Tema 3
Redes de Área de Sistema

Nicolás Calvo Cruz
Dpto. de Arquitectura y Tecnología de los Computadores
@ncalvocruz
ncalvocruz@ugr.es

Motivación

- Redes de **comunicación en computadores** paralelos.
- Hoy en día se están **sustituyendo** los **buses por redes** con conexiones **punto a punto** a todos los niveles:
  - Interno al chip
  - A nivel de tarjeta y placa
  - A nivel de chasis o caja
  - LAN y Router IP
- Conocer los algoritmos de **encaminamiento** y la **infraestructura** permite mejorar las **prestaciones**.

Objetivos

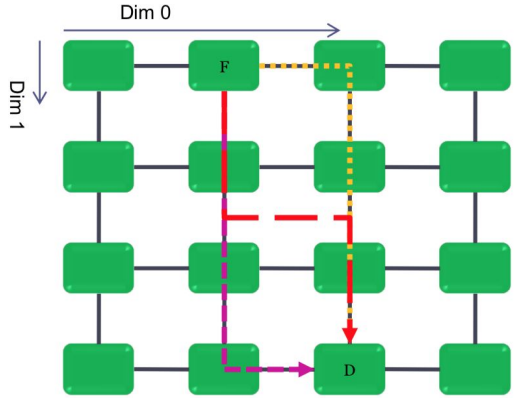
- **Distinguir** entre redes de **altas prestaciones y** redes **estándar**.
- **Conocer** la estructura general de **un conmutador**.
- **Estudiar** las **topologías** y nomenclaturas de las redes **de altas prestaciones**.
- Estudiar los **algoritmos de encaminamiento**.

Índice

5. Enrutamiento

Algoritmos de enrutamiento

5. Enrutamiento



¿Qué hace un algoritmo de enrutamiento?

Un algoritmo de enrutamiento o encaminamiento ha de cumplir 2 funciones:

- Función de encaminamiento: Identificar qué rutas se pueden seguir en la red para llegar desde un cierto origen a un destino.

- Función de selección: Escoger, para cada paquete, qué camino seguir.

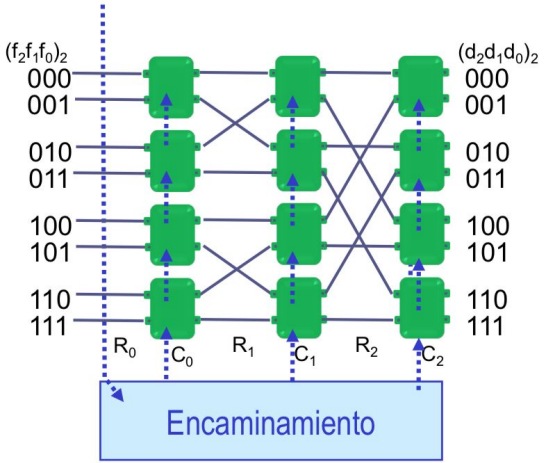
5. Enrutamiento

Diseño y clasificación según:

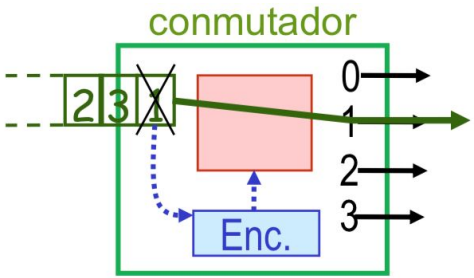
- Funcionalidad
- Decisión de encaminamiento
  - Centralizado
  - Fuente
  - Distribuido
  - Multifase
- Implementación
  - Con tabla de consulta
  - Sin tabla de consulta
- Selección del camino
  - Deterministas
  - Inconscientes
  - Adaptativos
- Canales candidatos y caminos alternativos (función de encaminamiento)

5. Enrutamiento

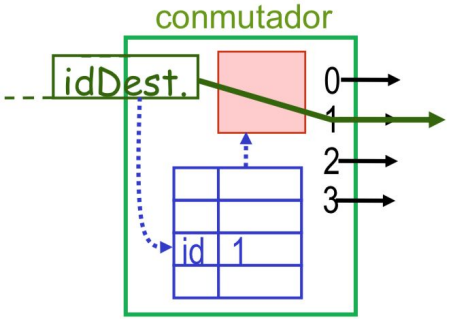
Decisión de enrutamiento:



<u>Centralizado</u>
Circuitería común genera las señales de control



<u>Fuente</u>
Los canales escogidos se fijan en la fuente



<u>Distribuido</u>
Cada conmutador decide localmente (p.ej. consultando una tabla)

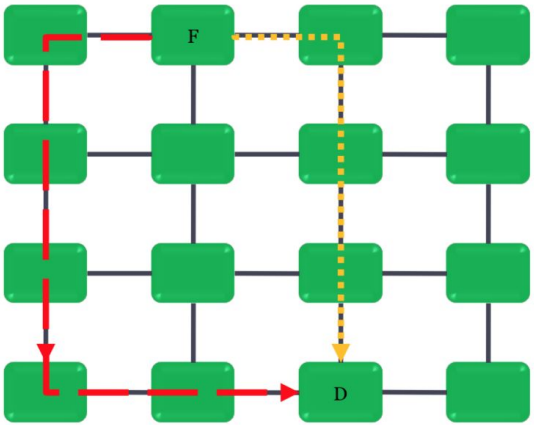
5. Enrutamiento

Selección del camino:

- Adaptativo progresivo o con retroceso (*backtraking*)
- Provechoso/mínimo o mal-enrutado/no-mínimo
- Completamente adaptativo o parcialmente adaptativo

Mínimo

No mínimo

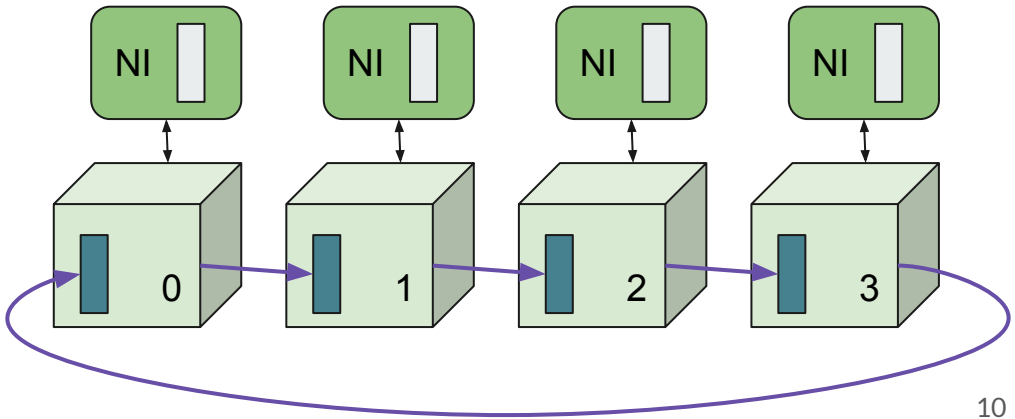


5. Enrutamiento

Enrutamiento en toro unidireccional: Distribuido, sin tablas, determinista

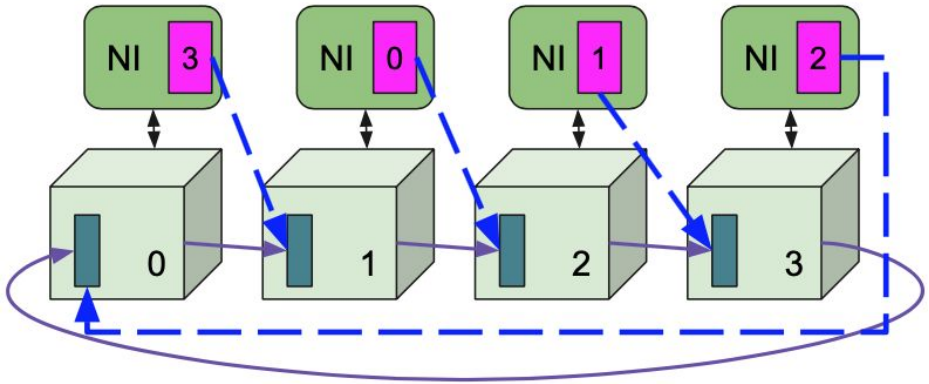
Entrada: Coordenada del nodo actual A y del nodo destino D
Salida: Canal de salida seleccionado (C, I)
Proceso:

```
if (A==D) then Canal = I
else Canal = C
```



5. Enrutamiento: Interbloqueo

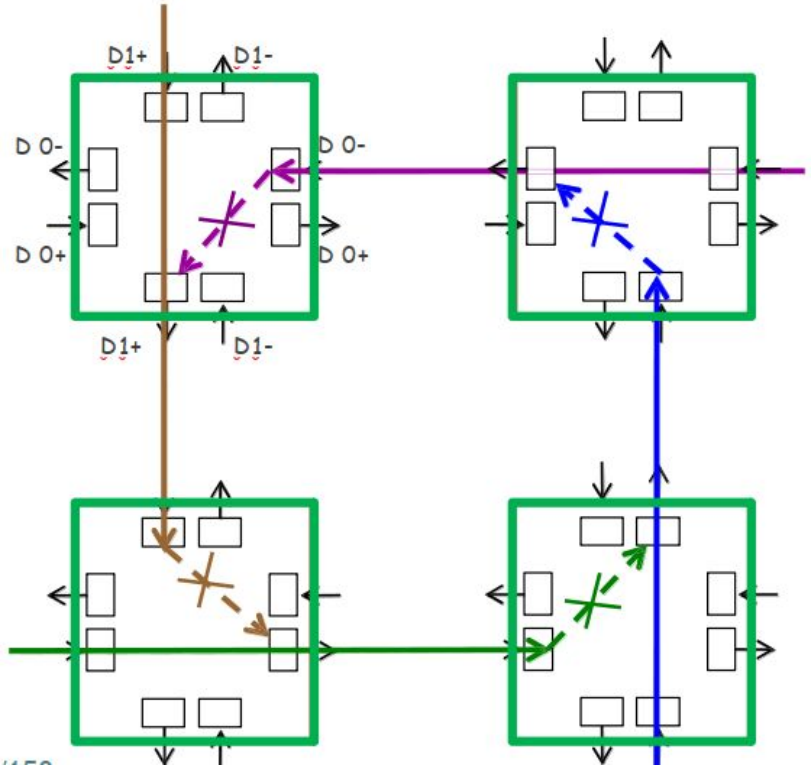
- Conjunto de paquetes bloqueados a la **espera** de que algún hueco (**recurso**) quede libre.
- Depende de:
  - **Topología**: Si no hay ciclos no hay riesgo de interbloqueo
  - Algoritmo de **enrutamiento**: Evitando ciclos, no habrá interbloqueos



5. Enrutamiento: Interbloqueo

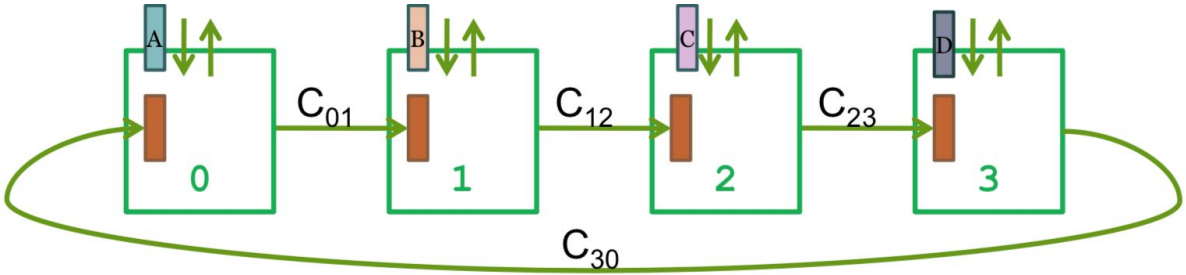
- Siempre que haya **ciclos** en la topología, puede haber **interbloqueos**.

Malla 2D:



5. Enrutamiento: Interbloqueo

- Grafo de dependencias: Grafo que incluye la lógica del algoritmo de enrutamiento y la topología de red para ver si sería posible un interbloqueo.



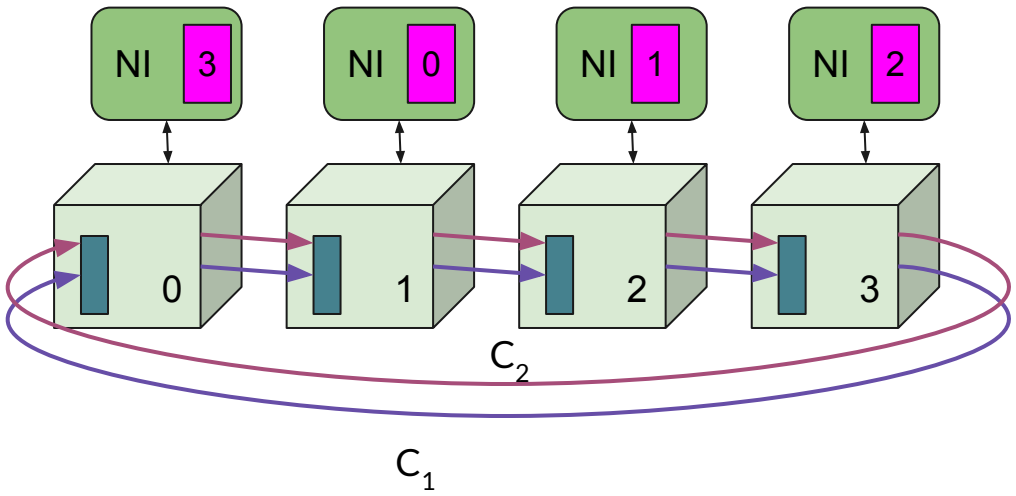
- Hay una dependencia entre el canal C1 y el C2 si un paquete que viene por el canal C1, puede encaminarse por el canal C2.

5. Enrutamiento: Interbloqueo

- **Tratamiento de Interbloqueos**:
  - Se pueden permitir, pero hay que **detectarlos y eliminarlos**.
- Detección:
  - Contador de saltos / tiempo
- Eliminación:
  - Suprimiendo paquetes del ciclo
  - Usando un camino alternativo
  - Introduciendo **canales virtuales**

5. Enrutamiento: Interbloqueo

Canales virtuales:



Entrada: Nodo actual, A y Nodo destino, D

Salida: Canal (C, I)

Proceso:

```
If(D>A) then Canal=C2;
If(D<A) then Canal=C1;
If(D=A) then Canal=I;
```

5. Enrutamiento

Objetivos:

- **Balancear** la **carga** evitando paradas y retrasos (Throughput)
- **Minimizar** los **tiempos** de enrutamiento (Latencia)
- **Minimizar** los **saltos** (Latencia)
- **Evitar ciclos** e interbloqueos (Latencia y Throughput)

Todos los objetivos se contradicen de alguna forma: **priorizar**.

5. Enrutamiento

Opciones:

1) Ignorar el tráfico y usar algoritmos inconscientes, que dan mal balanceo de carga, retrasos o interbloqueos. **Algoritmos inconscientes y/o deterministas.**

2) Tener en cuenta el tráfico y que el algoritmo balancee la carga aunque tarde más en enrutar y aumente la latencia media. **Algoritmos adaptativos**

La solución deberá buscar un **compromiso** entre extremos.

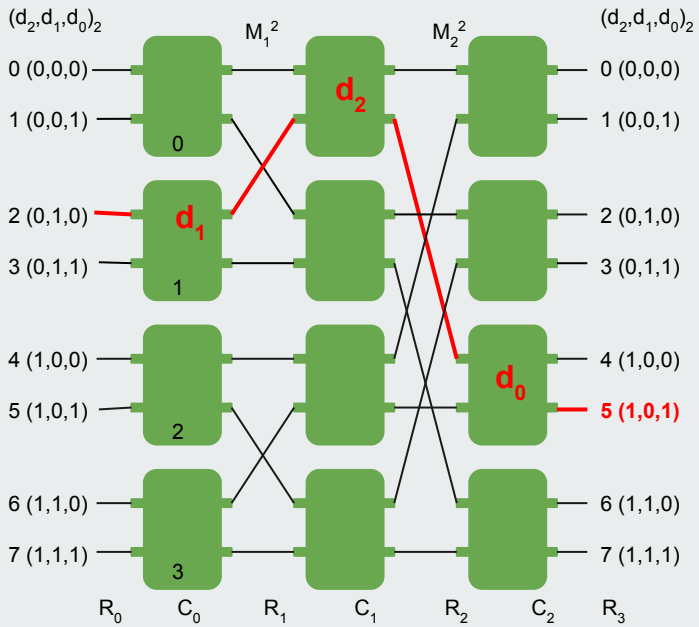
5. Enrutamiento: Alternativas

1) Greedy: Se escoge el camino más corto (en cualquier sentido). Si hay empate, se elige aleatoriamente.

2) Aleatorio uniforme: Se elige de forma uniformemente aleatoria en sentido horario o contrario.
3) Aleatorio no uniforme: Se le da cierta probabilidad al camino más corto, pero el otro también es posible (ponderación).

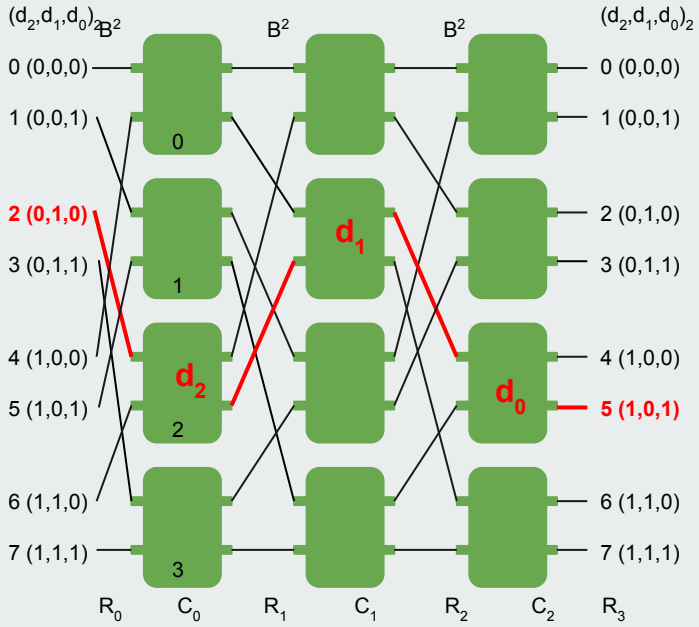
4) Adaptativo: Enviar el paquete por el sentido menos cargado según el recuento de paquetes en ese sentido en los últimos *T* ciclos de reloj.

5. Enrutamiento determinista en redes mariposa (unidireccionales) Enrutamiento-TAG



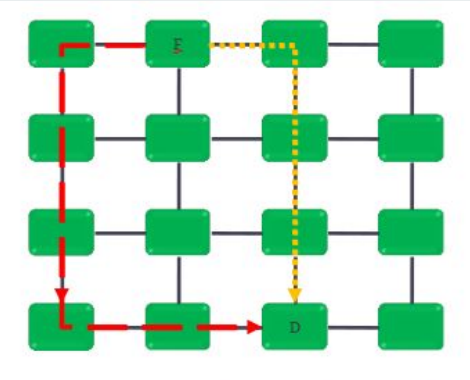
- Cada uno de los dígitos de la **dirección de destino** se utiliza **para seleccionar el puerto de salida de cada etapa** de conmutación de la red.
- El bit d_i controla la etapa i-1 y el bit d_0 la etapa n-1
- Ejemplo: destino 101 (nodo 5)
  - El d_2 =1 controla la etapa 2-1=1
  - El d_1=0 controla la etapa 0
  - El d_0=1 controla la etapa 2
- Ejemplo envío desde 2 hasta 5.
- Este algoritmo **no provoca ciclos porque la topología no tiene ciclos.**

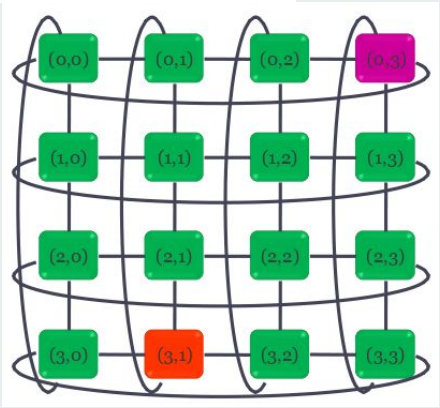
5. Enrutamiento determinista en redes omega (unidireccionales) Enrutamiento-TAG



- Cada uno de los dígitos de la **dirección de destino** se utiliza **para seleccionar el puerto de salida de cada etapa** de conmutación de la red.
- Red omega: El bit d_i controla la etapa n-1-i
- Ejemplo: destino 101 (nodo 5)
  - El d_2 =1 controla la etapa 3-1-2=0
  - El d_1=0 controla la etapa 3-1-1=1
  - El d_0=1 controla la etapa 3-1-0=2
- Ejemplo envío desde 2 hasta 5.
- Este algoritmo **no provoca ciclos porque la topología no tiene ciclos**.

5. Enrutamiento en redes estrictamente ortogonales





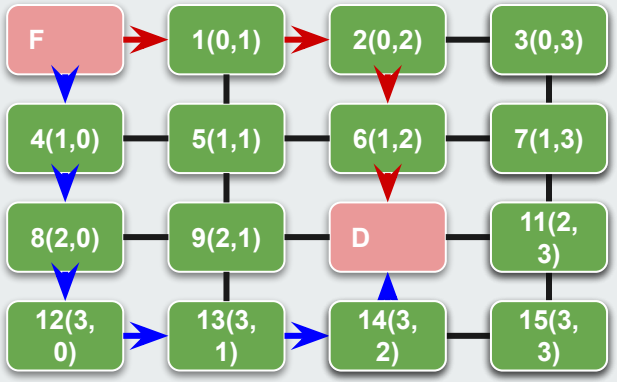
- **Encaminamiento ordenado por dimensión** creciente o decreciente:
  - Con implementación algorítmica igual para todas las entradas
  - Con implementación algorítmica diferente para todas las entradas
  - Algoritmo del intervalo, con implementación por tablas
- **Creciente**: los paquetes se enrutan siguiendo primero los **enlaces de** la **dimensión menor hacia** enlaces de la dimensión **mayor**.
- **Decreciente**: los paquetes se enrutan siguiendo primero los enlaces de la **dimensión mayor a la menor**.

5. Enrutamiento: Ordenado por Dimensión

Características:

- Envío hasta un destino $D=(d_{n-1}, d_{n-2}, \dots, d_2, d_1, d_0)_k$
- Proceso:
  - Calcular las distancias hasta el nodo D.
  - Encaminar anulando las distancias en cada dimensión según un orden creciente o decreciente
- Implementación:
  - Distribuido, sin tabla, determinista (XY)
  - Fuente, sin tabla, determinista (street sign)
  - Distribuido con tabla, determinista (Intervalo)
- Interbloqueos: la lógica del algoritmo **evita los ciclos en mallas e hipercubos**. **Para** redes **toro** se necesitan **canales virtuales** para eliminar los ciclos.

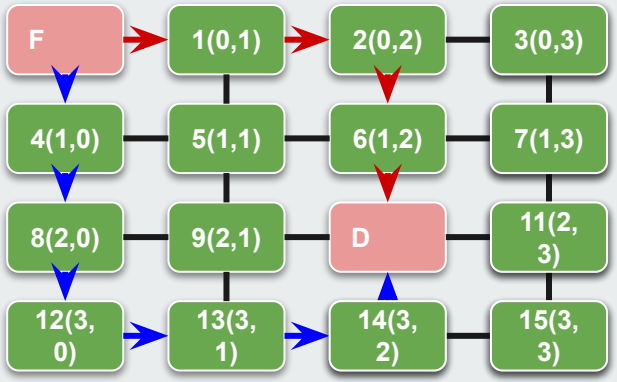
5. Enrutamiento: Ordenado por Dimensión



- Distribuido-sin tabla-determinista, algoritmo **común** a todas las entradas. **Sigue orden creciente**, se anula la distancia en la dimensión 0 y después en la dimensión 1.
- Entrada: Nodo actual, $A(a_1,a_0)$, Nodo destino $D(d_1,d_0)$.
- Salida: Canal de salida, CS = $D_0^+,D_0^-,D_1^+,D_1^-,I$
- Procedimiento:

```
dist0=d0-a0;
dist1=d1-a1;
if(dist0<0) cs=d0-;
if(dist0>0) cs=d0+;
if(dist0==0 && dist1<0) cs=d1-;
if(dist0==0 && dist1>0) cs=d1+;
if(dist0==0 && dist1==0) cs = I;
```

5. Enrutamiento: Ordenado por Dimensión



- Distribuido-sin tabla-determinista, algoritmo **diferente** para cada entrada. **Sigue orden creciente**, se anula la distancia en la dimensión 0 y después en la dimensión 1.
- Entrada: Nodo actual, $A(a_1,a_0)$, Nodo destino $D(d_1,d_0)$.
- Salida: Canal de salida, CS = $D_0^+, D_0^-, D_1^+, D_1^-, I$
- Procedimiento:

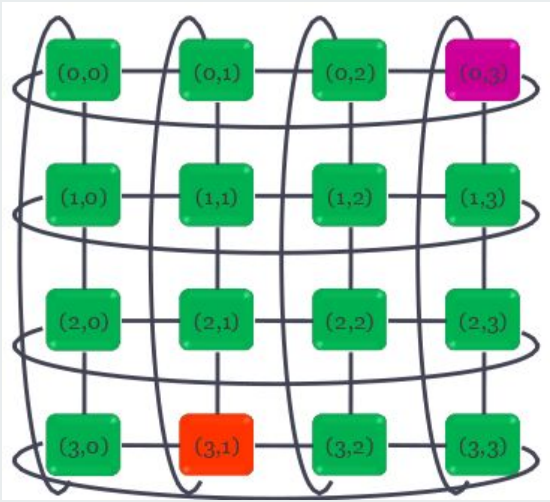
Canal D0-

```
Entrada: distancias (dist1, dist0)
Salida: Canal  cs=(D0-,D1-,D1+, I).
Procedimiento:
if ( dist0≠0 ) { cs = D0- ; dist0--;}
if ( dist0=0 & dist1<0 )
    {cs = D1- ; dist1--;}
if ( dist0=0 & dist1>0 )
   { cs = D1+; ; dist1--;}
if ( dist0=0 & dist1=0 ) cs = I ;
```

Canal D1+

```
Entrada: distancias (dist1)
Salida: Canal  cs=(D1+, I).
Procedimiento:
if ( dist1≠0 )  {cs = D1+ ; dist1--;}
if ( dist1=0 ) cs = I ;
```

5. Enrutamiento: Ordenado por Dimensión

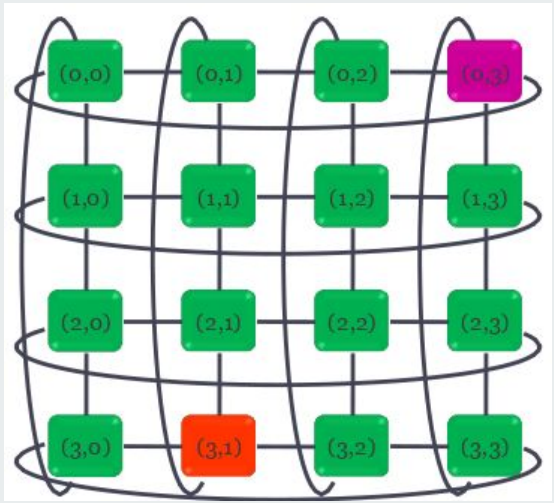


- Distribuido-sin tabla-determinista, algoritmo **igual para cada entrada**. Sigue orden creciente, se anula la distancia en la dimensión 0 y después en la dimensión 1.
- Entrada: Nodo actual, $A(a_1,a_0)$, Nodo destino $D(d_1,d_0)$.
- Salida: Canal de salida, $CS = D_0^+, D_0^-, D_1^+, D_1^-, I$
- Procedimiento:

Entrada: Actual $A = (a_1,a_0)_k$, $D = (d_1,d_0)_k$
Salida: Canal cs=(D0+,D0-,D1-,D1+, I).
Procedimiento:
dist0+=$(d_0 - a_0) \bmod 4$; dist0-=$(a_0 - d_0) \bmod 4$;
dist1+=$(d_1-a_1) \bmod 4$; dist1-=$(a_1-d_1) \bmod 4$;
if (dist0≠0 && dist0+>=dist0-) cs = D0- ;
if (dist0≠0 && dist0+<dist0-) cs = D0+ ;
if (dist0==0){
    if (dist1≠0 && dist1+>=dist1-) cs = D1- ;
    if (dist1≠0 && dist1+<dist1-) cs = D1+ ;
    if (dist1==0) cs = I;
}

- Interbloqueos: NO está libre de **interbloqueos**

5. Enrutamiento: Ordenado por Dimensión

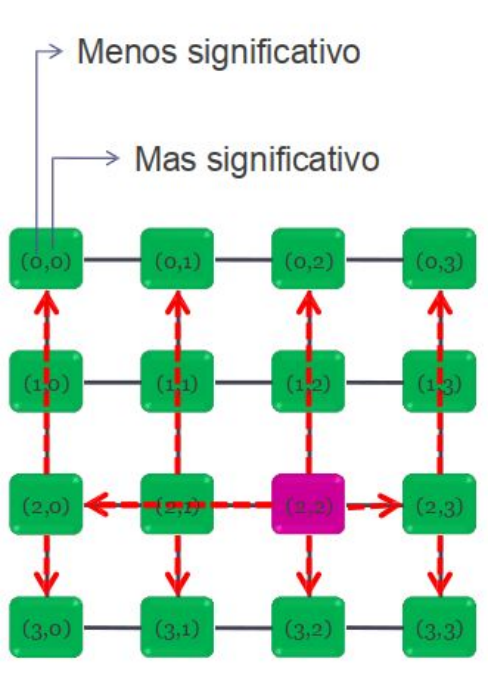


- Si incluimos los canales virtuales:

Entrada: Actual A = $(a_1,a_0)_k$, D = $(d_1,d_0)_k$
Salida: *Canal* cs=(D0+¹,D0-¹,D1-¹,D1+¹,D0+²,D0-²,D1-²,D1+²,I).
Procedimiento:

```
dist0+=(d0-a0) mod 4; dist0-=(a0-d0) mod4;
dist1+=(d1-a1) mod 4; dist1-=(a1-d1) mod4;
if ( dist0≠0 && dist0+>=dist0- )
        if(d0>a0) cs = D0-1 else cs=D0-2;
if ( dist0≠0 && dist0+<dist0- )
        if(d0<a0) cs = D0+1 else cs=D0+2;
if ( dist0==0){
        if ( dist1≠0 && dist1+>=dist1- )
                if(d1<a1) cs = D1-1 else cs=D1-2;
        if ( dist1≠0 && dist1+<dist1- )
        if(d1<a1) cs = D1+1 else cs=D1+2;
        if (dist1==0) cs = I;
}
```

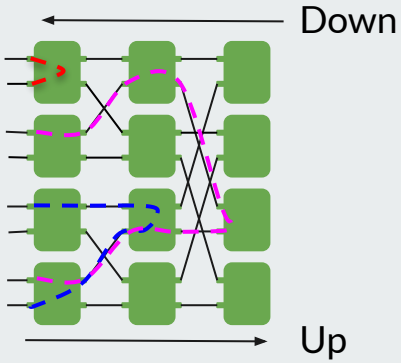
5. Enrutamiento: Ordenado por Dimensión

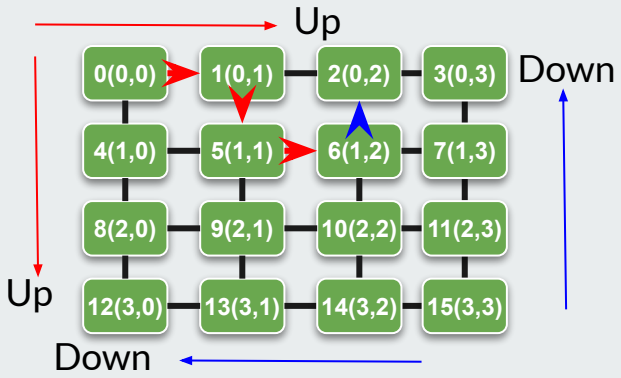


- Algoritmo del Intervalo: Distribuido, con tabla, algoritmo común a todas las entradas. orden creciente
- Se determina para cada nodo, la tabla de envíos indicando para cada enlace el intervalo de nodos a los que da acceso.

Canal	Intervalo
D0+	12-15
D0-	0-7
D1+	11-11
D1-	8-9
I	10-10

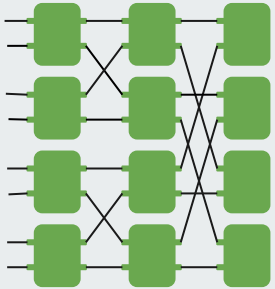
5. Enrutamiento: Up-Down

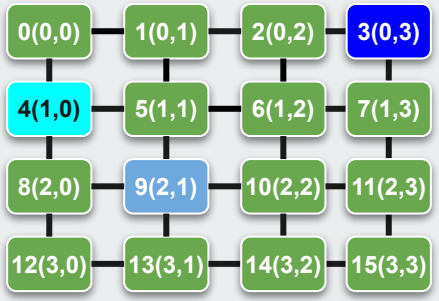




- **Algoritmo Up-Down**
- Se aplica a redes ortogonales y multietapa
- Implementación: Adaptativo o inconsciente.
- Interbloqueos: Evita interbloqueos si la topología no tiene ciclos. Si los tiene, evita los ciclos tomando solo enlaces Up o Down en orden, pero no alternándolos.
- Procedimiento:
  - Se asigna un sentido up o down a todos los enlaces siguiendo siempre el mismo patrón:
    - Redes ortogonales, Up (+) y Down (-)
    - Redes dinámicas bidireccionales Up (raíz) y down (hojas). Ofrece caminos alternativos
  - Se eligen canales **Up hasta que no haya alternativa y se comienzan a coger canales Down**.
- Implementación: Puede ser **adaptativo o** parcialmente adaptativo, **mínimo o no** mínimo.

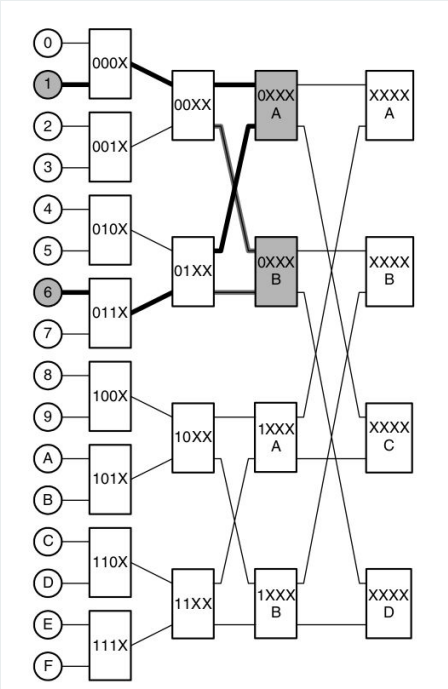
5. Enrutamiento: Valiant





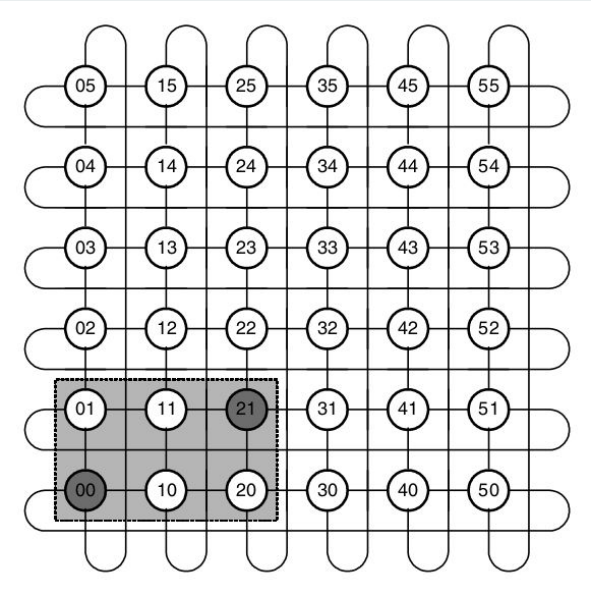
- **Algoritmo De Valiant**
- Se aplica a redes **ortogonales y multietapa**
- Es **inconsciente** y **balancea** la carga **muy bien**. Inconvenientes: **incrementa** la **latencia** media y produce una ocupación de canales que **puede saturar** la red.
- Procedimiento: Tiene **2 fases**:
  a. Se elige desde el nodo origen un paso intermedio cualquiera, elegido de forma aleatoria o con algún criterio.
  b. Desde ese nodo intermedio elegido, se enruta hacia el destino final.
- Implementación: Puede ser **mínimo o no** mínimo

5. Enrutamiento: Valiant mínimo

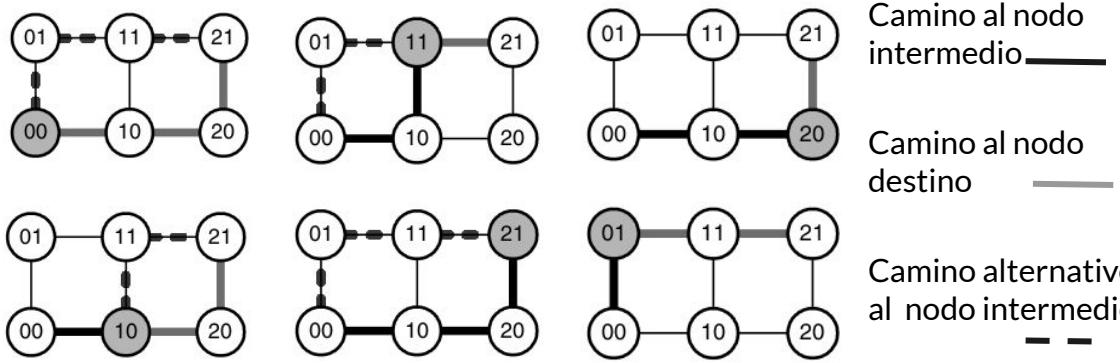


- Valiant, también se puede implementar un algoritmo **inconsciente** y que sea **mínimo**.
- Las **máscaras de cada conmutador fijan el camino** a seguir en la selección de caminos.
- Se selecciona como nodo **intermedio algún conmutador previo al destino** y desde ahí hasta el destino.
- Ej: Para ir **desde el nodo 1 al 6** (011X), se **selecciona** entre los conmutadores que están **antes** en el árbol y que son comunes al 1 y al 6, en este caso el A(0XXX) o el B(0XXX) y ya **desde ahí al destino** usando los bits de la dirección del nodo destino.

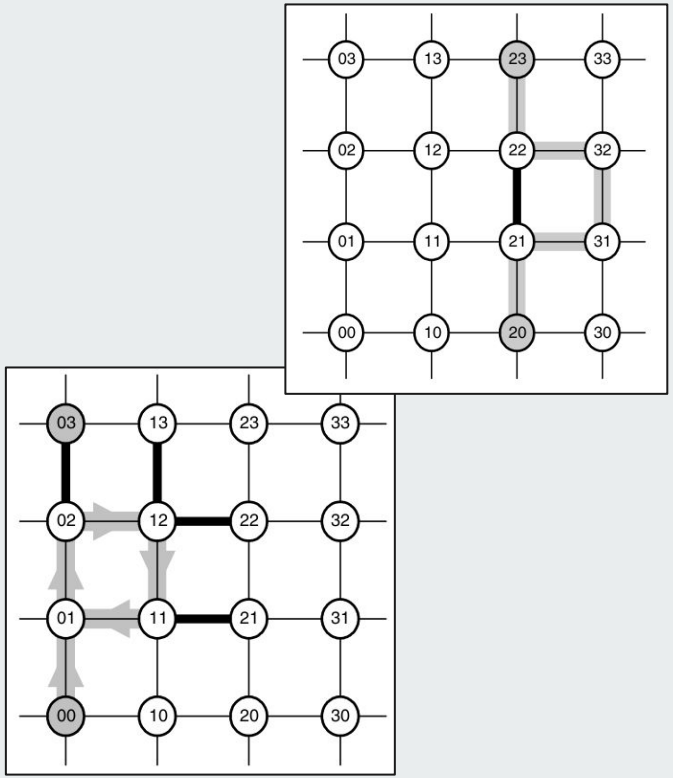
5. Enrutamiento: Valiant mínimo



- Para el caso de redes ortogonales:
  - Se elige el nodo **intermedio** de forma que pertenezca al conjunto de nodos encerrados **en el cuadrado mínimo que incluye al** nodo **origen y** al nodo **destino**. Para ello se calculan las distancias en cada dimensión entre origen y destino y la suma de dichas distancias será la dimensión de la subred entre la que hay que elegir el nodo intermedio.
  - **Se enruta hacia ese nodo** seleccionado **y después hacia el nodo destino**. **Todas** las opciones serán **caminos mínimos** posibles.



5. Enrutamiento: Adaptativo



- Cualquiera de estos últimos algoritmos se puede **implementar de forma adaptativa si**, **para seleccionar** de entre los caminos alternativos, **se tiene en cuenta el estado de la red**.
- Adaptación **global**, si tenemos información global de la red
- Adaptación **local**, si se utiliza sólo información del entorno más cercano a donde se está enrutando.

- Criterios que se pueden consultar para saber el estado de la red
  - **Ocupación de los buffers de salida**
  - **Ocupación media en los últimos T ciclos** de los buffers de salida
  - **Backpressure**, que es el **reflejo que pasado un tiempo** se nota **en los buffers anteriores a un enlace saturado**. Si los buffers son grandes, esta presión tarda en notarse. Si los buffers son pequeños, se nota pronto hacia atrás.

- **Ejemplo** de algoritmo totalmente adaptativo: **Para cada paquete, se calcula el camino mínimo a su destino** y **si el enlace tiene una ocupación menor a** un **umbral**, se enruta en ese salto por ahí. Si no, **se envía por el** canal **más vacío**, camino **mínimo o no**. Podría incluso **volverse con retroceso y ciclos**.

Índice

6. Técnicas de conmutación

6. Técnicas de conmutación

- Determina **cómo la información viaja** de un nodo fuente a otro destino de la red.
    - Almacenamiento y Reenvío:
        - Usado inicialmente en multicomputadores y redes WAN
    - Vermiforme (*Wormhole*):
        - Inicialmente destinado a multicomputadores
    - Virtual Cut-Through:
        - Propuesto para cualquier tipo de red
    - Conmutación de circuitos:
        - Originalmente propuesto en redes de telefonía

6. Técnicas de conmutación: Almacenamiento y reenvío

En una red con **conmutación** de **almacenamiento y reenvío** o ***de paquetes***:

- El conmutador espera a recibir (almacenar) la unidad de transferencia entre interfaces (paquete) antes de enrutar, y tras esto, se reenvía.
- En todo momento el paquete ocupa, como mucho, un canal de la red.
- El almacenamiento del conmutador debe permitir guardar todo el paquete.
- **Buffer y enlace** se asignan **a nivel de paquete**.

6. Técnicas de conmutación: Vermiforme (*Wormhole*)

En una red con **conmutación vermiforme** o ***wormhole***:

- El **enrutamiento** del conmutador se ejecuta al recibir la cabecera de la unidad de transferencia entre interfaces, **sin esperar al resto,** que seguirán a la cabecera en orden, según vayan llegando.
- En cada momento, **un paquete** se está transfiriendo **por múltiples canales en paralelo**. El camino se divide en **etapas**, de forma que los flits de un mismo paquete pueden estar ocupando simultáneamente varias etapas, que se separan mediante almacenamientos. Más etapas, más recursos usados por paquete y menor latencia.
- **Buffer y enlace** se asignan **a nivel de flit**.

6. Técnicas de conmutación: Virtual Cut-through

En una red con **conmutación *Virtual Cut-Through***:

- El **enrutamiento** es **en cuanto llega la cabecera**. La unidad de transferencia entre interfaces puede romperse (*cut-through*).

- Si no hay bloqueo, un paquete puede ir en paralelo por varios canales. Aunque si lo hay, **debe poder almacenarse en el conmutador**.

- Si lo mensajes no tienen un tamaño máximo, se deben dividir en unidades más pequeñas: Segmenta el camino en etapas enviando por el camino segmentado de un paquete como en vermiforme, aunque asignando **buffer y enlaces a nivel de paquete como en almacenamiento y reenvío**.

6. Técnicas de conmutación: Circuitos

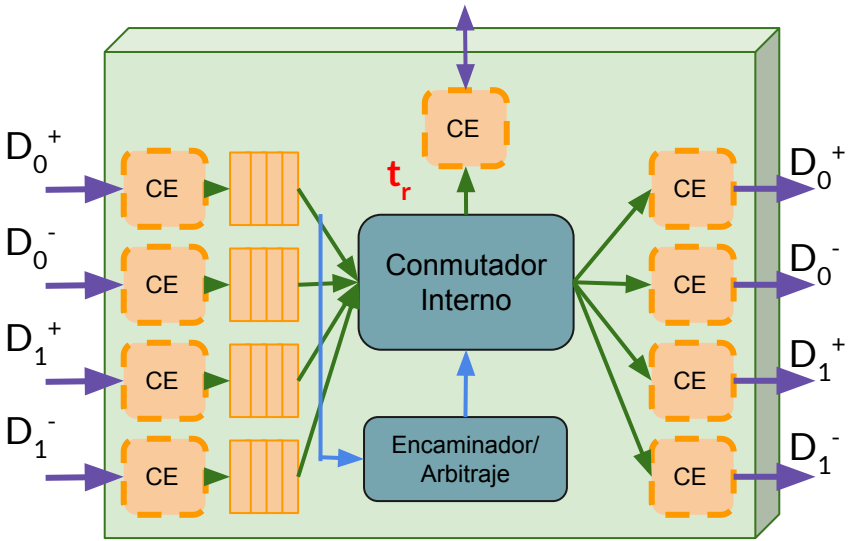
La **conmutación de circuitos** se hereda de los sistemas telefónicos clásicos:

- Reserva el camino entre interfaces que se usa en la transferencia:
  - Completo: Se manda una **sonda** (flit cabecera) que va reservando, y al llegar a destino, éste envía un ACK. Con el ACK en fuente, comienza un envío por canal de comunicación continuo (sin almacenamientos).
  - Segmentado/Etapas: El envío entre extremos es como en conmutación vermiforme, haciendo que los flits de un paquete ocupen varias etapas a la vez. El almacenamiento asociado a cada canal en los conmutadores es de un flit.

- La **cola** del paquete **va liberando** el camino reservado.

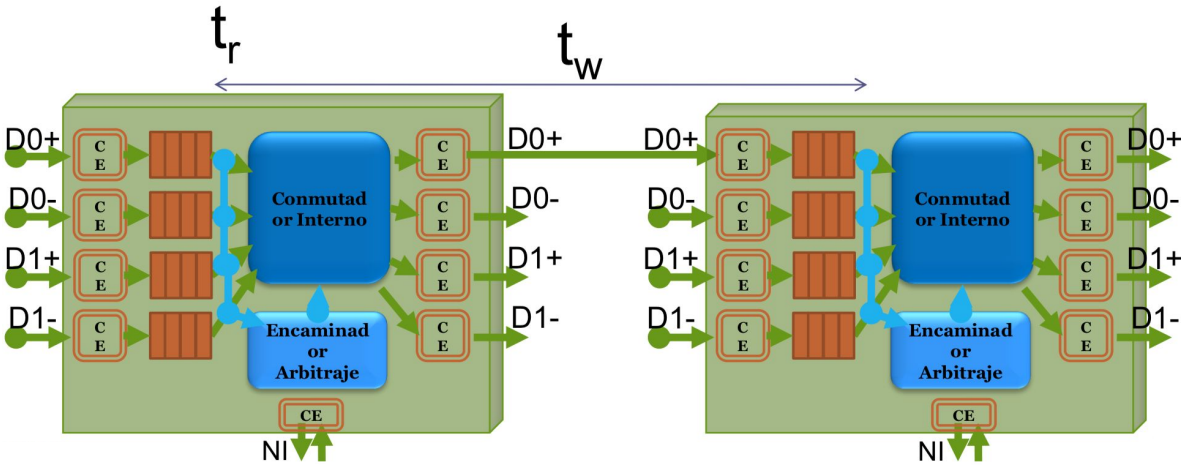
6. Técnicas de conmutación. Ejemplo práctico

- Situación base:
  - 1 flit = 1 phit = w bits
  - Tamaño de la cabecera = 1 flit
  - Tamaño de los datos L, L/w= 3 flits
  - Usaremos conmutadores sin buffer a la salida
  - D = número de parejas conmutador-enlace que unen fuente y destino.
  - Asumimos siempre un camino sin retenciones



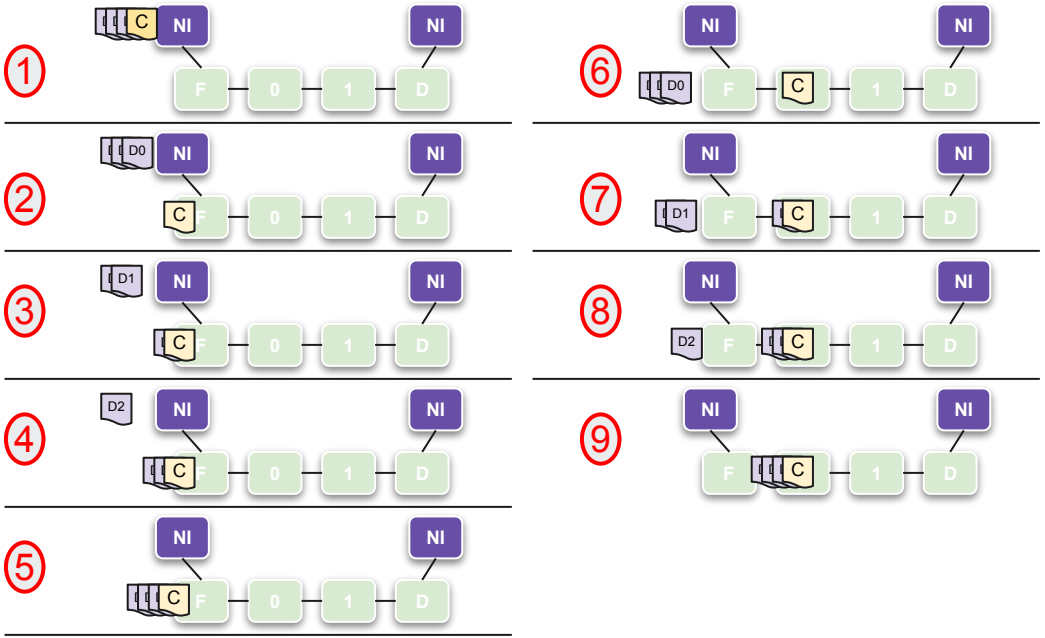
6. Técnicas de conmutación. Ejemplo práctico

- t_w: tiempo necesario para transportar w bits en una etapa conmutador-enlace.
- t_r: tiempo de enrutamiento.



6. Técnicas de conmutación. Almacenamiento y Reenvío

$$t_{AR} = D\left[t_r + t_w\left(\left\lceil\frac{L}{w}\right\rceil + 1\right)\right] + t_w\left(\left\lceil\frac{L}{w}\right\rceil + 1\right)$$

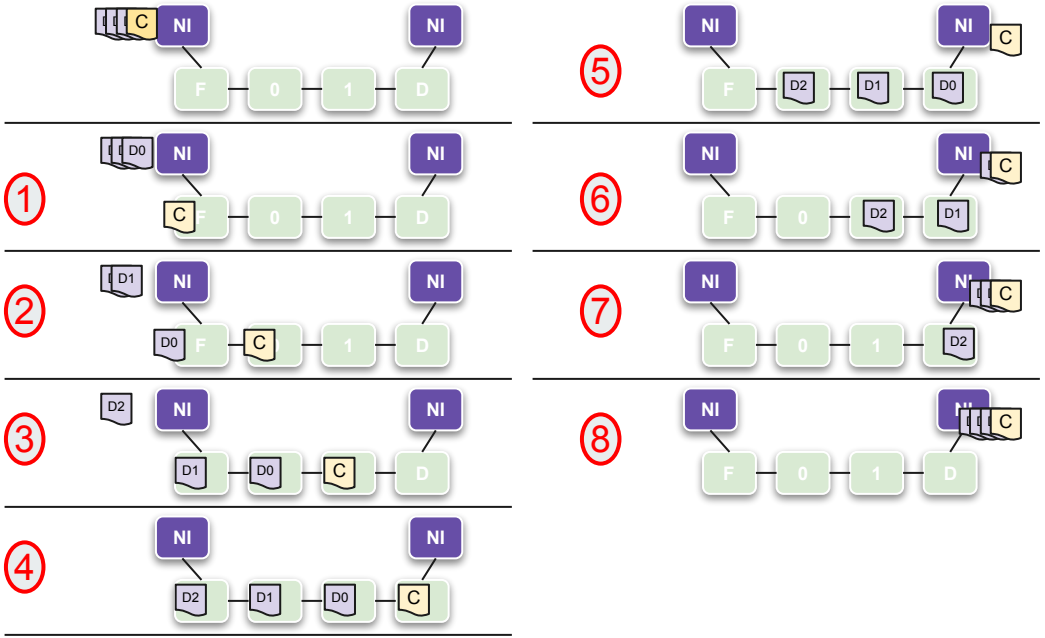


T	R	T
1		
2		t_w
3		t_w
4		t_w
5		t_w
6	t_r	t_w
7		t_w
8		t_w
9		t_w

Un **paquete** ocupará **sólo un enlace**, por lo que los retrasos provocados no serán nada más que los que haya esperando en ese enlace. La incidencia en el ancho de banda global es mínima.
Almacenamientos de como mínimo lo que ocupe un paquete.

6. Técnicas de conmutación. Vermiforme (Wormhole)

$$t_V = t_c + t_r = D\left[t_r + t_w\right] + t_w\left(\frac{L}{W}\right)$$



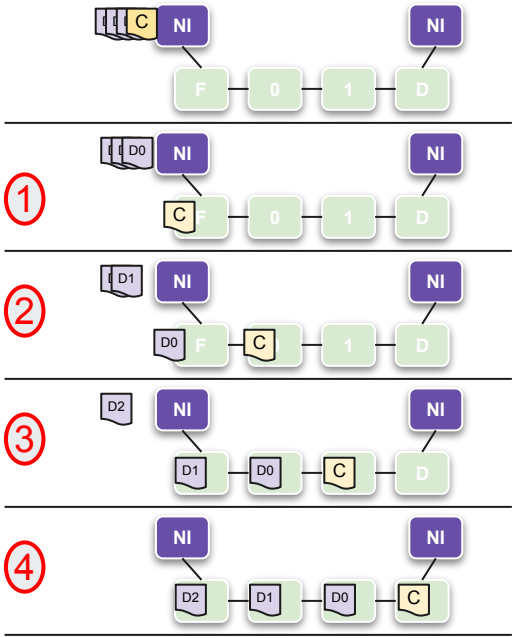
T	R	T
1		t_w
2	t_r	t_w
3	t_r	t_w
4	t_r	t_w
5	t_r	t_w
6		t_w
7		t_w
8		t_w

Un **paquete** ocupará **tantos enlaces** como necesite por su longitud, por lo que los **retrasos afectarán a varios conmutadores**.

Almacenamientos de como mínimo lo que ocupe un flit.
Si el paquete se retrasa, se queda extendido.
La incidencia en el ancho de banda global es significativa.

6. Técnicas de conmutación. Virtual Cut-Through

$$t_V = t_c + t_r = D\left[t_r + t_w\right] + t_w\left(\frac{L}{W}\right)$$

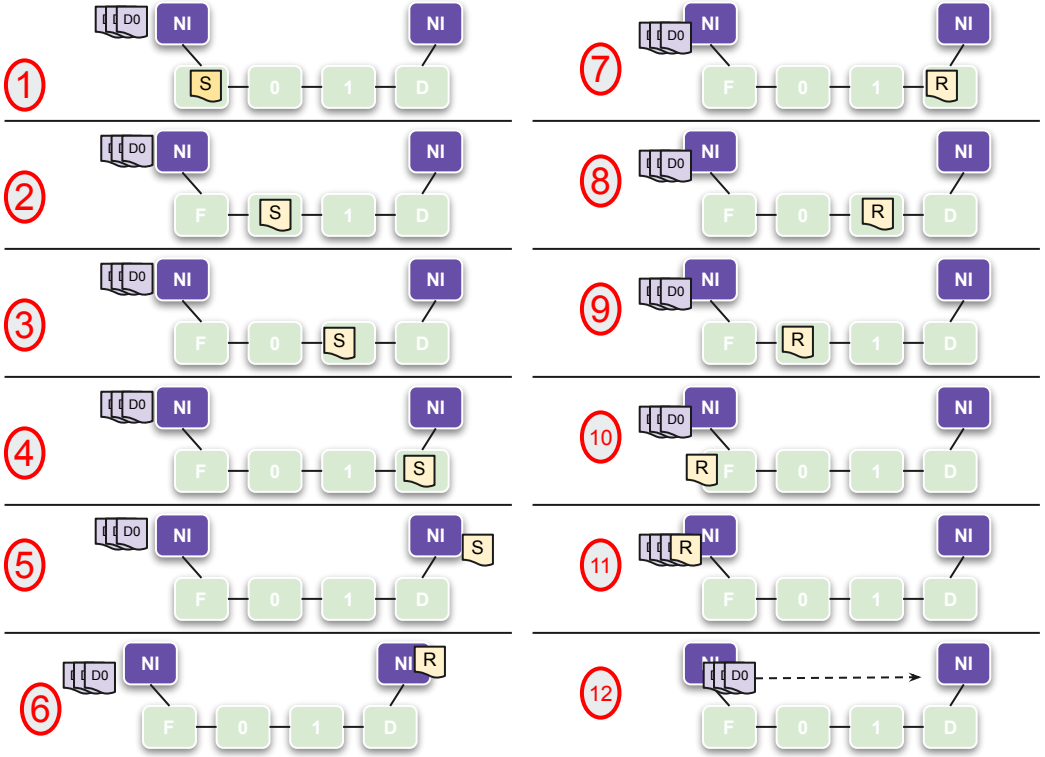


T	R	T
1		t_w
2	t_r	t_w
3	t_r	t_w
4	t_r	t_w
5	t_r	t_w
6		t_w
7		t_w
8		t_w

Se comporta **igual que Vermiforme, pero en caso de bloqueo o parada, el paquete se agrupa, dejando los enlaces libres.** La incidencia en el ancho de banda global es menor que en Vermiforme, y comparable a Almacenamiento y Reenvío.

6. Técnicas de conmutación. Com. de circuitos

$$t_V = t_s + t_r + t_d = D\left[t_r + t_w\right] + t_w + D \cdot t_w + t_w + B_{Canal}\left(\frac{L}{W}\right)$$



T	R	T
1		t_w
2	t_r	t_w
3	t_r	t_w
4	t_r	t_w
5	t_r	t_w
6		t_w
7		t_w
8		t_w
9		t_w
10		t_w
11		t_w
12	B_{Canal}*3	

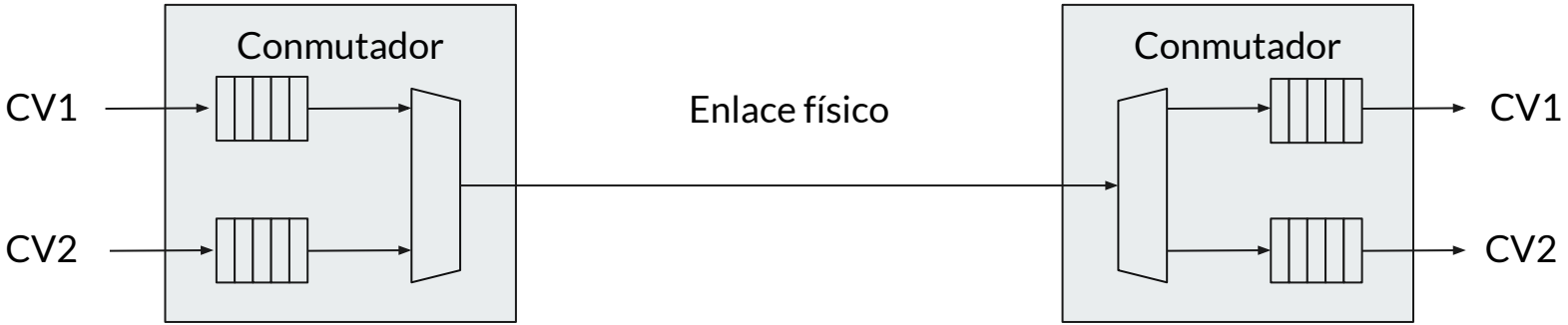
Se establece un **canal** y se **usa todo el ancho de banda** disponible.

El **almacenamiento mínimo** requerido es el **tamaño de la sonda**.

Su impacto en el ancho de banda global afecta a todos los canales reservados desde que se envía el reconocimiento hasta la llegada de todos los datos

6. Conmutación: Canales virtuales

- Si se llena el buffer de un enlace de entrada, el canal está ocupado aunque no transfiera. Este problema se evita asociando varios **canales virtuales** a un enlace físico, pudiendo así **multiplexar en el tiempo el uso del canal**.
- Los canales comparten enlace a nivel de flit: Los flits de distintos paquetes se envían por el enlace mezclados.



6. Conmutación: Canales virtuales

- ¿2 canales virtuales sobre uno físico? Velocidad = velocidad/2 ...
- En una red con conmutación vermiforme se reduce la probabilidad de:
  - Bloqueo de un paquete, aumentando el ancho de banda.
  - Paquetes pequeños esperen a paquetes grandes.
- Usar canales virtuales independiza la asignación de enlace de la asignación de almacenamiento.
- Si se bloquea un canal virtual, el enlace físico no se inutiliza, lo usan los otros. De esta forma, aumenta el tráfico que puede circular.

6. Conmutación: Canales virtuales

- Usar canales virtuales necesita aumentar el hardware asociado a cada enlace, pues un canal virtual necesita:
  - Almacenamiento
  - Señales de control de flujo
  - Hardware de multiplexado y demultiplexado
  - Arbitraje virtuales - físico

- Los canales virtuales mejoran el ancho de banda y la latencia global, pero hasta un límite, pues se añade **retardo** (arbitraje, multiplexores…), que **depende del número de canales virtuales**.

Índice

Ejemplos para lectura

- Louis Jenkings. Networks for High-Performance Computing
- Jesus Escudero-Sahuquillo & Co. Efficient and Cost-Effective Hybrid Congestion Control for HPC Interconnection Networks
- Francisco J. Andújar-Muñoz &Co. N-Dimensional Twin Torus Topology
- Benito M. & Co. Analysis and improvement of Valiant routing in low-diameter networks.


Gracias.