# Arquitectura de routers

*¿Cómo funcionan los routers?*
*+ introducción al scheduling*

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

Redes
4º Ingeniería Informática

# Hoy...

1. Introducción a las redes
2. Tecnologías para redes de área local
3. Conmutación de circuitos
4. Tecnologías para redes de área extensa y última milla
5. Encaminamiento
6. **Arquitectura de conmutadores de paquetes**

- Control de acceso al medio
- Transporte extremo a extremo

# Problemas en el nivel de red

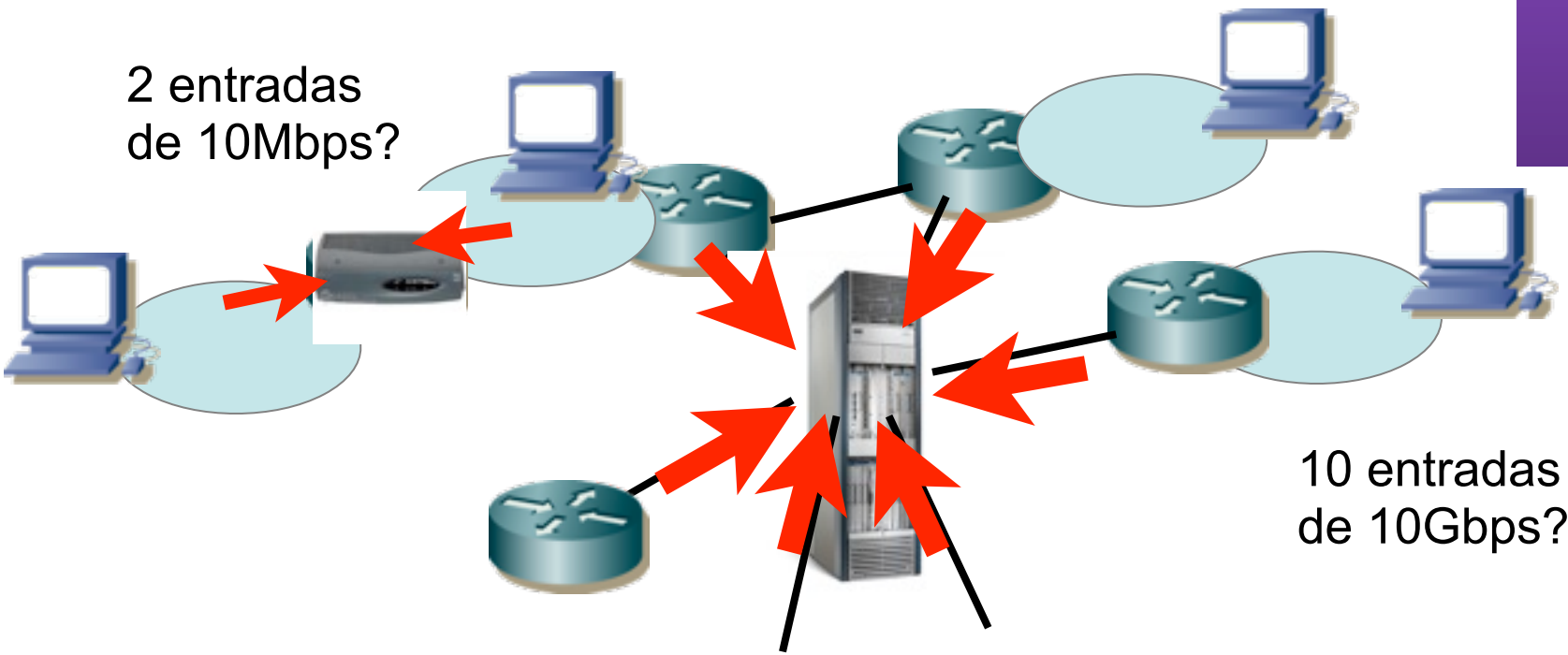
- Problema de enrutamiento:

  Cómo sé a qué vecino debo reenviar?

- **Construcción de routers**

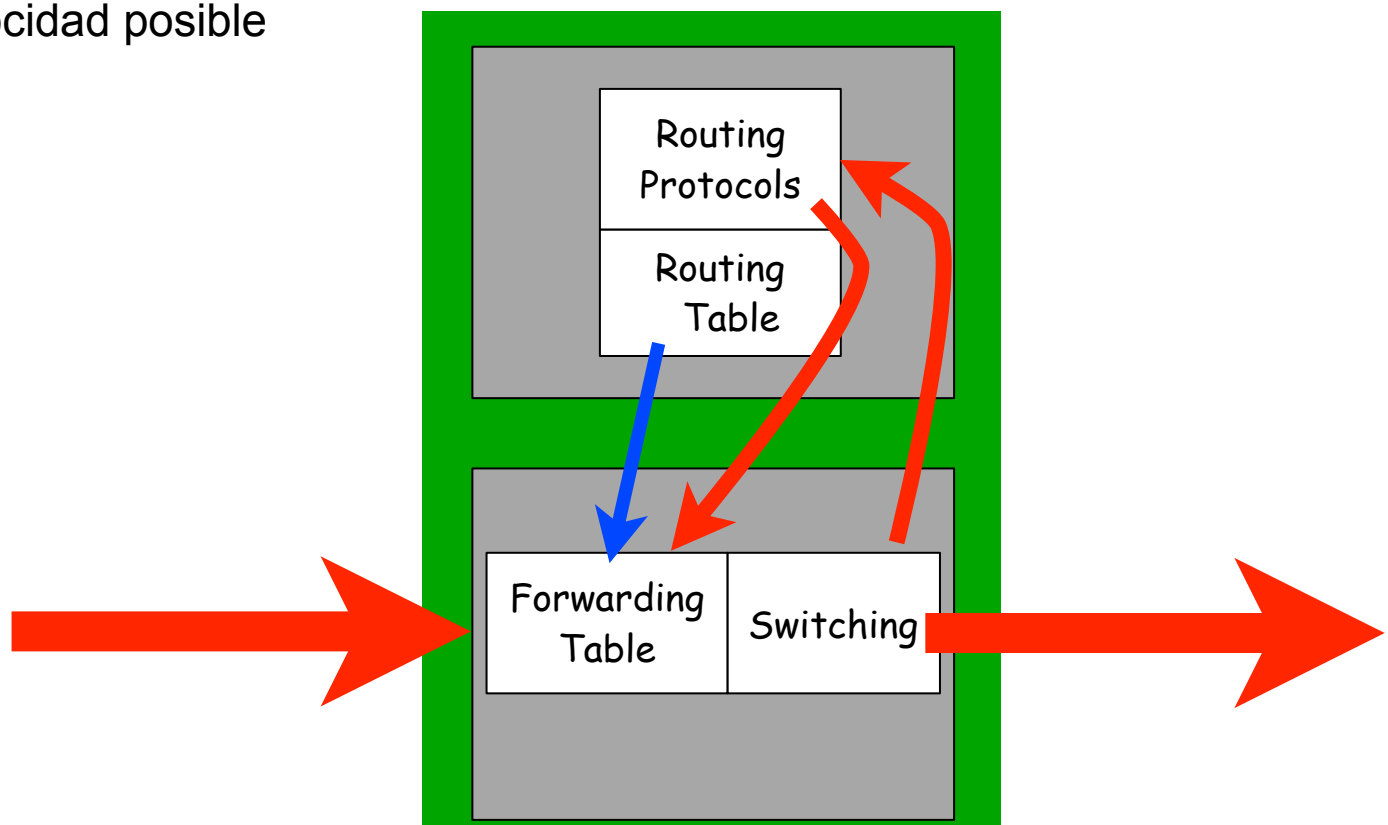
  Se pueden construir conmutadores/encaminadores de paquetes?

  Cómo de rápidos/eficientes? cuántos paquetes por segundo puedo reenviar? Qué funcionalidades necesitan?



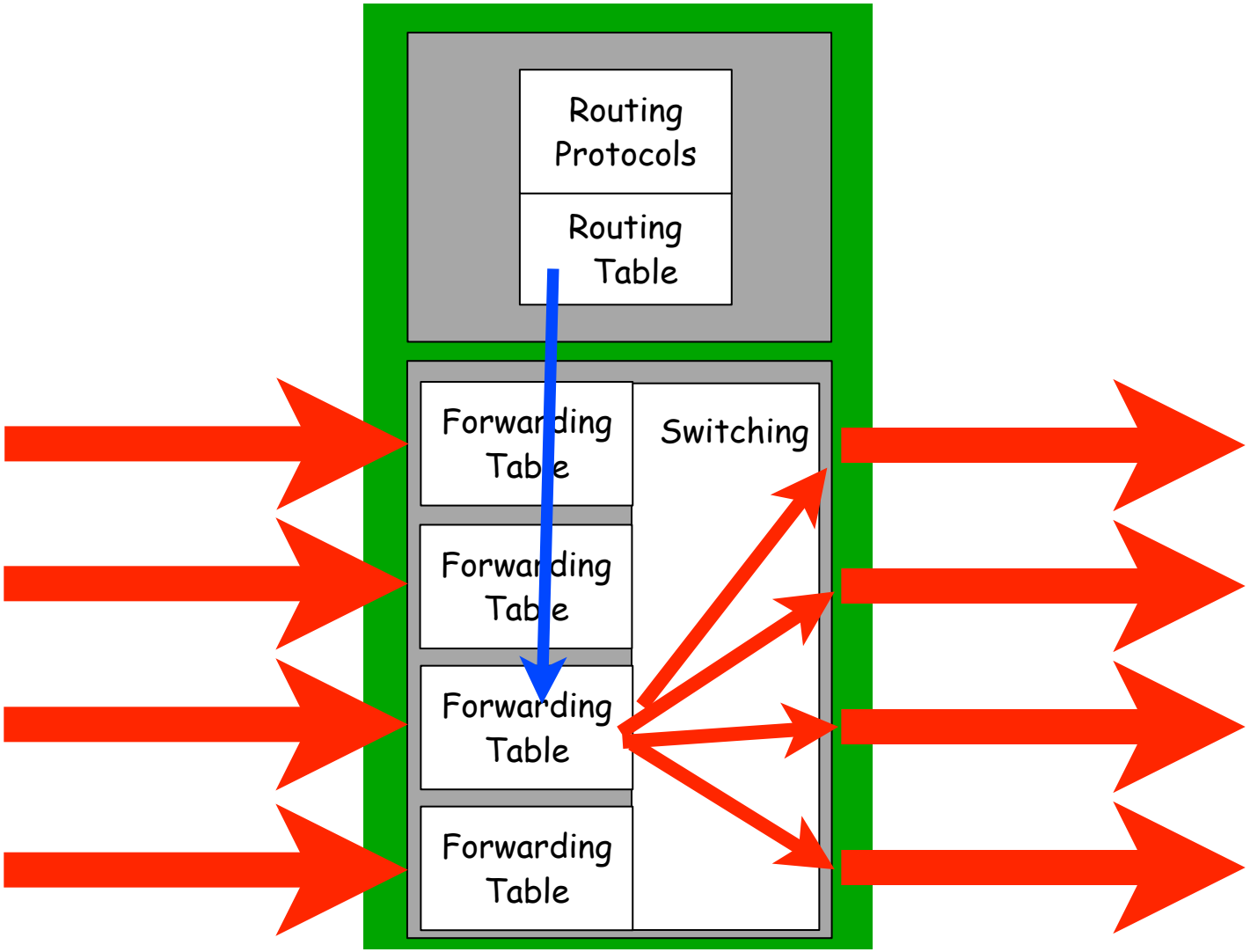
# Arquitectura básica de un router IP

- **Plano de control**
  - Los protocolos de enrutamiento y otra información de control pueden enviar y recibir información de la red
  - Construyen la tabla de rutas y configuran el plano de datos
- **Plano de datos**
  - Los paquetes que pasan por el router deben atravesarlo a la máxima velocidad posible



# Arquitectura básica de un router IP

- **Plano de datos**
  - No es una entrada y una salida sino varias entradas y salidas



# Proceso por paquete

¿Qué hay que hacer por cada paquete?

- **Recibir** a nivel de enlace.
- **Lookup**: buscar la dirección destino del paquete en la tabla de reenvío (forwarding) para identificar por donde debe salir
- **Header processing**: manipular cabecera IP: decrementar TTL, recalcular checksum
- **Switching**: llevar el paquete a la tarjeta de salida correspondiente
- **Buffering**: almacenar el paquete durante los tiempos que deba esperar
- **Transmitir** a nivel de enlace

- La velocidad a la que se pueda hacer da las prestaciones

# Las dificultades

- En media debemos hacer todo el proceso para cada paquete en el tiempo en el que la tarjeta de red recibe el siguiente

  De lo contrario estamos acumulando paquetes

- Tiempo para enviar un paquete de 29+14bytes

  (ping de 1 byte) a 10Gbps : **34ns**

# Address lookup

- Buscar una dirección IP en una tabla de rutas

  Longest prefix match no vale con encontrar uno que cumpla

  La tabla puede tener ~150000 entradas

- Se usan estructuras prefix tree

  Ejemplo: almacena los prefijos

  {*,000*,0000*,0001*,100*,110*,111*}

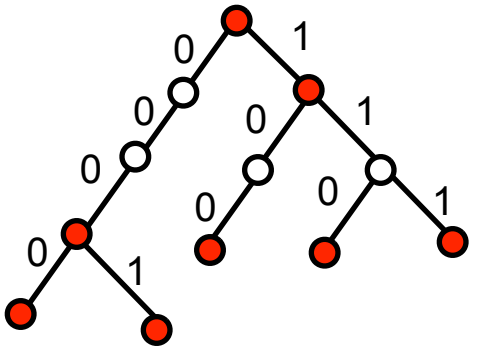
  busquedas de tipo

  00010010 -> 0001*

  10101001 -> 1*

  00101010 -> *

  11010101 -> 110*




Se hace en hardware (NSE network search engines)

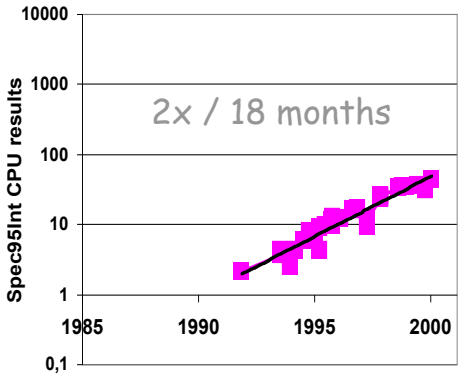
(Se pueden encontrar circuitos que hacen 100000000 busquedas por segundo ~10ns)

# Almacenamiento y cálculo

- No es trivial almacenar y recuperar paquetes en 30ns velocidades de acceso a memoria?
- No es trivial manipular la cabecera

  Hardware especifico para routers de alta velocidad
- Problemas con la ley de Moore
  - Velocidad de procesado se duplica cada 18 meses
  - Capacidad de la fibra se duplica cada año
  - Velocidad de la memoria solo se multiplica por 1.1 cada 18 meses

Packet Processing Power




- El tráfico de Internet crece más deprisa

  **Capacidad típica de los routers comerciales**

  Capacidad 1992 ~ 2Gb/s

  Capacidad 1995 ~ 10Gb/s

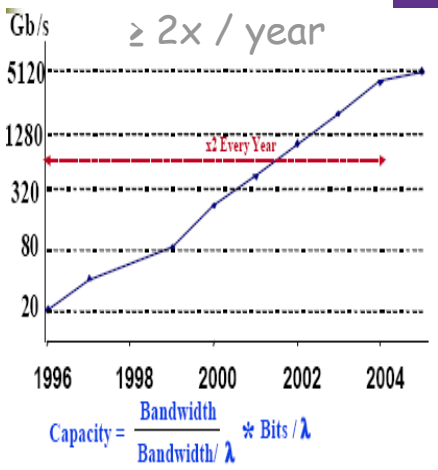
  Capacidad 1998 ~ 40Gb/s

  Capacidad 2001 ~ 160Gb/s

  Capacidad 2003 ~ 640Gb/s

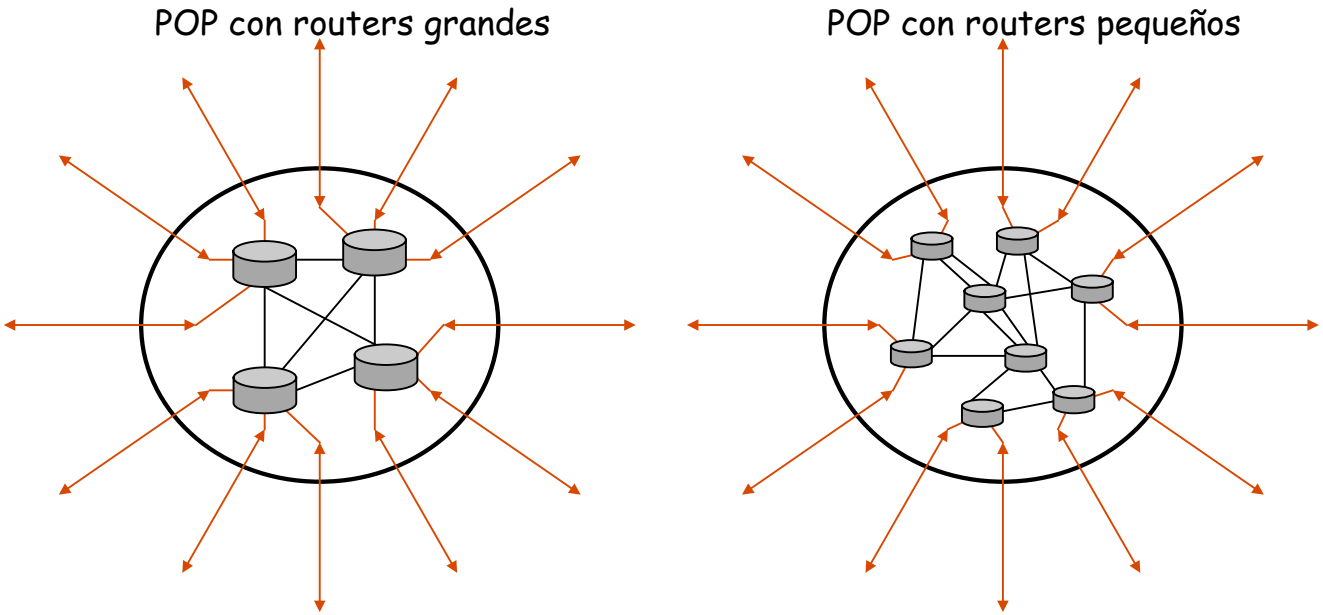
  Media approx 2.2x / 18meses

Single Fiber Capacity (commercial)



# Distribuyendo la carga

- Siempre podemos usar mas routers e interconexiones para distribuir el tráfico
- Interesa más tener pocos routers de gran potencia
  - Gestión más simple
  - Menos consumo y menos coste: lo que cuesta son los puertos




- Puertos: precio >$50k,  consumo > 400W.
- Alrededor de  50-60% de los puertos es para interconexión

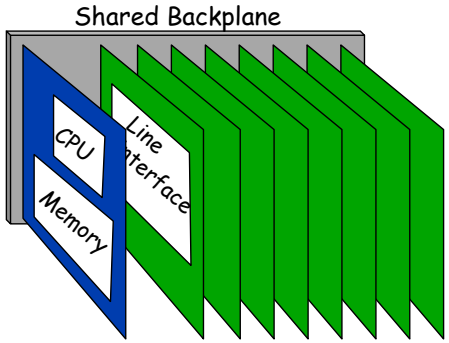
# Switching

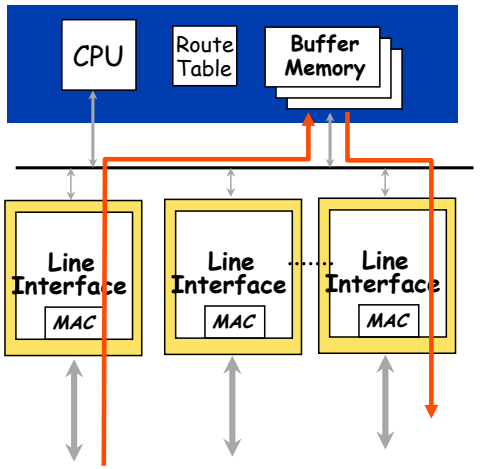
- Problema fundamental
- ¿Como enviamos los paquetes de la entrada a la salida correspondiente?

  Matriz de conmutaciñon / switch fabric
- El problema aparece en conmutación de circuitos primero
  - En telefonía es más fácil el circuito se establece con antelación. Una vez establecido todos los datos que llegan por un puerto de entrada van al mismo puerto de salida hasta que se libera
- Necesitamos el mismo tipo de sistemas pero que sean capaces de cambiar de estado en el tiempo de un paquete
  - La velocidad a la que los paquetes atraviesen esta matriz de conmutación es fundamental

# Switching architectures

- Memoria central y bus
  - Ordenador tradicional conmutación bajo control de la CPU
  - El paquete se copia del puerto de entrada  a la memoria
  - El paquete se copia de la memoria al puerto de salida
  - Gran flexibilidad y fácil de programar
  - El paquete pasa 2 veces por el bus
  - La máxima velocidad a la que reenviamos paquetes es la mitad de la velocidad del bus
  - Los paquetes deben esperar en las tarjetas de entrada (Contención)




  - Routers de primera generación

    Tipica velocidad agregada < 0.5Gbps

# Switching architectures

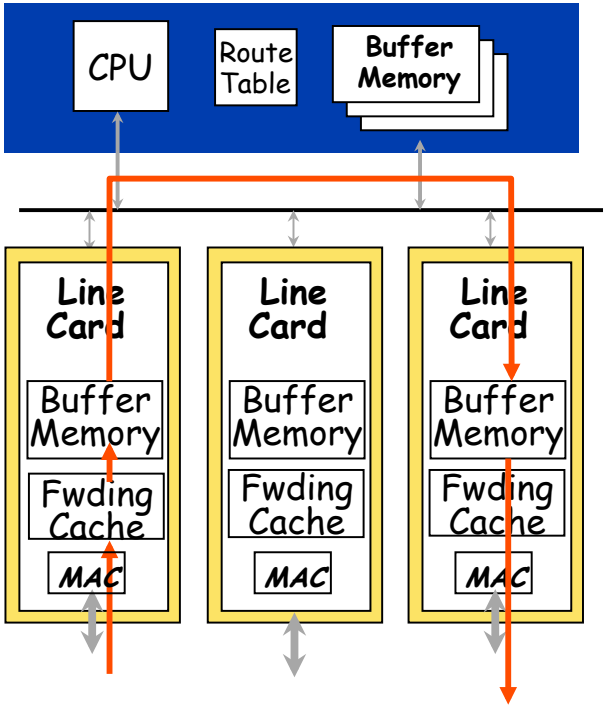
- Bus compartido y puertos independientes
  - El paquete se copia de la memoria del puerto de entrada a la del puerto de salida
  - Más complejo
  - El paquete pasa 1 vez por el bus. La máxima velocidad agregada es la del bus
  - Routers de segunda generación

    Tipica velocidad agregada < 5Gbps
  - Pero en un instante como mucho

    pasa 1 paquete por el bus

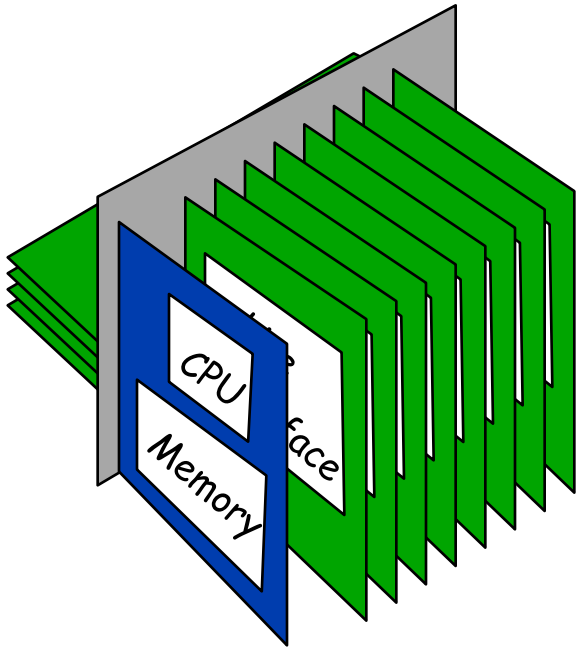
    si otro puerto de entrada tiene un paquete

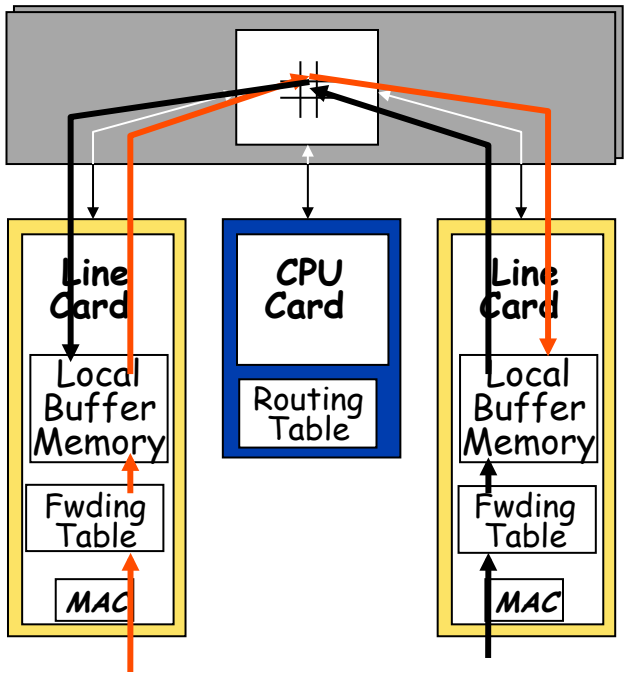
    debe esperar...



# Switching architectures

- Tercera generacion ++
  - Matriz de conmutación más compleja capaz de tener varias transferencias a la vez (switched backplane)
  -

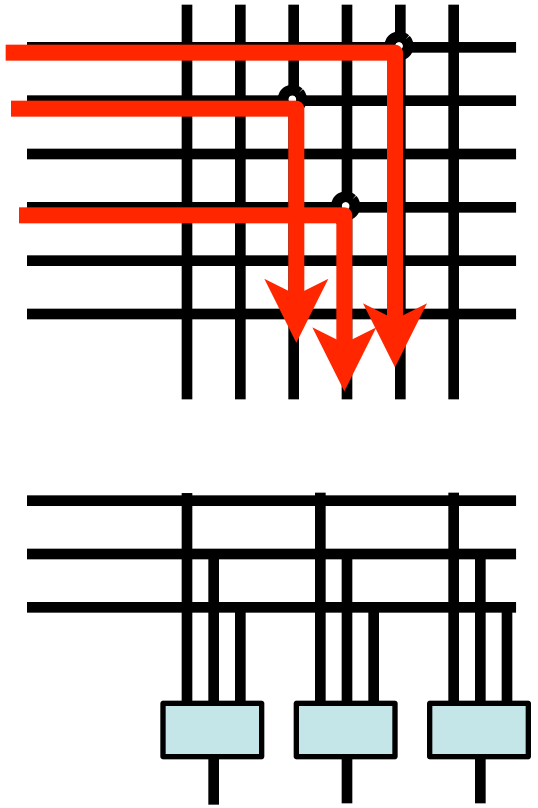


# Switching

- Switch fabrics capaces de intercambiar varios flujos al mismo tiempo
- **Crossbar (space-switching)**
  - N veces mas rapido que un bus
  - Pero hay problemas si dos quieren enviar un paquete a la vez al mismo destino (contencion)

    O bien el acceso al puerto de salida es N veces más rápido o bien puede haber bloqueo
- **Broadcast**
  - N veces mas rapido que un bus
  - El acceso al puerto de salida N veces mas rápido o bloqueo
- Necesita muchos recursos
- El cuello de botella es la entrada al puerto de salida
- Escalan mal con el numero de puertos



# Switching

- **Redes Banyan**

  Facil de encaminar paquetes por la electronica

  Electronica a la velocidad del puerto de entrada

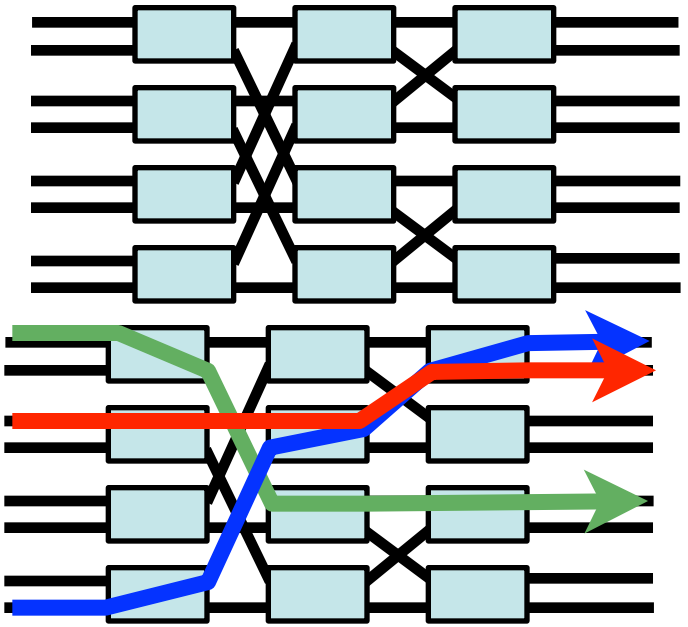
  Pero por si sola tiene probabilidad de bloqueo interno (bloqueo incluso aunque no quieran salir por el mismo puerto)

- El bloqueo interno se puede eliminar si ordeno las entradas

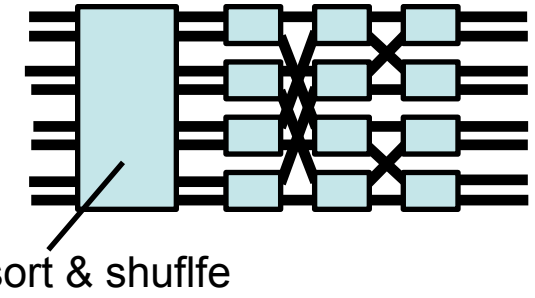
  Batcher-Banyan networks

- Pero si dos paquetes quieren ir a la vez a la misma salida solo hay dos posibilidades
  - Retardarlo con buffers
  - Hacer el puerto de salida mas rápido

Banyan network de 3 bits



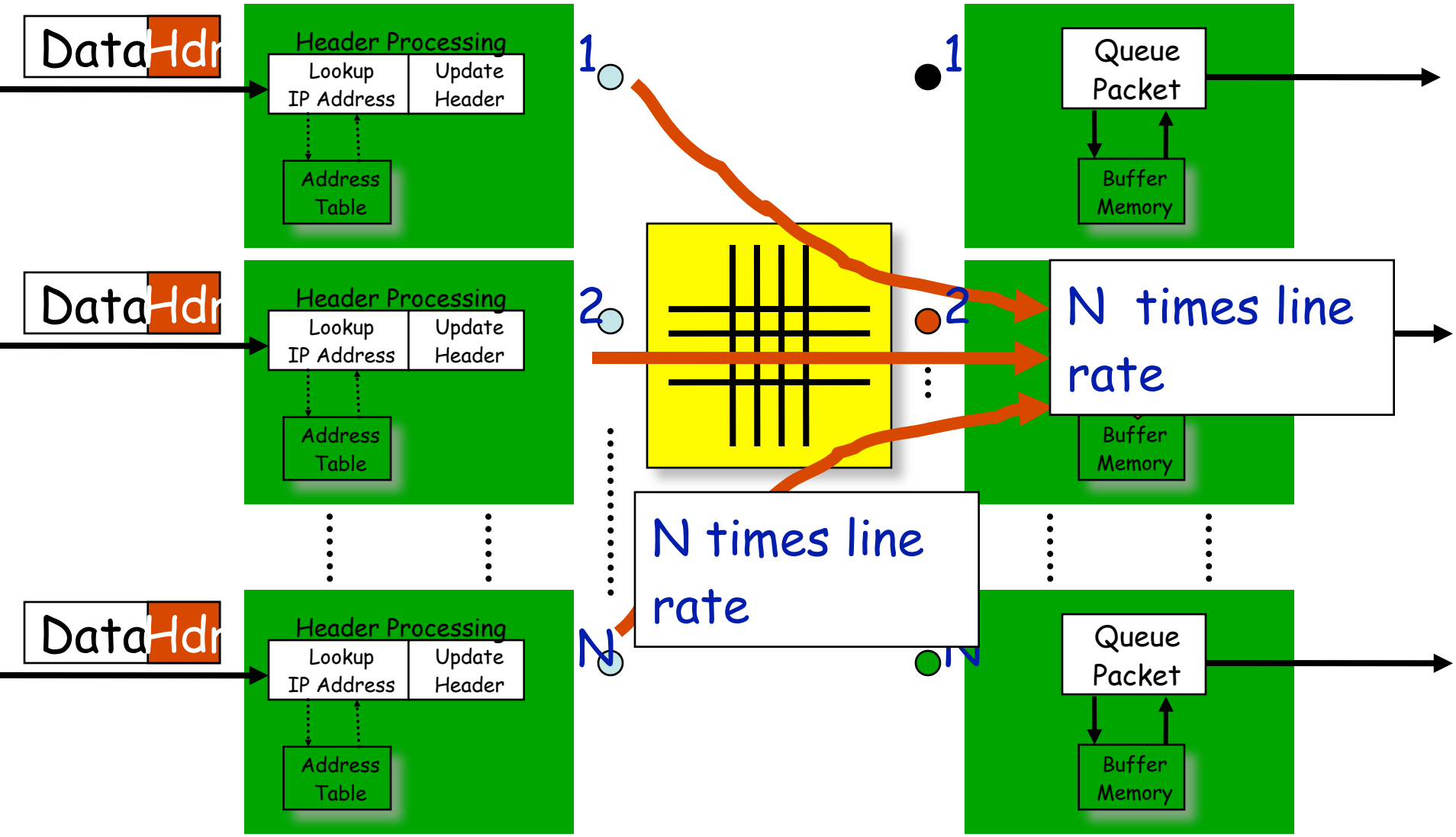
Batcher-Banyan



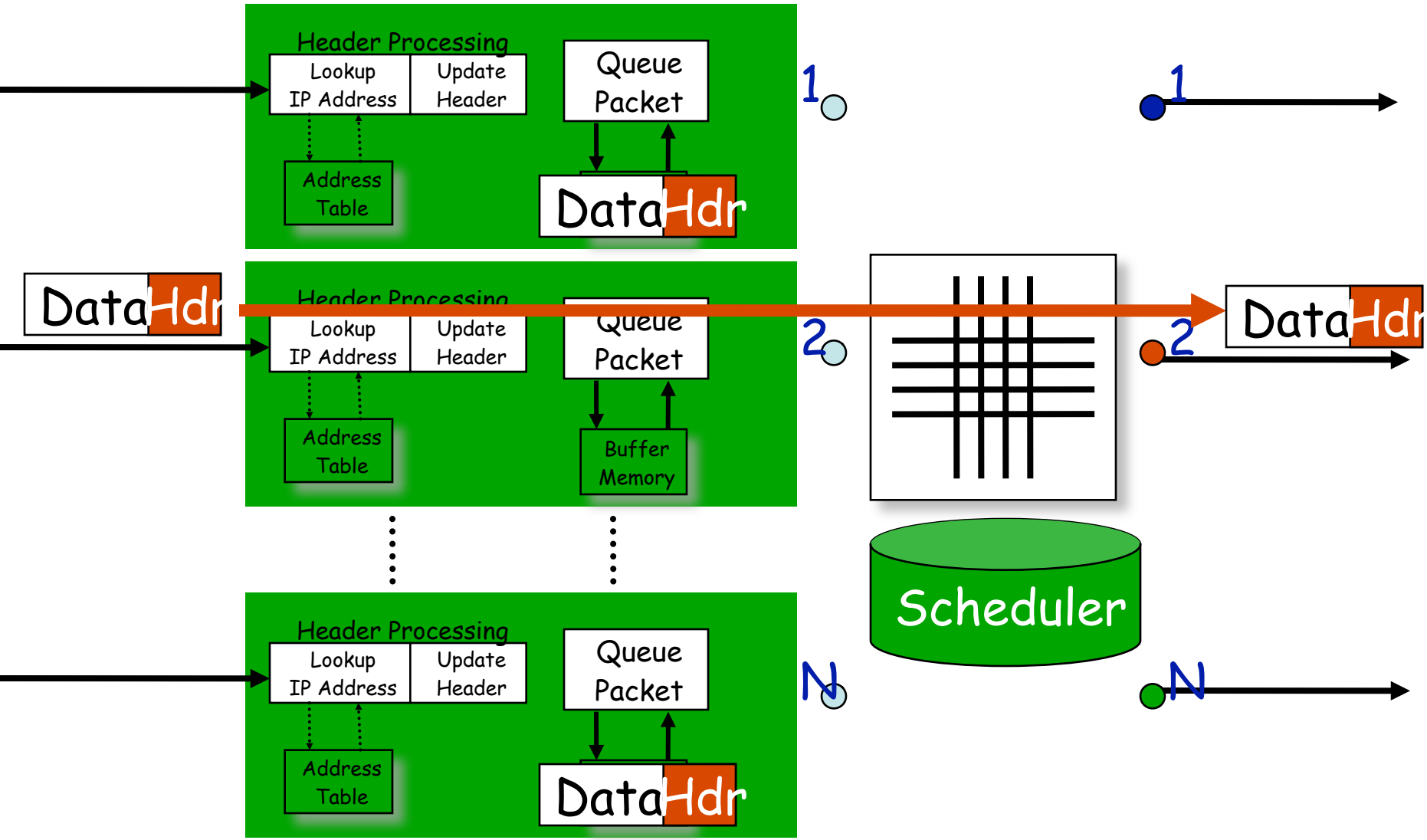
# Arquitecturas de routers

- Dos filosofias
  - Colas a la entrada
  - Colas a la salida
-

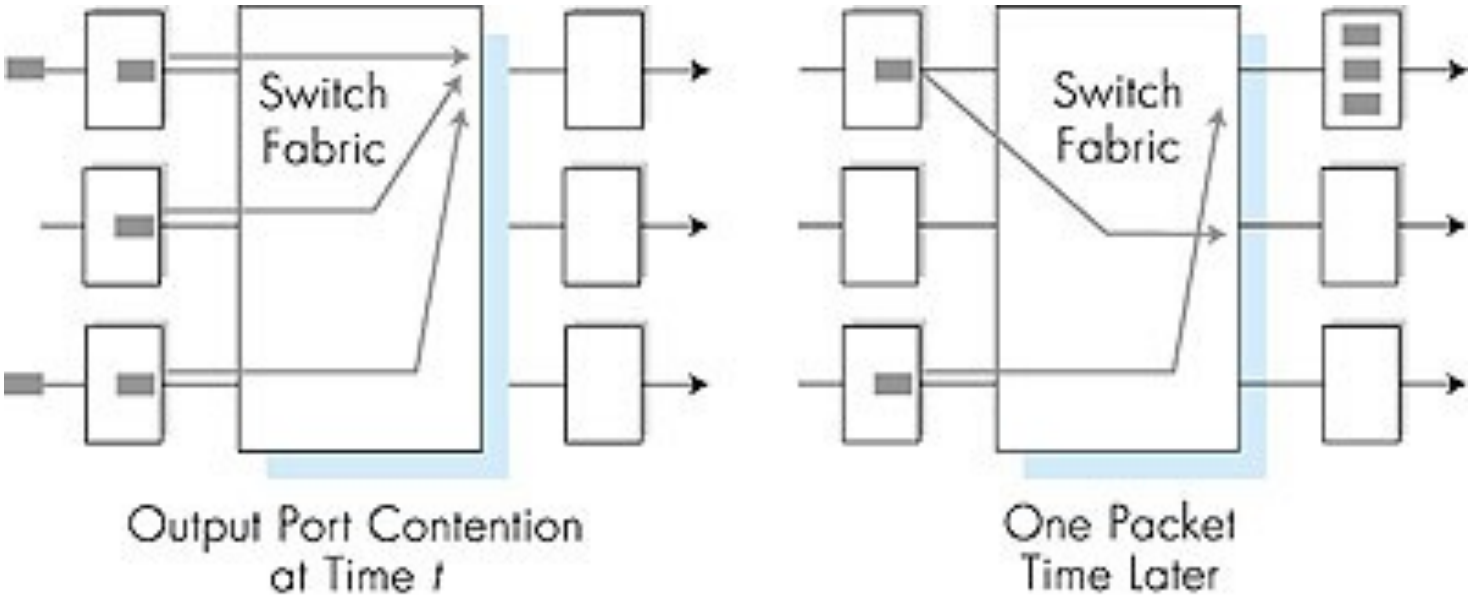
# Colas a la salida



# Colas a la entrada



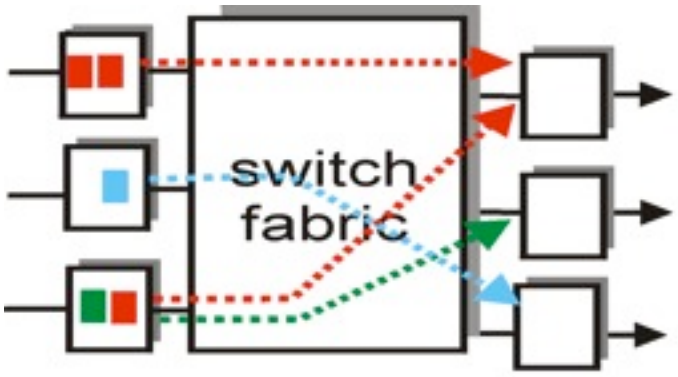
# Usando colas en los puertos de salida



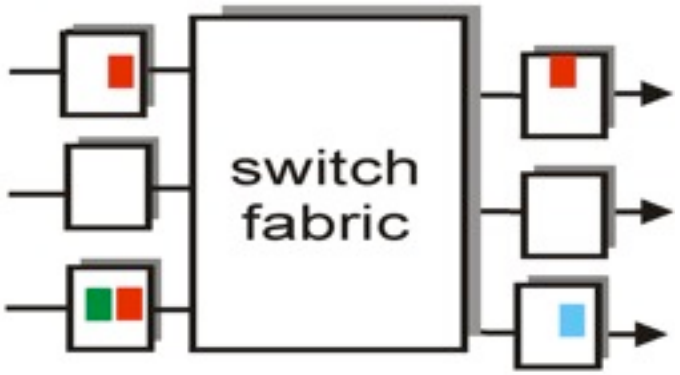

- Cuando la velocidad de llegada a traves de la matriz de conmutación es superior a la velocidad de salida se almacenan en buffer (cola de salida)
- *almacenamiento (retraso) y pérdidas debidas al desbordamiento del buffer del puerto de salida*
- Problema: *matriz de conmutación debe ser N veces mas rápida que el puerto de entrada/salida*

# Usando colas en los puertos de entrada

- Matriz de conmutación más lenta que la suma de los puertos de entrada -> hay que almacenar en el puerto de entrada
- Bloqueo Head-of-the-Line (HOL): un paquete que no puede ir a un puerto bloquea al resto de paquetes en el puerto de entrada, aunque podrían ser servidos
- *Retraso y pérdidas debidas a desbordamiento de puerto de entrada*
- Combinación de las dos técnicas para conseguir poco HOL con matrices de conmutación rápidas pero no tanto como la suma de los puertos




output port contention
at time t - only one red
packet can be transferred




green packet
experiences HOL blocking

# Arquitecturas de routers

- Colas a la salida
  - Al puerto de salida pueden llegar durante un tiempo mas paquetes de los que puede servir
  - Se plantea el problema de la elección de que paquete envío cuando se me acumulan
  - El problema se llama planificación (scheduling)
- Colas a la entrada
  - Al puerto de salida llegan paquetes a la velocidad de linea por lo que no tiene mucho donde elegir...
  - La planificación es necesaria para decidir cual de los puertos de entrada debo dejar que pase a la salida

- **Planificación/scheduling es un problema fundamental de redes**
  - Como elijo que cliente/usuario/entrada/flujo debo servir
  - Como gestiono la cola de peticiones

# Scheduling why?

- No es suficiente con elegir el primero?

  **FCFS first come first served**? cola unica elijo el primer paquete de la cola

  **DropTail** si voy a colocar un paquete en la cola y no hay sitio lo descarto

  – Puede no existir el concepto de el primero en llegar

    en el caso de colas a la entrada tenemos que elegir entre paquetes en varios puertos de entrada... cuál es el primero en llegar?

  – Podemos querer tratar de forma diferente a diferentes aplicaciones

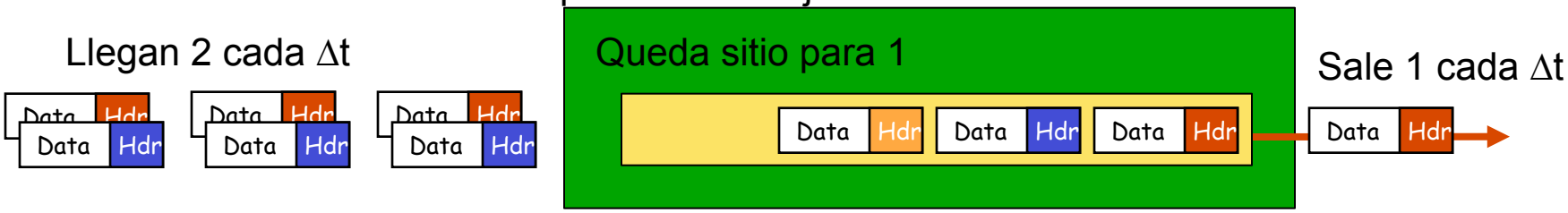
    **Aplicaciones elásticas** (mail, web, transferencia de ficheros)

    *No les importa mucho tardar unos pocos milisegundos mas en salir (best-effort)*

    **Aplicaciones con restricciones de tiempo** (VoIP, videoconf, juegos)

    *Milisegundos de mas marcan la diferencia entre funcionar o no*

  – Podemos querer tratar a todos los paquetes por igual (Fairness) y si no tenemos cuidado FCFS puede crear injusticias



# Principios de scheduling

- Requisitos
  - Facilidad de implementación

    La decisión debe tomarse por cada paquete

    Debe escalar con el numero de entradas/flujos/conexiónes

    Debe tener poco estado interno

    Normalmente se hará en hardware
  - Equidad(fairness) and protection

    Debe tratar por igual a las entradas

    incluso en condiciones en las que las entradas no se comportan correctamente no debe penalizar a las que si lo hacen
  - Requisitos de prestaciónes

    Ser capaz de garantizar limites arbitrarios de prestaciones para algunas entradas/flujos

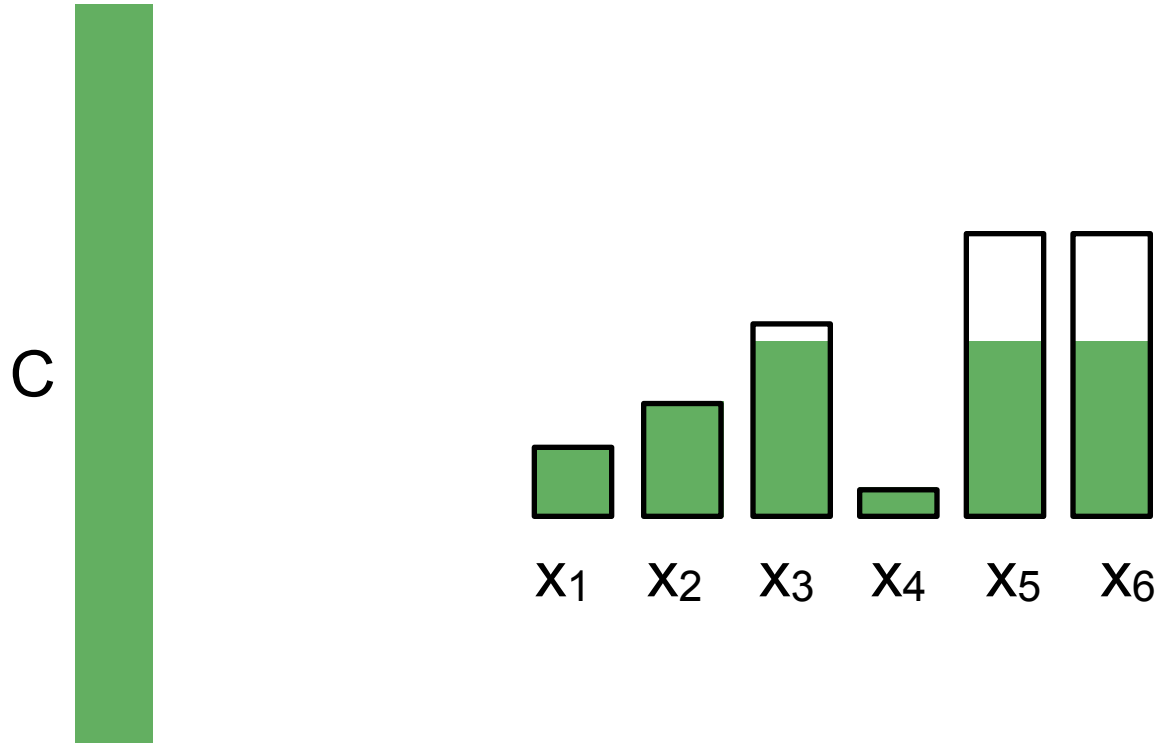
    Estos limites se garantizaran asignando recursos
  - Facilidad de control de admisión

    Ser capaz de decidir si permitir una nueva entrada/flujo permitira seguir cumpliendo los limites o no, para decidir si aceptarla

# Equidad max-min (max-min fairness)

- Propiedad deseable para un reparto de un recurso escaso
- El recurso tiene una capacidad a repartir C
- Las demandas de recurso de los diferentes usuarios son

  $\{x_1, x_2, x_3 ...\}$ tales que en general $x_1+x_2+x_2+... > C$
- Intuitivamente querríamos conceder su petición a los usuarios que piden poco y repartir por igual entre los usuarios que piden mucho



# Equidad max-min (max-min fairness)

- Algoritmo
- Ordenar las demandas de menor a mayor
- Supondremos que $\{x_1,x_2,x_3...\}$ están ordenadas de forma que $x_1<x_2<x_2<...$
- Asignamos C/N a cada usuario

  Si $x_1<C/N$ nos sobra $C/N-x_1$ para repartir entre el resto

  Sumamos $(C/N-x_1)/(N-1)$ a cada uno del resto

  $x_1$ ya tiene asignado la cantidad correcta tomamos el siguiente y repetimos el proceso

- Maximiza el mínimo asignado a las demandas que no son satisfechas

  No penaliza a los usuarios que se comporten bien y piden poco incluso en presencia de usuarios que piden mucho.

# Scheduling más simple

- FCFS + DropTail
- No hacemos distinciones

  Primero en entrar primero en salir

  Al colocar un paquete en cola si no hay sitio se elimina

- Utilización media del servidor (independiente del scheduling utilizado)

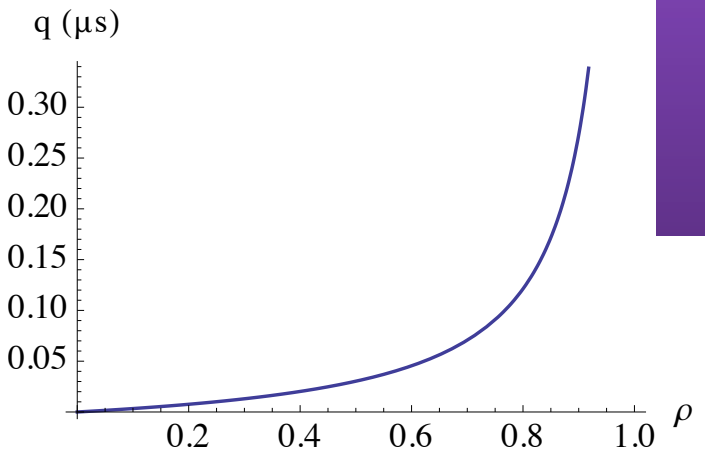
  **ρ=λ x**

  **λ** paquetes por unidad de tiempo

  **x** tiempo medio de servicio del paquete
- **q** : Tiempo medio de espera en la cola

  depende de la utilización

  cómo depende se discute en teoría de colas




- FCFS no hace distinción entre paquetes y todos tienen que esperar en media eso. ¿Podemos hacer otras disciplinas de scheduling que consigan que algunos paquetes esperen menos?

# Ley de conservacion

- Si que se puede conseguir menos tiempo de espera si separamos las llegadas y las tratamos de forma diferente

  En las próximas clases se discutirán métodos

  Pero... hay un limite

- Ley de conservación

  Si tenemos clases de usuarios i=0,1,2...

  Cada clase genera una carga $\rho_i=\lambda_i x_i$

  Cada clase obtiene un tiempo medio de espera $\mathbf{q_i}$

  Y el sistema es **conservativo de trabajo**=si hay paquetes para servir los sirve

  Se cumple que para cualquier disciplina de scheduling

$$\sum_{i=0}^{N} \rho_i q_i = constante$$

Para FCFS $$q \sum_{i=0}^{N} \rho_i = constante$$

- No podemos obtener un retardo medio menor que el que obtiene FCFS. Sólo podemos reducir el tiempo medio de espera de una clase aumentando el de otras

# Conclusiones

- Arquitectura de routers con plano de control y plano de datos
- Problemas que deben resolverse para construir el plano de datos
  - Address lookup (prefix-trees)
  - Acceso a memoria y calculo para manipulación de cabeceras
  - Conmutación de paquetes en el plano de datos

    Switching fabrics: buses, crossbar, broadcast, banyan...
  - Colas a la entrada o a la salida
  - El problema del scheduling

    Introducción: max-min fairness

    FCFS y ley de conservación

- Proxima clase: scheduling