

# Tecnología Digital IV: Redes de Computadoras

## Clase 21: Nivel de Enlace - Parte 3

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital  
Universidad Torcuato Di Tella

29 de Octubre de 2025

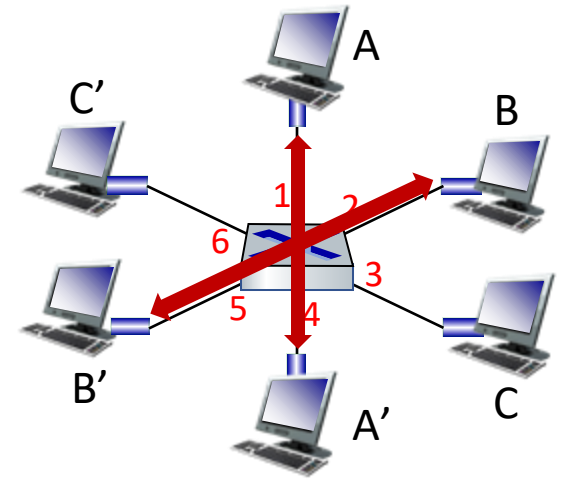
# Switches

# Switches en Ethernet

- Un **switch** es un dispositivo de nivel de enlace: toma un rol **activo**
  - Almacena y reenvía frames Ethernet
  - Inspecciona las MACs de los frames entrantes y los reenvía selectivamente a uno (o más) de los enlaces de salida
- Es **transparente**: los hosts no están al tanto de su presencia
- Es ***plug-and-play*** y “autodidacta”
  - No requieren configuración

# Switch: múltiples transmisiones en simultáneo

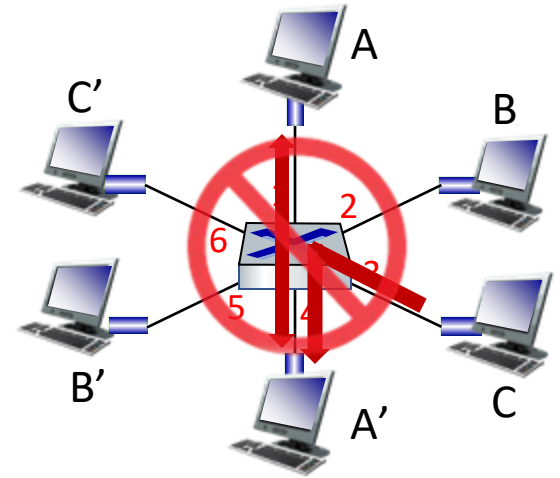
- Los hosts tienen una conexión dedicada y directa hacia un switch
- Los switches almacenan frames
- Se utiliza el protocolo Ethernet en cada enlace:
  - **Sin colisiones:** *full-duplex*
  - Cada enlace es un **dominio de colisión**
- **Switching:** pueden coexistir transmisiones de A hacia A' y de B hacia B', sin colisiones



switch con seis interfaces  
(1,2,3,4,5,6)

# Switch: múltiples transmisiones en simultáneo

- Los hosts tienen una conexión dedicada y directa hacia un switch
- Los switches almacenan frames
- Se utiliza el protocolo Ethernet en cada enlace:
  - **Sin colisiones:** *full-duplex*
  - Cada enlace es un **dominio de colisión**
- **Switching:** pueden coexistir transmisiones de A hacia A' y de B hacia B', sin colisiones
  - Pero **no** de A hacia A' y de C hacia A'



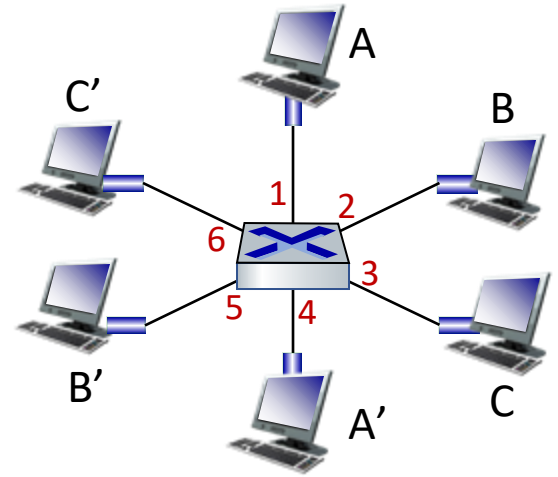
switch con seis interfaces  
(1,2,3,4,5,6)

# Tabla de *forwarding* en switches

¿Cómo sabe el switch que A' es alcanzable vía su interfaz 4 y B' vía la 5?

Cada switch tiene una **tabla**:

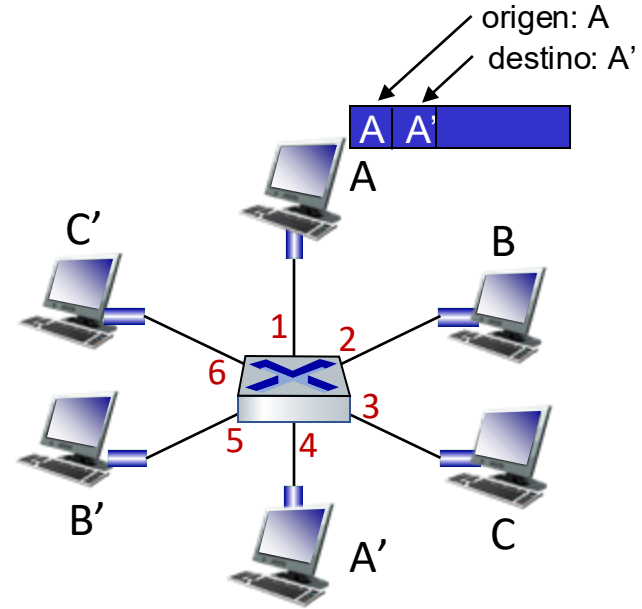
- Entradas de la forma  
(dirección MAC, interfaz, *timestamp*)
- Similar a una tabla de ruteo



¿Cómo se generan y administran las entradas en las tablas?

# Autoaprendizaje

- Los switches **infieren** qué hosts pueden alcanzarse a través de cuáles interfaces
- Al recibir un frame, el switch aprende la ubicación del emisor: el segmento de la LAN por el que llega el frame
- Guarda el par (emisor, ubicación) en su tabla



MAC	interfaz	TTL
A	1	60

tabla del switch  
(inicialmente vacía)

# Algoritmo de *forwarding* en switches

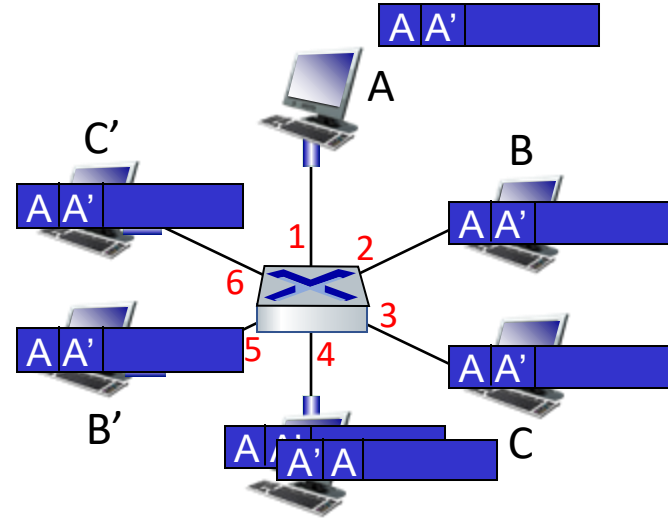
Al recibir un frame,

1. Registrar la MAC y el enlace de entrada del emisor
2. Indexar la tabla de *forwarding* empleando la MAC destino
3. Si existe una entrada en la tabla {  
    Si el destino está en el mismo segmento de entrada {  
        Descartar frame  
    } Si no {  
        Reenviar frame por la interfaz indicada en la entrada  
    }  
} Si no {  
    *Flooding*: reenviar el frame en todas las interfaces  
    (excepto la de entrada)  
}



# Autoaprendizaje y *forwarding*: ejemplo

- Destino del frame, A', desconocido  
*flood*
- Destino del frame, A, conocido  
*envío selectivo en un único enlace*

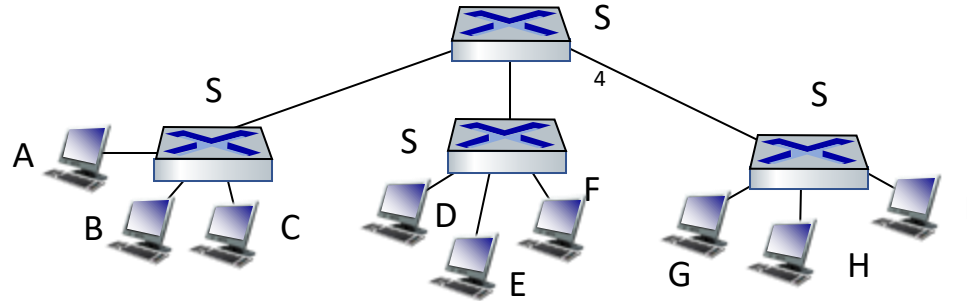


MAC	interfaz	TTL
A	1	60
A'	4	60

tabla del switch  
(inicialmente vacía)

# Interconexión de switches

Los switches pueden interconectarse:

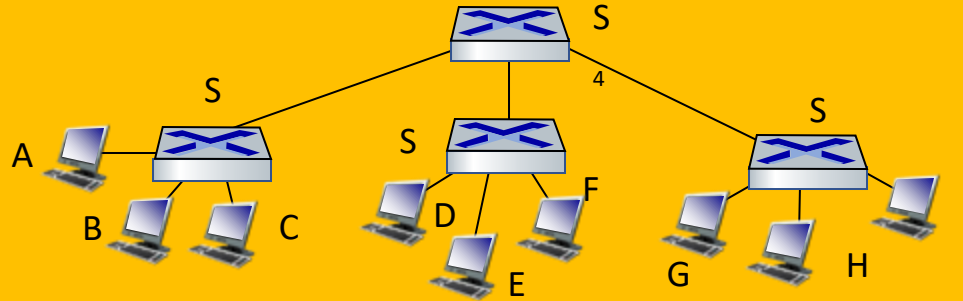


Al enviar un frame de A hacia G,  
¿cómo sabe  $S_1$  que debe reenviar el frame hacia  $S_4$ ?

- Vía **autoaprendizaje** (funciona exactamente igual que antes)
- ¿Qué sucede si la MAC destino no se encuentra en la red?

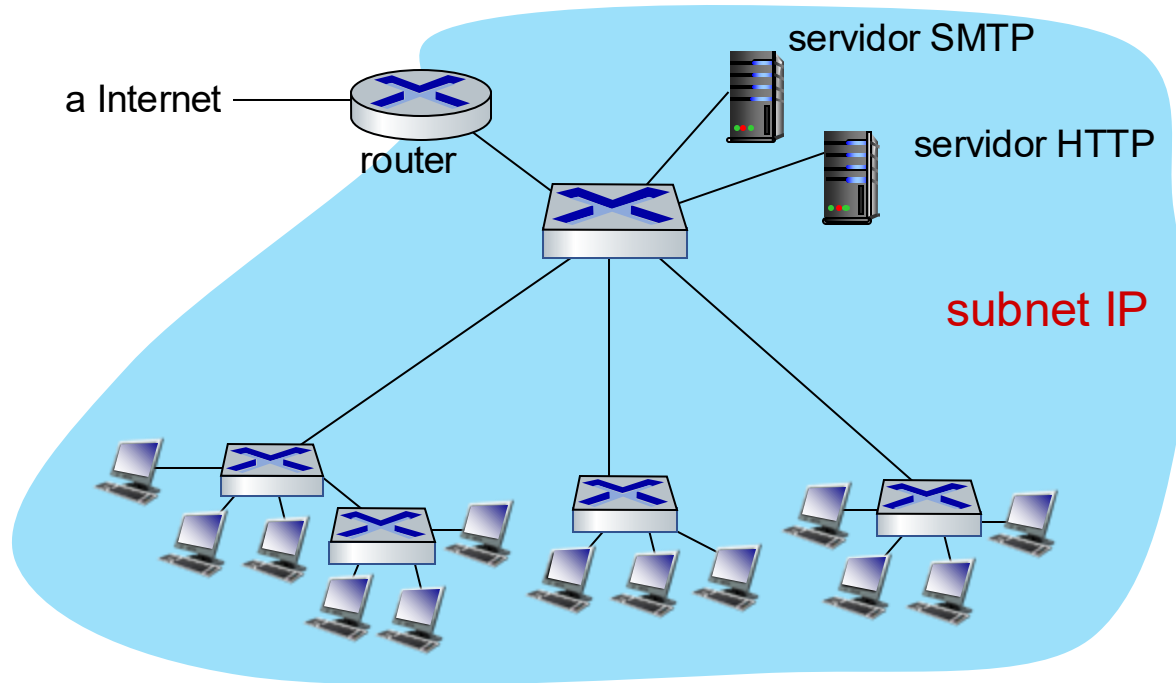
# Ejercicio!

Supongamos que C envía un frame a I y que éste le responde



Mostrar las tablas de *forwarding* y el reenvío de paquetes en los cuatro switches de la LAN

# Ejemplo: red corporativa *switchheada*



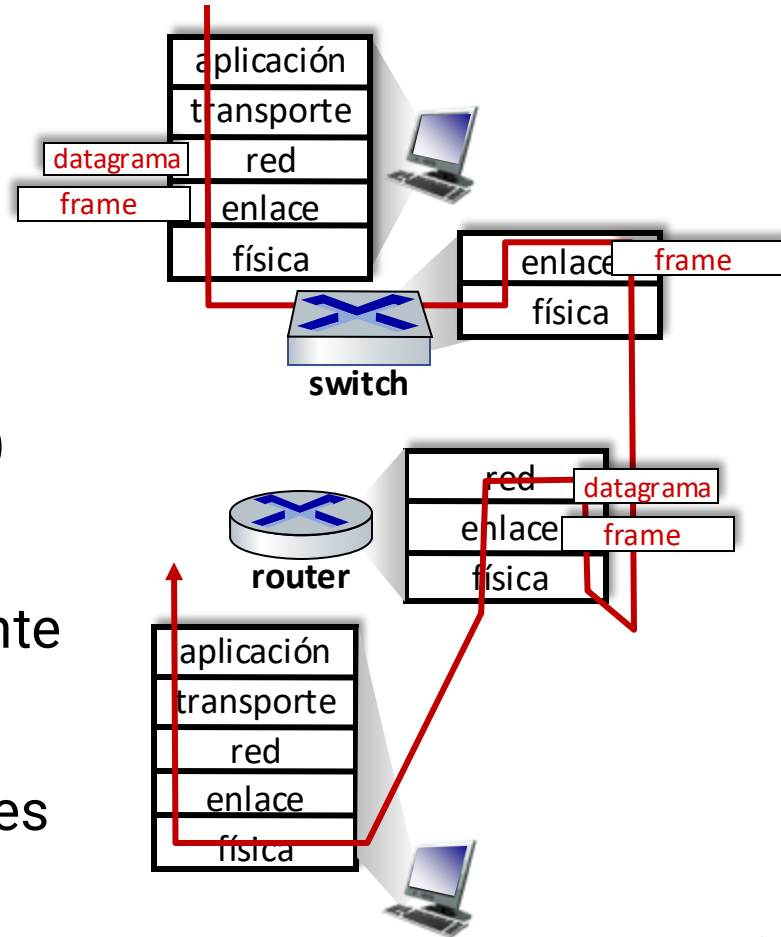
# Switches vs. routers

Ambos son *store-and-forward*

- Routers: dispositivos de nivel de red (examinan *headers* de red)
- Switches: dispositivos de nivel de enlace (examinan *headers* de enlace)

Ambos tienen *tablas de forwarding*

- Routers: computan las tablas mediante algoritmos de ruteo y direcciones IP
- Switches: aprenden las tablas vía *flooding*, autoaprendizaje y direcciones MAC



# Redes de datacenters

# Redes de datacenters

Decenas o cientos de miles de hosts acoplados y en proximidad

- Comercio electrónico (e.g. Amazon)
- Servidores de contenido (e.g., YouTube, Akamai, Apple, Microsoft)
- Motores de búsqueda (e.g., Google)

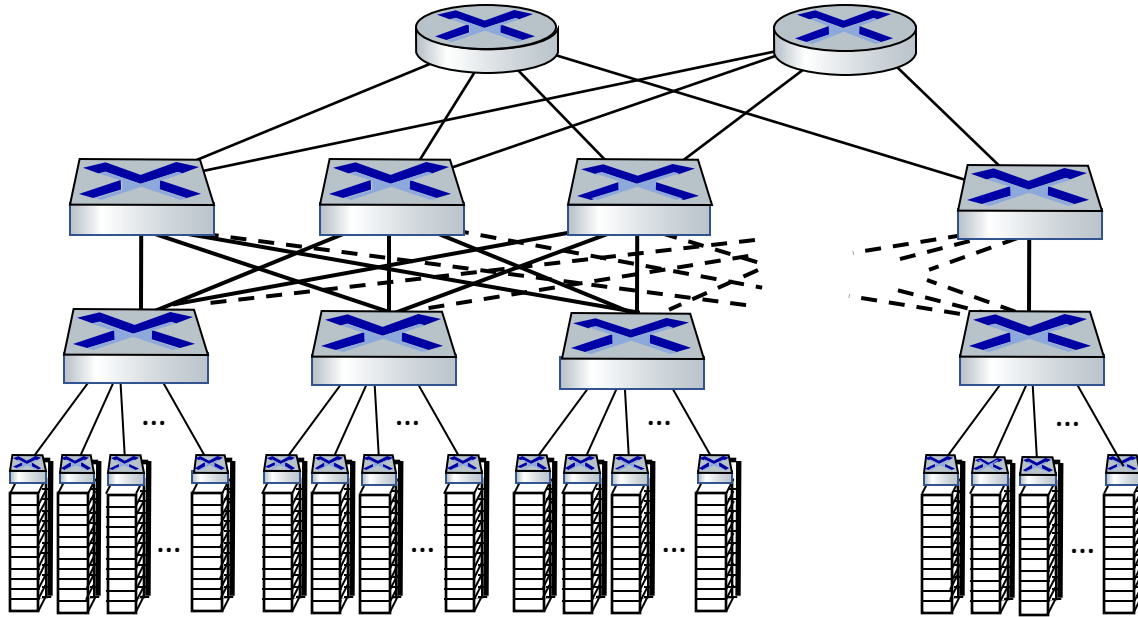
Desafíos:

- Múltiples aplicaciones, cada una sirviendo una alta cantidad de clientes
- Confiabilidad
- Balanceo de carga, evasión de *bottlenecks* (e.g. de *networking*)



Datacenter de Microsoft

# Redes de datacenters: topología



## *Border routers*

- Conectividad hacia el exterior

## *Switches tier-1*

- Conectando ~16 T-2s debajo

## *Switches tier-2*

- Conectando ~16 TORs debajo

## *Switch TOR (Top of Rack)*

- Uno por rack
- Ethernet de 40-100Gbps

## *Racks de servidores*

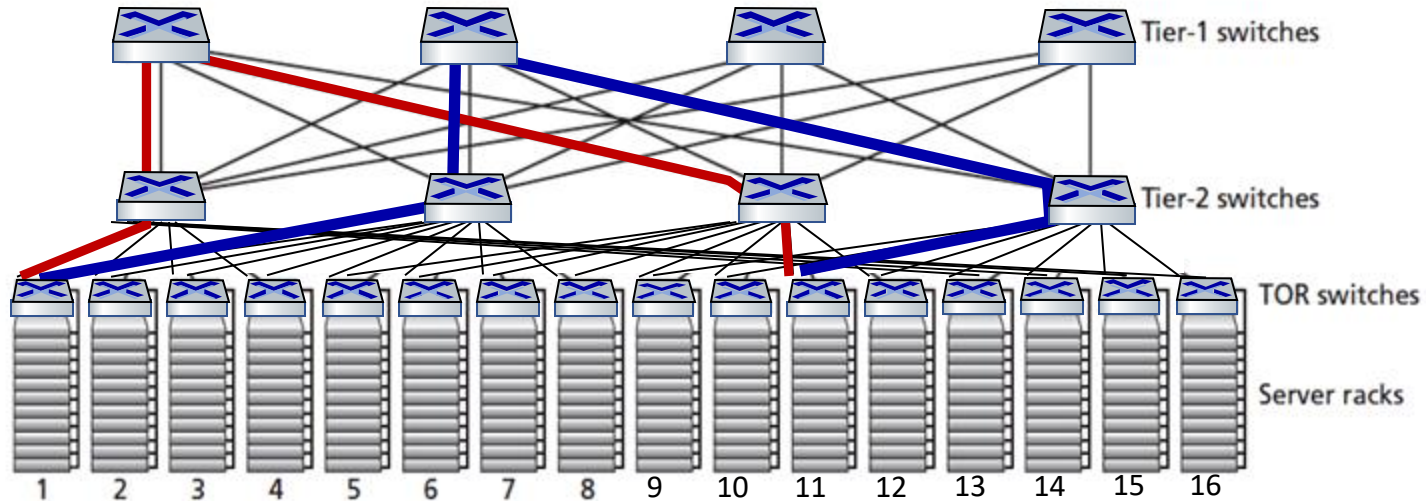
- 20-40 *blades*: hosts





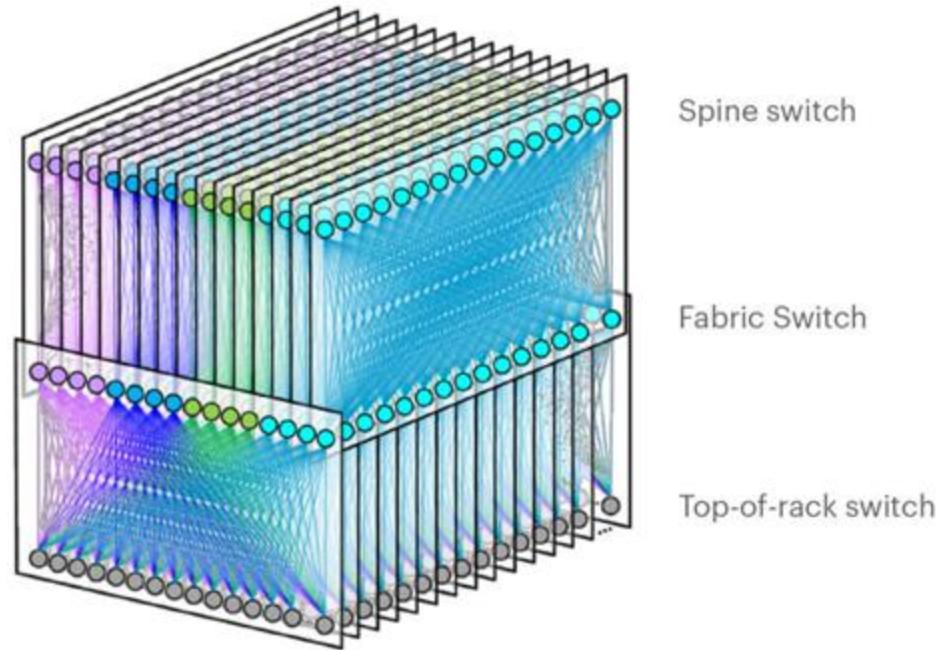
# Redes de datacenters: *multipath*

- Interconexión abundante entre switches y racks
  - Más throughput entre *racks* (múltiples rutas posibles)
  - Más confiabilidad por redundancia



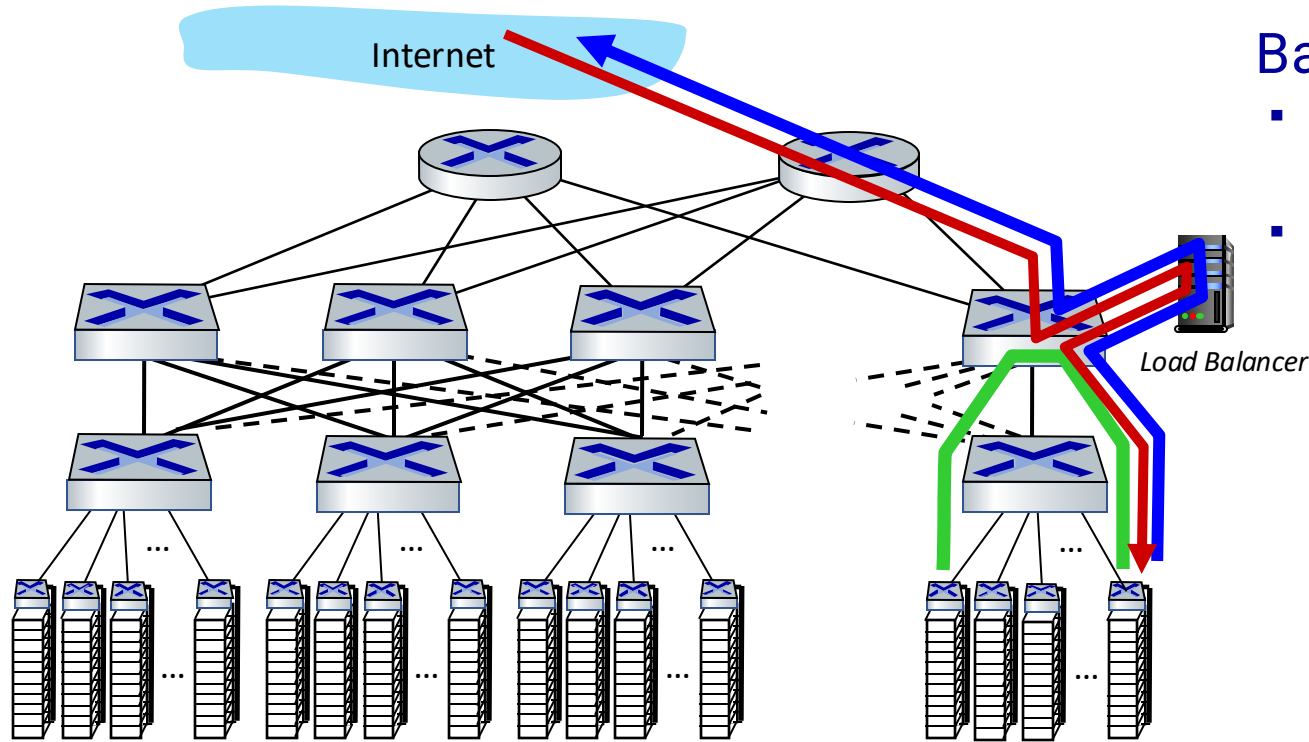
Dos caminos **disjuntos** entre los *racks* 1 y 11

# Ejemplo: red F16 (Facebook)



<https://engineering.fb.com/data-center-engineering/f16-minipack/> (marzo de 2019)

# Redes de datacenters: ruteo a nivel aplicación



## Balancedador de carga

- Recibe *requests* de clientes externos
- Dirige el tráfico dentro del datacenter
- Devuelve los resultados al cliente (escondiendo la complejidad del datacenter del cliente)