

upna

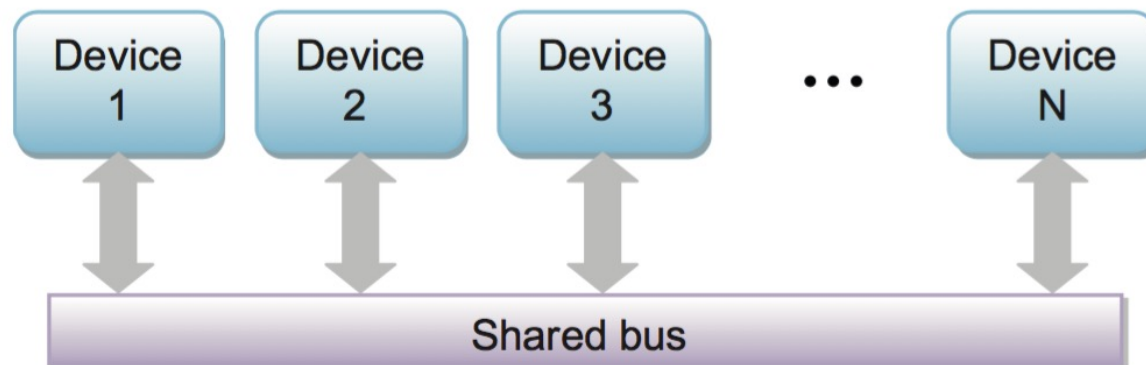
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

Arquitectura de conmutadores

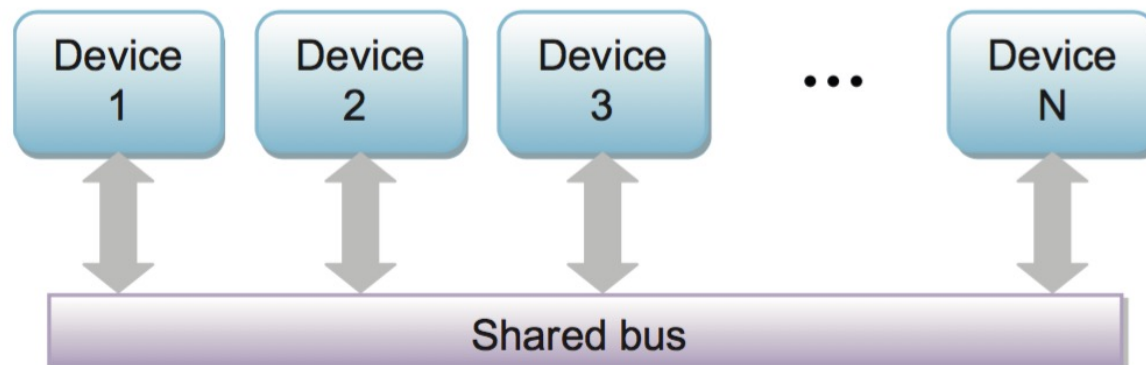
Shared Bus

- Típico en primeros sistemas con múltiples procesadores
- Cuando hay una gran cantidad de dispositivos no es económico una malla
- Un bus interconecta en modo compartido a los dispositivos
- Más económico, sencillo añadir dispositivos
- Requiere resolver la contienda y colisiones
- Ejemplos: Ethernet (CSMA/CD), PCI, I²C, etc
- Un bus puede consistir en múltiples señales en paralelo



Shared Bus

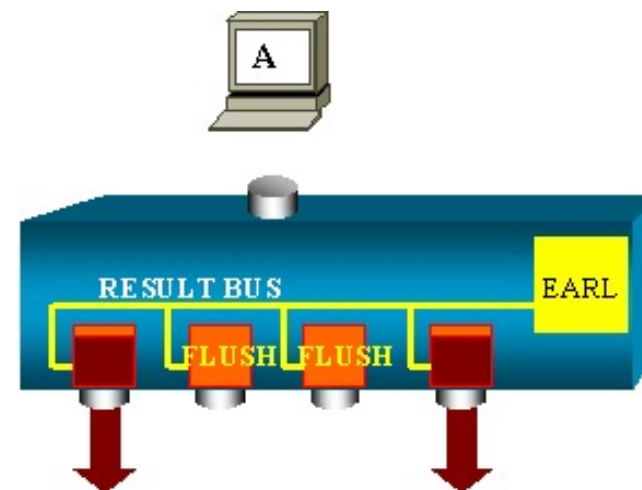
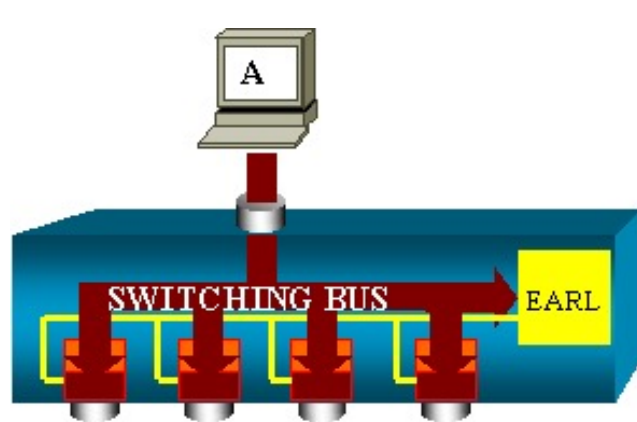
- La capacidad del bus está compartida, aunque cada dispositivo debe poder enviar a la máxima velocidad del mismo
- Se puede aumentar la velocidad aumentando la anchura del bus o su frecuencia
- Aumentar la anchura aumenta el número de pines
- Aumentar la frecuencia aumenta la posibilidad de que se desfasen y de que se interfieran
- Para altas velocidades la industria se ha movido hacia interfaces serie



Ejemplo

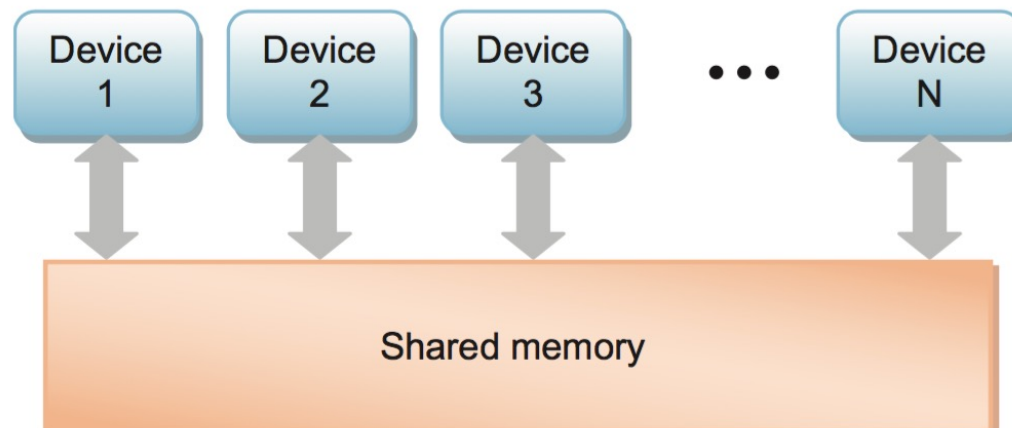
Catalyst 5500/5000 y 6500/6000

- El paquete recibido se transmite por el bus interno
- Todos los puertos almacenan el paquete en buffers internos suyos
- La Encoded Address Recognition Logic (EARL) también recibe la cabecera
- La EARL calcula un resultado que envía a los puertos por el bus
- Este resultado permite a los puertos saber si deben enviar o descartar el paquete



Shared Memory

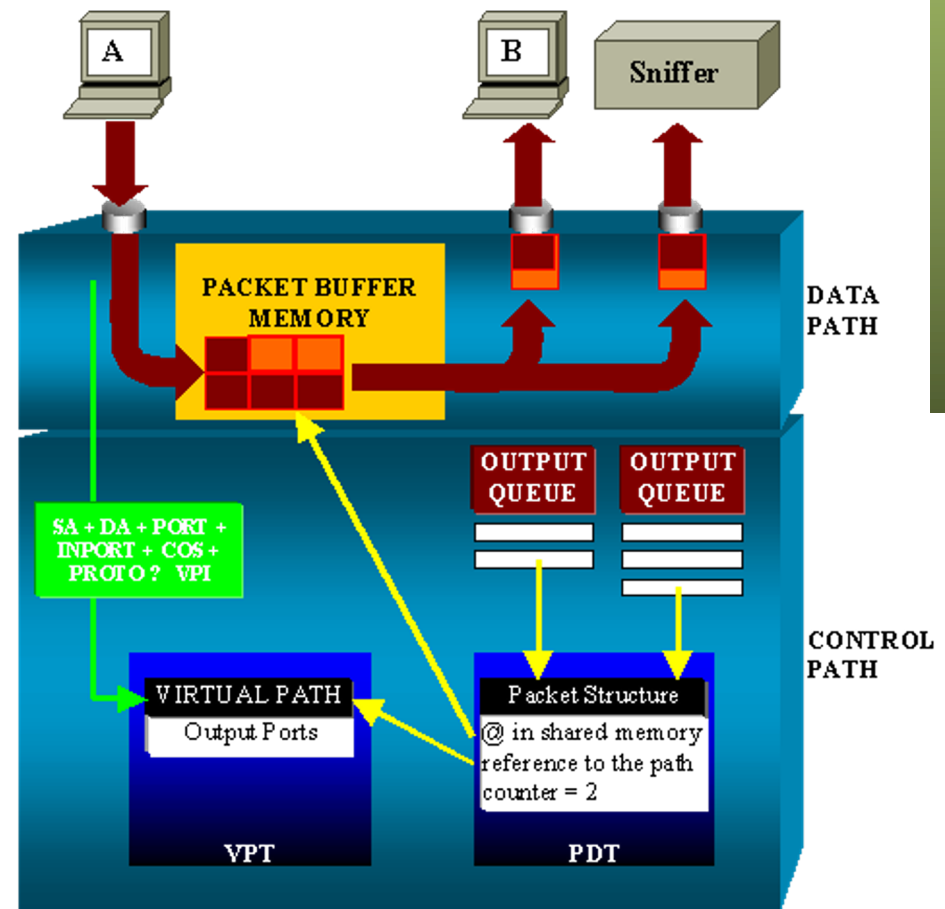
- Intercambian la información leyendo y escribiendo en una memoria compartida
- Las conexiones a la memoria son punto-a-punto y cada dispositivo puede emplear todo el ancho de banda de ellos
- Si hay N dispositivos con interfaces a una tasa R necesitamos una memoria capaz de escribir a una tasa NxR
- Con puertos de dispositivo full-duplex necesitamos ese mismo número de lecturas, así que debe soportar una tasa $2xNxR$
- Si la memoria es un cuello de botella necesitamos un árbitro para el acceso a la misma
- También requiere una memoria con una gran cantidad de pines



Ejemplo

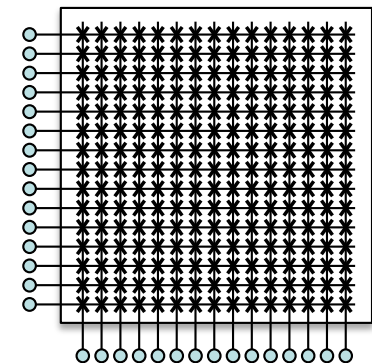
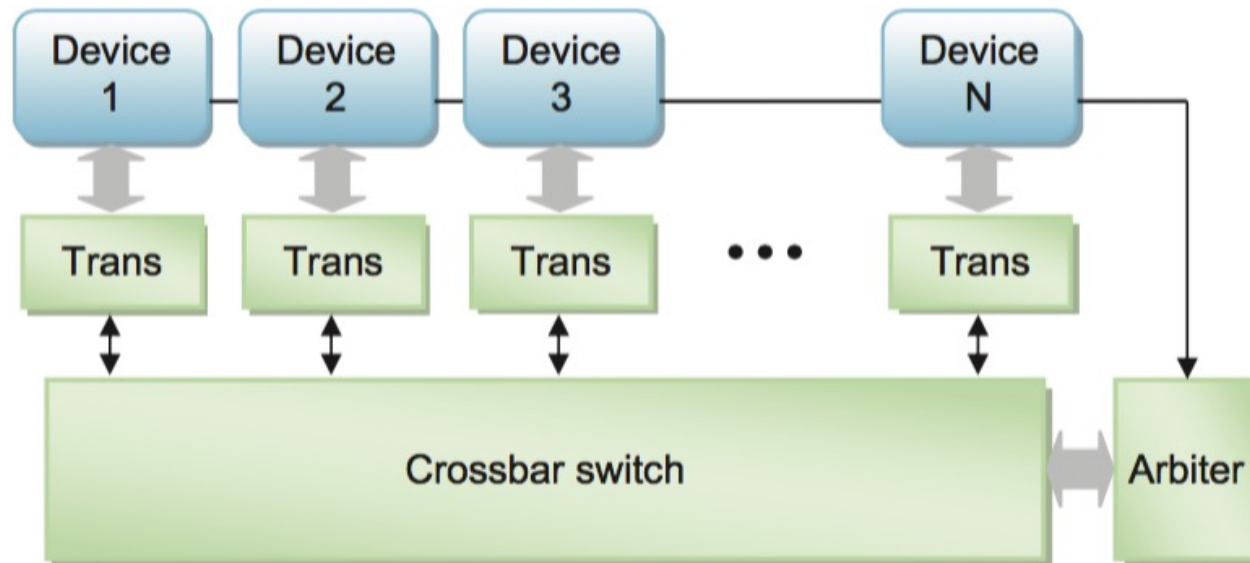
Catalyst 4500/4000

- Shared memory almacena paquete
- Se crea una estructura en la Packet Descriptor Table (PDT)
- Calcula hash con: srcaddr, dstaddr, VID, proto, inputPort, CoS
- Localiza con él entrada en la Virtual Path Table (VPT)
- Eso da los puertos de salida
- La estructura se añade a las colas de los puertos de salida
- Cuando se ha enviado por todos esos puertos se libera el buffer



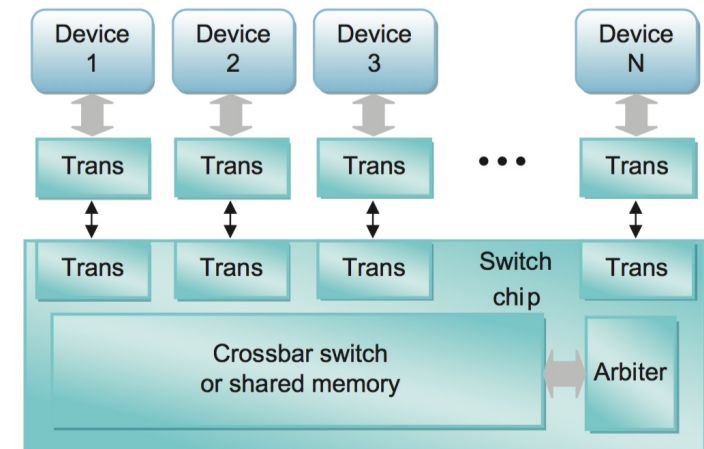
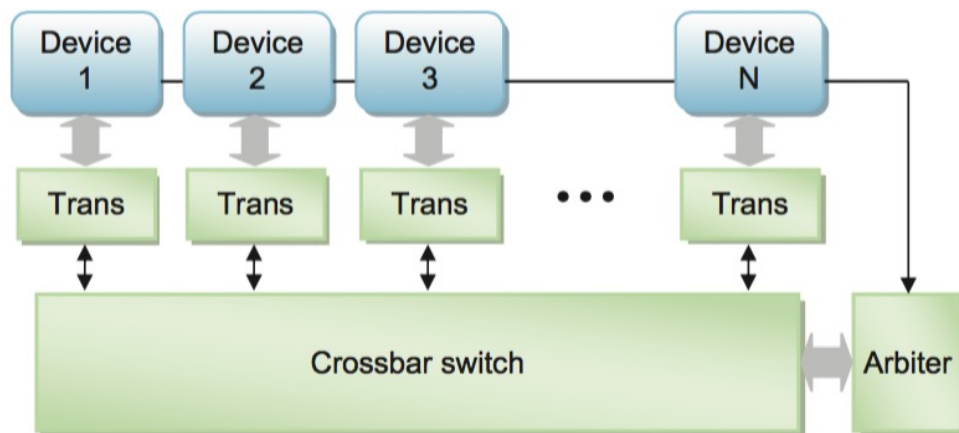
Crossbar Switch

- En los 90s surgen los primeros SerDes en el rango de 1Gbps
- Esto permite emplear chips de conmutación crossbar asíncrona
- Los datos se serializan y entran por un puerto del crossbar
- Un árbitro se encarga de programar el crossbar
- Puede unir cualquier entrada con cualquier salida (una a la vez)
- No hay bloqueo interno
- No hay re-sincronización, se des-serializa a la salida



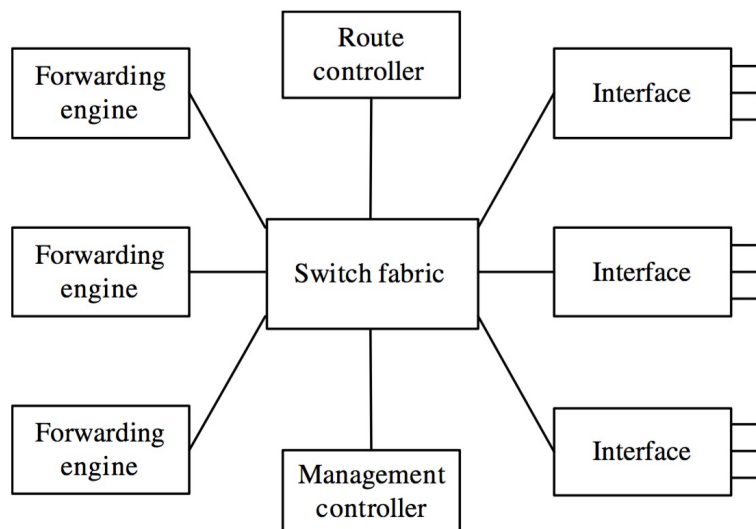
Crossbar Switch

- Suelen segmentarse los datos en “celdas” de cara a gestionar la configuración temporal del crossbar
- Eso implica que se requieran memorias y lógica de segmentación y reensamblado (SAR)
- Los buffers también son necesarios en caso de bloqueo externo
- El tiempo de enganche del PLL del receptor a la señal del transmisor es un tiempo desaprovechado
- A finales de los 90s se mejoran estas arquitecturas con una conmutación síncrona



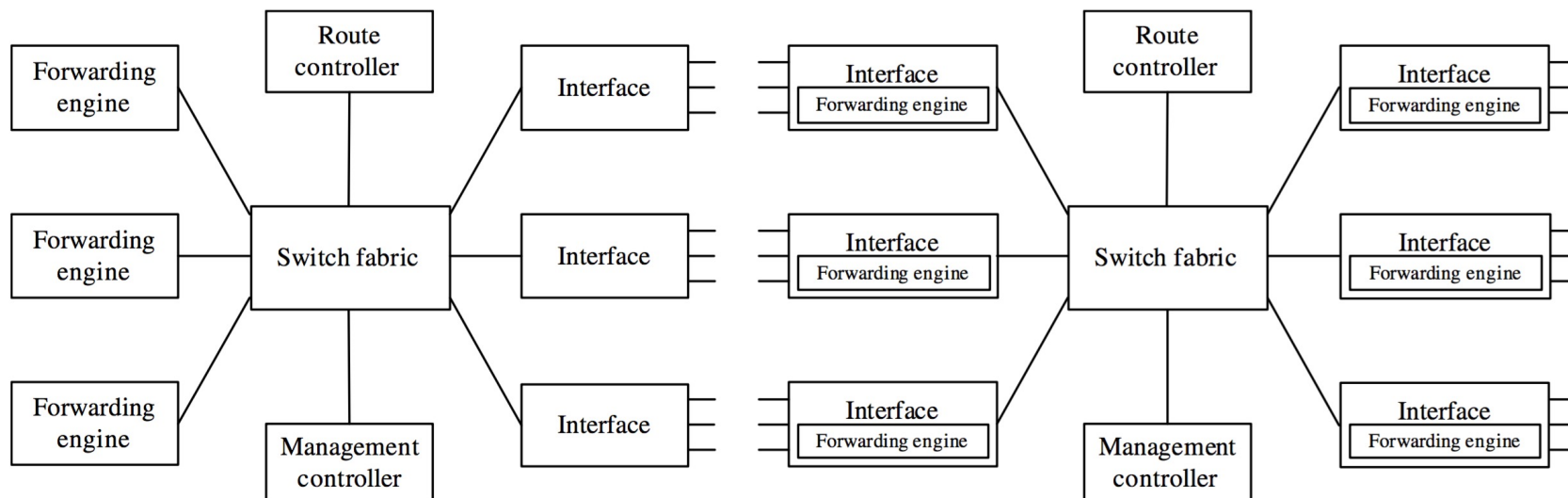
Distribuido o centralizado

- Arquitectura centralizada
 - Los interfaces entregan los paquetes a los motores de reenvío
 - Éstos toman la decisión y pasan el paquete al interfaz correspondiente
 - Puede que esos interfaces sean tarjetas con múltiples puertos
 - Puede que solo se envíe al FE la cabecera y cuando toma la decisión se entregue el paquete al interfaz de destino
 - Las rutas son calculadas por otros elementos



Distribuido o centralizado

- Arquitectura distribuida
 - Los interfaces son capaces de tomar las decisiones de reenvío
 - Las rutas calculadas se configuran en los interfaces
 - Mejora el rendimiento

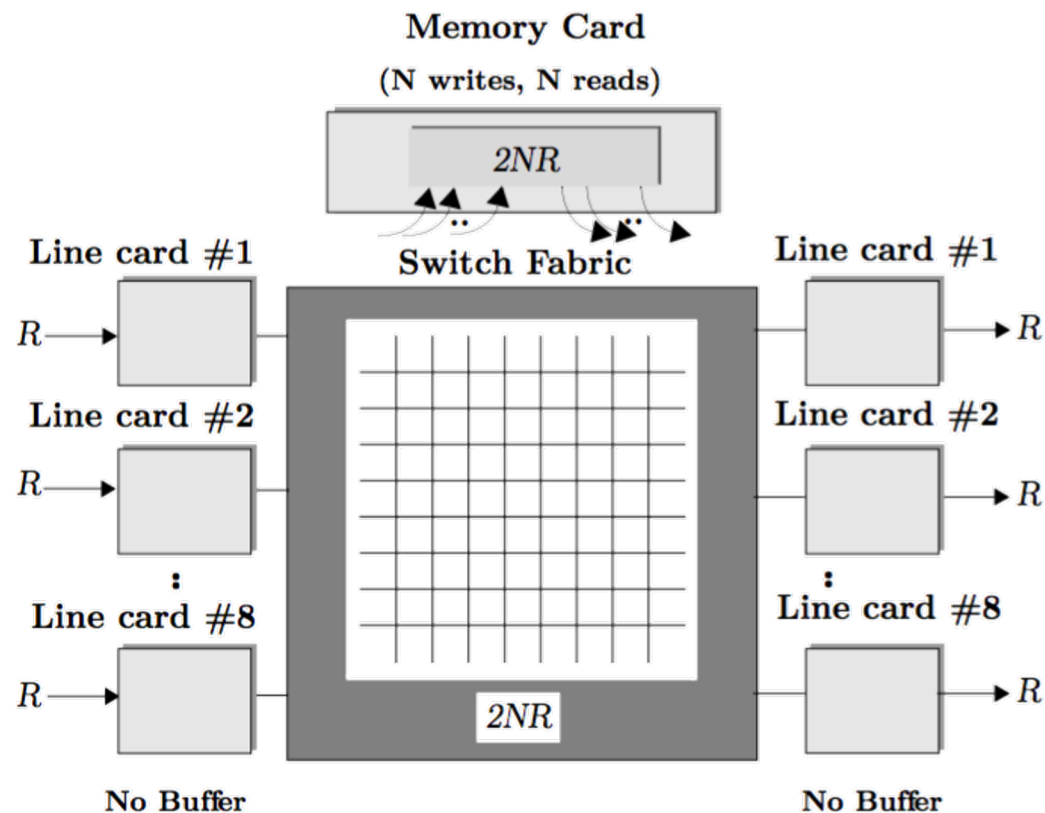


Forwarding: CAM vs TCAM

- *Content Addressable Memory*
 - En lugar de dar una dirección a la memoria se le da directamente el contenido
 - La CAM devuelve la dirección en que está almacenado
 - Más rápido que una búsqueda en RAM
 - Empleado en tablas MAC (layer 2)
- *Ternary Content Addressable Memory*
 - Al almacenar bits pueden indicar que no importa el valor de algunas posiciones
 - Eso permite (si se ordenan bien las entradas) hacer búsquedas equivalentes a longest-prefix-match
 - Empleadas en tablas de rutas IP (MPLS, QoS, ACLs, etc) (layer 3)
- Tamaños pequeños, elevados consumos

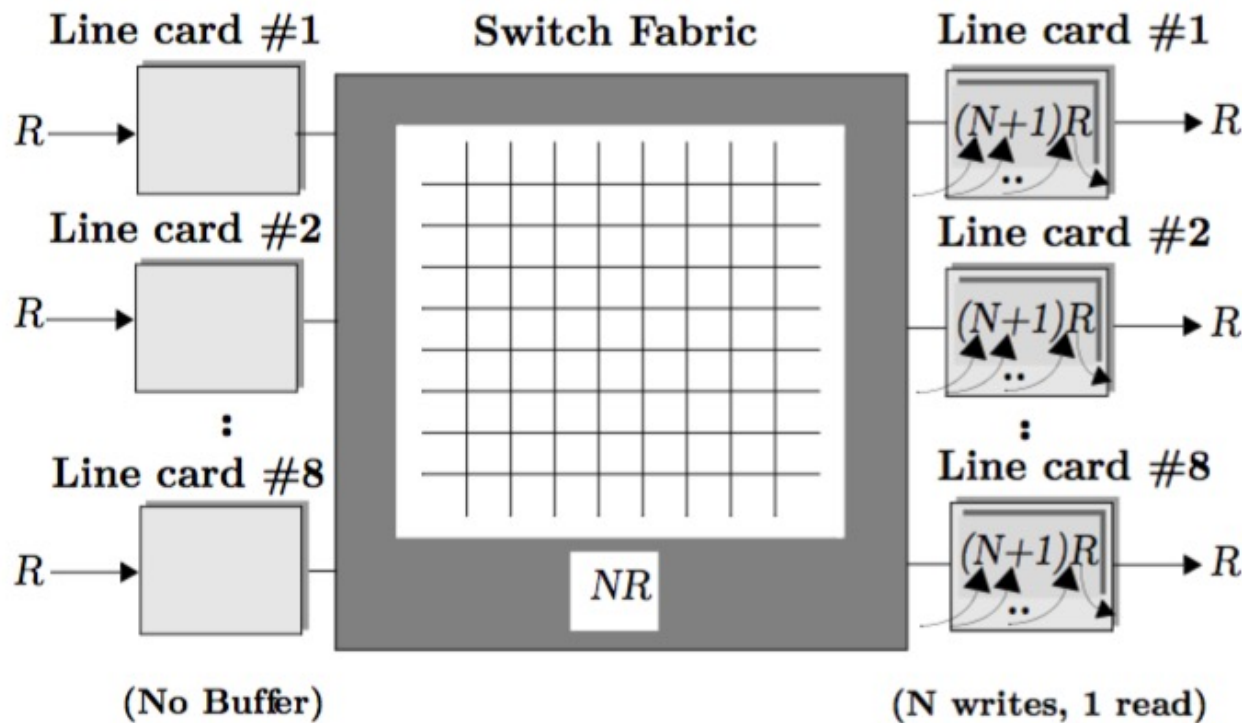
Output Queueing

- El conmutador con memoria compartida centralizada es un ejemplo de conmutador con colas a la salida
- Los paquetes se colocan directamente en lo que sería la cola de la salida
- A partir de ahí solo tienen que esperar a que se envíen los paquetes que estén encolados por delante para esa salida



Output Queueing

- Puede implementarse con una memoria por cada salida
- Con N entradas una de estas memorias debe permitir en el peor caso N escrituras y 1 lectura por slot, o una tasa $(N+1) \times R$
- Esto reduce a la mitad los requerimientos para cada memoria pero fragmenta su uso
- Sigue sin ser una gran reducción

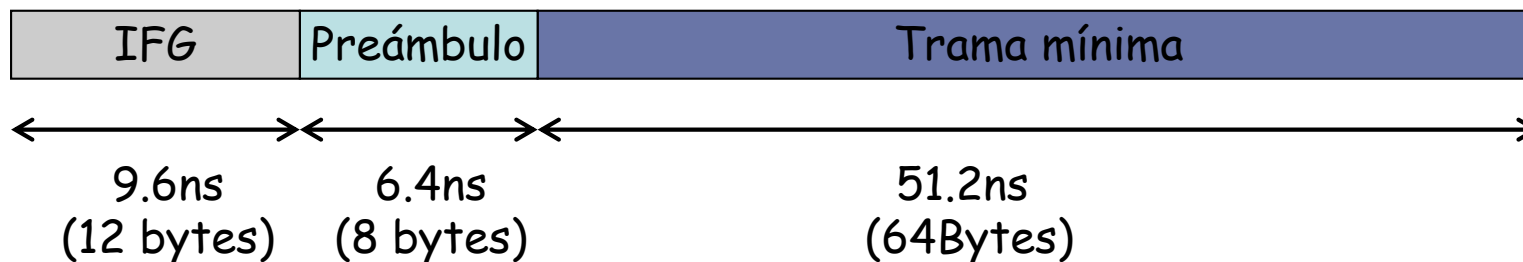


Limitaciones de la memoria

- DRAM
 - Diseñada pensando en el menor coste por byte y aumentar la capacidad
 - Tiempos de acceso en las decenas de nanosegundos
- SRAM
 - Diseñada pensando en el menor tiempo de acceso
 - Tiempos de acceso en los ns (un orden de magnitud inferior)
 - Más cara (mayor nº de transistores por bit)
- ¿Qué órdenes de magnitud necesitamos?

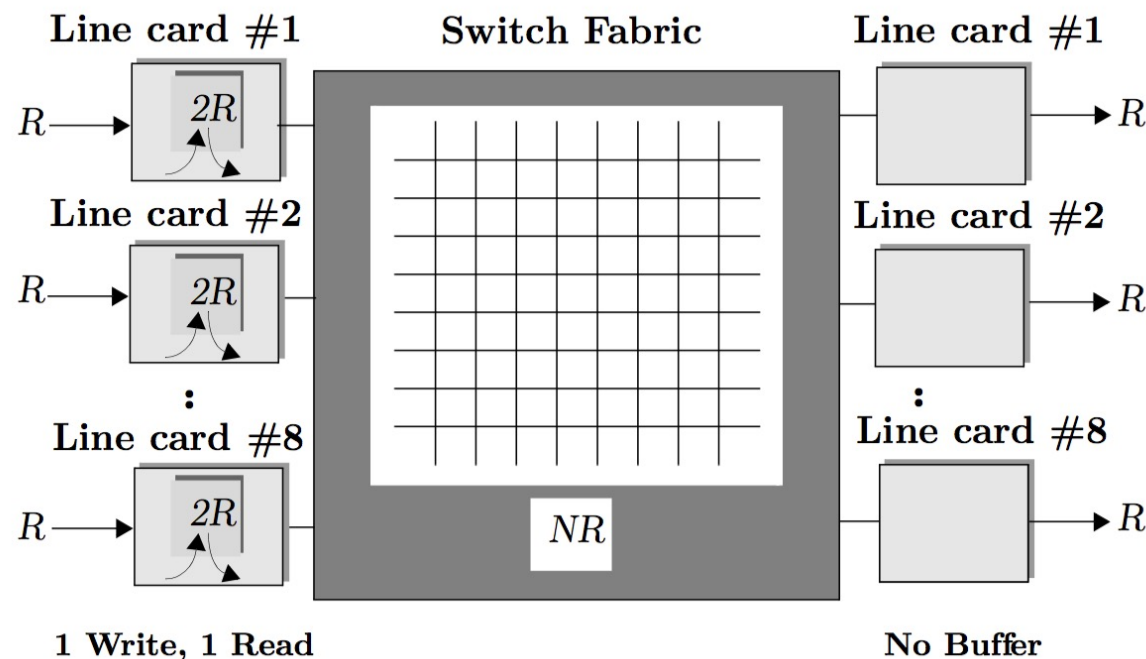
10G Ethernet

- En el peor caso tramas de 64 bytes
- A 10Gbps eso es una trama cada 67ns
- Los tiempos de acceso a DRAM están en ese orden de magnitud
- Eso quiere decir que en el tiempo que tarda en llegar otra trama podemos hacer muy pocos accesos a memoria
- Necesitamos al menos leer la dirección MAC destino, seguramente también etiquetas 802.1Q
- Necesita por ejemplo memorias que hagan *pre-fetching*
- O con menor tiempo de respuesta (SRAM, memoria on-chip)
- ¿Y si tenemos un conmutador con varios puertos 10G?
- Una “solución” es poner varias memorias en paralelo



Input Queueing

- En los 90s deja de ser viable la arquitectura de memoria compartida y se populariza la arquitectura de colas a la entrada
- Los paquetes se almacenan en los dispositivos de entrada
- La memoria pasa a estar distribuida
- Un crossbar switch interconecta los dispositivos
- La memoria tiene requisitos más ligeros pues no depende del número de dispositivos



upna

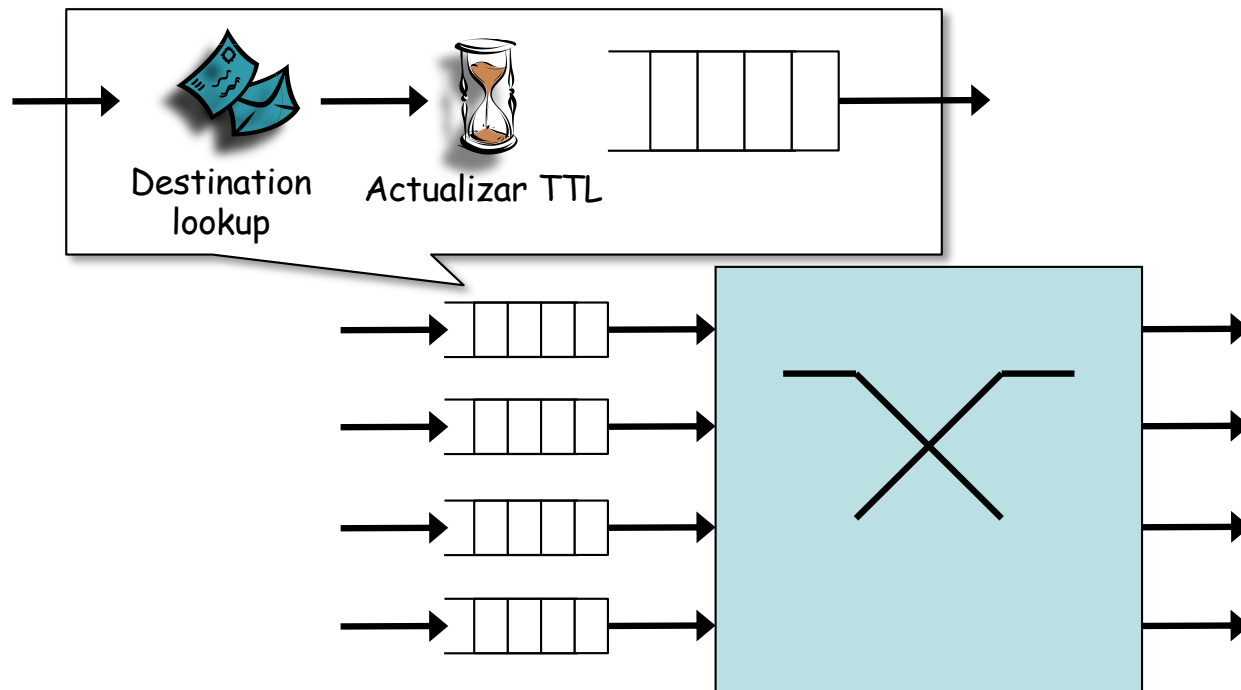
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

Input Queueing

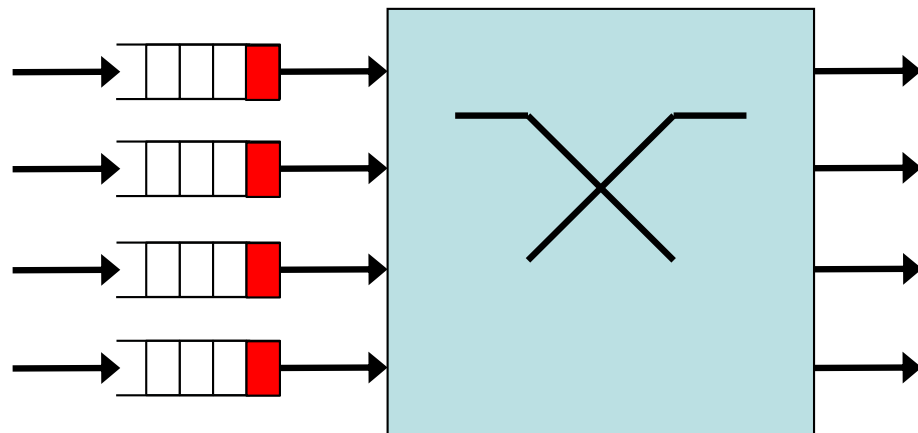
Input Queueing

- No requiere memorias tan rápidas
- Memorias distribuidas entre los interfaces de entrada
- Colas FIFO
- Distribuye también las tablas para el *lookup*
- (...)



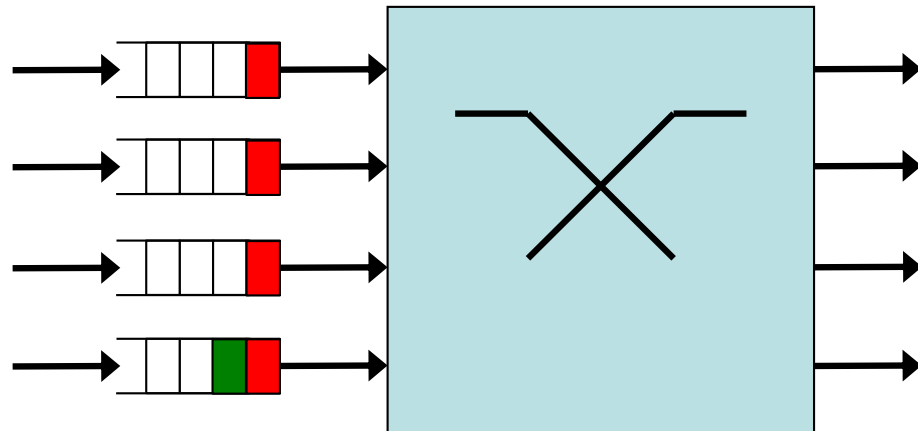
Input Queueing

- No requiere memorias tan rápidas
- Memorias distribuidas entre los interfaces de entrada
- Colas FIFO
- Distribuye también las tablas para el *lookup*



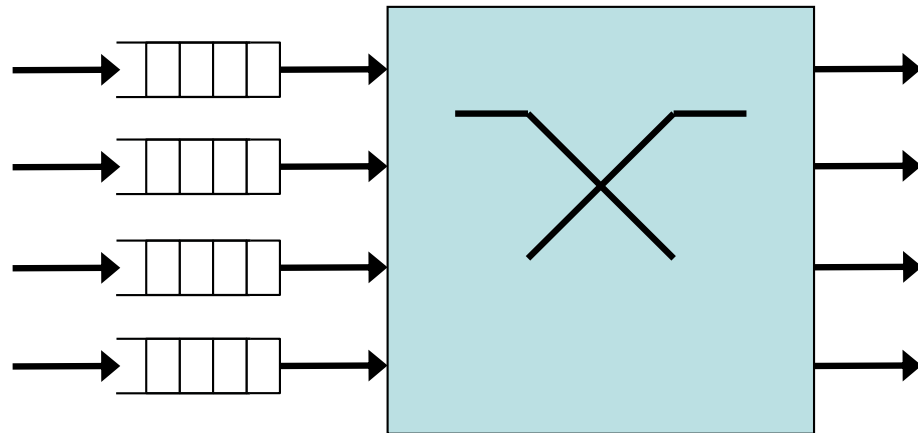
HOL blocking

- *Head Of Line blocking*
- Ejemplo:
 - El paquete verde va dirigido al primer puerto
 - No puede salir hasta que se envíe el paquete anterior (cola FIFO)
 - Aunque el puerto de salida al que va está libre (...)
 - *Work Conserving?*



HOL blocking

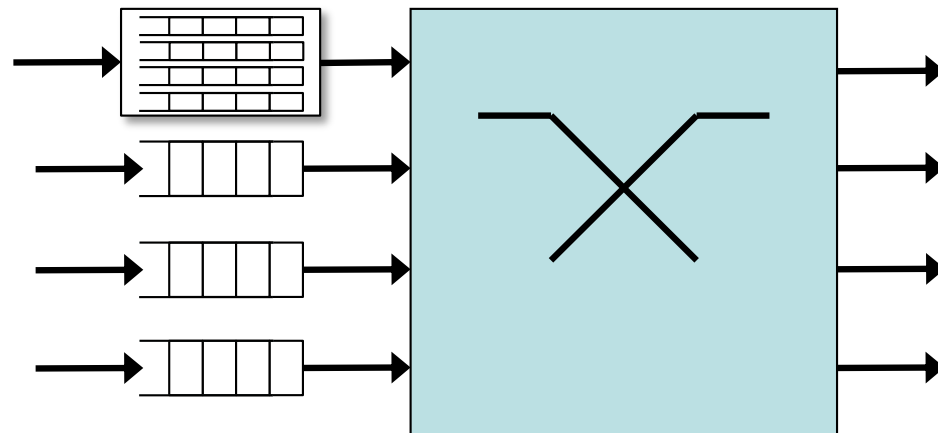
- Degrada el rendimiento
- Con tráfico distribuido uniformemente por las salidas está limitado a un throughput del 58.6% [1]



[1] Mark J. Karol, Michael G. Hluchyj, Samuel P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch", IEEE Trans. On Comm., Vol. COM-35, No. 12, Dic. 1987

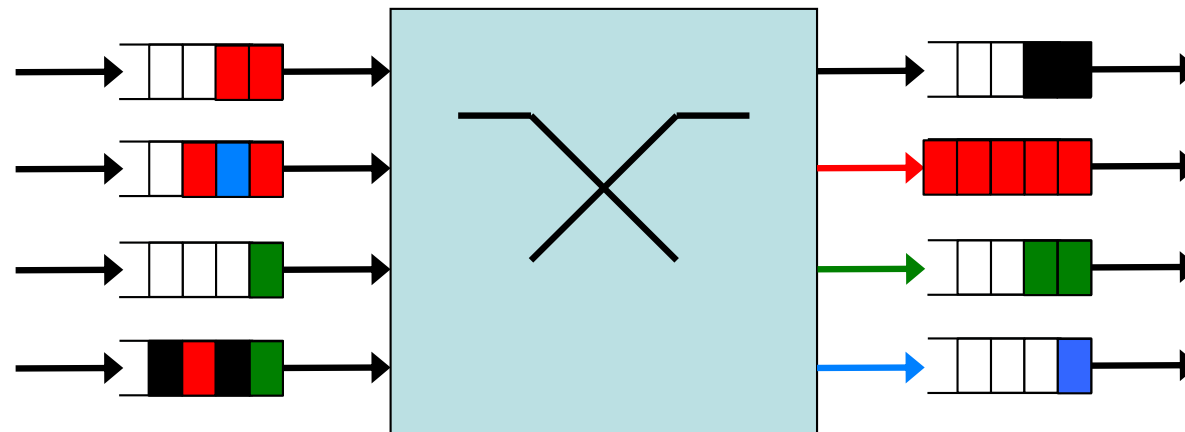
HOL blocking y VOQ

- Se elimina empleando *Virtual Output Queues (VOQs)*
- Cada buffer de puerto de entrada separa el tráfico en tantas VOQs como puertos de salida
- Aunque una VOQ no pueda ser servida porque el puerto de salida esté ocupado se puede servir de otra
- El algoritmo de planificación de la conmutación interna se complica



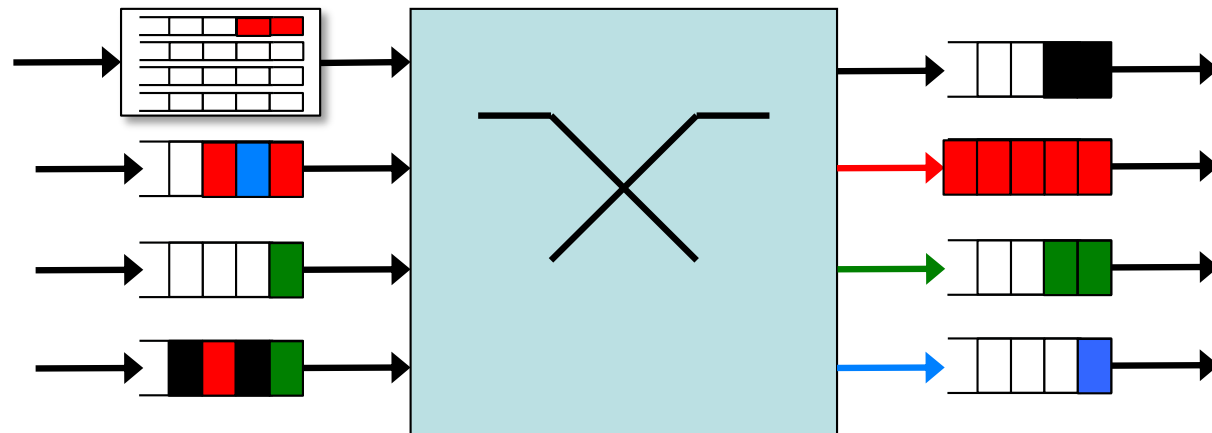
Speedup

- Se pueden acumular paquetes en los buffers de entrada
- Puede haber múltiples paquetes para el mismo puerto
- En un intervalo de tiempo (*slotted*) solo atravesaría la matriz un paquete para cada puerto
- Pueden atravesar la matriz varios paquetes en una ranura de tiempo si trabaja a una velocidad superior a los puertos
- Se dice que tiene un *speedup* (xN)
- Pero entonces requiere buffers a la salida, si no son puertos de velocidad superior



CIOQ

- *Combined Input and Output Queueing*
- Podemos añadirle VOQ
- Está comprobado que para prácticamente cualquier tipo de tráfico se comporta aproximadamente como un FIFO-OQ con tal de que el *speedup* sea de al menos $x2$



upna

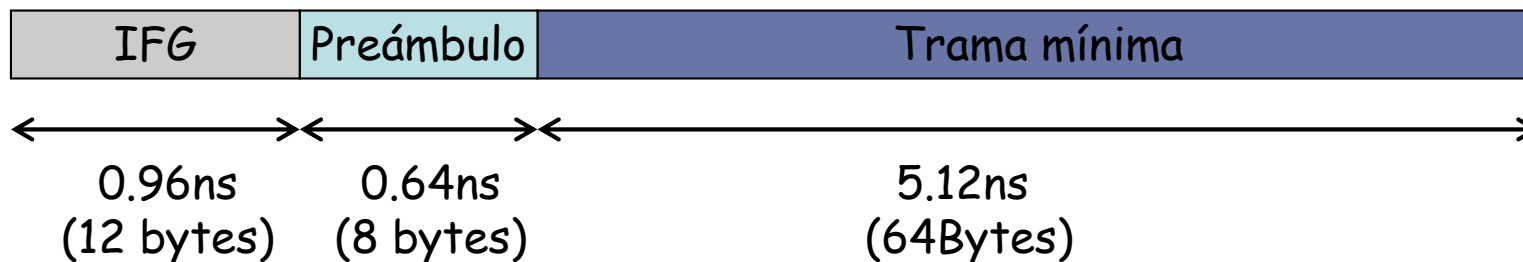
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Redes de Nueva Generación
Área de Ingeniería Telemática

Input Queueing

>10G Ethernet

- 40Gbps, 100Gbps
- A 100Gbps la trama de 64bytes tarda 6.4ns
- Las memorias más rápidas responden en el rango de 1-2ns
- Pero de nuevo eso es solo con un puerto, con varios puertos hay que poder atender a varias peticiones
- La limitación es más seria pues la luz recorre en un 1ns...
- $3 \times 10^8 \text{m/s} \times 10^{-9} \text{s/ns} = 0.3 \text{m/ns} = 30 \text{cm/ns}$



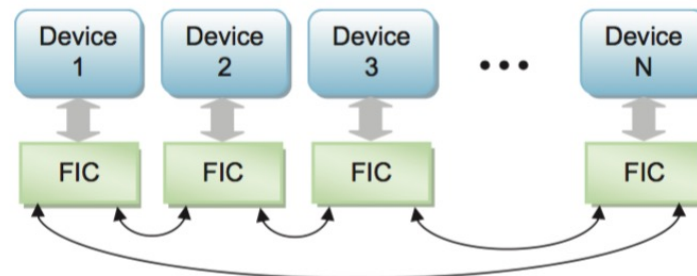
Topologías

- Un data center requiere decenas de miles de puertos
- Un chip puede ofrecer en el orden de decenas, tal vez un centenar
- Quedan entonces las soluciones multi-etapa
 - Soluciones multi-conmutador
 - Y soluciones multi-chip dentro del conmutador
- Las topologías básicas son las mismas (...)

Topologías

Anillos

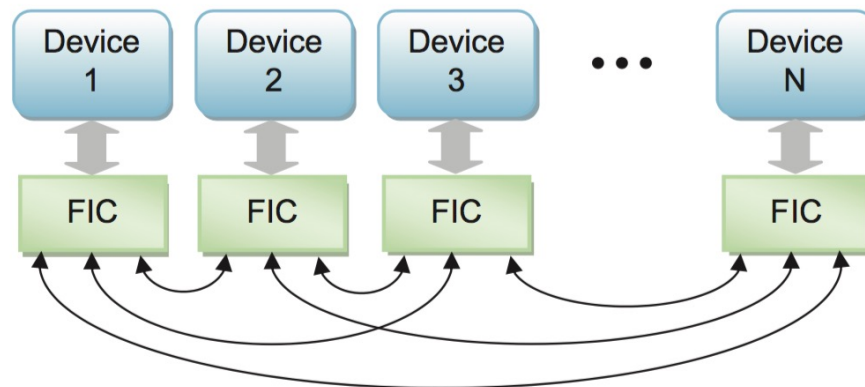
- Elementos sencillos (2 puertos)
- Ofrece protección
- Pero un fallo en un dispositivo rompe el anillo
- Fácil añadir nuevos dispositivos aumentando el anillo
- Pero al aumentar los dispositivos aumenta la carga
- En el entorno LAN/WAN hemos tenido Token Ring, FDDI y por ejemplo SDH forma típicamente anillos
- O por ejemplo apilamos conmutadores formando un anillo
- E igualmente podemos tener un anillo entre dispositivos, internamente al conmutador



Topologías

Mesh

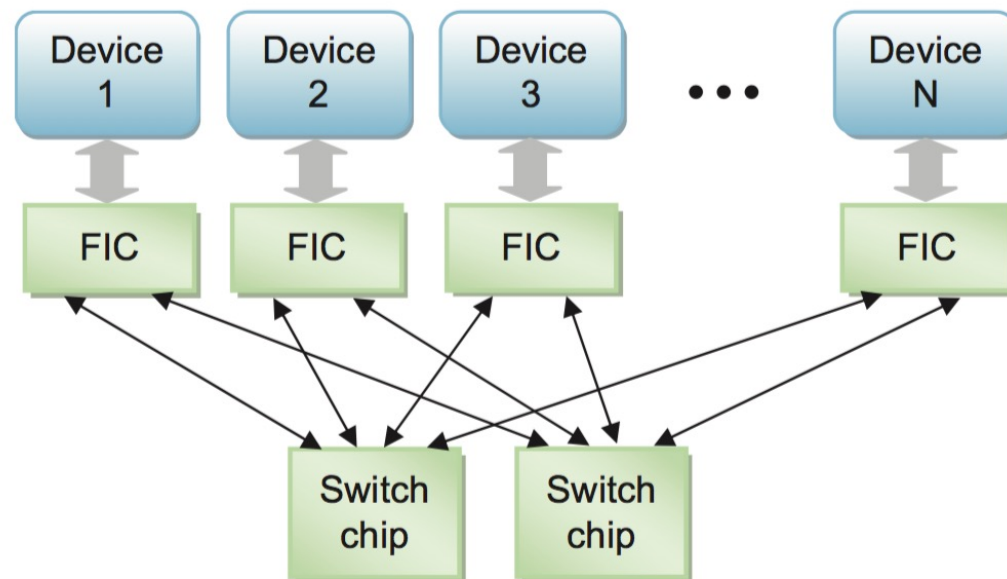
- Enlace dedicado entre cada pareja de dispositivos
- Mejora el rendimiento
- Requiere gran cantidad de puertos por dispositivo
- Requiere más decisión de encaminamiento que el anillo



Topologías

Estrella

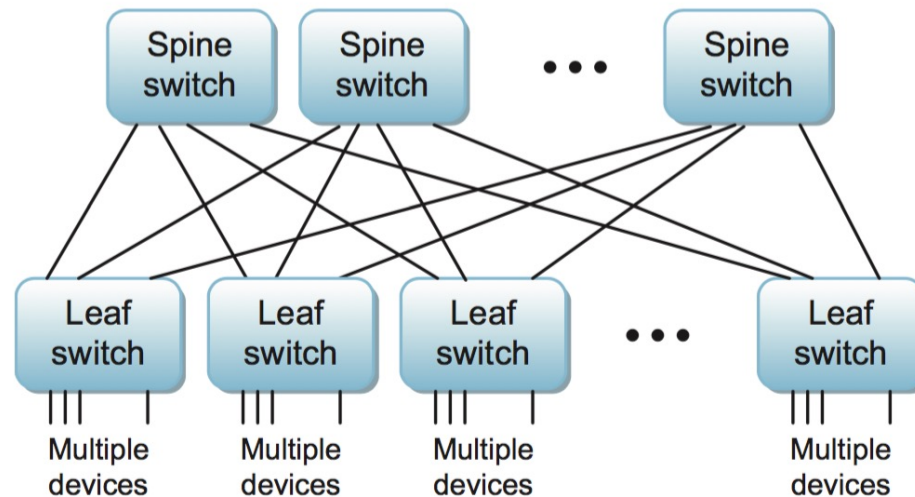
- Es la topología típica en conmutadores modulares
- El chasis contiene elementos de conmutación



Topologías

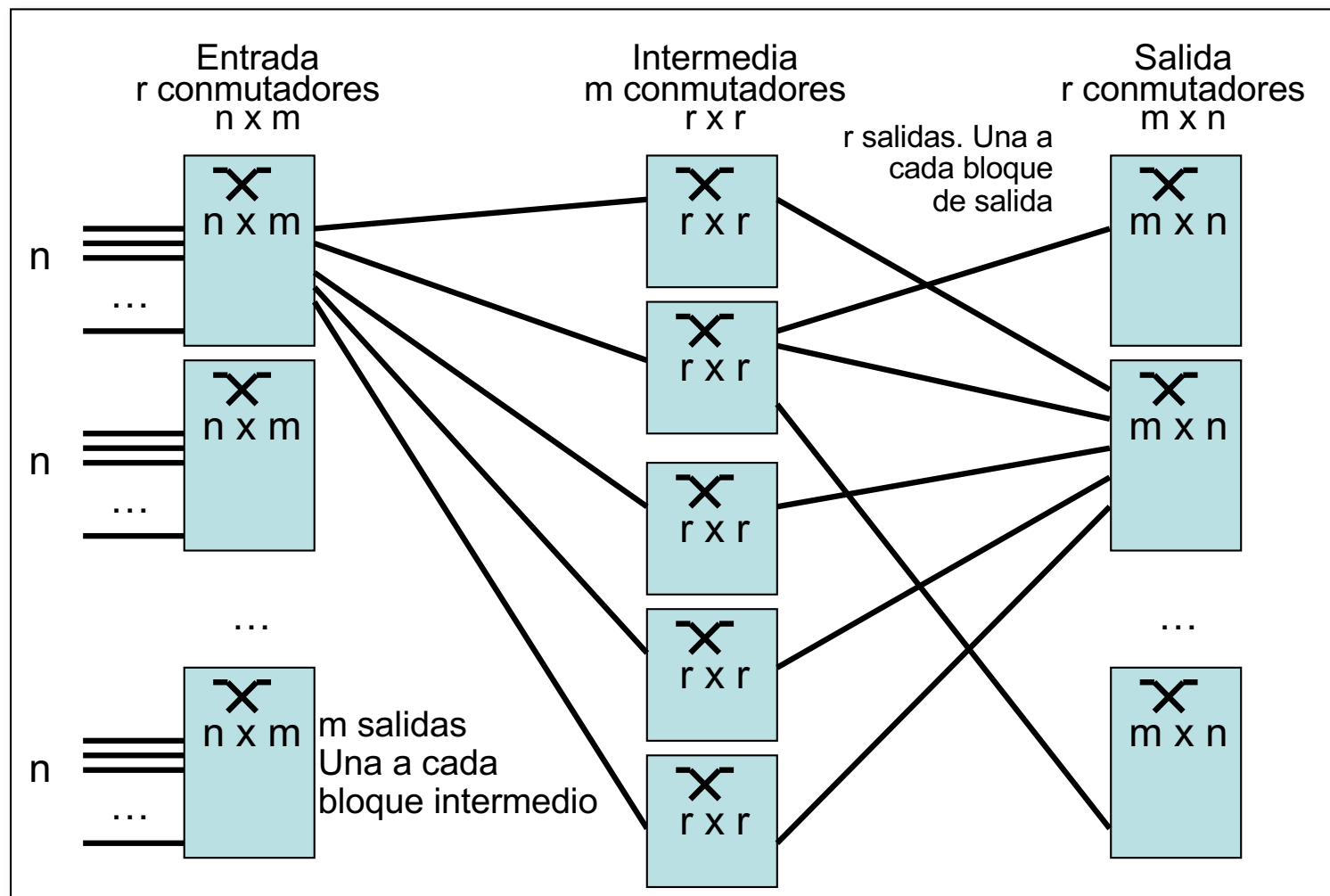
Fat Tree

- Evoluciona de la topología en estrella con más nodos centrales
- En realidad esta topología es muy conocida en el entorno de conmutación de circuitos...



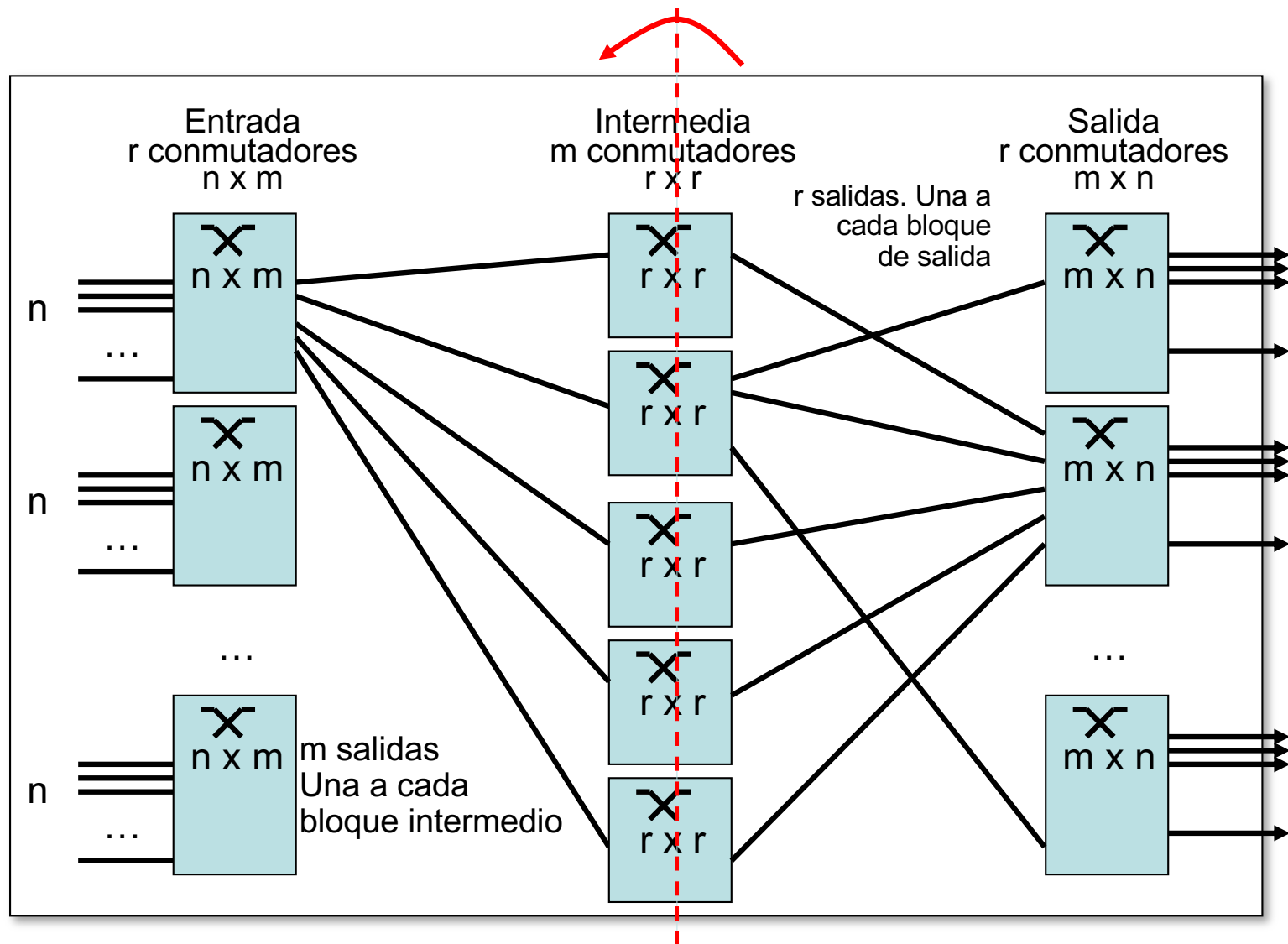
Redes de (Charles) Clos

- Básico en arquitectura de conmutadores de circuitos
- Hay múltiples etapas y los elementos de la etapa x se conectan solo con los de $x-1$ y los de $x+1$ pero no con los de x



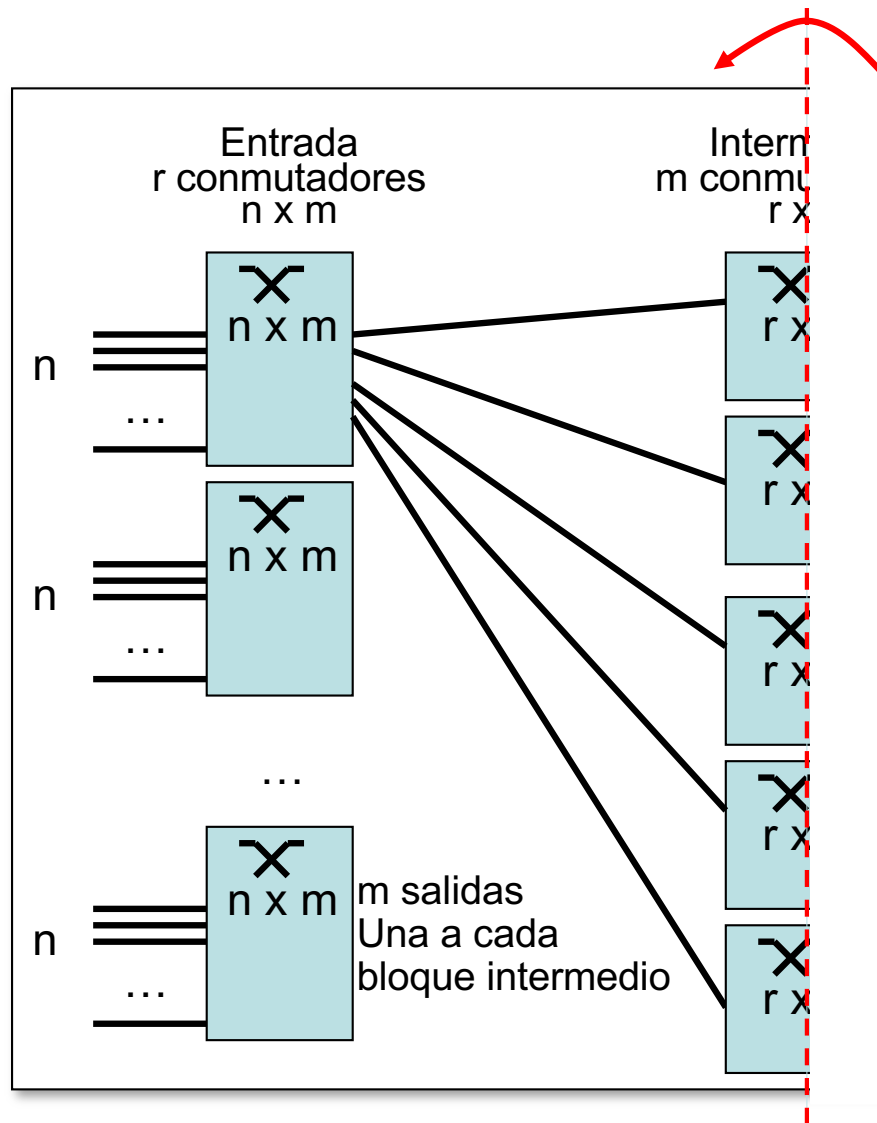
Folded Clos / Fat Tree

- En realidad este diseño es para enlaces unidireccionales
- Con enlaces bidireccionales se “dobla” esta topología (...)



Folded Clos / Fat Tree

- En realidad este diseño es para enlaces unidireccionales
- Con enlaces bidireccionales se “dobla” esta topología (...)



Folded Clos / Fat Tree

- Aquí hemos hablado de la arquitectura interna del conmutador pero volveremos a esta topología al hablar de la interconexión de ellos

