado hasta que el último flit del mensaje lo abandone. En worm-hole, los sucesivos flits que contienen el resto del mensaje son inyectados en la red, uno a la vez, inmediatamente después de que la cabeza del mensaje es introducida en la misma. (Ver 13.19)

5.3.3 Pipeline circuit switching: (PCS)

En esta técnica la cabeza del mensaje es primeramente ruteada hasta obtener un camino entre el nodo fuente y el destino, reservando buffers en los nodos que atraviesa. Cuando esto sucede, el destino retorna un asentimiento hacia el nodo fuente utilizando el camino ya reservado. Luego, los siguientes flits son inyectados en la red.

5.3.4 Acknowledged pipeline circuit switching (APCS)

Esta es una técnica muy similar a *pipeline circuit switching*, diferenciándose por el agregando de un asentimiento al final de la transmisión desde el destino hasta el nodo fuente.

5.3.5 Scouting

En los dos casos anteriores los flits de datos no son inyectados en la red hasta que se establece en forma completa un camino entre el origen y el destino. Esta técnica de comunicación resulta en un aumento importante del overhead y del tiempo de entrega de cada mensaje (latencia).

En scouting, la cabeza del mensaje siempre va adelante del primer flit de datos en al menos k ejes (este número es llamado distancia de scouting). Entonces, cuando la cabeza progresa, un asentimiento positivo es retornado a través del camino reservado. Cuando el número de asentimientos recibidos en un nodo intermedio es igual a la distancia de scouting los flits pueden avanzar. Si la cabeza tiene la necesidad de retroceder, esta puede enviar asentimientos negativos.

Obsérvese que este caso presentado puede ser interpretado como uno más general que abarca *Worm-hole* y *Acknowledged pipeline circuit switching*. Tomando la distancia de Scouting k igual a 1, se está en presencia de *Worm-hole*. Si se toma k igual al diámetro de la red sería equivalente a *Acknowledged pipeline circuit switching*.

5.4. Comparación entre modelos de mensajes

5.4.1 Latencia

En las figuras 12.2.1 hasta la 12.2.4 del capítulo "Gráficos", se presentan diagramas en el cual se puede visualizar la evolución de los flits que componen un mensaje a través del tiempo.

En la figura 12.2.1, correspondiente al modelo *worm-hole*, la cabeza avanza permanentemente y reserva exitosamente un canal. A medida que se produce este movimiento los flits de datos fluyen por los canales ya reservados a través de la red.

En la figura 12.2.2 (*pipeline circuit switching*) se puede observar que en cada movimiento de la cabeza un buffer es exitosamente reservado. La brecha entre la cabeza y el primer flit de datos se va incrementando hasta llegar a destino. Luego se envía un asentimiento que fluye en la dirección opuesta hasta el origen. A partir de ese momento los flits de datos fluyen por los buffers ya reservados a través de la red.

En la figura 12.2.3 (*acknowledged pipeline circuit switching*) simplemente se agrega a la figura anterior un último asentimiento desde el destino hacia el origen.

Finalmente, en la figura 12.2.4 se presenta el modelo de *scouting* con una distancia k igual a 3. A cada paso un buffer es exitosamente reservado por la cabeza y luego se envía un asentimiento positivo. Este asentimiento fluye en la dirección opuesta. La brecha entre la cabeza y el primer flit de datos se va incrementando hasta que la cantidad de asentimientos recibidos por el origen sea igual a k . A partir de ese momento los flits de datos fluyen por los buffers ya reservados a través de la red.

Si se toma como L la longitud de mensajes medida en flits, D el número de ejes en el camino y K la distancia de scouting, se puede derivar la expresión para cada uno de los modelos respecto a la latencia de los mensajes:

Latencia en *worm-hole*: $D+L$

Latencia en *Scouting*: $D+2K-1+L$

Latencia en *pipeline circuit switching*: $3D+L-1$

Latencia en *acknowledgment pipeline circuit switching*: $4D+L-1$

5.4.3 Fallas

Para un mejor análisis se presentarán dos modelos de fallas que pueden ser asumidas. Luego se expondrán características para cada modelo de comunicación y para cada modelo de fallas teniendo en cuenta características y problemas para el desarrollo de algoritmos de comunicación tolerantes a fallas.

Modelos de fallas (5.4.3.1)

En el *modelo de fallas estático* únicamente los ejes de comunicación pueden fallar y sólo cuando no están siendo utilizados.

De esta forma en *packet-switching* cualquier paquete pueden encontrar la falla dentro de la red. Para todos los modelos pipelines las cabezas de los mensajes son los únicos flits que pueden encontrar ejes dañados.

En un *modelo de fallas dinámicas* en cualquier momento y en cualquier estado pueden producirse fallas de ejes en la red. Este modelo es mucho más complicado de analizar en modelos pipelines. Si se produce la caída de un eje ocupado se puede llegar a situaciones de deadlock, dado que los buffers reservados no son liberados. Posteriormente, se plantearán soluciones a este problema.

Criterios tomados (5.4.3.2)

Mientras la detección de fallas no es un problema trivial, en estos casos se supone un mecanismo de detección de las mismas, con el objeto de diseñar protocolos de ruteo robustos. Si un eje falla, es detectado y marcado como fallado. Si algún nodo falla, éste y todos los ejes que acceden y parten del mismo, son marcados como fallados. Si un eje es marcado como fallado no puede ser usado para la transmisión de flits. Luego, en un modelo de fallas estático, los ejes son marcados como fallados antes que cualquier cabeza de mensaje lo utilice. En el modelo de fallas dinámico, los ejes pueden fallar en cualquier momento y pueden interrumpir el progreso a través de la red de los datos. Dado que la cabeza de un mensaje es la única que contiene el destino, en el caso de que ocurra una falla entre ésta y los flits de datos que la suceden, se producirá un bloqueo y no podrán ser re-ruteados. Los recursos ocupados por los mensajes cuando ocurre este tipo de bloqueos pueden acarrear situaciones de deadlocks. La liberación de los recursos se presenta como principal objetivo para encontrar una solución de los mismos.

Comparación en modelo de fallas estático entre los modelos de comunicación (5.4.3.3)

En *Packet-switching* deviene una situación de fácil resolución el problema de rutear datos ante una red que presenta fallas. Aquí cualquier paquete no puede ser perdido por la aparición repentina de una falla en medio de un ruteo desde el nodo origen a un destino dado, por lo tanto se puede tomar otro camino que evite el eje dañado o se lo puede descartar. La elección dependerá de la forma de ruteo y de sus características (minimal, full-adaptivo, etc.)

Dada la naturaleza optimista de *worm-hole*, la presencia de ejes caídos puede producir situaciones en la que los mensajes no logren progresar a través de la red ocasionando de esta forma situaciones de deadlock.

Sin embargo *pipeline circuit*, *routing* y *APCS* no son susceptibles a este tipo de problemas. Si la cabeza, ante la presencia de fallas, no progresa, retorna por el camino ya seleccionado para intentar acceder al destino por medio de otro.

Por último, en *scouting* si la cabeza progresa ante la presencia de fallas, retorna por el camino ya seleccionado para intentar por alguno alternativo. En este caso, a diferencia con el anterior la cantidad de pasos que podrá retroceder dependerá del número de *scouting* k , pudiendo enfrentar el mismo tipo de problemas que se presentó en *worm-hole* o evitarlo como en *pipeline circuit routing* o *APCS*.

Modelo de fallas dinámico (5.4.3.4)

Como ya se adelantó en *Packet-switching* es una situación de fácil resolución el problema de rutear datos ante una red que presenta algún modelo de fallas. Aquí cualquier paquete se descartará completamente sin dejar rastro del mismo lo que no produce inconveniente.

Una situación muy distinta se presenta en las técnicas *pipelines* al producirse la caída de un eje que afecta al camino ya formado entre la cabeza del mensaje y el último flit de datos.

Para evitar este tipo de problemas se han presentado diferentes técnicas que se detallarán a continuación.

Compressionless routing (CR) es un protocolo que intenta detectar y recuperar situaciones de deadlock utilizando temporizadores. CR permite que se produzcan situaciones de deadlock, para luego remover los mensajes de la red, siempre y cuando un tiempo suficiente haya pasado desde que un flit experimento algún progreso.

Otro mecanismo que es totalmente distribuido y no requiere temporizadores, es el denominado *Kill flits* (Ver 13.6). En orden de liberar los recursos, el controlador del eje introduce un *Kill Flit* dentro de la red (Retroceso) que sigue por todo el trayecto de buffers reservados para esta comunicación hasta llegar al nodo fuente donde es removido de la red, liberando a su paso los recursos reservados. Además el controlador del eje introduce otro *Kill Flit* dentro de la red (Avance) que continúa por todo el trayecto de eje reservados hacia el destino hasta llegar a éste o encontrar la cabeza ruteadora donde es removido de la red, liberando a su paso los recursos reservados. (Figura 5.4.3.4.a)

Se observa que es posible encontrar múltiples fallas en la red y pueden llegar a existir colisiones de *kill flits*. Por ello se establece que si un *kill flit* de retroceso choca con un *kill flit* de avance, ambos son removidos de la red.

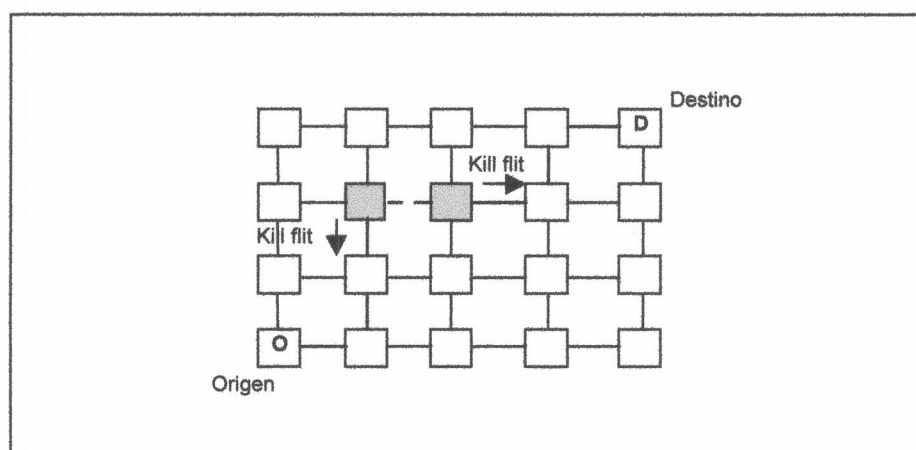


Figura 5.4.2.4.A: Generación de kill flits ante la presencia de una falla.

En orden de comparar este mecanismo con el mecanismo de temporizadores, cabe señalar que el método de kill flits no es simple de implementar dado que aumenta notablemente la complejidad del modelo de detección de fallas teniendo que generar flits para que liberen los recursos. Además, se sobrecarga la complejidad de cada nodo teniendo que agregarle una función que distinga entre dos tipos distintos de datos. Finalmente, otro inconveniente es que, en redes donde la comunicación no es bilateral, los kill flits no pueden realizar el camino de regreso hacia el nodo que originó el mensaje.

Pero, con este método se asegura que un nodo fuente de un mensaje pipeline recibe un kill flit si y solo si el mensaje es interrumpido por una o más fallas. Además que todo canal virtual reservado por el mensaje tarde o temprano va a ser liberado si éste es interrumpido. Los kill flits no producen deadlocks.

El mecanismo de temporizadores resulta de simple implementación en cada nodo, pero este exige la existencia de un tiempo global. Las diversas técnicas de sincronización tienen un cierto overhead.

Además hay que tener en cuenta el problema de tener que detectar entre congestión y deadlock. Si una zona de la red se encuentra muy congestionada afecta el tiempo de entrega de los mensajes. Luego pueden vencerse los temporizadores de los mensajes y como consecuencia, se active mecanismo de recuperación de deadlocks.

6. Redes de Bruijn Direccionales - Función de Ruteo

A continuación se presentará la función de ruteo para redes de Bruijn Direccionales.

Como se ha mencionado anteriormente, el algoritmo de ruteo consta de dos partes:

1. La función de ruteo en la que se especifica a qué nodo adyacente debe direccionar cada nodo al recibir un mensaje.
2. El grafo acíclico direccional - DAG - (capítulo 7) que conecta los diferentes buffers entre cada par de nodos adyacentes, asegurando que la función de ruteo presentada no entre en deadlock.

Se comenzará especificando nuevamente la red de Bruijn Direccional:

La topología Bruijn, notada como $B(n)$, consiste en 2^n nodos llamados N y 2^{n+1} ejes direccionales. Cada nodo corresponde a un vector $V(2,n)$. Dos nodos $\alpha=(\alpha_0, \dots, \alpha_{n-1})$ y $\beta=(\beta_0, \dots, \beta_{n-1})$ están unidos por un eje direccional notado como $L(\alpha, \beta)$ si:

$$1) \alpha_i = \beta_{i-1} \text{ para } 0 < i \leq n-1$$

y

$$2) 0 \leq \beta_{n-1} \leq 1.-$$

Al conjunto de ejes que pertenecen a $B(n)$ se notará $LG(B(n))$.

Se comienza por demostrar algunas interesantes propiedades de estas redes. Luego de unas definiciones auxiliares, se caracterizará el cálculo de la longitud del camino mínimo que conecta dos nodos cualesquiera α y β . Luego se demostrará que existe un único camino mínimo entre esos nodos, para finalmente presentar la función de ruteo.

6.1 Propiedad de la conservación de bits

Intuitivamente, la siguiente propiedad dice que en una red de Bruijn luego de avanzar una cantidad de c ejes (c menor al diámetro de la red n) se conservan los $n-c$ bits más a la derecha, desplazados c posiciones hacia la izquierda.

A modo de ejemplo, se puede ver en el siguiente gráfico dónde se reemplazó el valor de los bits que conforman la dirección del nodo por la posición de cada bit en el nodo inicial:

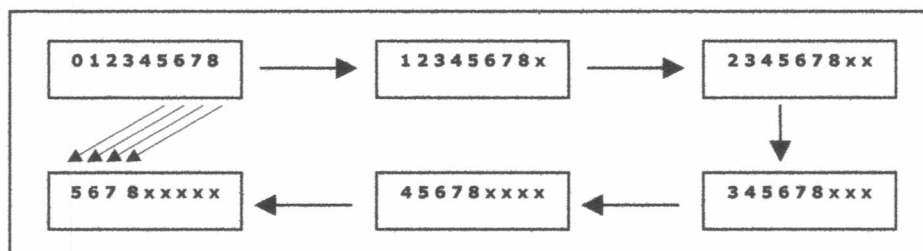


Figura 6.1.A: Desplazamiento de Bits

Propiedad

Dado un nodo $\alpha \in B(n)$ y c , $0 \leq c < n$, luego de cruzar c ejes, se llega a un nodo β tal que

$$\alpha_i = \beta_{i-c} : c \leq i < n$$

Demostración

Se demostrará por inducción en la longitud del camino c (se notará $|c|$).

Caso base: $|c| = 1$

La longitud del camino del nodo α al nodo β .

Se quiere probar que $\alpha^i = \beta^{i-1} : 1 \leq i < n$, pero esto es cierto por la definición de la construcción de las redes de Bruijn.

Caso General:

Se toma un camino cualquiera de longitud c tal que cumple con $c < n$.

Luego se tomará como verdadero lo expuesto anteriormente para todos los valores de los caminos menores a c .

HI: Se plantea la HI: $\alpha^i = \beta^{i-c'} : c' \leq i < n$ para todo camino c' en $B(n)$ tal que $|c'| = |c| - 1$, entonces, $\alpha^i = \beta^{i-(c-1)} : c-1 \leq i < n$.

Sea un camino c de longitud $|c|$ que parte de un nodo α y llega a un nodo β cualquiera, tal que $1 \leq |c| < n$.

$c(\alpha, \beta) : \alpha \rightarrow \dots \rightarrow \beta' \rightarrow \beta$

Sea c' el camino obtenido de c sin el ultimo eje ($\beta' \rightarrow \beta$), entonces:

1. $|c'| = |c| - 1$
2. Por **HI** es verdadero que $\alpha^i = \beta^{i-c'+1} : c \leq i < n$
3. y además $\beta^i = \beta^{i-1} : 1 \leq i \leq n-1$ por construcción de Bruijn

Se obtiene:

$$\alpha^i = \beta^{i-c+1} = \beta^{(i-c+1)-1} : c \leq i < n$$

$$\text{entonces } \alpha^i = \beta^{i-c} : c \leq i < n ///.$$

6.2 Cálculo de la longitud de los caminos mínimos

Se define la subcadena izquierda - $\text{SubL}(\alpha, c)$ - de un nodo α cuya longitud es c como los primeros c bits del identificador del nodo. De igual manera se define la subcadena derecha - $\text{SubR}(\alpha, c)$ - del nodo α con longitud c a los últimos c bits del identificador del nodo. Estas servirán para demostrar que la longitud de un camino mínimo que parte de un nodo α hacia un nodo β es $n-c$ tal que c es el máximo valor que cumple la condición de $\text{SubR}(\alpha, c) = \text{SubL}(\beta, c)$.

Al buscar los caminos mínimos para la topología de Bruijn, se deben localizar los segmentos de bits que se repiten en idéntico orden en los identificadores de los nodos origen y destino. Pero no cualquier segmento de bits con estas características es útil. Intuitivamente, supóngase que existe algún bit o varios de ellos que se repiten en ambos nodos origen y destino, pero que el bit más significativo del nodo origen α no pertenece a ninguno de ellos. Dicho bit debe ser eliminado, pero la única manera de hacerlo en esta topología, es insertando n bits. A modo de ejemplo, se expone el siguiente gráfico:

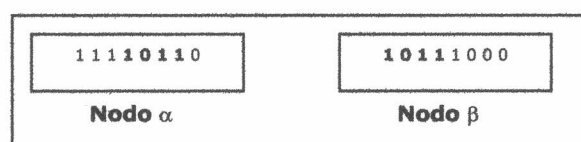


Figura 6.2.A: Ventana no volcada a la derecha en nodo origen

Un segmento idéntico en ambos nodos es la secuencia **1011**, pero el bit de la posición 7 del nodo α, con valor 0, no corresponde con el bit de la posición 4 del nodo destino β. Por lo tanto esta secuencia no puede ser tomada en cuenta sino sólo aquellas secuencias volcadas hacia la derecha en el nodo de origen α.

Ahora, supóngase que existe una secuencia de bits que se repite idénticamente en ambos nodos, pero no está volcada a izquierda:

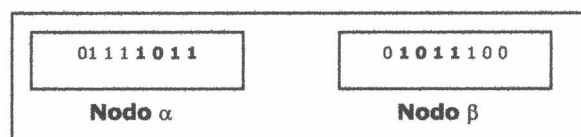


Figura 6.2.B: Ventanas no volcada a la izquierda en nodo destino

Sería interesante utilizar la secuencia **1011**, insertando los bits 100, pero se llegaría al nodo **11011100**. El bit menos significativo debe ser eliminado, y la única manera de hacerlo es insertando un nuevo bit. La topología al ser direccional, no permite desplazar hacia la derecha, sino sólo hacia la izquierda, por lo tanto no es posible utilizar esta ventana.

A esta altura se aprecia que sólo es posible utilizar aquel segmento de bits volcado a la derecha en el nodo de origen y a la izquierda en el nodo destino idéntico. Demás esta decir que el segmento óptimo será aquel que cumpla estas condiciones pero con cantidad máxima de bits.

Definición (6.2.1)

SubL : Subcadena izquierda

$\text{SubL}(\alpha, c) = \beta$, $0 \leq c < n$, con $\alpha \in V(2, m)$ siendo $\alpha = (\alpha_0, \dots, \alpha_{n-1})$, $\beta \in V(2, c)$ siendo $\beta = (\beta_0, \dots, \beta_{c-1})$ con $\alpha_i = \beta_i$ para $0 \leq i < c$.

Definición (6.2.2)

SubR : Subcadena derecha.

$\text{SubR}(\alpha, c) = \beta$, $0 \leq c < n$, con $\alpha \in V(2, m)$ siendo $\alpha = (\alpha_0, \dots, \alpha_{n-1})$, $\beta \in V(2, c)$ siendo $\beta = (\beta_0, \dots, \beta_{c-1})$ y $\alpha_{n-c+i} = \beta_i$ para $0 \leq i < c$

Teorema (6.2.3)

La longitud mínima de un camino que parte del nodo α y llega al nodo β es n-c tal que

$\text{SubR}(\alpha, c) = \text{SubL}(\beta, c)$ para el máximo c.

Demostración

Siendo $\text{SubR}(\alpha, c) = \text{SubL}(\beta, c)$ entonces $\alpha_{n-c+i} = \beta_i$ para i tq $0 \leq i < c$.

Supongamos que existe un camino de longitud $n-c'$, $n-c' < n-c$

Luego de $n-c'$ ejes se obtiene el nodo β . Por la propiedad 6.1:

$\alpha_i = \beta_{i-(n-c')} : n-c' \leq i < n \Rightarrow \alpha_i = \beta_{i-n+c'} : n-c' \leq i < n \Rightarrow$ Sea $j = i-n+c'$ entonces $i = j+n-c'$,
 $\alpha_{j+n-c'} = \beta_j : n-c' \leq j+n-c' < n \Rightarrow \alpha_{j+n-c'} = \beta_j : 0 \leq j < c' \Rightarrow$

$\text{SubR}(\alpha, c') = \text{SubL}(\beta, c')$

Siendo $n-c' < n-c \Rightarrow c' > c$ lo cual es absurdo ya que c es el máximo///.

6.3 Teorema del camino único

A continuación se demostrará que existe un único camino de longitud mínima para las redes de Bruijn Direccionales.

Intuitivamente, supóngase que existen más de un camino mínimo. Luego de un nodo cualquiera bifurcan. La única manera de hacerlo es insertando un valor j para el primer camino y un valor i para el segundo, con $i \neq j$. Como son caminos mínimos y el diámetro de la red es n , la longitud de éstos debe ser menor o igual a n . En consecuencia el bit insertado con diferente valor se conserva en el nodo destino. De hecho se conserva en la misma posición en ambos caminos, ya que tienen igual longitud. Pero entonces se ha llegado a dos nodos distintos.

Teorema

Dados α y $\beta \in B(n)$, existe un único camino de longitud mínima entre dos nodos cualesquiera

Demostración

Supongamos que existen dos caminos de longitud mínima $c'(\alpha, \beta)$ y $c''(\alpha, \beta)$ tal que $c'(\alpha, \beta) \neq c''(\alpha, \beta)$, por ende bifurcan luego de algún nodo k , pasando por el nodo γ' y γ'' respectivamente tal que $\gamma' \neq \gamma''$. Se debe notar que $\gamma'_i = \gamma''_i$ para $1 \leq i < n$.

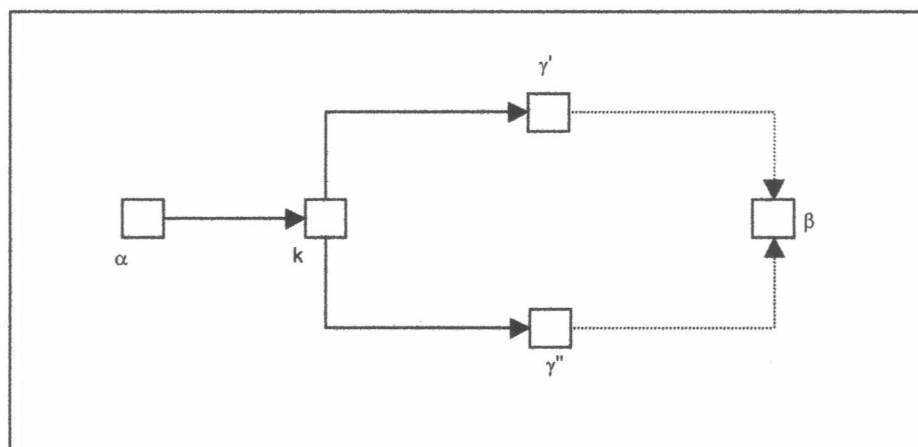


Figura 6.3.A: Bifurcación en caminos

Sin embargo, la longitud del camino mínimo entre γ' y β es igual a la longitud del camino mínimo entre γ'' y β , llamado t .

Siendo el diámetro de $B(n) = n \Rightarrow t < n$ ya que son nodos intermedios del camino mínimo entre α y β cuya longitud máxima puede ser n , del cual por lo menos ya se ha cruzado un eje.

Por la anterior propiedad :

$$\gamma^i = \beta^{i-t}; t \leq i < n$$

$$\text{y también } \gamma''^i = \beta^{i-t}; t \leq i < n$$

En particular $\gamma^{n-1} = \beta^{n-1-t}$ y $\gamma''^{n-1} = \beta^{n-1-t}$ pero entonces $\gamma' = \gamma''$. De esta forma se llega a un absurdo por suponer que existen entre dos nodos dos caminos mínimos distintos.///.

6.4 Función de Ruteo

La presente función de ruteo permite encaminar un paquete inyectado en un nodo cualquiera $\alpha \in B(m)$ con destino a un nodo cualquiera $\beta \in B(m)$ mediante un camino de longitud mínima en una red de Bruijn.

Para todo nodo α' que recibe un paquete con destino al nodo β :

1. Para i tal que $0 \leq i < n - 1$:
 - 1.1. Desplazar hacia la izquierda el identificador de nodo de α' la cantidad de i bits
 $(\alpha'0 \dots \alpha'n-1) \Rightarrow (\alpha'^i \dots \alpha'n-i \text{ xxx}) = \underline{\alpha}$
 - 1.2. Aplicar la función or exclusivo (XOR) entre $\underline{\alpha}$ y el identificador del nodo destino β
 $(\beta 0 \dots \beta n-1)$ para los bits cuya posición esta comprendida entre los valores 0 e i .
 - 1.3. Contar la cantidad de 0 consecutivos comenzando desde la posición menos significativa (desde $\alpha'0$). Llamaremos a este valor $\phi(\alpha', i)$
2. Elegir el máximo $\phi(\alpha', i)$ tal que $\phi(\alpha', i) = n-i$
3. Según i del $\phi(\alpha', i)$ elegido:
 - 3.1. Si es igual a n , entonces se llegó al nodo destino (ya que este valor sólo se puede lograr si la cantidad de bits desplazados i es igual 0).
 - 3.2. Para cualquier otro valor, desplazar el mensaje al nodo α'' , tal que $\alpha''^{n-1} = \beta^{n-i}$
 - 3.3. Si no existe ningún $\phi(\alpha', i)$ tal que $\phi(\alpha', i) = n - i$ entonces desplazar el mensaje al nodo α'' tal que $\alpha''^{n-i} = \beta 0$.

Ejemplo:

Como ejemplo se mostrará el camino que construirá el algoritmo para rutear un mensaje insertado en el nodo α cuya dirección es 001010 con destino al nodo β rotulado 101011.

En el nodo 001010, se realizan las siguientes operaciones:

Resultado del paso 1.1.	Resultado del paso 1.2.	Resultado del paso 1.3.
001010	100001	$\phi(\alpha,0) = 0$
01010X	11111X	$\phi(\alpha,1) = 0$
1010XX	0000XX	$\phi(\alpha,2) = 4$
010XXX	111XXX	$\phi(\alpha,3) = 0$
10XXXX	00XXXX	$\phi(\alpha,4) = 2$
0XXXXX	1XXXXX	$\phi(\alpha,5) = 0$

En el paso 2 son candidatos a ser elegidos $\phi(\alpha,2)$ y $\phi(\alpha,4)$, porque todos cumplen con que $\phi(\alpha,i) = n-i$. Pero se elegirá $\phi(\alpha,2)$ ya que este tiene el valor 4 que es igual a $n-i$ (con $n = 6$ e $i = 2$), por lo tanto en el paso 3 se ruteará el mensaje al nodo 010101 ya que $\beta n-i = \beta 4-2 = 1$.

En el nodo 010101, se realizan las siguientes operaciones:

Resultado del paso 1.1.	Resultado del paso 1.2.	Resultado del paso 1.3.
010101	11110	$\phi(\alpha,0) = 0$
10101X	00000X	$\phi(\alpha,1) = 5$
0101XX	1111XX	$\phi(\alpha,2) = 0$
101XXX	000XXX	$\phi(\alpha,3) = 3$
01XXXX	11XXXX	$\phi(\alpha,4) = 0$
1XXXXX	0XXXXX	$\phi(\alpha,5) = 1$

En el paso 3 los candidatos a ser elegidos son $\phi(\alpha,1)$, $\phi(\alpha,3)$ y $\phi(\alpha,5)$. Se elegirá $\phi(\alpha,1)$ y se ruteará al nodo 101011.

En el nodo 101011, se realizan las siguientes operaciones:

Resultado del paso 1.1.	Resultado del paso 1.2.	Resultado del paso 1.3.
101011	000000	$\phi(\alpha,0) = 6$
01011X	11110X	$\phi(\alpha,1) = 0$
1011XX	0001XX	$\phi(\alpha,2) = 3$
011XXX	110XXX	$\phi(\alpha,3) = 0$
11XXXX	01XXXX	$\phi(\alpha,4) = 0$
1XXXXX	0XXXXX	$\phi(\alpha,5) = 1$

En el paso 2 los candidatos a ser elegidos son $\phi(\alpha,0)$, $\phi(\alpha,3)$ y $\phi(\alpha,5)$. Se elegirá $\phi(\alpha,0)$ y en el paso 3 se detectará que se llegó al nodo destino.

7. Grafo dirigido acíclico (DAG) para Bruijn Direccional.

Como se mencionó anteriormente una de las formas para garantizar la no existencia de deadlock en el ruteo de mensajes se logra mediante la definición de un orden estricto monotónico de los buffers de manera tal que no se produzcan ciclos y la función de ruteo pueda cumplir con el mismo.

El DAG presentado en esta sección será *escalable*, es decir que cada nodo tendrá un número constante de buffers y proveerá full-adaptabilidad para caminos mínimos.

7.1. Idea intuitiva

En el presente capítulo se establece un orden total entre todos los identificadores de nodo de una topología de Bruijn Direccional. Se plantean nodos con cuatro buffers cada uno, que forman en la topología cuatro niveles de buffers. Comenzando desde el nivel inferior, todo mensaje que es ruteado utilizando caminos mínimos debe cumplir con el orden para ese nivel de buffer. En caso contrario, se debe continuar con el nivel de buffer siguiente. Por lo tanto, el orden planteado debe permitir que los cuatro niveles de buffers alcancen para rutear cualquier mensaje que utilice caminos mínimos.

Como ya se ha indicado, cada nodo α en Bruijn Direccional está identificado por un vector $(\alpha_0, \dots, \alpha_n)$. Se define el número natural $W \equiv n/2$ y *vector-W* al vector de tamaño W (que está incluido en el identificador del nodo) cuyo valor natural es el máximo. Se observa que pueden existir más de una ocurrencia del vector-W en un identificador de un nodo. Cada uno tiene una posición inicial, tomándose como cero el bit de más a la izquierda.

En un camino mínimo en Bruijn si en algún paso j el valor natural del vector-W aumenta, el nuevo vector-W se encontrará ubicado en el extremo derecho del identificador del segundo nodo. De otra forma, el valor del vector-W existía en el identificador del nodo anterior al paso j . A medida que el camino mínimo es recorrido, el vector-W se desplaza paso por paso hacia la izquierda hasta que ocurre una de las siguientes condiciones:

- Se llega a origen
- Aparece un nuevo vector-W de mayor valor (ubicado en el extremo derecho).
- Surge un nuevo vector-W de menor valor natural, desapareciendo el anterior por el extremo izquierdo.

Si en un camino mínimo aumenta el valor natural del vector-W, éste se ubica en el extremo derecho. Si surge un nuevo vector-W de menor valor, entonces el anterior desapareció por el extremo izquierdo. Por lo tanto, considerando el tamaño del vector-W, se producen $n/2$ movimientos del mensaje, es decir que se corrigen $n/2$ bits de la dirección origen.

Se plantea hacer uso del primer nivel de buffers hasta que aparezca un vector-W de mayor valor. Al ocurrir esta condición se comienza a utilizar el segundo nivel de buffer, hasta que surja un vector-W menor. En conjunto se habrán corregido en estos dos primeros niveles $n/2$ bits. Realizando el mismo procedimiento con los otros dos niveles, se estará corrigiendo en total n bits que es el diámetro de la red.

Por lo tanto, el **primer componente** del orden estará dado por el valor natural decreciente del vector-W para los niveles primero y tercero (de esta forma, se abandona el nivel cuando aparece un vector-W de mayor valor en el extremo derecho) y decreciente para los niveles segundo y cuarto.

El problema principal es ordenar los identificadores que tienen igual valor natural del vector-W. Para resolverlo se efectúa un análisis más profundo del comportamiento de los vector-Ws en caminos mínimos en Bruijn Direccional.

Se define al período mínimo de un vector como el menor número de bits (mayor que cero) en el que el vector se repite .

En el punto 7.4 se demostrará que, si existen más de una ocurrencia del vector-W en un identificador, entonces éstos se encuentran distanciados por el período mínimo.

Esto permite concluir que:

- En un camino mínimo en Bruijn, en cada paso las ocurrencias del vector-W están distanciadas por el período mínimo de éste. La aparición y desaparición de otras ocurrencias se producen en forma escalonada.
- Si la posición del vector-W ubicado más a la izquierda es menor al período mínimo, entonces puede suceder que en algún paso reciente del camino mínimo haya desaparecido una ocurrencia del vector-W.
- Si el menor período del vector-W es igual al menor período del vector formado desde la posición de la ocurrencia del vector-W ubicado más a la derecha hasta el extremo derecho del identificador, entonces aún puede ingresar una nueva ocurrencia del vector-W.

Finalmente, se establece que los identificadores de nodo para idéntico vector-W, se ordenan por el promedio de la cantidad de ocurrencias. El comienzo será:

- *Cero* en caso de que exista la posibilidad que una ocurrencia del vector-W haya desaparecido recientemente y que no pueda aparecer otra.
- *La posición inicial de la ocurrencia del vector-W ubicada más a la izquierda* en otro caso.

Este valor es el **segundo componente** del orden y permite ordenar todos los identificadores de igual valor de vector-W, haciendo posible que, en un camino mínimo una vez que un vector-W aparezca por el extremo derecho y hasta que surja otro de valor natural menor, no se abandona el nivel de buffer en el que se encuentra.

7.2. Resultados Conocidos

Los siguientes teoremas proveen condiciones necesarias y suficientes para la creación de algoritmos de ruteos oblivious o adaptivos, packet-switching o modelo pipelines, deadlock-free.

Teorema 1: Dado cualquier algoritmo de ruteo oblivio o adaptivo, packet-switching o modelo pipeline, si existe un orden de los buffers tal que todos los mensajes en cada paso son movidos hacia el buffer del orden más alto o entregado a destino, entonces el algoritmo es deadlock-free. (Ver 13.9)

Teorema 2: Dado cualquier algoritmo de ruteo packet-switching o modelo pipeline, que sea oblivio o adaptivo y deadlock-free, existe un orden total de los buffers tal que todos los mensajes en cada paso son movidos hacia el buffer del orden más alto o entregado a destino. (Ver 13.8)

7.3. Definiciones

Dado $B(n)$ se consideran las siguientes definiciones:

1): **Relación de orden.** Se definirá la siguiente relación de orden entre dos vectores α, β pertenecientes a $V(2,n)$ y se utilizará de aquí en adelante para comparar dos elementos cualesquiera del conjunto dado.

Dado cualquier $\alpha \in V(2,n), \beta \in V(2,n)$, se dirá que:

i) $\alpha < \beta$ si existe un $j, 0 \leq j \leq n-1$, tal que $\alpha_j < \beta_j$ y $\alpha_t = \beta_t$ para todo $t, 0 \leq t < j$.

ii) $\alpha = \beta$ si $\alpha_t = \beta_t$ para todo $t, 0 \leq t < n-1$.

2): **$W()$:** $W(\alpha, k, i)$ retorna un vector de longitud k desde la posición i del vector α .

Dado k, i donde $1 \leq k \leq n, 0 \leq i \leq n-k$ y $\alpha \in V(2,m)$, luego $W(\alpha, k, i) = \beta$ donde $\beta \in V(2,k)$ y $\beta_j = \alpha_{j+i}$ para todo $j, 0 \leq j \leq k-1$.

3): **Maxval:** Retorna el mayor vector (en el orden de los naturales) de longitud k que se encuentre en α .

Dado $k, 1 \leq k \leq n, \alpha \in V(2,m)$ luego $\text{maxval}(\alpha, k) = W(\alpha, k, i)$ donde $0 \leq i \leq n-k$ y para todo $j \neq i, 0 \leq j \leq n-k, W(\alpha, k, j) \leq W(\alpha, k, i)$.

4): **W y V :**

$W \in \mathbb{N}, w = \lceil n/2 \rceil$.

$V \in \mathbb{N}, v = \lfloor n/2 \rfloor$.

Notar que $v+w = n$ y que $v \leq w \leq v+1$

5) **Val :** Retorna el máximo vector de longitud W que se puede encontrar en el vector α . A partir de este momento se denominará **vector- W** .

$\text{val}(\alpha) = \text{maxval}(\alpha, W)$

6) **Pos :** Retorna la posición en el vector α donde comienza algún vector- W de α .

$\text{pos}(\alpha) = s$ con $s \in \mathbb{N}$ tal que $\text{val}(\alpha) = W(\alpha, w, s)$.

7): **PosR:** Retorna la posición en el vector α del vector- W más a la derecha.

$\text{posr}(\alpha) = s$ con $s \in \mathbb{N}$ tal que $s = \text{pos}(\alpha)$ y no existe j tal que $j = \text{pos}(\alpha)$ y $j > s$.

8): **PosL:** Retorna la posición en el vector α del vector- W más a la izquierda.

$\text{posl}(\alpha) = s$ con $s \in \mathbb{N}$ tal que $s = \text{pos}(\alpha)$ y no existe j tal que $j = \text{pos}(\alpha)$ y $j < s$.

9): **Resto:** Retorna el vector que se encuentra en un vector α que va desde PosR hasta el final hacia la derecha

$\text{Resto}(\alpha) = W(\alpha, m - \text{PosR}(\alpha), \text{PosR}(\alpha))$ con $\alpha \in V(2,m)$.

10) **P**: Retorna el período de un vector. Entiéndase como período al número de posiciones hacia un lado donde el vector se vuelve a repetir

$$P(\alpha) = p \text{ con } p \in \mathbb{N}, p > 0, \text{ tal que } \alpha_j = \alpha_{j+p} \text{ para todo } 0 \leq j \leq n-p-1$$

11) **Per**: Retorna el menor período de un vector dado.

$$\text{Per}(\alpha) = p \text{ tal que } p = P(\alpha) \text{ y no existe } j \text{ tal que } j = P(\alpha) \text{ y } j < p.$$

12) **Ord**: Retorna el orden de aparición de izquierda a derecha de un vector-W dado.

$\text{Ord}(\alpha, \beta) = j$ tal que $j \in \mathbb{N}$, $\beta = \text{Val}(\alpha) = W(\alpha, w, s)$ y si $j > 1$ existen exactamente $j-1$ β' distintos tal que $\beta' = \text{Val}(\alpha) = W(\alpha, w, r)$ con $r < s$.

13) **CantVW**: Retorna la cantidad de Vector- W s que contiene el vector α .

14) **PerCte**: Retorna verdadero si el valor del menor período del vector- W de α es igual al valor del menor período del resto de α . En otro caso retorna falso.

$\text{PerCte}(\alpha)$	Verdadero	si $\text{Per}(\text{val}(\alpha)) = \text{Per}(\text{Resto}(\alpha))$
	Falso	en otro caso.

7.4. Características de los vector- W s y su relación con el período

A continuación se demuestra una proposición que va a ser utilizada en el resto del presente trabajo para poder construir el grafo de dependencias acíclico.

En la siguiente proposición se asegura que si existen más de una ocurrencia del vector- W , éstas van a estar distanciadas por el período mínimo del vector- W .

Proposición:

Dado un vector $\alpha \in V(2, n)$, para toda ocurrencia del vector- W β tal que $\beta = W(\alpha, j, W)$. entonces

$$1) j = \text{PosR}(\alpha)$$

o

2) Existe un vector- W β' tal que $\beta' = W(\alpha, j + \text{Per}(\beta), W)$. y no existe un vector- W β'' tal que $\beta'' = W(\alpha, t, W)$ con $j < t < j + \text{Per}(\beta)$.

Demostración:

Dado un vector $\alpha \in V(2, n)$, se toma una ocurrencia del vector- W β cualquiera tal que $\beta = W(\alpha, j, W)$. Se supone que existe un vector- W β' tal que $\beta' = W(\alpha, t, W)$, $j < t$, $j + \text{Per}(\beta) \neq t$ y no existe un vector- W β'' tal que $\beta'' = W(\alpha, s, W)$ con $j < s < t$. Por lo tanto, se supone que existe un vector- W es el primero desplazándose hacia la derecha y se desliza en un número distinto al período.

Luego la demostración se divide en dos casos

$$a) j + \text{Per}(\beta) < t.$$

En este caso también se divide el estudio en dos partes:

a.1) Si $\text{Módulo}(t-j)_{\text{Per}(\beta)} = 0$

Se sabe que dos vector-Ws están montados entre sí (o sea que el inicio de uno está entre el inicio y el fin de otro) o en posiciones exactamente contiguas (o sea que el inicio de uno es el final más uno del otro)

Este caso es absurdo porque entre el vector-W β y el vector-W β' existiría uno o más vector-Ws β'' .

Luego como ejemplo se toma $\beta'' = [\alpha t + \text{per}(\beta), \dots, \alpha t + \text{Per}(\beta) + W]$.

a.2) Si $\text{Módulo}(t-j)_{\text{Per}(\beta)} < 0$

Esto es absurdo dado que β y β' no serían la mayor ventana de tamaño W en α .

En este punto, se puede asegurar que

1) Si $\text{Módulo}(t-j)_{\text{Per}(\beta)} > \text{Per}/2$ entonces $\beta' > \beta$

2) Si $\text{Módulo}(t-j)_{\text{Per}(\beta)} < \text{Per}/2$ entonces $\beta'' = [\alpha t - \text{Módulo}(t-j)_{\text{Per}(\beta)}, \dots, \alpha t - \text{Módulo}(t-j)_{\text{Per}(\beta)} + W] > \beta$

b) $j + \text{Per}(\beta) > t$

En este caso el vector-W β' va a comenzar entre el inicio del vector-W β y el inicio del vector-W β más el período de β .

O sea $j < t < j + \text{Per}(\beta)$.

Luego $\alpha j + s = \alpha t + s$ para $0 \leq s \leq W - t$ dado que en j y t comienzan 2 vector-Ws iguales. Por lo tanto existiría un período menor que el menor período dado $\text{Per}(\beta)$.

Finalmente se llega a un absurdo por suponer que el primer vector-W β' más a la derecha que otro vector-W β comienza antes de la posición de β mas el período del mismo.

Luego se concluye que $j + \text{Per}(\beta) = t$

Por lo tanto, dado un vector $\alpha \in V(2, n)$, tomando un vector-W β cualesquiera tal que $\beta = W(\alpha, j, W)$ si existe un vector-W β' tal que $\beta' = W(\alpha, t, W)$, $j < t$, y no existe un vector-W β'' tal que $\beta'' = W(\alpha, s, W)$ con $j < s < t$, entonces $j + \text{Per}(\beta) = t$. En el caso de no existir tal vector-W β' entonces β es la vector-W más hacia la derecha por lo tanto $j = \text{PosR}(\alpha)$.///

Finalmente se concluye que $j + \text{Per}(\beta) = t$. Por lo tanto, si existe un vector-W más hacia la derecha éste tiene que estar desplazado exactamente en el $\text{Per}(\beta)$. Si no existe un vector-W más hacia la derecha entonces $j = \text{PosR}(\alpha)$

Con lo demostrado se puede decir que si el período se mantiene desde el primer vector-W más a la izquierda hasta el final a la derecha, es factible que ingrese un nuevo vector-W de igual valor.

7.5. Funciones de comparación

En el punto 7.4 se presentó una interesante propiedad de las ocurrencias del vector-W. En ésta se demuestra que están distanciadas por el menor período del mismo.

En el presente punto se presentará un conjunto de funciones. Al tomar un vector cualquiera obtienen como resultado el promedio de las ocurrencias del vector-W. Estos valores servirán para ordenar a los vectores de igual valor natural de sus vector-Ws con el fin de formar el DAG en Bruijn.

La función AvgW2 tomará el promedio de los valores dados por la función PosAvg2.

La idea intuitiva de la función AvgW2 es tomar el promedio de las ocurrencias del vector-W a partir de la posición de inicio de la ubicada más a la izquierda.

Para facilitar la definición de la función PosAvg1, se definirá la función PosIni. Esta retornará uno de dos valores posibles.

- a) Cero si el valor de la posición inicial de la ocurrencia del vector-W ubicada más a la izquierda es menor que el período (siendo posible que en un nodo anterior del camino mínimo haya desaparecido una ocurrencia del vector-W) y el menor período del vector-W es distinto al menor período del vector formado desde la posición de la ocurrencia del vector-W ubicado más a la derecha hasta el extremo derecho del identificador del nodo (Si ya no pueden ingresar vector-W de igual valor a los que se tienen en ese momento)
- b) El valor de la posición inicial de la ocurrencia del vector-W ubicado más a la izquierda, en otro caso.

La función AvgW1 tomará el promedio de los valores dados por la función PosAvg1.

La idea intuitiva de la función AvgW1 es tomar el promedio de las ocurrencias del vector-W a partir del valor dado por PosIni.

7.5.1 Definiciones

Dados $\alpha \in V(2,n)$ y $\beta = \text{val}(\alpha)$ se define:

- $\text{PosIni}(\alpha) = \begin{array}{ll} \text{i) } 0 & \text{Si } \text{PosL}(\alpha) < \text{Per}(\beta) \text{ y NOT } \text{PerCte}(\alpha). \\ \text{ii) } \text{PosL}(\alpha) & \text{Sino.} \end{array}$
- $\text{PosAvg1}(\alpha, \beta) = (\text{Pos}(\beta) - ((\text{Ord}(\beta) - 1) * (\text{Per}(\beta) - 1))) - \text{PosIni}(\alpha).$
- $\text{PosAvg2}(\alpha, \beta) = \text{Pos}(\beta) - ((\text{Ord}(\beta) - 1) * (\text{Per}(\beta) - 1)).$

Luego se define el conjunto de naturales **S** y **T** como:

- i) Para todo $\beta = \text{Val}(\alpha)$ entonces $\text{PosAvg1}(\alpha, \beta) \in S.$
- ii) Para todo $\beta = \text{Val}(\alpha)$ entonces $\text{PosAvg2}(\alpha, \beta) \in T.$

En el siguiente diagrama se expone una idea visual de lo presentado en este punto. Se trata de un caso con gran cantidad de ocurrencias de vector-Ws en un camino mínimo (El vector-W es 1010 y el período es 2). Con una X se presentan marcados los valores de las posiciones de las ocurrencias del vector-W para las definiciones de PosAvg1 y PosAvg2 respectivamente.

Se puede observar que cuando dos nodos contiguos de la función PosAvg1 son iguales, los valores de la función PosAvg2 no lo son. En este caso, estos segundos valores disminuyen.

Vector-W	PosAvg1	PosAvg2
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0 0 0 0 1 0 1 0	- - - - X - - -	- - - - X - - -
0 0 0 1 0 1 0 1	- - - X - - - -	- - - X - - - -
0 0 1 0 1 0 1 0	- - X X - - - -	- - X X - - - -
0 1 0 1 0 1 0 1	- X X - - - - -	- X X - - - - -
1 0 1 0 1 0 1 0	X X X - - - - -	X X X - - - - -
0 1 0 1 0 1 0 0	X X - - - - - -	- X X - - - - -
1 0 1 0 1 0 0 0	X X - - - - - -	X X - - - - - -
0 1 0 1 0 0 0 0	X - - - - - - -	- X - - - - - -
1 0 1 0 0 0 0 0	X - - - - - - -	X - - - - - - -

7.5.2 Definiciones: AvgW1, AvgW2.

AvgW1: Avg(S)

AvgW2: Avg(T).

(Avg es la notación de la función promedio para un conjunto de naturales).

7.5.3 Orden total de los nodos

Para definir un orden total entre los identificadores de Bruijn se utiliza una cuatro-upla. Cada uno de los componentes está definido por una función tomando como parámetro el identificador del nodo.

Así mismo, se definen cuatro órdenes:

En el *primer orden* se suceden en forma creciente los identificadores cuyo valor literal del vector-W decrezca (Primer componente de la cuatro-upla). Si el vector-W de los identificadores es igual, se utilizan los valores de las funciones presentadas en el punto anterior. Se ordena en forma descendente según el valor de la función AvgW1 (Segundo componente de la cuatro-upla) y si ésta continúa siendo igual, se ordena en forma descendente según el valor de la función AvgW2 (Tercer componente de la cuatro-upla). Si los tres primeros componentes de la cuatro-upla son iguales, se ordena en forma ascendente según el valor literal del identificador del nodo. (Cuarto componente de la cuatro-upla)

En el *segundo orden* se suceden en forma creciente los nodos cuyo valor literal del vector-W crezca (Primer componente de la cuatro-upla). En los casos en que los nodos tengan un mismo valor se aplicará la misma técnica del orden anterior. (Segundo, tercero y cuarto componente de la cuatro-upla)

Los órdenes *tercero* y *cuarto* son idénticos a los órdenes primero y segundo respectivamente.

7.5.4 Definiciones de los órdenes: Orden1, Orden2, Orden3, Orden4.

Dados $\alpha \in V(2,n)$ se definirá la tupla:

Tupla: $T(\alpha) = (t_{\alpha 1}, t_{\alpha 2}, t_{\alpha 3}, t_{\alpha 4})$ donde :

$$t_{\alpha 1} = \text{val}(\alpha)$$

$$t_{\alpha 2} = \text{AvgW1}(\alpha)$$

$$t_{\alpha 3} = \text{AvgW2}(\alpha)$$

$$t_{\alpha 4} = \text{valor literal de } \alpha.$$

Dados $\alpha \in V(2,n)$ y $\beta \in V(2,n)$, $\alpha \neq \beta$, $T(\alpha), T(\beta)$ se definen dos órdenes de cuatro-uplas:

Orden1: $\alpha <_1 \beta$ sii

- i) $t_{\alpha 1} > t_{\beta 1}$ o
- ii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} > t_{\beta 2}$ o
- iii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} = t_{\beta 2}$ y $t_{\alpha 3} > t_{\beta 3}$ o
- iiii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} = t_{\beta 2}$ y $t_{\alpha 3} = t_{\beta 3}$ y $t_{\alpha 4} < t_{\beta 4}$.

Orden2: $\alpha <_2 \beta$ sii

- i) $t_{\alpha 1} < t_{\beta 1}$ o
- ii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} > t_{\beta 2}$ o
- iii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} = t_{\beta 2}$ y $t_{\alpha 3} > t_{\beta 3}$ o
- iiii) $t_{\alpha 1} = t_{\beta 1}$ y $t_{\alpha 2} = t_{\beta 2}$ y $t_{\alpha 3} = t_{\beta 3}$ y $t_{\alpha 4} < t_{\beta 4}$.

Orden3: $\alpha <_3 \beta$ sii $\alpha <_1 \beta$

Orden4: $\alpha <_4 \beta$ sii $\alpha <_2 \beta$

7.6. Comparaciones según el comportamiento

En esta sección se demuestra un conjunto de propiedades que servirán para comparar las funciones dadas anteriormente, de acuerdo a la entrada, salida o movimiento de las ocurrencias del vector-W en caminos mínimos en Bruijn.

Se toma K cualquiera pertenecientes a N.

7.6.1 Propiedad de los valores contiguos constantes.

Esta propiedad servirá para comparar el promedio de un conjunto de valores contiguos pertenecientes a los naturales con el promedio de los mismos valores decrecidos en uno.

Como ejemplo se expone el promedio del conjunto $\{3,4,5,6\} = 4.5$ es mayor que el promedio del conjunto $\{2,3,4,5\} = 3.5$.

Gráficamente:

Valores	0	1	2	3	4	5	6
Conjunto 1				X	X	X	X
Conjunto 2			X	X	X	X	

$$\frac{\sum_{i=0}^s (i+(K+1))}{s+1} > \frac{\sum_{i=0}^s (i+K)}{s+1}$$

Demostración:

Se puede observar que cada componente de la primer sumatoria es, elemento a elemento, mayor que los de la segunda.///

7.6.2 Propiedad de los valores contiguos crecientes.

Gracias a esta propiedad es posible comparar el promedio de un conjunto de valores contiguos pertenecientes a los naturales mayores que cero, con el promedio de los mismos valores más un valor que resulta de restar uno al menor valor del primer conjunto. Como ejemplo se expone el promedio del conjunto $\{3,4,5,6\} = 4.5$ es mayor que el promedio del conjunto $\{2,3,4,5,6\} = 4$.

Gráficamente:

Valores	0	1	2	3	4	5	6
Conjunto 1				X	X	X	X
Conjunto 2			X	X	X	X	X

Dado $n > 0$, $0 \leq k < n$, se define:

$$\frac{\sum_{i=k+1}^n (i)}{n-k} > \frac{\sum_{i=k}^n (i)}{n-k+1}$$

(α)

Demostración:

Dividiendo el primer término se tiene que:

$$\frac{\sum_{i=k}^n (i)}{n-k+1} = \frac{\sum_{i=k+1}^n (i)}{n-k+1} + \frac{k}{n-k+1}$$

(β) (δ)

Luego en las sumas de (α) y de (β) hay $n - k$ términos.

Un término cualquiera de (α) será, para $K < i \leq n$:

$$\frac{i}{n-k}$$

Luego se transforma la suma de (β) y (δ) en otra (π) dividiendo (δ) en $n-k$ partes y sumando estos subtérminos a cada término de la suma en (β) . Por lo tanto, un término cualquiera de (π) será, para $K < i \leq n$:

$$\frac{k}{(n-k+1)(n-k)} + \frac{i}{(n-k+1)} = \frac{k+in+ik}{(n-k+1)(n-k)}$$

Entonces si:

$$\frac{k+in+ik}{(n-k+1)(n-k)} < \frac{i}{(n-k)} ; \quad k+i(n-k) < i(n-k)+i ; \quad k < i$$

se tiene que cada término de la suma de (α) es mayor que cada término de la suma de (π) . Como todos los términos son mayores que cero, (α) es menor que (π) . Como (π) es igual a (β) más (δ) , la desigualdad queda demostrada.///

7.6.3 Propiedad de los valores contiguos decrecientes.

Esta propiedad es útil a la hora de comparar el promedio de un conjunto de valores contiguos pertenecientes a los naturales con inicio en cero con el promedio de los mismos valores menos el número de mayor valor del primer conjunto. Como ejemplo se expone el promedio del conjunto $\{0,1,2,3\} = 2$ es mayor que el promedio del conjunto $\{0,1,2\} = 1$.

Gráficamente:

Valores	0	1	2	3	4	5	6
Conjunto 1	X	X	X	X			
Conjunto 2	X	X	X				

Dado $s \geq 0$, se define:

$$\frac{\sum_{i=0}^{s+1} (i)}{s+2} > \frac{\sum_{i=0}^s (i)}{s+1}$$

(α)

Demostración:

Se tiene que:

$$\frac{\sum_{i=0}^{s+1} (i)}{s+2} = \frac{\sum_{i=0}^s (i)}{s+2} + \frac{s+1}{s+2}$$

(β) (δ)

Luego en las sumas de (α) y de (β) se tiene $s+1$ términos.

Se tiene que un término cualquiera de (α) será, para $0 \leq i < s+1$:

$$\frac{i}{s+1}$$

Luego se transformará la suma de (β) y (δ) en otra (π) dividiendo (δ) en $s+1$ partes y sumando estos subterminos a cada término de la suma en (β) . Por lo tanto, un término cualquiera de (π) será, para $0 \leq i < s+1$:

$$\frac{i}{s+2} + \frac{s+1}{(s+2)(s+1)} = \frac{is+i+s+1}{(s+2)(s+1)}$$

Entonces si:

$$\frac{i}{(s+1)} < \frac{is+i+s+1}{(s+2)(s+1)} ; i < \frac{is+i+s+1}{(s+2)} ; is+i2 < is+i+s+1 ; i < s+1.$$

se tiene que cada término de la suma de (α) es menor que cada término de la suma de (π) . Como todos los términos son mayores que cero, (α) es menor que (π) . Como (π) es igual a (β) más (δ) , la desigualdad queda demostrada.///

7.7. Análisis del comportamiento de las ocurrencias del vector-W

En esta sección se expondrán algunas características en caminos mínimos para poder deducir la cantidad de buffers necesarios en ruteos mínimos en Bruijn

Dado un *camino mínimo de longitud t* que parte de un nodo α llegando a un nodo β en $B(n)$, notado por $\alpha=\alpha_0, \alpha_1, \dots, \alpha_{t-1}=\beta$ se dice que:

7.7.1 Propiedad de la posición del vector-W creciente.

Si el valor del vector-W crece en algún paso de un camino mínimo entonces el nuevo vector-W va a estar ubicado en el extremo más a la derecha del identificador del nodo.

Propiedad:

Si $\text{val}(\alpha_i) < \text{val}(\alpha_{i+1})$ para algún i $0 \leq i < t-1$ entonces $\text{posL}(\alpha_{i+1}) = n-W$.

Demostración:

Se supone que si $\text{val}(\alpha_i) < \text{val}(\alpha_{i+1})$ entonces $\text{posL}(\alpha_{i+1}) \neq n-W$.

Primeramente se descarta la suposición de $\text{posL}(\alpha_{i+1}) > n-W$ dado que la longitud del vector-W es de W , por lo tanto, no puede superar esa posición.

En el otro caso, $\text{posL}(\alpha_{i+1}) < n-W$, y por la forma de construcción de Bruijn, el vector-W de α_{i+1} que se encuentra en esa posición también existiría en el vector α_i en la posición $\text{posL}(\alpha_{i+1})+1$. Luego $\text{val}(\alpha_i) = \text{val}(\alpha_{i+1})$.

Por lo tanto se llega a un absurdo con $\text{val}(\alpha_i) < \text{val}(\alpha_{i+1})$ por suponer que $\text{posL}(\alpha_{i+1}) \neq n-W$.

Entonces $\text{posL}(\alpha i+1) = n-W$. ///

7.7.2 Propiedad de la cantidad de ocurrencias del vector-W en un camino mínimo.

En esta propiedad se demuestra que si el valor natural del vector-W se mantiene igual al realizar algún paso de un camino mínimo, entonces la cantidad de ocurrencias del vector-W va a diferir en a lo sumo uno

Propiedad:

Si $\text{val}(\alpha i) = \text{val}(\alpha i+1)$ para algún i $0 \leq i < t-1$ entonces la $\text{Abs}(\text{CantVW}(\alpha i) - \text{CantVW}(\alpha i+1)) \leq 1$

(Abs es la función que retorna el valor absoluto de un entero)

Demostración:

Esta propiedad se comprueba por la forma de construcción de Bruijn. En ésta se observa que dos nodos α y β cualquiera están unidos por un eje si:

$$\alpha i = \beta i-1 \quad \text{para } 0 < i \leq m-1 \quad \text{y} \\ 0 \leq \beta m-1 < 1.$$

Todos los bits de α se mantienen en β en una posición menor. Luego, todas las ocurrencias del vector-W en α se mantienen en β excepto si se encuentra uno que comienza en la posición menos significativa. En este caso existiría una diferencia de uno a favor de α .

En β puede ser que con la aparición de un nuevo bit en la posición más significativa, se genere otra ocurrencia del vector-W. En este caso existiría una diferencia de uno a favor de β .

Por lo tanto la diferencia de vector-Ws entre α y β es a lo sumo uno.///

7.7.3 Propiedad del comportamiento de vector-W en caminos mínimos.

En esta propiedad se analiza el comportamiento, en caminos mínimos, de los valores de las funciones de comparación de los identificadores de los nodos.

Para mayor claridad se divide el estudio en partes y subpartes según corresponda y solamente utilizando el Orden1 y Orden2 dado que son similares al Orden3 y Orden4 respectivamente.

Parte 1 (7.7.3.1)

A continuación se analizan los pares de nodos que están unidos por un eje en Bruijn. En el primero de ellos, la posición de inicio de la ocurrencia del vector-W ubicado más a la izquierda es mayor o igual que el período del mismo.

Bajo esta suposición en el segundo nodo la cantidad de ocurrencias del vector-W puede ser igual o mayor en uno, si apareciere una nueva.

Primer caso (cantidad de ocurrencias de vector-W iguales):

7.7.3.1.1 Si $\text{PosL}(\alpha i) \geq \text{Per}(\text{val}(\alpha i))$ y $\text{CantVW}(\alpha i) = \text{CantVW}(\alpha i+1)$ para algún i $0 \leq i < t-1$ entonces $\alpha i <_1 \alpha i+1$ y $\alpha i <_2 \alpha i+1$.

Demostración:

Tomando la hipótesis de que $\text{CantVW}(\alpha i) = \text{CantVW}(\alpha i+1)$, $\text{val}(\alpha i) = \text{val}(\alpha i+1)$, y teniendo en cuenta que $\text{PosIni}(\alpha i) = \text{PosL}(\alpha i)$ y aplicando el punto 7.6.1 se demuestra que $\text{AvgW1}(\alpha i) > \text{AvgW1}(\alpha i+1)$ por lo tanto $\alpha i <_1 \alpha i+1$ y $\alpha i <_2 \alpha i+1$.///

Segundo caso (Cantidad de ocurrencias del vector-W difiere en uno):

7.7.3.1.2) Si $\text{PosL}(\alpha_i) \geq \text{Per}(\text{val}(\alpha_i))$ y $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_{i+1}) + 1$ para algún i $0 \leq i < t-1$ entonces $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.

Demostración:

Tomando la hipótesis de que $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_{i+1})$, $\text{val}(\alpha_i) = \text{val}(\alpha_{i+1})$, y teniendo en cuenta que $\text{PosIni}(\alpha_i) = \text{PosL}(\alpha_i)$ y aplicando el punto 7.6.2 se demuestra que $\text{AvgW1}(\alpha_i) > \text{AvgW1}(\alpha_{i+1})$ por lo tanto $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.///

Parte 2 (7.7.3.2)

Se analizan los pares de nodos que están unidos por un eje en Bruijn donde, en el primero la posición de inicio de la ocurrencia del vector-W ubicado más a la izquierda es menor que el período mínimo del mismo.

Bajo esta suposición se pueden plantear dos casos: cuando en el primer nodo del eje el período mínimo del vector-W es distinto al período mínimo del resto o cuando dicho período es igual.

Para el primer caso (El período mínimo del vector-W es distinto al período del Resto):

7.7.3.2.1) Se divide el estudio en dos partes. Si la cantidad de ocurrencias del vector-W es igual o disminuye en uno en el segundo nodo.

Primer caso (Cantidad de ocurrencias del vector-Ws es igual):

7.7.3.2.1.1) Si $\text{PosL}(\alpha_i) > 0$, $\text{PosL}(\alpha_i) < \text{Per}(\text{val}(\alpha_i))$, $\text{Not PerCte}(\alpha_i)$ y $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_{i+1})$ para algún i $0 \leq i < t-1$ entonces $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.

Demostración:

Tomando la hipótesis dada, entonces $\text{PosIni}(\alpha_i) = 0$, $\text{PosIni}(\alpha_{i+1}) = 0$ y aplicando el punto 7.6.1 se demuestra que $\text{AvgW2}(\alpha_i) > \text{AvgW2}(\alpha_{i+1})$ por lo tanto $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.///

Segundo caso (Cantidad de ocurrencias del vector-W disminuye en uno)

7.7.3.2.1.2) Si $\text{PosL}(\alpha_i) = 0$, $\text{PosL}(\alpha_i) < \text{Per}(\text{val}(\alpha_i))$, $\text{Not PerCte}(\alpha_i)$ y $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_{i+1}) + 1$ para algún i $0 \leq i < t-1$ entonces $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.

Demostración:

Tomando la hipótesis dada, entonces $\text{PosIni}(\alpha_i) = 0$, $\text{PosIni}(\alpha_{i+1}) = 0$, y aplicando el punto 7.6.3 se demuestra que $\text{AvgW1}(\alpha_i) > \text{AvgW1}(\alpha_{i+1})$ por lo tanto $\alpha_i <_1 \alpha_{i+1}$ y $\alpha_i <_2 \alpha_{i+1}$.///

Segundo caso (Período mínimo del vector-W es igual al período mínimo del Resto):

7.7.3.2.2) Se divide el estudio en tres partes. Si la cantidad de ocurrencias del vector-W es igual, disminuye en uno o aumenta en uno en el segundo nodo.

Primer caso (Cantidad de ocurrencias del vector-W igual):

7.7.3.2.2.1) Si $\text{PosL}(\alpha_i) > 0$, $\text{PosL}(\alpha_i) < \text{Per}(\text{val}(\alpha_i))$, $\text{PerCte}(\alpha_i)$ y $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_i+1)$ para algún i $0 \leq i < t-1$ entonces $\alpha_i <_{\alpha_i+1}$ y $\alpha_i <_{\alpha_i+1}$.

Demostración:

Tomando la hipótesis dada, entonces $\text{PosIni}(\alpha_i) = \text{PosL}(\alpha_i)$, $\text{PosIni}(\alpha_i+1) = \text{PosL}(\alpha_i+1)$ o $\text{PosIni}(\alpha_i+1) = 0$, y aplicando el punto 7.6.1 se demuestra que $\text{AvgW1}(\alpha_i) > \text{AvgW1}(\alpha_i+1)$ por lo tanto $\alpha_i <_{\alpha_i+1}$ y $\alpha_i <_{\alpha_i+1}$.///

Segundo caso (Cantidad de ocurrencias del vector-W disminuye en uno):

7.7.3.2.2.2) Si $\text{PosL}(\alpha_i) = 0$, $\text{PosL}(\alpha_i) < \text{Per}(\text{val}(\alpha_i))$, $\text{PerCte}(\alpha_i)$ y $\text{CantVW}(\alpha_i)+1 = \text{CantVW}(\alpha_i+1)$ para algún i $0 \leq i < t-1$ entonces $\alpha_i <_{\alpha_i+1}$ y $\alpha_i <_{\alpha_i+1}$.

Demostración:

Tomando la hipótesis dada, entonces $\text{PosIni}(\alpha_i) = \text{PosL}(\alpha_i)$, $\text{PosIni}(\alpha_i+1) = \text{PosL}(\alpha_i+1)$ o $\text{PosIni}(\alpha_i+1) = 0$, y aplicando el punto 7.6.3 se demuestra que $\text{AvgW1}(\alpha_i) > \text{AvgW1}(\alpha_i+1)$ por lo tanto $\alpha_i <_{\alpha_i+1}$ y $\alpha_i <_{\alpha_i+1}$.///

Tercer caso (Cantidad de ocurrencias del vector-W aumenta en uno):

7.7.3.2.2.3) Si $\text{PosL}(\alpha_i) > 0$, $\text{PosL}(\alpha_i) < \text{Per}(\text{val}(\alpha_i))$, $\text{PerCte}(\alpha_i)$ y $\text{CantVW}(\alpha_i) = \text{CantVW}(\alpha_i+1)+1$ para algún i $0 \leq i < t-1$ entonces $\alpha_i \text{ not } <_{\alpha_i+1}$ y $\alpha_i \text{ not } <_{\alpha_i+1}$.

Demostración:

Hay que notar que se está en un caso que no es camino mínimo, dado que por el segundo nodo ya he pasado. En este caso $\alpha_i \text{ not } <_{\alpha_i+1}$ y $\alpha_i \text{ not } <_{\alpha_i+1}$, por lo tanto se produciría un cambio de nivel.///

7.7.4 Propiedad de la cantidad de movimientos.

En esta propiedad se asegura que, dado un vector-W ubicado en el extremo derecho de un identificador de un nodo, si en un camino mínimo aparece un vector-W menor, entonces se corrigieron por lo menos W posiciones

Demostración

La demostración es trivial, dado que el único movimiento permitido en $B(n)$ es el desplazamiento de una posición en cada paso de derecha a izquierda. Como el vector-W es el vector de tamaño W más grande, para que aparezca uno más chico es necesario que el actual desaparezca. Por lo tanto tendrán que realizarse $n-W$ movimientos para que el vector-W se ubique en la posición 0 y un paso más para que pueda entrar un vector-W inferior.

Luego como $W = \lfloor n/2 \rfloor$ se tiene que

Cantidad de pasos = $(n - W) + 1 = (n - (\lfloor n/2 \rfloor)) + 1 = (n/2)+1$ Si n es par

Cantidad de pasos = $(n - W) + 1 = (n - (\lfloor n/2 \rfloor)) + 1 = (n/2)$ Si n es impar

7.8. DAG en Bruijn

El algoritmo de ruteo en Bruijn, para que sea deadlock-free (según lo expresado en el punto 7.2) ruteará sobre un DAG que se forma de la siguiente manera:

En cada nodo existen cuatro buffers identificados con los números 1,2,3 y 4, y se notan como δ_i con $0 < i \leq 4$.

El ruteo entre los dos nodos contiguos en un camino mínimo es dado por la función desarrollada en el punto 6 y se el buffer del segundo nodo se obtiene con la función Fbff.

Dados $B(n)$, $\alpha \in V(2,n)$, $\beta \in V(2,n)$, $L(\alpha,\beta) \in LG(B(n))$ se define :

Fbff: $\alpha, \beta, \delta_i \rightarrow \delta'_j$ con $0 < i \leq 4$, $i \leq j \leq 4$, $\delta \in \text{buffers de } \alpha$ y $\delta' \in \text{buffers de } \beta$.

Luego:

- Fbff(α, β, δ_i)
- i) δ_i si $\alpha < \beta$.
 - ii) δ_{i+1} si $\alpha \geq \beta$.

Los mensajes inyectados en la red se ubican en el buffer 1 del nodo donde se generan. Mientras los identificadores de los nodos sean comparables en forma ascendente según el orden1, el mensaje se desplazará por el primer nivel de buffers. Cuando esto no sea factible, entonces se comenzará a utilizar el siguiente nivel de buffers y continuará con su camino mientras se cumpla con el orden2. De idéntica manera se procederá con los restantes niveles de buffers.

7.9. Análisis de la cantidad de buffers necesarios según el DAG presentado.

A lo largo de este capítulo se presentaron cuatro órdenes totales para los identificadores de nodos de $B(n)$, se demostraron importantes propiedades de los mismos para caminos mínimos y se construyó con ellos un DAG entre los cuatro buffers de cada nodo.

Finalizando, en este punto se demuestra que estos cuatro buffers son suficientes en caminos mínimos en Bruijn, para rutear un mensaje desde cualquier nodo origen hasta cualquier nodo destino.

Dado un camino mínimo de longitud t que parte de un nodo α hacia un nodo β en $B(n)$, notado por $\alpha = \alpha_0, \alpha_1, \dots, \alpha_{t-1} = \beta$:

7.9.1 Abandono del nivel de buffers 1

Los casos en los que se abandona el nivel de buffers 1 al nivel 2 son:

7.9.1.1) Si $\text{val}(\alpha_i) < \text{val}(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.1 el siguiente vector-W se ubica en el extremo más a la derecha del identificador del nodo α_{i+1}

7.9.1.2) Si $\text{val}(\alpha_i) \geq \text{val}(\alpha_{i+1})$ y $\text{AvgW1}(\alpha_i) < \text{AvgW1}(\alpha_{i+1})$ y $\text{AvgW2}(\alpha_i) < \text{AvgW2}(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.3.2.2.3 no se está en presencia de caminos mínimos.

7.9.2 Abandono del nivel de buffers 2

Teniendo en cuenta que se comienza a utilizar el segundo nivel de buffers, el vector-W está ubicado en el extremo derecho del identificador, se dirá que los casos en los que se abandona el nivel de buffers 2 al 3 son:

7.9.2.1) Si $\text{val}(\alpha_i) > \text{val}(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.4 en este nivel se han corregido por lo menos W posiciones.

7.9.2.2) Si $val(\alpha_i) \leq val(\alpha_{i+1})$ y $AvgW1(\alpha_i) < AvgW1(\alpha_{i+1})$ y $AvgW2(\alpha_i) < AvgW2(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.3.2.2.3 no se está en presencia de caminos mínimos.

7.9.3 Abandono del nivel de buffers 3

Los casos en los que se abandona el nivel de buffers 3 al nivel 4 son:

7.9.3.1) Si $val(\alpha_i) < val(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.1 el siguiente vector- W se ubica en el extremo más a la derecha del identificador del nodo α_{i+1}

7.9.3.2) Si $val(\alpha_i) \geq val(\alpha_{i+1})$ y $AvgW1(\alpha_i) < AvgW1(\alpha_{i+1})$ y $AvgW2(\alpha_i) < AvgW2(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.3.2.2.3 no se está en presencia de caminos mínimos.

7.9.4 Abandono del nivel de buffers 4

Teniendo en cuenta que en el orden 4 la ocurrencia vector- W ubicado en el extremo derecho del identificador, se dirá que los casos en los que se abandona el orden 4 son:

7.9.4.1) Si $val(\alpha_i) > val(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.4 en este nivel se corrigieron por lo menos W posiciones.

7.9.4.2) Si $val(\alpha_i) \leq val(\alpha_{i+1})$ y $AvgW1(\alpha_i) < AvgW1(\alpha_{i+1})$ y $AvgW2(\alpha_i) < AvgW2(\alpha_{i+1})$ para algún i $0 \leq i < t-1$.

Según lo demostrado en la propiedad 7.7.3.2.2.3 no se está en presencia de caminos mínimos.

Luego, queda demostrado que en cada par de órdenes, Orden1-Orden2 y Orden3-Orden4, se corregirá como mínimo $\lceil n/2 \rceil$ posiciones. Por lo tanto:

$$\lceil n/2 \rceil + \lceil n/2 \rceil = n \quad \text{Si } n \text{ es par}$$

$$\lceil n/2 \rceil + \lceil n/2 \rceil = n+2 \quad \text{Si } n \text{ es impar}$$

Luego, cuatro buffers por nodo son suficientes para todo camino mínimo en redes de Bruijn.///

8 Redes de Bruijn Bidireccional

A continuación se presentarán los algoritmos de ruteo para redes de Bruijn Bidireccionales.

Como se ha mencionado anteriormente, el algoritmo de ruteo consta de dos partes: la función de ruteo que se desarrolla en el presente capítulo y el gráfico acíclico dirigido (capítulo 9)

Nuevamente se comienza especificando la red de Bruijn Bidireccional:

La topología Bruijn Bidireccional, notada como $UB(n)$, consiste en 2^n nodos. Cada nodo corresponde a un único vector identificado $V=(v_0, \dots, v_{n-1}) \in V(2, n)$. Dos nodos $\alpha=(\alpha_0, \dots, \alpha_{n-1})$ y $\beta=(\beta_0, \dots, \beta_{n-1})$ están unidos por un eje direccional notado como $L(\alpha, \beta)$ si:

$$i) \quad \alpha_i = \beta_{i-1} \text{ para } 0 < i \leq n-1 \quad \text{y} \quad 0 \leq \beta_{n-1} < 1.-$$

o

$$ii) \quad \alpha_i = \beta_{i+1} \text{ para } 0 \leq i < n-1 \quad \text{y} \quad 0 \leq \beta_0 < 1.-$$

Al conjunto de ejes que pertenecen a $UB(n)$ se lo notará $ULG(UB(n))$ divididos en dos tipos $ULGL(UB(n))$ y $ULGR(UB(n))$ definidos en i) y ii) respectivamente.

Se demostrará que la función de ruteo minimal presentada en este trabajo es correcta teniendo la propiedad de ser full-adaptiva.

Se comienza probando que si un camino mínimo cualquiera que une dos nodos α y β con una longitud menor que el diámetro de la red, entonces existe una secuencia de bits que se repite idénticamente en ambos nodos α y β . Luego se demostrará que si el camino es mínimo alguna secuencia de bits se conserva íntegra, es decir que en cualquier nodo intermedio perteneciente al camino se podrá hallar dicha secuencia.

Se continúa analizando el comportamiento de las redes de Bruijn Bidireccionales cuando existen más de una secuencia de bits idénticos en ambos nodos origen y destino. Se llega a la conclusión que a la función de ruteo le basta con analizar de a una secuencia de bits independientemente de las otras ocurrencias. Sólo se puede utilizar una única secuencia para llegar al nodo destino.

Finalmente se presenta una función que permite calcular la longitud del camino mínimo entre dos nodos y la función de ruteo minimal full-adaptiva.

8.1 Propiedades preliminares de las Redes de Bruijn Bidireccionales

8.1.1 Definición de ventana

Esta definición especifica formalmente una secuencia de bits idénticos que se repite en ambos nodos origen y destino de ahora en adelante llamada ventana. Asimismo agrega el concepto que dicha secuencia es maximal, no siendo ésta una subventana de otra.

Una ventana $V(\alpha, \beta)$ es un segmento de bits idéntico que se repite en los nodos α y β de longitud máxima, no importando la posición de dichos segmentos.

$$V(\alpha, \beta) = (a_j \dots a_{j+k}) \text{ si } \exists c, d : (0 \leq c, d \leq n-k) \text{ tq } \alpha_{i+c} = \beta_{i+d} \quad \forall i : 0 \leq i \leq k$$

$$\text{y } (c = 0 \text{ o } d = 0 \text{ o } \alpha_{c-1} \neq \beta_{d-1}) \text{ y } (c+k = n-1 \text{ o } d+k = n-1 \text{ o } \alpha_{c+k+1} \neq \beta_{d+k+1})$$

8.1.2 Existencia de ventanas en caminos mínimos

La siguiente proposición dice que de existir un camino entre los nodos α y β pertenecientes a $UB(n)$ cuya longitud sea menor al diámetro de la red n , entonces existe una ventana entre los nodos α y β .

Intuitivamente se puede observar que si se recorre c ejes, insertando c bits, c menor que n , entonces existe una ventana entre los nodos α y β , estos bits son aquellos que no fueron eliminados (Se puede pensar que no fueron ingresados).

Proposición

Si existe un camino c entre los nodos α y β de longitud L menor al diámetro de la red (n) entonces existe por lo menos una ventana $V(\alpha, \beta)$.

Demostración

Supóngase que existe un camino c entre los nodos α y β de longitud L menor que n y no existe ninguna ventana V entre los nodos α y β .

Por la construcción de la red de Bruijn Bidireccional, en cada movimiento se cambia un único bit.

Sea $L = L_{izq} + L_{der}$, donde L_{izq} son los bits insertados por el extremo izquierdo y L_{der} los bits insertados del extremo derecho para arribar al nodo destino β . La cantidad máxima de bits modificados es igual al máximo de $\{L_{der}, L_{izq}\}$ (esto se debe a que al insertar por un extremo se pierden los últimos bits del extremo opuesto).

El número de bits máximos a ser modificados se obtiene insertando bits sólo por un extremo, ya que maximizar la función $\max \{L_{der}, L_{izq}\}$ resulta igual a L que es la cantidad máxima de bits que se modifican.

Dado que $L < n$ se conserva un segmento de $n-L$ bits idénticos de la dirección del nodo α y del nodo β sin modificar, por lo tanto existe una ventana. Con esta afirmación se llega a un absurdo de la suposición planteada.///

8.1.3 Corolario de la conservación de la ventana

El presente corolario dice que en un camino mínimo de longitud c menor que n , alguna ventana se conserva íntegra a lo largo del mismo, es decir que ninguno de los bits que la conforman se pierde por desplazamiento. Intuitivamente se ve que de perderse por lo menos algún bit, el mismo debería volver a ser insertado, por lo tanto el camino no sería mínimo ya que se podría haber evitado a priori cruzar dos ejes, el primero para eliminarlo por algún extremo y el segundo al ser ingresado nuevamente por el mismo extremo que fue eliminado.

Corolario

Alguna ventana $V(\alpha, \beta)$ se conserva íntegra a lo largo de todo un camino mínimo c de longitud $< L$.

Demostración

Supongamos que no se conserva íntegra a lo largo de todo el camino mínimo c , sino que sólo se conservan k bits de la ventana, $0 \leq k \leq n - L$ a lo largo de todo el camino.

Entonces existe un nodo δ perteneciente al camino c tal que no contiene a toda la ventana, sino por lo menos a una subventana SB de $V(\alpha, \beta)$ de longitud menor que $n-L$ bits.

Como toda la secuencia de bits que pertenecen a la ventana también pertenecen al nodo destino β , estos bits que no se conservan, tendrán que ser insertados nuevamente, por lo tanto se redefinirá:

- $L_{der} = L'_{der} + L''_{der}$ donde L'_{der} son los bits insertados por la derecha que no pertenecían a la ventana $V(\alpha, \beta)$ y donde L''_{der} son los bits insertados por la derecha que pertenecían a la ventana $V(\alpha, \beta)$.
- $L_{izq} = L'_{izq} + L''_{izq}$ donde L'_{izq} son los bits insertados por la izquierda que no pertenecían a la ventana $V(\alpha, \beta)$ y donde L''_{izq} son los bits insertados por la izquierda que sí pertenecían a la ventana $V(\alpha, \beta)$, donde $L''_{izq} + L''_{der} \geq 1$.

Por lo tanto la cantidad de bits nuevos insertados es $L' = L'_{izq} + L'_{der}$, resultando $L' < L$. La cantidad máxima de bits modificados es L' que también resulta de maximizar la función $\max(L'_{izq}, L'_{der})$. Luego c no es un camino mínimo ya que existía una ventana $V'(\alpha, \beta)$ que se obtenía con un camino que sólo modifica L' bits.

Se llega a un absurdo por suponer que alguna ventana no se conserva íntegra a lo largo de todo un camino mínimo.///

8.2 Conjunto de Ventanas

8.2.1 Definición de secuencia

Una secuencia S es una secuencia de n ventanas entre dos nodos α y β .

$$S = \{V_0(\alpha, \beta), \dots, V_l(\alpha, \beta), \dots, V_{n-1}(\alpha, \beta)\}$$

8.2.2 Definición de configuración

Una configuración C es una secuencia de n ventanas que se repite en ambos nodos, sin solaparse.

$$C = \{V_0(\alpha, \beta), \dots, V_l(\alpha, \beta), \dots, V_{n-1}(\alpha, \beta)\}$$

Donde si $V_h(\alpha, \beta) = (a_h \dots a_{h+k}) \Rightarrow$ para todo $V_l(a_c \dots a_{c+z}) \in C$ tq $l \neq h$:

$$c > h+k \text{ o bien } h+k < c$$

Se trabajará con configuraciones ya que al no estar solapadas las ventanas es posible reutilizar los bits de éstas para obtener la dirección del nodo destino.

8.2.3 Definición de orden en configuraciones

Orden: Si la posición de comienzo de una ventana V_a es menor que la posición de comienzo de una ventana V_b en el nodo α también lo es en el nodo β .

$$\text{Sea } C = \{V_0(\alpha, \beta), V_a(\alpha, \beta), \dots, V_b(\alpha, \beta), \dots, V_l(\alpha, \beta), \dots, V_{n-1}(\alpha, \beta)\}$$

$$\text{Si } V_a(\alpha, \beta) = (a_j \dots a_{j+k}) \text{ si } \exists c, d, 0 \leq c, d \leq n-k \text{ tq } \alpha \mid c = \beta \mid d \quad \forall i: 0 \leq i \leq k$$

$$\text{y } (c=0 \text{ o } d=0 \text{ o } \alpha c-1 \neq \beta d-1) \text{ y } (c+k=n-1 \text{ o } d+k=n-1 \text{ o } \alpha c+k+1 \neq \beta d+k+1)$$

$y Vb(\alpha, \beta) = (aj' \dots aj' + k')$ sii $\exists c', d' 0 \leq c', d' \leq n-k'$ tq $\alpha i + c' = \beta i + d' \quad \forall i : 0 \leq i \leq k'$
 $y (c'=0 \text{ o } d'=0 \text{ o } \alpha c'-1 \neq \beta d'-1) \text{ y } (c'+k'=n-1 \text{ o } d'+k'=n-1 \text{ o } \alpha c'+k'+1 \neq \beta d'+k'+1)$
 entonces si $c \leq c' \Rightarrow d \leq d'$

8.2.4 Definición de posición en configuraciones

Posición: Las configuraciones estarán en Posición si sus ventanas están idénticamente distribuidas en ambos nodos, es decir que si la distancia que separa a las ventanas Va y Vb es igual en ambos nodos.

Sea $C = \{ V 0(\alpha, \beta), Va(\alpha, \beta), \dots, Vb(\alpha, \beta), \dots, V l(\alpha, \beta), \dots, V n-1(\alpha, \beta) \}$

Si $Va(\alpha, \beta) = (aj \dots aj + k)$ sii $\exists c, d 0 \leq c, d \leq n-k$ tq $\alpha i + c = \beta i + d \quad \forall i : 0 \leq i \leq k$
 $y (c=0 \text{ o } d=0 \text{ o } \alpha c-1 \neq \beta d-1) \text{ y } (c+k=n-1 \text{ o } d+k=n-1 \text{ o } \alpha c+k+1 \neq \beta d+k+1)$
 $y Vb(\alpha, \beta) = (aj' \dots aj' + k')$ sii $\exists c', d' 0 \leq c', d' \leq n-k'$ tq $\alpha i + c' = \beta i + d' \quad \forall i : 0 \leq i \leq k'$
 $y (c'=0 \text{ o } d'=0 \text{ o } \alpha c'-1 \neq \beta d'-1) \text{ y } (c'+k'=n-1 \text{ o } d'+k'=n-1 \text{ o } \alpha c'+k'+1 \neq \beta d'+k'+1)$
 entonces $|c - c'| = |d - d'|$

8.3 Propiedad de las ventanas en orden y posición

Intuitivamente, esta propiedad dice que en una secuencia de ventanas en orden y posición cualquier par de ventanas están separadas entre sí por lo menos por un bit que no pertenece a ninguna de éstas. De no cumplirse esta propiedad, como están en orden y posición, son adyacentes y no son ventanas (Por su definición son de longitud máxima), siendo ambas reemplazadas por una nueva. Esta última es la concatenación de las dos ventanas anteriores.

Propiedad

En una configuración C en orden y posición, las ventanas están separadas entre sí con un segmento de por lo menos un bit que no pertenece a ninguna de las dos ventanas.

Demostración

Supongamos que existen dos ventanas V' y V'' consecutivas llamadas $V'(\alpha, \beta)$ y $V''(\alpha, \beta)$ tal que la posición del bit de comienzo de V' es menor que la posición del bit de comienzo de la ventana V'' . Como son consecutivas:

$$V'(\alpha, \beta) = (aj \dots aj + k) \text{ y } V''(\alpha, \beta) = (at \dots at + h) \text{ donde } j + k + 1 = t$$

Luego V' no es ventana ya que para serlo debe cumplir con la siguiente definición:

$$V(\alpha, \beta) = (aj \dots aj + k) \text{ sii } \exists c, d 0 \leq c, d \leq n-k \text{ tq } \alpha i + c = \beta i + d \quad \forall i : 0 \leq i \leq k$$

$$y (c=0 \text{ o } d=0 \text{ o } \alpha c-1 \neq \beta d-1) \text{ y } (c+k=n-1 \text{ o } d+k=n-1 \text{ o } \alpha c+k+1 \neq \beta d+k+1)$$

pero $c+k < n-1$ y $d+k < n-1$ (Ya que luego comienza la segunda ventana) y $\alpha c+k+1 = at$ y $\beta d+k+1 = at$ entonces $\alpha c+k+1 = \beta d+k+1$.

8.4 Teorema de configuraciones sin orden

Cualquier algoritmo que encuentre caminos mínimos en una red de Bruijn, obtendrá los mismos caminos mínimos considerando sólo aquellas configuraciones en orden.

Demostración

La única forma de modificar el orden de las ventanas en las redes de Bruijn es descartar la ventana fuera de orden e insertarla nuevamente en el orden correcto.

Al no conservarse la ventana primeramente encontrada y tener que ingresarla como si no existiera en el nodo origen, se puede afirmar que no es necesario considerarla.

8.5 Teorema de configuraciones sin posición

Cualquier algoritmo que encuentre caminos mínimos en una red de Bruijn, obtendrá los mismos caminos mínimos considerando sólo aquellas configuraciones donde las ventanas estén en posición.

Demostración

La única forma de modificar la posición de las ventanas en las redes de Bruijn es desplazar los bits del identificador del nodo para introducirla en posición como si nunca hubiese existido. Eventualmente incluso habrá que eliminarla.

Al no ser la ventana primariamente encontrada aquella que es posicionada como corresponde puede no ser tomada en cuenta.

8.6 Ventana única

Según lo demostrado en los dos teoremas anteriores, sólo aquellas configuraciones que estén en orden y posición deberán ser tomadas en cuenta en los algoritmos que obtengan caminos mínimos en redes de Bruijn.

Como se ha demostrado oportunamente, en las configuraciones con orden y posición, las ventanas están separadas entre sí por lo menos con un segmento de bits. Este segmento de bits tendrá que ser corregido por la función de ruteo, cualquiera sea ésta. Para corregir este segmento es necesario eliminar alguna de las dos ventanas adyacentes a él. Por lo tanto sólo una ventana de una secuencia podrá ser utilizada para obtener un camino mínimo.

8.7 Cálculo de la longitud de un camino

A continuación se presentará una fórmula que permite calcular la longitud mínima entre dos nodos α y β en una red de Bruijn Bidireccional a partir de la utilización de una única ventana. Se comienza desarrollando una propiedad la cual dice que cualquier camino mínimo en una red de Bruijn Bidireccional, tiene a lo sumo tres fases. Se entiende por fase al conjunto de movimientos contiguos hacia la misma dirección. Es decir, los movimientos sólo tendrán alguna de las siguientes formas:

- Ingreso de secuencia de bits por la posición menos significativa, ingreso de secuencia de bits por la posición mas significativa y finalmente, ingreso de secuencia de bits por la posición menos significativa.

- Ingreso de secuencia de bits por la posición mas significativa, ingreso de secuencia de bits por la posición menos significativa y finalmente, ingreso de secuencia de bits por la posición mas significativa.
- Ingreso de secuencia de bits por la posición menos significativa e ingreso de secuencia de bits por la posición mas significativa.
- Ingreso de secuencia de bits por la posición mas significativa e ingreso de secuencia de bits por la posición menos significativa.
- Sólo Ingreso de secuencia de bits por la posición menos significativa.
- Sólo Ingreso de secuencia de bits por la posición mas significativa.

Obsérvese que algunos caminos mínimos requieren tres fases como el siguiente ejemplo: del nodo 1xxxxxx1 se desea rutear al nodo 0xxxxxx0. Se han marcado con x los bits que corresponden a la ventana entre estos nodos. Existen ocho caminos mínimos para conectar este par de nodos:

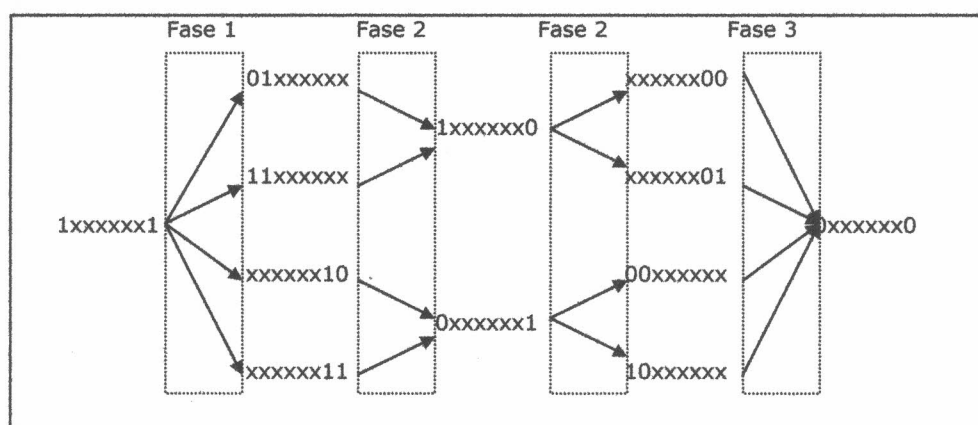


Figura 7.7.A: Ejemplo de caminos mínimos

Como se ha demostrado anteriormente, sólo se puede trabajar con una única ventana para hallar un camino mínimo que una dos nodos α y β . Es por esto que, de no estar volcada la ventana seleccionada en el nodo de origen en alguno de los extremos, no se puede comenzar a formar una nueva ventana, ya que existiría un segmento de bits en el medio de ambas. La única alternativa es hacerla llegar a alguno de los extremos, donde concluiría la primera fase, habiendo ingresado f_1 bits. En caso que originalmente la ventana este sobre algún extremo, la fase puede ser nula, no cruzándose ningún eje.

Luego se debe comenzar a agrandar el tamaño de la ventana, ingresando f_2' bit por el extremo que está volcada. Luego de esta fase se puede haber llegado al nodo destino. De no ser así, se debe continuar en esta fase, ingresando f_2'' bits cualquiera hasta hacer llegar la ventana al extremo opuesto.

Se continúa en la tercera fase ingresando f_3 bits por el mismo extremo que en la primera, pero esta vez se extiende la ventana. Obsérvese que $f_3 = f_2''$, en caso contrario se perdería alguno de los f_2' bits ingresados en la primer etapa de la fase 2, y que se ha extendido la longitud de la ventana en $f_2' + f_3$ bits. De hecho se ha llegado al nodo destino ya que la ventana está volcada en ambos extremos.

De existir una cuarta fase se ingresarían f_4 bits, pero se eliminaría por lo menos 1 bit de los ingresados en la fase 3. Sin embargo para ingresar ese bit no sólo se cruzó un eje en la fase 3, sino también se posicionó la ventana para aceptarlo, cruzándose un eje en la segunda etapa de la fase 2. Por lo tanto el camino encontrado no es mínimo ya que se han cruzado $f_1 + f_2' + f_2'' + f_3 + f_4$ ejes, pero se podrían haber cruzado $f_1 + (f_2' + 1) + (f_2'' - 1) + (f_3 - 1) + f_4$ ejes.

La figura 7.7.B presenta los parámetros de la función que permite calcular la longitud de un camino mínimo en una red de Bruijn Bidireccional utilizando únicamente una ventana.

- c : Indica el comienzo de la ventana en el nodo de origen α
- d : Indica el comienzo de la ventana en el nodo destino β
- l : Indica la longitud de la ventana. Este valor puede ser calculado a partir de los nodos α y β y los valores c y d .

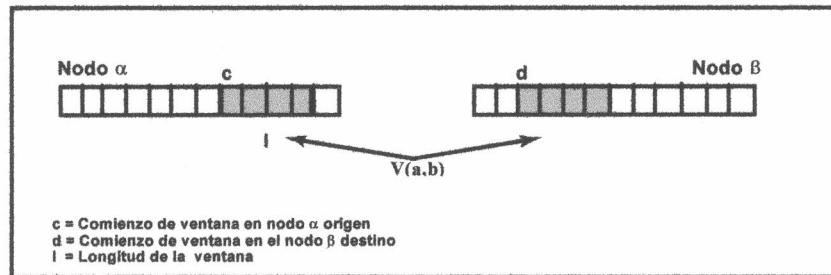


Figura 8.7.B: Parámetros de función de ruteo

Se presentarán primeramente diferentes casos y luego una fórmula genérica:

Caso 1: La ventana se encuentra volcada en extremos opuestos.

La longitud del camino mínimo en este caso es de $(n-l)$ ya que se ingresará esta cantidad de bits para hacer llegar esta ventana a su posición final.

Los bits ingresados corresponden con la dirección de destino. Utilizando esta ventana, el camino mínimo es único.

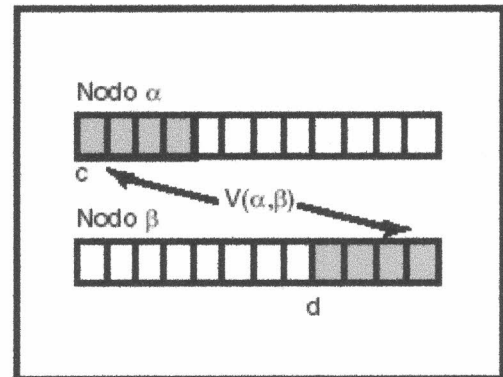
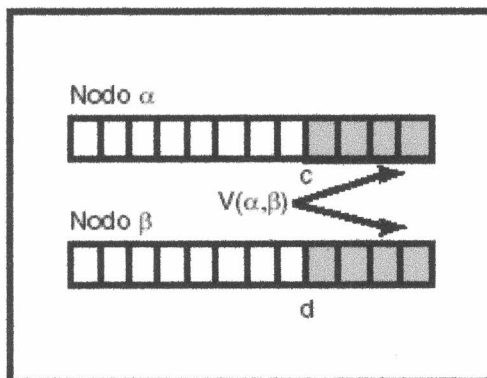


Figura 8.7.C: Extremos opuestos

Figura 8.7.D: Mismo extremo



Caso 2: La ventana se encuentra volcada en ambos nodos en el mismo extremo.

La longitud del camino mínimo en este caso es de $2(n-l)$, siendo $2(n-l) < n$ que es el diámetro de la red. De no cumplirse esta condición, no puede ser tomada en cuenta esta ventana, ya que consume menos pasos generar la dirección del nodo destino como si no existiese dicha ventana. Se deben ingresar $n-l$ bits por el extremo donde se encuentra volcada la ventana hasta hacerla llegar

al otro extremo, para luego ingresar los $n-l$ bits que corresponden a la dirección del nodo destino. El hecho de poder ingresar en la primer fase cualquier secuencia de bits permite afirmar que el camino mínimo en este caso no es único.

Caso 3 : La ventana no se encuentra en un extremo en alguno o los dos nodos.

En este caso existen dos alternativas. Ambas son similares y están compuesta de tres fases que sólo difieren en la dirección de los movimientos de cada una:

1. Desplazar la ventana hasta que llegue a un extremo, ingresando cualquier secuencia de bits.
2. Insertar los bits correspondientes al nodo hasta llegar a alguno de los dos extremos del nodo destino y luego insertar cualquier secuencia de bits hasta que la ventana llegue al extremo opuesto
3. Insertar la secuencia correspondiente de bits del nodo destino por el extremo donde se encuentra situada la ventana hasta llegar al nodo destino

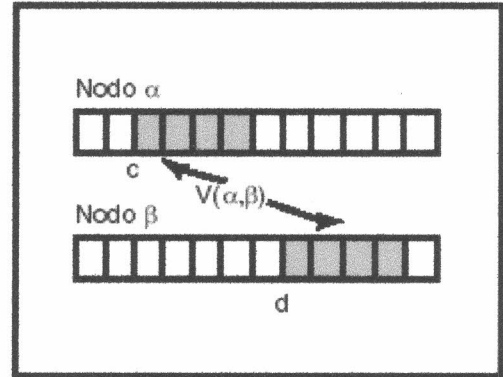


Figura 8.7.E:Ventanas libres

La secuencia de bits insertados en la fase 1, así como la secuencia ingresados en la segunda etapa de la fase 2, nos permite afirmar que también en este caso existe más de un camino mínimo.

La cantidad de bits insertados es :

	Primer Movimiento Izquierda	Primer Movimiento Derecha
Fase 1	c bits	$n - (c+1)$ bits
Fase 2	$n-1$ bits	$n-1$ bits
Fase 3	$n-(d+1)$ bits	d bits
Total	$c + (n-1) + n-(d+1) =$ $2(n-1) + c - d$	$n - (c+1) + (n-1) + d =$ $2(n-1) - c + d$ bits
	$2(n-1) - c-d $ bits	

Forma general :

Dada una ventana $V(\alpha, \beta)$ que de longitud l , tal que comienza en el bit c en el nodo α y en el bit d en el nodo destino β :

$$F(n, c, d, l) = 2(n-1) - |c-d|, \text{ siendo } F(n, c, d, l) \leq n \text{ ó } n$$

Es la misma fórmula presentada en el caso 3.

Se debe observar que en el caso 1, $|c-d| = n-1$ por lo tanto $F(n, c, d, l) = n-1$. En el caso 2, $c=d$, siendo $|c-d| = 0$ resultando $F(n, c, d, l) = 2(n-1)$.

Por lo tanto, la función para calcular el camino mínimo en una red de Bruijn Bidireccional es:

$$\text{Min}(\forall c, d : F(n, c, d, l)) = \text{Min}(\forall c, d : 2(n-1) - |c-d|)$$

Esta función, calcula la longitud del camino mínimo. Es correcta ya que no existe ninguna ventana entre los nodos α y β que no sea contemplada.

8.8 Función de Ruteo

El presente algoritmo de ruteo permite encaminar un paquete inyectado en un nodo α con destino a un nodo β mediante cualquier camino de longitud mínima en una red de Bruijn Bidireccional.

Para todo nodo α' que recibe un paquete con destino al nodo β :

1. Calcular la función de camino mínimo

$$\Psi : \text{Min}(\forall c,d: F(n,c,d,l)) = \text{Min}(\forall c,d : 2(n-l) - |c-d|)$$

2. Si $\Psi = 0$ Se llegó al nodo destino.

3. Si $\Psi > n$:

Insertar por el extremo izquierdo el bit con valor $\beta[n-1]$, o bien

Insertar por el extremo derecho el bit con valor $\beta[0]$

Si $\Psi = n$:

Insertar por el extremo izquierdo el bit con valor $\beta[n-1]$, o

Insertar por el extremo derecho el bit con valor $\beta[0]$

O bien continuar al punto 4

4. Según c, d :

Si $d < c$:

(4.1) Si $n = c + l$ insertar por el extremo derecho el bit con valor $\beta[d+l]$.

($d < c \Rightarrow d+l < c+l = n$ entonces $d+l < n$)

(4.2) Caso contrario, insertar cualquier bit por el extremo izquierdo

Si $d > c$:

(4.3) Si $c = 0$ insertar por el extremo izquierdo el bit con valor $\beta[d-1]$.

($d > c = 0 \Rightarrow d \geq 0$)

(4.4) Caso contrario, insertar cualquier bit por el extremo derecho

Si $d = c$:

(4.5) Si $d = 0$ insertar cualquier bit por la izquierda

(4.6) Si $d + l = n$ insertar cualquier bit por la derecha

(4.7) Caso contrario, insertar cualquier bit por cualquiera de los dos extremos.

El algoritmo anterior genera todos los caminos de longitud mínima en una red de Bruijn. A continuación, se demostrará que en cada paso el mensaje es ruteado a un nodo, para el cuál existe un camino de longitud mínima menor que el anterior hasta llegar al nodo destino.

En el **paso 2** se asegura que el algoritmo termina dado que llega al nodo destino

En el siguiente, **paso 3**, de no ser $k \leq n$, al insertar el bit correspondiente en cualquiera de las dos alternativas se obtiene un camino de longitud mínima de $n-1$.

Para las distintas alternativas del **paso 4**:

(4.1) Luego de insertar por el extremo derecho el bit con valor $\beta[d+l]$ se obtiene $c'=c-1$ y $l'=l+1$, $d'=d$ y siendo $d < c \Rightarrow d' < c'$:

$$f(n, c', d', l') = 2(n-l') - (c'-d) = 2(n-l-1) - (c-1-d) = 2(n-l) - (c-d) - 2 + 1 = f(n, c, d, l) - 1$$

(4.2) Al insertar cualquier bit por el extremo izquierdo se obtiene $c'=c+1$, $l'=l$, $d'=d$, y siendo $d < c \Rightarrow d' < c'$:

$$f(n, c', d', l') = 2(n-l) - (c'-d) = 2(n-l) - (c+1-d) = 2(n-l) - (c-d) - 1 = f(n, c, d, l) - 1$$

(4.3) Luego de insertar por el extremo izquierdo el bit con valor $\beta[d-1]$, se obtiene $c'=c=0$, $l'=l+1$, $d'=d-1$ y siendo $d > c \Rightarrow d' \geq c'$:

$$f(n, c', d', l') = 2(n-l') - (d'-c) = 2(n-l-1) - (d-1-c) = 2(n-l) - (d-c) - 2 + 1 = f(n, c, d, l) - 1$$

(4.4) Luego de insertar cualquier bit por el extremo derecho, se obtiene $c'=c-1$, $l'=l$, $d'=d$ y siendo $d > c \Rightarrow d' > c'$:

$$f(n, c', d', l) = 2(n-l) - (d-c') = 2(n-l) - (d-c+1) = 2(n-l) - (d-c) - 1 = f(n, c, d, l) - 1$$

(4.5) Luego de insertar cualquier bit por el extremo izquierdo $c'=c+1$, $d'=d$, $l'=l$ y siendo $c=d \Rightarrow c' > d'$:

$$f(n, c', d', l) = 2(n-l) - (c'-d) = 2(n-l) - (c+1-d) = 2(n-l) - (c-d) - 1 = f(n, c, d, l) - 1$$

(4.6) Luego de insertar cualquier bit por el extremo derecho se obtiene $c'=c-1$, $d'=d$, $l'=l$ y siendo $c=d \Rightarrow c' < d'$:

$$f(n, c', d', l') = 2(n-l) - (d-c') = 2(n-l) - (d-c+1) = 2(n-l) - d(d-c) - 1 = f(n, c, d, l) - 1$$

(4.7) Si se inserta cualquier bit por el extremo izquierdo, se obtiene $c'=c+1$, $d'=d$ y $l'=l$ y siendo $c=d \Rightarrow c' > d'$:

$$f(n, c', d', l) = 2(n-l) - (c'-d) = 2(n-l) - (c+1-d) = 2(n-l) - (c-d) - 1 = f(n, c, d, l) - 1$$

De insertar cualquier bit por el extremo derecho, se obtiene $c'=c-1$, $d'=d$ y $l'=l$ y siendo $c=d \Rightarrow c' < d'$:

$$f(n, c', d', l) = 2(n-l) - (d-c') = 2(n-l) - (d-c+1) = 2(n-l) - (d-c) - 1 = f(n, c, d, l) - 1$$

Finalmente, se debe notar que en todos los casos del punto 4, la ventana se mantiene íntegra y en alguno de los casos aumenta su longitud, hasta llegar al nodo destino.

9. Grafo dirigido acíclico (DAG) para Bruijn Direccional.

9.1. Introducción

Como se mencionó anteriormente una de las formas para garantizar la no existencia de deadlock en el ruteo de mensajes se logra mediante la definición de un orden estricto monotónico de los buffers de manera tal que no se produzcan ciclos y la función de ruteo pueda cumplir con el mismo.

El DAG presentado en esta sección será *escalable*, es decir que cada nodo tendrá un número constante de buffers y proveerá full-adaptabilidad para caminos mínimos.

En la siguiente definición se toman como recursos críticos a los buffers de los nodos, y se establece con éstos un grafo acíclico dirigido (DAG).

El presente punto se basa fuertemente en el DAG desarrollado en el capítulo 7 y en las características de los caminos mínimos en Bruijn Bidireccional desarrolladas en el punto 7 del capítulo 8 que expresa la cantidad de fases (movimientos contiguos y en el mismo sentido) de éstos.

9.2. Definiciones

Fase: Es el conjunto de ejes contiguos de un camino mínimo entre dos nodos α y β pertenecientes a $UB(n)$ tal que todos pertenecen a $ULGL(UB(n))$ o bien a $ULGR(UB(n))$.

FaseL Es el conjunto de ejes contiguos de un camino mínimo entre dos nodos α y β pertenecientes a $UB(n)$ tal que todos pertenecen a $ULGL(UB(n))$.

FaseR Es el conjunto de ejes contiguos de un camino mínimo entre dos nodos α y β pertenecientes a $UB(n)$ tal que todos pertenecen a $ULGR(UB(n))$.

9.3. Cantidad necesaria de movimientos contiguos

En este punto se nombran las características de caminos mínimos de Bruijn Bidireccional desarrolladas en el **punto 8.7**. En éste se establece como en todo camino mínimo entre dos nodos cualquiera en $UB(n)$ existe a lo sumo 3 fases.

Por lo tanto en cada camino mínimo se tendrá a lo sumo:

- a) FaseL + FaseR + FaseL
- o
- b) FaseR + FaseL + FaseR.

9.5 Presentación del DAG en Bruijn Direccional Inverso

Lo expuesto en el punto 7 se basó en redes de Bruijn Direccionales con ejes del siguiente formato

De cada nodo $V = (v_0, \dots, v_{n-1})$ parte un eje direccional a un nodo $W = (w_0, \dots, w_{m-1})$ sii

$$v_i = w_{i-1} \text{ para } 1 \leq i \leq n-1$$

y

$$0 \leq w_{n-1} \leq 1$$

por lo tanto, se supone un desplazamiento de la dirección de derecha a izquierda.

Si se plantea una topología de Bruijn con el siguiente tipo de ejes (Bruijn Direccional Inverso):

De cada nodo $V = (v_0, \dots, v_{n-1})$ parte un eje direccional a un nodo $W = (w_0, \dots, w_{m-1})$ si

$$v_i = w_{i+1} \text{ para } 0 \leq i \leq n-2$$

y

$$0 \leq w_0 \leq 1$$

se supone un desplazamiento de la dirección de izquierda a derecha.

En Bruijn Bidireccional se usan ejes en ambos sentidos. El DAG de Bruijn Direccional Inverso es similar al desarrollado en el capítulo 7. El mismo no es expuesto dado que no aporta ningún valor adicional al presente trabajo.

9.6 DAG en Bruijn Bidireccional

En el algoritmo de ruteo en Bruijn Direccional se utilizaron cuatro buffers por nodo.

Como es de fácil deducción, cada fase puede ser tomada como un ruteo dentro de una topología de Bruijn Direccional o Direccional Inverso. Dependiendo del sentido de la fase (FaseL o FaseR) se utiliza el DAG correspondiente.

Por lo tanto, para el DAG en Bruijn Bidireccional se utilizan 4 capas de 4 buffers cada una. Estos, buffers y capas se identifican como $\delta_{i,j}$ con $0 \leq i \leq 4$, $0 \leq j \leq 4$, donde i implica el número de buffer y j el número de capa.

El DAG formado por los buffers de los nodos $\in UB(n)$, sobre el cual ruteará la función dada en el capítulo 8, es dado por la siguiente función:

Dados $UB(n)$, $\alpha \in V(2,n)$, $\beta \in V(2,n)$, $L(\alpha,\beta) \in ULG(UB(n))$ se define :

$$Fbffu: \alpha, \beta, \delta_{i,j} \rightarrow \delta'_{s,t} \text{ con } 0 \leq i \leq 4, 1 \leq s \leq 4, 0 \leq j \leq 4, 1 \leq t \leq 4, \delta_{i,j} \in \alpha \text{ y } \delta'_{s,t} \in \beta.$$

Luego, para $j=1$ o $j=3$.

$Fbffu(\alpha, \beta, \delta_{i,j})$	i) $\delta_{i,j}$	si $\alpha < \beta$ y $L(\alpha, \beta) \in ULGL(UB(n))$.
	ii) $\delta_{i,j+1}$	si $\alpha < \beta$ y $L(\alpha, \beta) \notin ULGL(UB(n))$.
	iii) $\delta_{i+1,j}$	si $\alpha \geq \beta$ y $L(\alpha, \beta) \in ULGL(UB(n))$.
	iiii) $\delta_{i+1,j+1}$	si $\alpha \geq \beta$ y $L(\alpha, \beta) \notin ULGL(UB(n))$.

Luego, para $j=2$ o $j=4$.

$Fbffu(\alpha, \beta, \delta_{i,j})$	i) $\delta_{i,j}$	si $\alpha < \beta$ y $L(\alpha, \beta) \in ULGR(UB(n))$.
	ii) $\delta_{i,j+1}$	si $\alpha < \beta$ y $L(\alpha, \beta) \notin ULGR(UB(n))$.
	iii) $\delta_{i+1,j}$	si $\alpha \geq \beta$ y $L(\alpha, \beta) \in ULGR(UB(n))$.
	iiii) $\delta_{i+1,j+1}$	si $\alpha \geq \beta$ y $L(\alpha, \beta) \notin ULGR(UB(n))$.

El algoritmo ubicará todos los nuevos mensajes en el buffer 1 de la capa 1 del nodo en el cual se generan.

Si el camino mínimo entre los nodos comienza en faseL, entonces utiliza la técnica direccional descrita en el punto 7.8. En cambio, si comienza en faseR, automáticamente se comienza a utilizar los buffers de la segunda capa con la técnica descrita en el punto 9.5. Al cambiar de fase se procede a utilizar el buffer 1 de la siguiente capa con la técnica adecuada.

10. Conclusiones

En el presente trabajo, primeramente se realizó una introducción al los algoritmos y arquitecturas paralelas. En ésta se presentaron diferentes topologías con sus características y algoritmos de ruteo con sus propiedades que fueron útiles para compararlos y medir su performance. El resultado de esta primera sección no sólo es una introducción al tema, sino también, es una considerable recopilación de los aportes presentados en los últimos trabajos del área.

La segunda sección constituye el aporte original de la presente tesis de licenciatura. En ésta, para las redes de Bruijn Direccionales, se ha demostrado que sólo existe un único camino de longitud mínima exponiendo una fórmula para calcular su longitud, se presentó la función de ruteo y se desarrolló un DAG escalable que solo utiliza cuatro buffers por nodo. Como resultado de todo esto se obtuvo un algoritmo de ruteo minimal, escalable, full-adaptivo, worm-hole.

También se trabajó sobre redes de Bruijn Bidireccionales. Se demostró que cualquier algoritmo de ruteo sólo debe considerar una única ventana para rutear mensajes sobre caminos mínimos. Otra importante característica presentada es la existencia de a lo sumo tres fases. El DAG para éstas redes plantea la utilización de un DAG direccional independiente para cada una de estas fases. Finalmente se llegó a un algoritmo de ruteo minimal, escalable, full-adaptivo, worm-hole, para topologías de Bruijn Bidireccionales, siendo éste también un aporte original, dado que hasta el momento no se han publicado trabajos que conjugan esas características.

Para terminar, en el presente trabajo se dejan planteados algunos puntos para posibles futuras extensiones. Estos pueden servir para mejorar y profundizar algunas áreas, dándole así posibilidades a otros grupos de trabajo.

Finalizando, deseamos destacar a modo personal, el conocimiento ganado en el transcurso de la elaboración del trabajo, más allá del propio de la materia. Los desafíos sorteados al escribir una tesis original, donde tomando el primer camino no siempre se llegó al resultado esperado, la incertidumbre resuelta con persistencia y mucho trabajo de no saber si se lograría completar el trabajo en su totalidad, mas allá de los resultados particulares y la ardua tarea de formalizar el contenido del mismo, salvado con la gran paciencia de nuestro director y jurados.

11. Extensiones posibles al presente trabajo

Dada la similitud existente entre Bruijn y Shuffle Exchange, el presente trabajo podría ser utilizado como base para lograr un algoritmo de ruteo para la segunda topología con idénticas características.

Otra extensión posible a ser analizada en mayor profundidad es desarrollar una nueva técnica de ruteo tolerante a fallas para las redes de Bruijn Bidireccionales. A continuación se presenta intuitivamente:

En el punto 5.3, se han descripto las más usuales técnicas de ruteo. En Packet-switching los mensajes son de tamaño constante y se pueden almacenar íntegramente en un buffer de un nodo y en técnicas pipelines los mensajes no tienen una longitud fija, sino que son divididos en una secuencia de *flits*.

En el punto 5.4.3 se presentaron los dos modelos de fallas: estáticas y dinámicas.

La función de ruteo presentada en el punto 8.8, otorga full-adaptabilidad para caminos mínimos. Por esta razón, existen nodos intermedios en caminos mínimos donde se puede optar por rutear el mensaje hacia más de una salida (Puntos de bifurcación) y otros donde sólo existe una alternativa. Una idea interesante es plantear una técnica nueva de ruteo pipeline, donde la cabeza ruteadora avance sola desde un punto de bifurcación hasta llegar a otro. Allí se puede enviar un asentimiento para que el resto de los flits avancen desde su posición actual hasta éste punto. En los nodos donde sólo existe un único camino a seguir, pueden continuar todos los flits juntos. Para una mayor performance ante fallas, la cantidad de puntos de bifurcación que avanza la cabeza ruteadora pueden ser mayor a uno.

12. Gráficos

12.1. Topologías

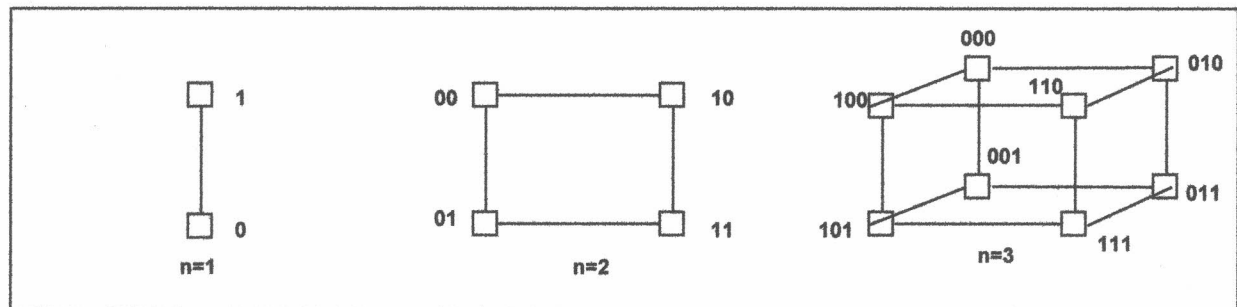


Figura 12.1.1: Hipercubos n-dimensionales para n igual a 2, 3 y 4.

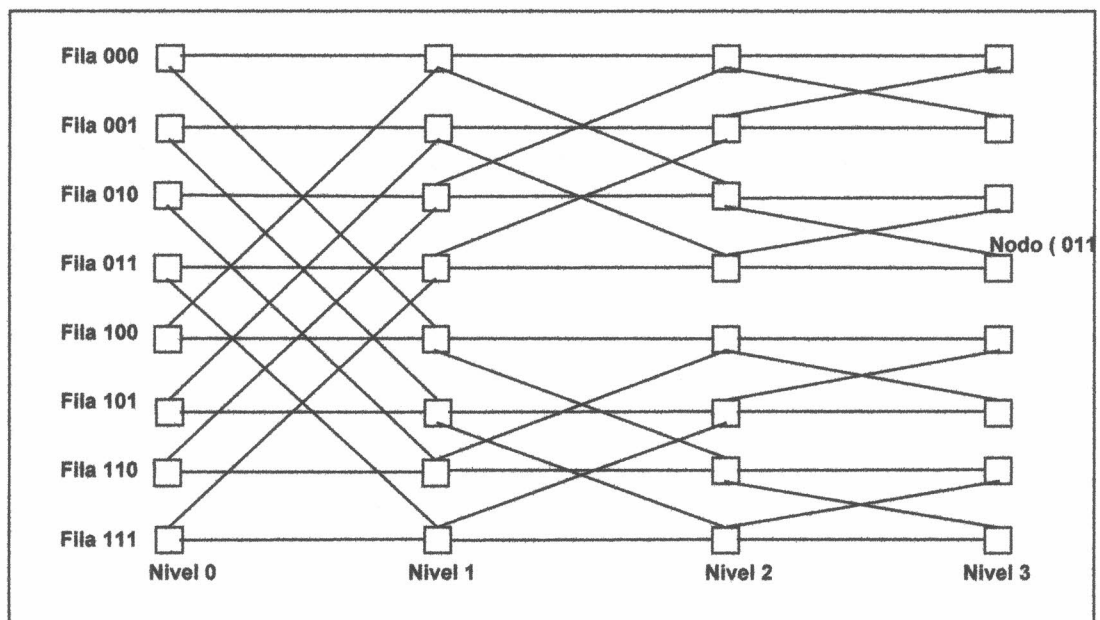


Figura 12.1.2: Butterfly 3-dimensional.

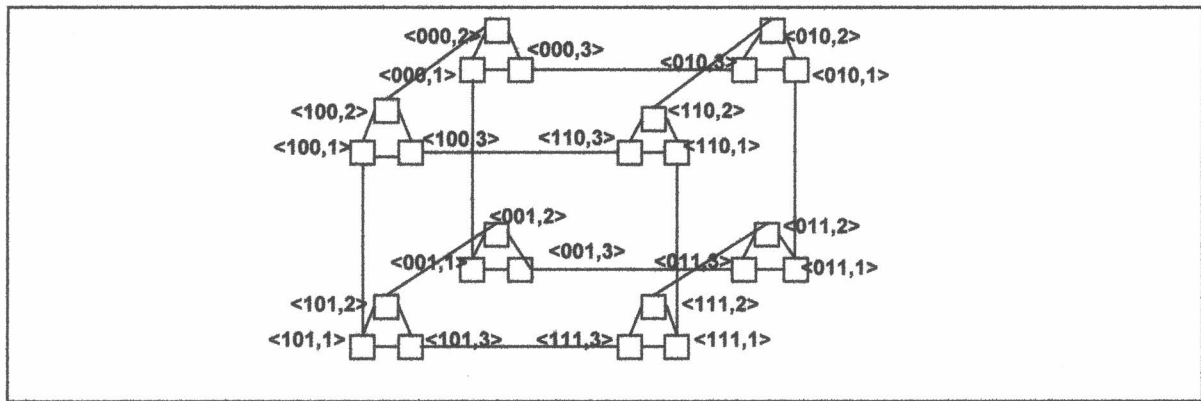


Figura 12.1.3: CCC 3-dimensional.

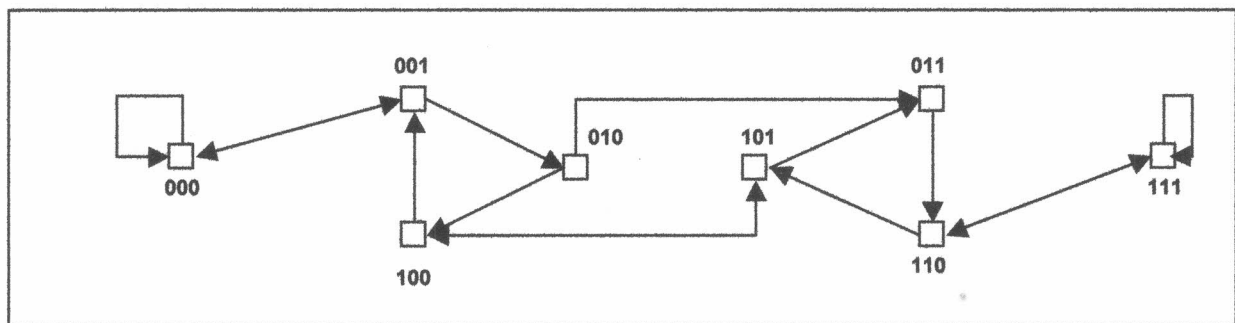


Figura 12.1.4: Suffle exchange 3-dimensional.

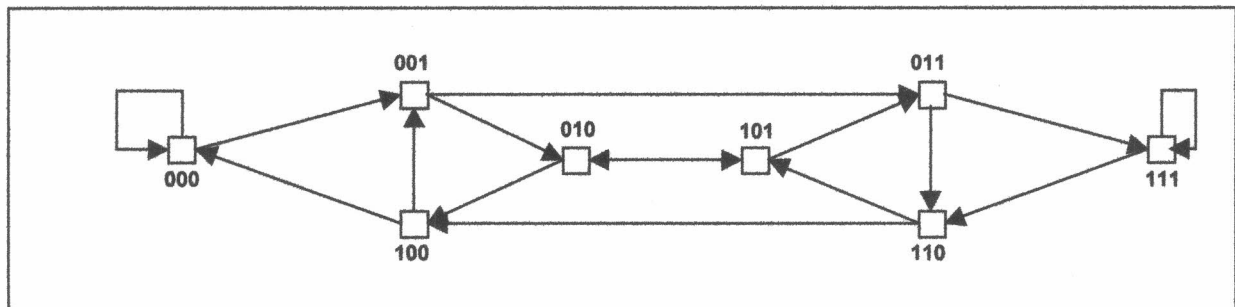


Figura 12.1.5: Bruijn 3-dimensional.

12.2. Comparación de técnicas de ruteo

Ruteo worm-hole

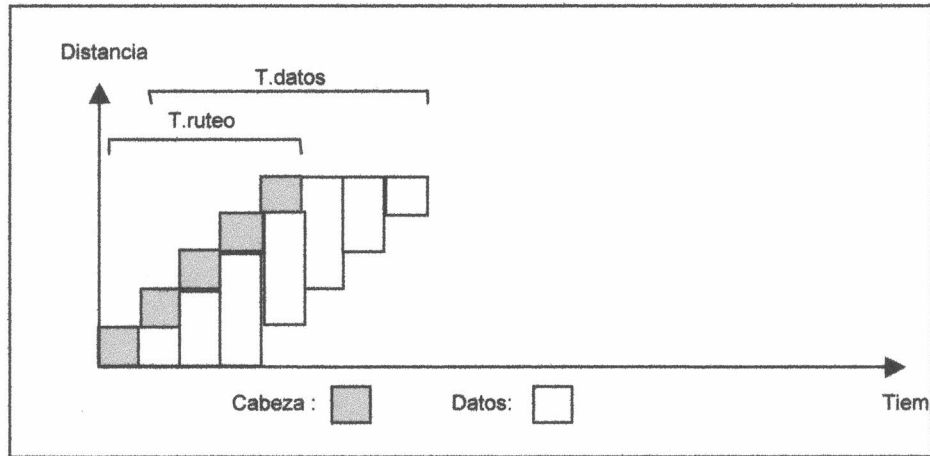
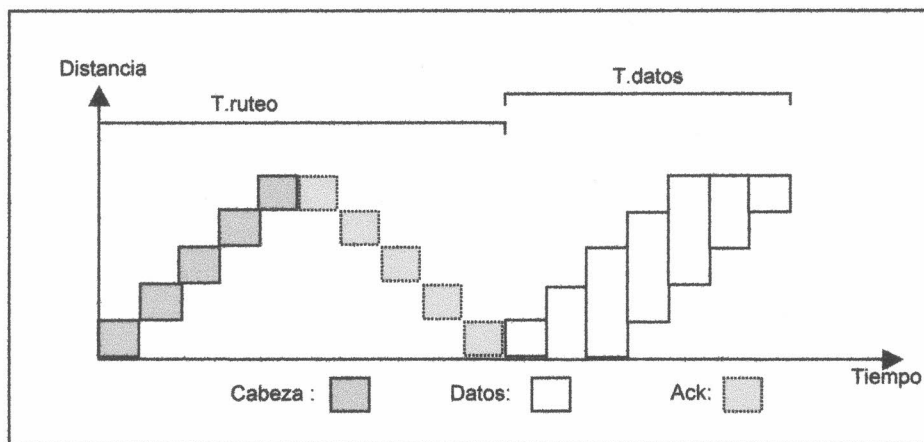
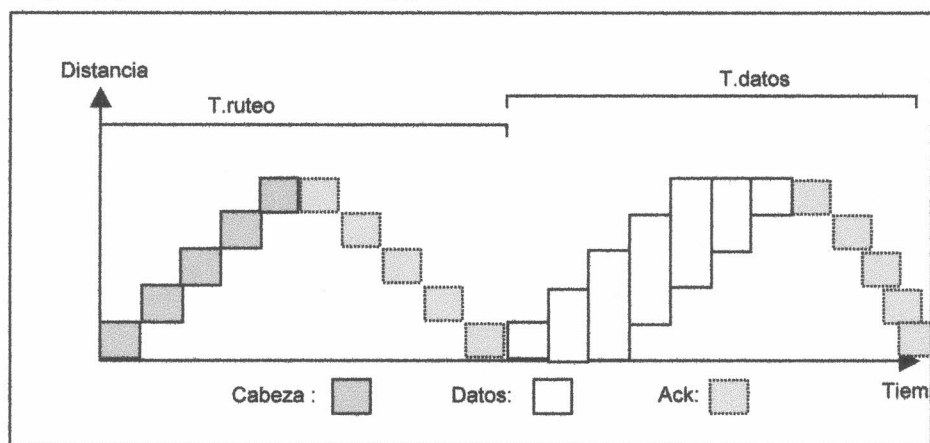
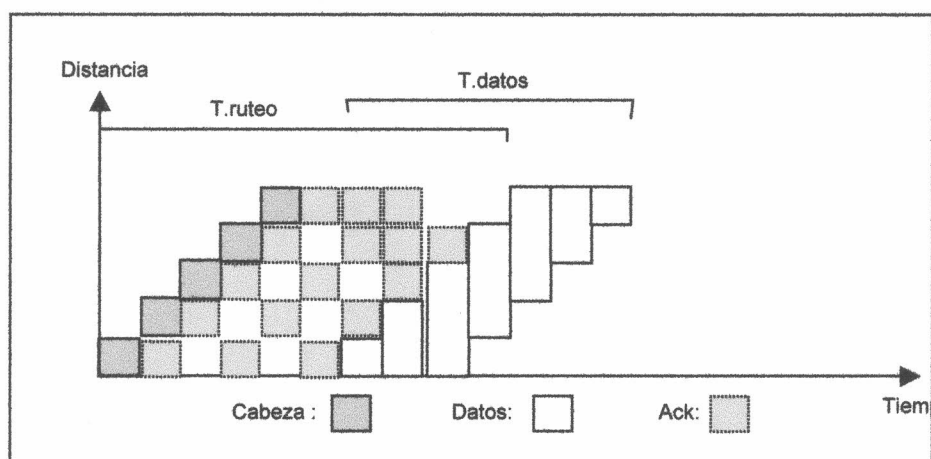


Figura 12.2.1: Diagrama de tiempo espacio para ruteo worm-hole.



Ruteo pipeline circuit switching

Figura 12.2.2: Diagrama de tiempo espacio para ruteo pipeline circuit switching

Acknowledged pipeline circuit switching (APCS)**Figura 12.2.3 Diagrama de tiempo espacio para ruteo acknowledged pipeline circuit switching****Ruteo scouting****Figura 12.2.4: Diagrama de tiempo espacio para ruteo scouting.**

12.3. Cuadros de identificadores ordenados

Nodo	Val	Per(Val)	Per(Resto)	PosIni	AvgW1	AvgW2
00111	111	1	1	2	2	2
10111	111	1	1	2	2	2
01111	111	1	1	1	1.5	1.5
01110	111	1	4	1	1	1
11111	111	1	1	0	1	1
11110	111	1	5	0	0.5	0.5
11100	111	1	5	0	0	0
11101	111	1	5	0	0	0
00110	110	3	3	2	2	2
10110	110	3	3	2	2	2
01101	110	3	3	1	1	1
01100	110	3	4	0	0	1
11000	110	3	5	0	0	0
11001	110	3	4	0	0	0
11010	110	3	5	0	0	0
11011	110	3	3	0	0	0
00101	101	2	2	2	2	2
01010	101	2	2	1	1	1
10101	101	2	2	0	0,5	0,5
01011	101	2	4	0	0	1
10100	101	2	5	0	0	0
00100	100	3	3	2	2	2
01001	100	3	3	1	1	1
01000	100	3	4	0	0	1
10000	100	3	5	0	0	0
10001	100	3	4	0	0	0
10010	100	3	3	0	0	0
10011	100	3	5	0	0	0
00011	011	3	3	2	2	2
00010	010	2	2	2	2	2
00001	001	3	3	2	2	2
00000	000	1	1	0	1	1

Cuadro 12.3.1: Identificadores pertenecientes a $V(2,5)$ ordenados según $<_1$ y $<_3$

Nodo	Val	Per(Val)	Per(Resto)	PosIni	AvgW1	AvgW2
00000	000	1	1	0	1	1
00001	001	3	3	2	2	2
00010	010	2	2	2	2	2
00011	011	3	3	2	2	2
00100	100	3	3	2	2	2
01001	100	3	3	1	1	1
01000	100	3	4	0	0	1
10000	100	3	5	0	0	0
10001	100	3	4	0	0	0
10010	100	3	3	0	0	0
10011	100	3	5	0	0	0
00101	101	2	2	2	2	2
01010	101	2	2	1	1	1
10101	101	2	2	0	0,5	0,5
01011	101	2	4	0	0	1
10100	101	2	5	0	0	0
00110	110	3	3	2	2	2
10110	110	3	3	2	2	2
01101	110	3	3	1	1	1
01100	110	3	4	0	0	1
11000	110	3	5	0	0	0
11001	110	3	4	0	0	0
11010	110	3	5	0	0	0
11011	110	3	3	0	0	0
00111	111	1	1	2	2	2
10111	111	1	1	2	2	2
01111	111	1	1	1	1,5	1,5
01110	111	1	4	1	1	1
11111	111	1	1	0	1	1
11110	111	1	5	0	0,5	0,5
11100	111	1	5	0	0	0
11101	111	1	5	0	0	0

Cuadro 12.3.2: Identificadores pertenecientes a $V(2,5)$ ordenados según $<_2$ y $<_4$

13. Bibliografía

- 1) On Optimal Placements of Processors in Tori Networks. Blaum, Bruck, Pifarre y Sanz. . Ha sido enviado para su aceptación a la conferencia 8° IEEE Symposium on Parallel and Distributed Processing.
- 2) Fully Adaptive Routing in The Torus Network. Felperin, Laffitte, Pifarre y Sanz. En proceso de su adaptación para presentación en Conferencia.
- 3) Routing techniques for massively parallel communication. Sergio A. Felperin, Luis Gravano, Gustavo D. Pifarre, Jorge L.C. Sanz. Proceedings of the IEEE, Special Issue on Massively Parallel Computers, Vol. 79, No. 4, Abril 1991. Trabajo de Pasantía en ESLAI.
- 4) Fully-Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes, and Other Networks: Algorithms and Simulations. Gustavo D. Pifarre, Luis Gravano, Sergio A. Felperin and Jorge L.C. Sanz. IEEE Transactions on Parallel and Distributed System, Vol. 5, No. 3, March 1994.
- 5) Scouting: Fully adaptive, Deadlock-Free Routing in Faulty pipelined Networks. J. Duato, B.V. Dao, P.T. Gaughan and S. Yalamanchili.
- 6) Distribute, Deadlock-Free Routing in Faulty, Pipelined, Direct Interconnection Networks. P.T. Gaughan, Binh V. Dao, S. Yalamanchili and David E. Schimmel.
- 7) Compressionless routing: A framework for adaptive and fault tolerant routing. J.H. Kim, Z. Liu, and A.A. Chien. 21st Annual International Symposium on Computer Architecture, Abril 1994.
- 8) Requirements for Deadlock-Free, adaptive packet routing, R. Cypher y L. Gravano, SIAM . On computing, vol 23, no. 6, pp. 1266-1274, 1994.
- 9) Some complexity results in the design of deadlock-free packet switching networks, SIAM S. Toueg y K. Steiglitz, On computing. Vol 10, pp. 702-712, 1981.
- 10) Minimal, deadlock free routing in Hypercubic and arbitrary networks, Robert Cypher, 7th IEEE symposium on parallel and distributed processing, 1995.
- 11) Introduction to parallel algorithms and architectures: arrays, trees, hypercubes. F. Thomson leighton
- 12) Optimum Broadcasting and Personalized Communication in Hypercubes, Johnsson and Ho, IEEE Transactions on Computers, 1989, September, 1249--1268, c-38(9).
- 13) An Adaptive Communications Protocol for Network Computer. Badr, Gelenter and Podar. Performance Evaluation. 1986. March. 35--51. Vol 6.
- 14) A Performance Comparison of Routing Algorithms for Hierarchical Hypercube Multicomputer Networks. Dandamundi. Proc. 1990 Int. Conf. Parallel Processing. 1990. August. 281--285.
- 15) The Turn Model for Adaptive Routing. Glass and Ni. Proc. of the 19th Annual International Symposium on Computer Architecture. 1992. May. 278--287.
- 16) Performance Analysis of Multiprocessor Mesh Interconnecton Networks with Worm-hole Routing. Adve and Vernon. 1992. June.
- 17) Adaptive Deadlock- and Livelock-Free Routing in the Hypercube Network. Denicolay, Gravano, Pifarre and Sanz. IEEE Transaction on Parallel and Distributed Systems.
- 18) Adaptive Deadlock- and Livelock-Free Routing With All Minimal Paths in Torus Network. Berman, Gravano, Pifarre and Sanz. IEEE
- 19) A theory of worm-hole routing in parallel computers. Raghavan and Upfal. IBM Research.

- 20) A survey of routing techniques in worm-hole networks. Ni and McKinley. Department of Computer Science, Michigan State University.
- 21) Efficient Schemes for Parallel Communication. Upfal. JACM. 31. 3. 1984.
- 22) General Purpose Parallel Architectures. Valiant. Handbook of Theoretical Computer Science.
- 23) Parallel algorithms to set up the Benes Permutation Network. Nassimi and Sahni. IEEE Transactions on Computers. c-31.
- 24) Parallel Permutation and Sorting Algorithms and a New Generalized Connection Network. Nassimi and Sahni. JACM.
- 25) Routing, Merging and Sorting on Parallel Models of Computation Borodin and Hopcroft. Symposium on Theory of Computing.
- 26) An $O(\log N)$ deterministic packet routing scheme. Upfal. 21 Annual ACM-SIGACT Symposium on Theory of Computing.