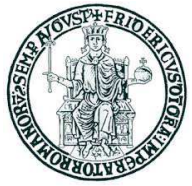
Cloud e Datacenter Networking

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
Laurea Magistrale in Ingegneria Informatica

Prof. Roberto Canonico

Datacenter: l'infrastruttura di networking

Parte II

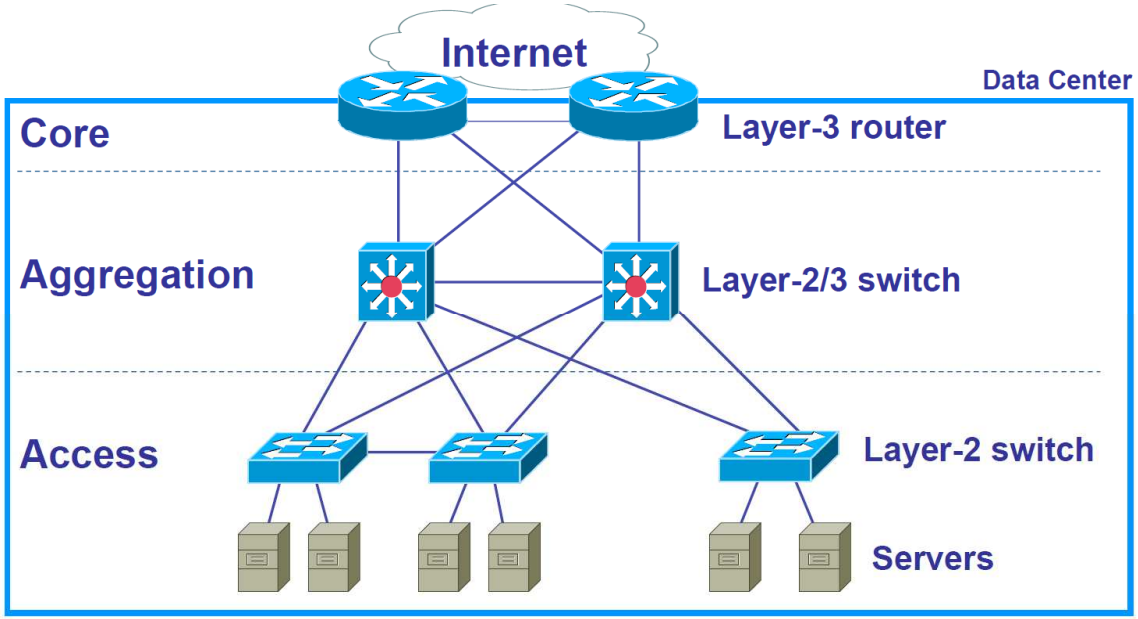


Argomenti della lezione

- Gestione dei loop in reti Ethernet
- STP
- Limiti delle topologie tradizionali di rete di un datacenter
- Reti di datacenter: sfruttamento di percorsi alternativi
- TRILL
- ECMP

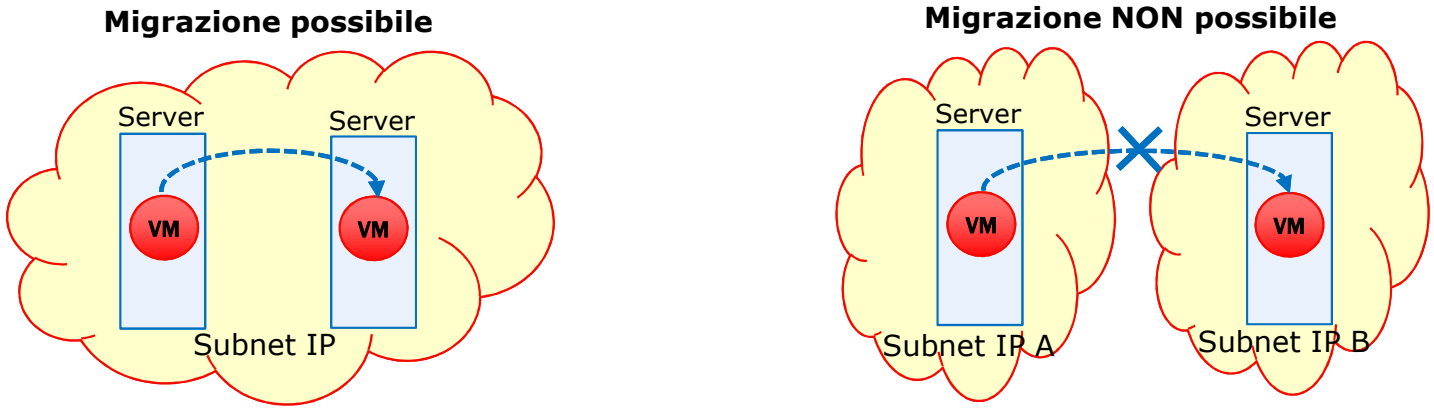
Architettura di networking di un DC

- I computer di un datacenter sono organizzati in rack per semplicità di gestione e cablaggio e per un utilizzo efficiente dello spazio
- L'infrastruttura di rete del datacenter è tipicamente organizzata in maniera gerarchica
- Le schede di rete dei server (2/4 per server) sono collegate ad una infrastruttura detta *access layer*
- Gli switch dell'*access layer* sono, a loro volta, collegati tra loro da una infrastruttura detta *aggregation layer*
- L'intero datacenter è collegato ad Internet attraverso una infrastruttura detta *core layer* che opera tipicamente a livello 3 (*routing IP*)



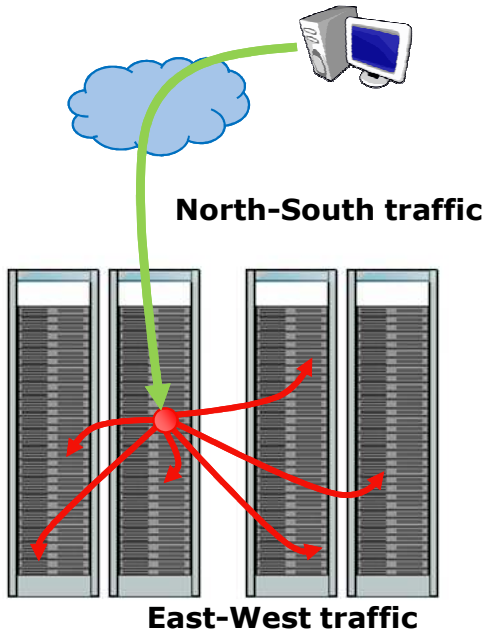
Datacenter networking e virtualizzazione: primi cenni

- Un utilizzo efficiente delle risorse computazionali dei server oggi si ottiene grazie ad un uso intensivo delle tecnologie di host virtualization
  - KVM, Xen, Vmware, …
- Un singolo host può ospitare anche decine di macchine virtuali (VM) ciascuna dotata di una o più interfacce di rete virtuali identificate da propri MAC address ed indirizzi IP
- Le moderne tecnologie di virtualizzazione consentono la migrazione di una VM tra host fisici differenti con trascurabili tempi di downtime (*live migration*)
- Requisito per la live migration:
  - la VM deve conservare il proprio indirizzo IP nella nuova posizione
- Se l'architettura di rete del datacenter partiziona l'intera topologia in cluster assegnati a subnet IP distinte, questo rappresenta un impedimento alla migrazione delle VM



Datacenter traffic analysis

- Osservazione: nei datacenter di grandi dimensioni il traffico tra macchine all'interno del DC (*East-West traffic* o *Machine-to-Machine traffic, m2m*) è preponderante rispetto al traffico che il DC scambia con l'esterno (*North-South traffic*)
  - Facebook: il traffico m2m raddoppia in meno di un anno
- Spiegazione: nelle moderne applicazioni cloud un'unica interazione generata dall'utente (client) produce molteplici query ed elaborazioni lato server
  - Es. suggerimenti nei box di ricerca, inserti pubblicitari "mirati", mashup di servizi ed informazioni estratte da varie basi di dati, ecc.
  - Solo una frazione dei dati prodotti si traducono in contenuti restituiti al client
  - Esempio osservato sulla rete di FB (*):
    una singola HTTP request ha prodotto
    88 cache lookups (648 KB),
    35 database lookups (25.6 KB), e
    392 remote procedure calls (257 KB)
- Conclusione: l'infrastruttura di collegamento tra i rack NON deve essere un collo di bottiglia

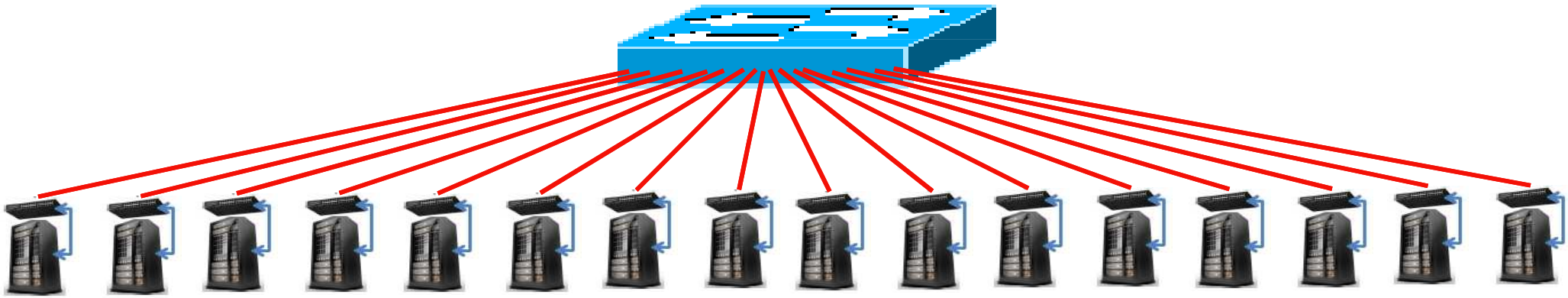


(*) N. Farrington and A. Andreyev. ***Facebook's data center network architecture***. In *Proc. IEEE Optical Interconnects*, May 2013

Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. **Inside the Social Network's (Datacenter) Network.** *SIGCOMM Computer Communications Review,* 45, 4 (August 2015), pp. 123-137

Aggregation layer: schema ideale

- Per consentire il massimo troughput nelle comunicazioni tra le macchine del datacenter, l'aggregation layer, idealmente dovrebbe essere un unico grande switch non bloccante che collega tutti gli switch dell'access layer
  - Soluzione non scalabile per l'elevato *throughput* richiesto alla matrice di commutazione
  - Le architetture di DC networking sono gerarchiche
- Se l'interconnessione tra i rack non garantisce il throughput necessario, le prestazioni del sistema degradano



- Esempio: si può realizzare un cluster di 1280 server organizzati in 32 rack da 40 macchine, con un uplink dagli switch ToR formato da 4 link da 10 Gb/s ed un unico switch aggregation con 128 porte da 10 Gb/s (costo ≈ USD 700000 nel 2008)
  - Se i server hanno link da 1 Gb/s l'oversubscription totale è 1:1 cioè la rete è non bloccante

Topologie ad albero multi-livello

- Per limitare i fenomeni di congestione, occorre limitare l'*oversubscription*
  - Ciò si realizza realizzando i collegamenti *aggregation* - *core* mediante fasci di link paralleli
- La figura illustra una topologia ad albero
  - Ogni switch è collegato ad un solo switch di livello gerarchico superiore
  - Gli switch di livello gerarchico superiore devono avere molte porte

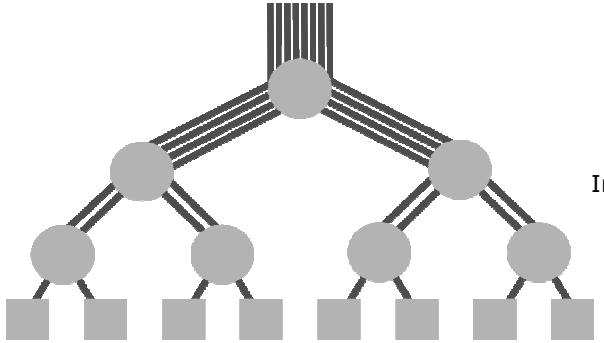
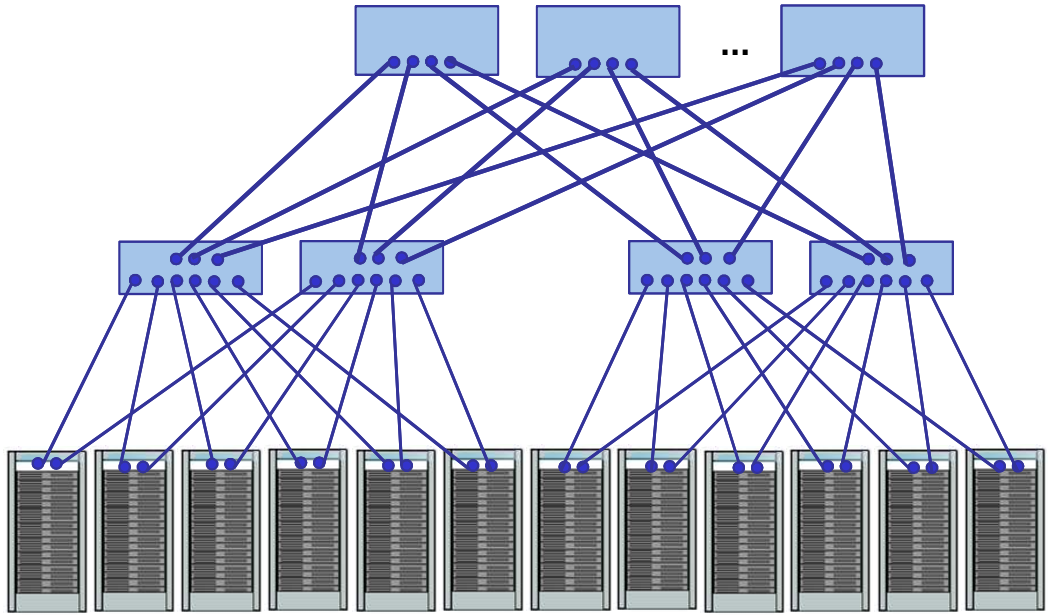


Immagine di Konstantin S. Solnushkin (www.clusterdesign.org)

- Per sfruttare la banda resa disponibile da link paralleli occorrono soluzioni di link aggregation (es. IEEE 802.3ad)
- I collegamenti tra livelli distinti sono soggetti ad un certo grado di *oversubscription* N:1 (N>1)
  - Ne consegue, per talune matrici di traffico, la possibilità che si verifichino congestioni
- Ciò impone dei vincoli nel deployment delle applicazioni sui server del DC
  - Server che comunicano più intensamente devono essere "più vicini"
- Requisito: avere una capacità distribuita uniformemente, senza vincoli sul traffico (*elasticity*)
- Un numero di livelli maggiore comporta una maggiore latenza
  - Impatto negativo sul throughput di TCP

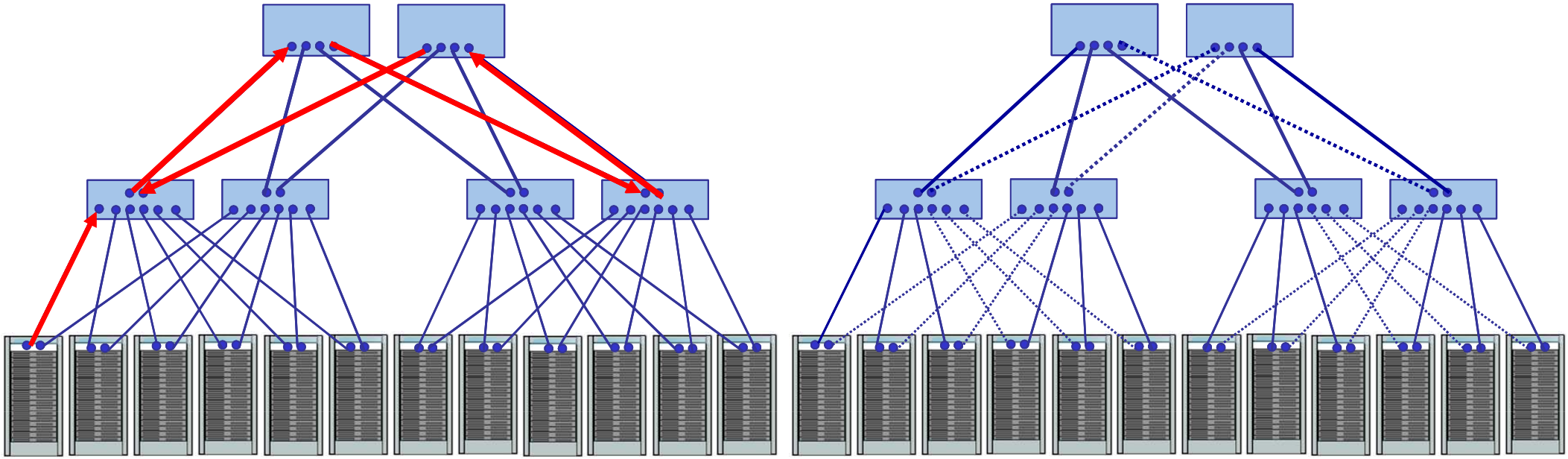
Evoluzione della architettura di rete dei DC

- L'approccio basato su un livello di aggregazione formato da pochi "grandi" switch con un elevato numero di porte non è scalabile
  - Se l'oversubscription tra i livelli diventa eccessivo, il sistema può essere soggetto a congestioni
- Da topologie ad albero si è passati a *topologie multi-rooted con molteplici percorsi alternativi tra due end-system,* in cui uno switch è collegato
  - ad uno switch di livello superiore con molteplici collegamenti uplink paralleli
  - a molteplici switch di livello superiore
- In questo modo:
  1. si può distribuire il traffico in uplink su più collegamenti distinti (ovvero si riduce l'*oversubscription*)
  2. si aumenta la resistenza ai guasti: esistono percorsi alternativi in caso di guasto di uno switch/link



Topologie di rete e loop

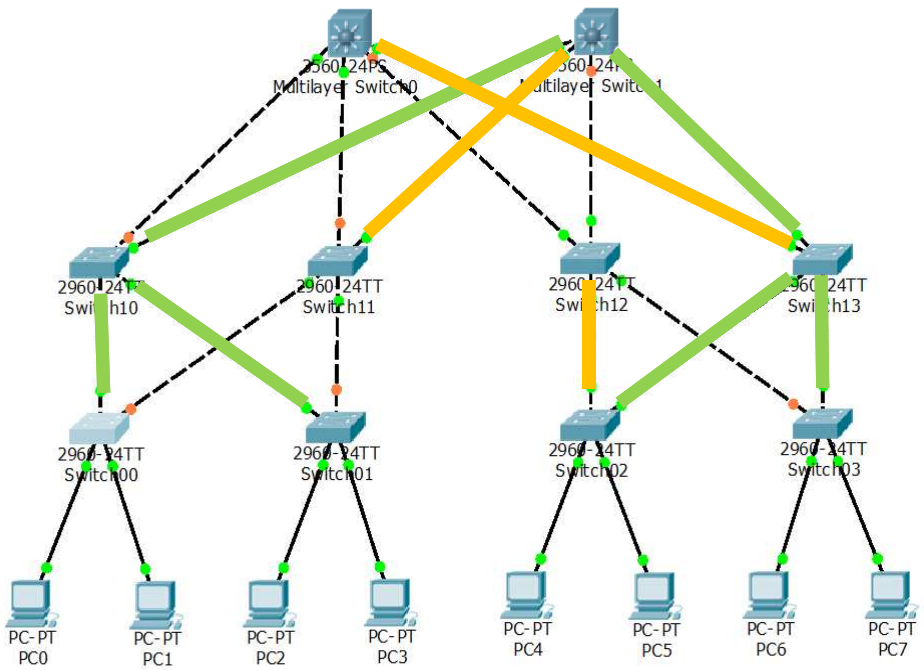
- La topologia in figura ha un problema: presenta dei loop !

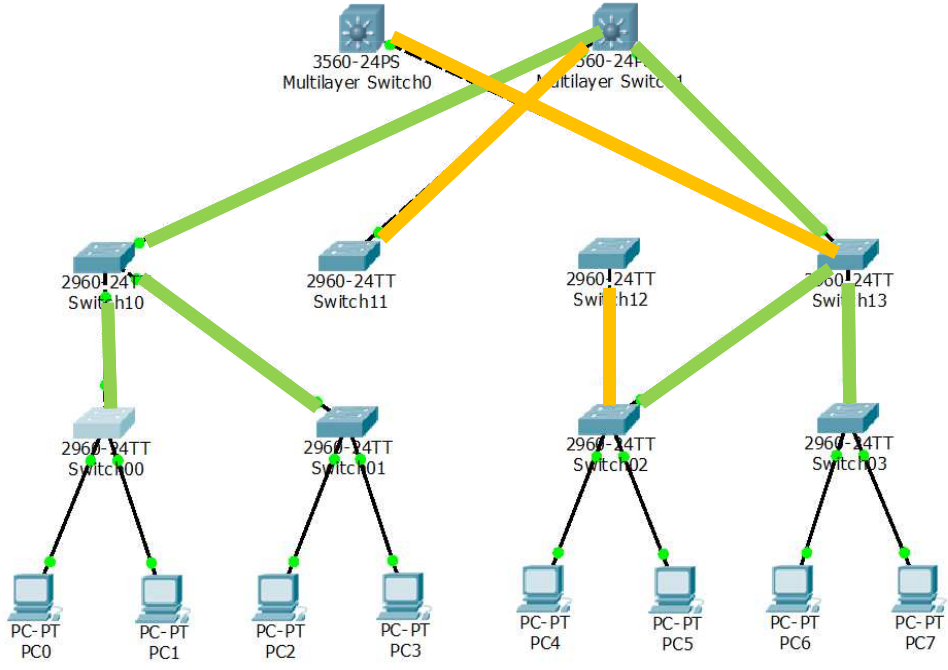


- Gli switch operano a livello 2 (no TTL)
- Occorrono meccanismi che, in presenza di loop, evitino la circolazione all'infinito dei pacchetti
- Approccio tradizionale per le reti Ethernet: STP
  - protocollo standard IEEE 802.1D inventato da Radia Perlman
  - per eliminare loop, la topologia della rete si riduce ad un albero (topologia loop-free) disattivando un sottoinsieme dei link della rete
  - Inconveniente: solo una frazione della capacità della rete risulta effettivamente utilizzabile e non si risolvono i problemi di oversubscription
- Soluzioni alternative: TRILL, FabricPath (Cisco), VCS (Brocade), M-LAG, QFabric, SPB, ...

Example of a DC network with STP in action

- Switches decide which interfaces should be switched off to prevent loops
- One end of a disabled link is turned off while the other is still on
- This results in a *spanning tree* connecting all the end systems as well as all the switches without any loop





STP: Spanning Tree Protocol

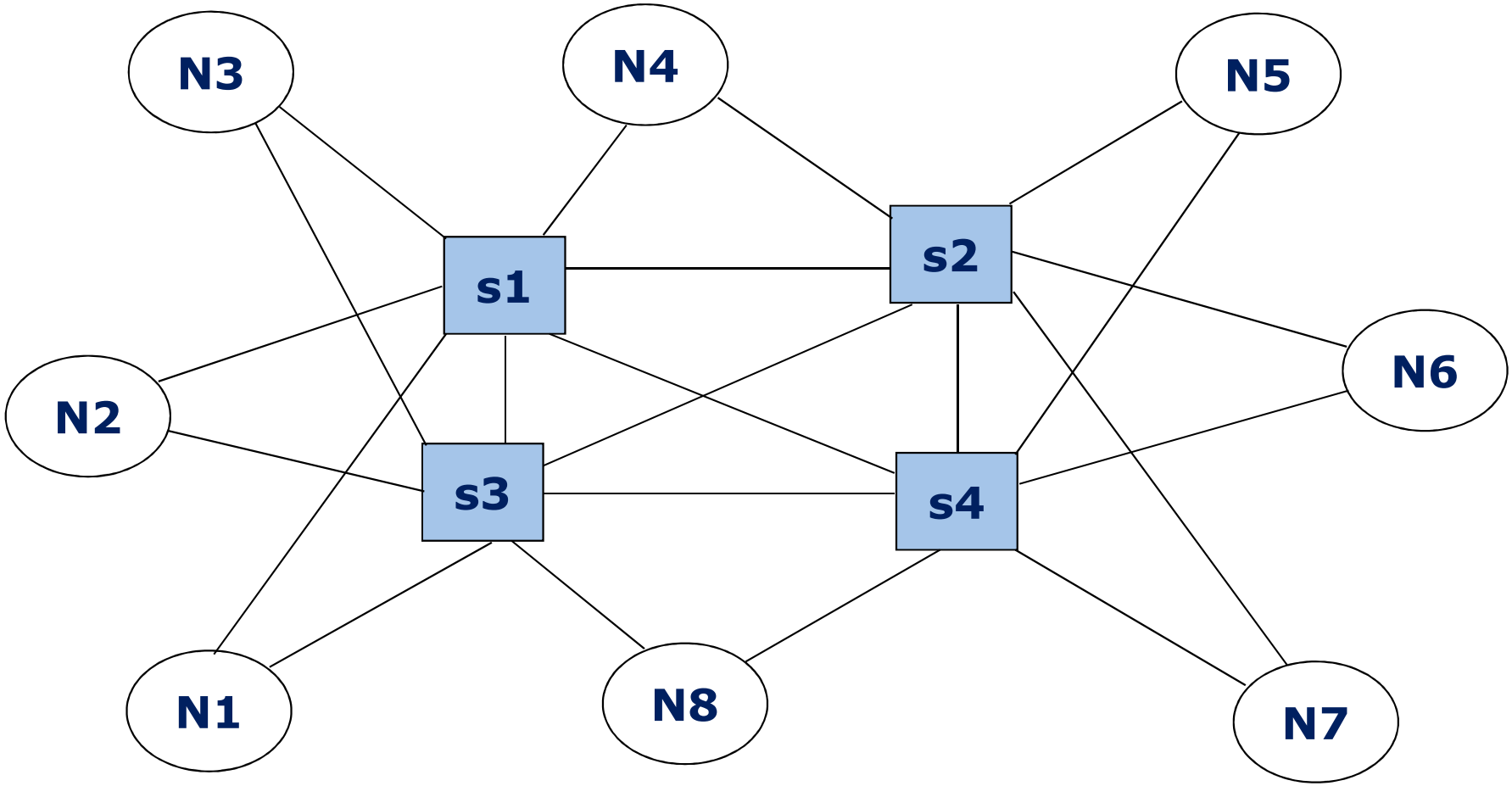
- Switches periodically exchange Configuration *Bridge Protocol Data Units* (BPDUs) to build the topology database
  - BPDU's are forwarded out all ports every 2s, to the dedicated MAC multicast address of 01:80:C2:00:00:00
  - Configuration BPDUs contain the switch *bridge ID*
- Each bridge starts out thinking it is the Root bridge
- Eventually, all switches agree that the Root bridge is the switch with smallest bridge ID
- Through BPDU exchanges, tree converges, which means all switches have same view of the spanning tree
- Each port of a switch may be in one of the following states:
  - Forwarding, Blocking, Listening, Learning

STP: how it works

STP creates a tree that provides a single unique path to each destination as follows:

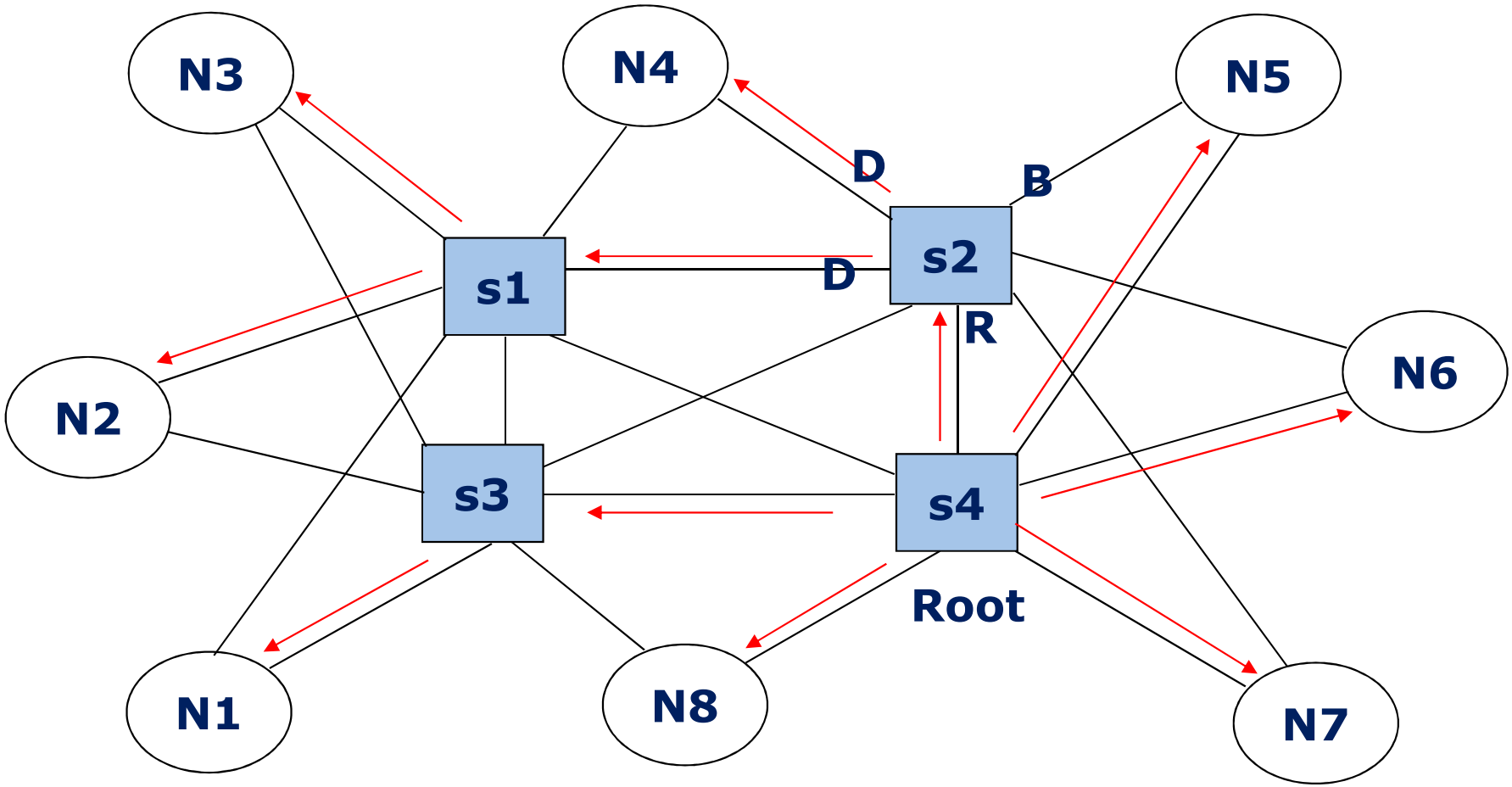
- switches elect a root bridge acting as the root of the spanning tree
- each bridge calculates the distance of the shortest path to the root bridge
- each bridge determines a *root port*, which will be used to send packets to root
- for each segment, a *designated port* is identified, i.e. the port closest to the root
- root ports and designated ports are set to *forwarding* state;
  all other ports are set to *blocking* state
  - Packets will not be received or forwarded on blocked ports

Spanning Tree Protocol in action (1)



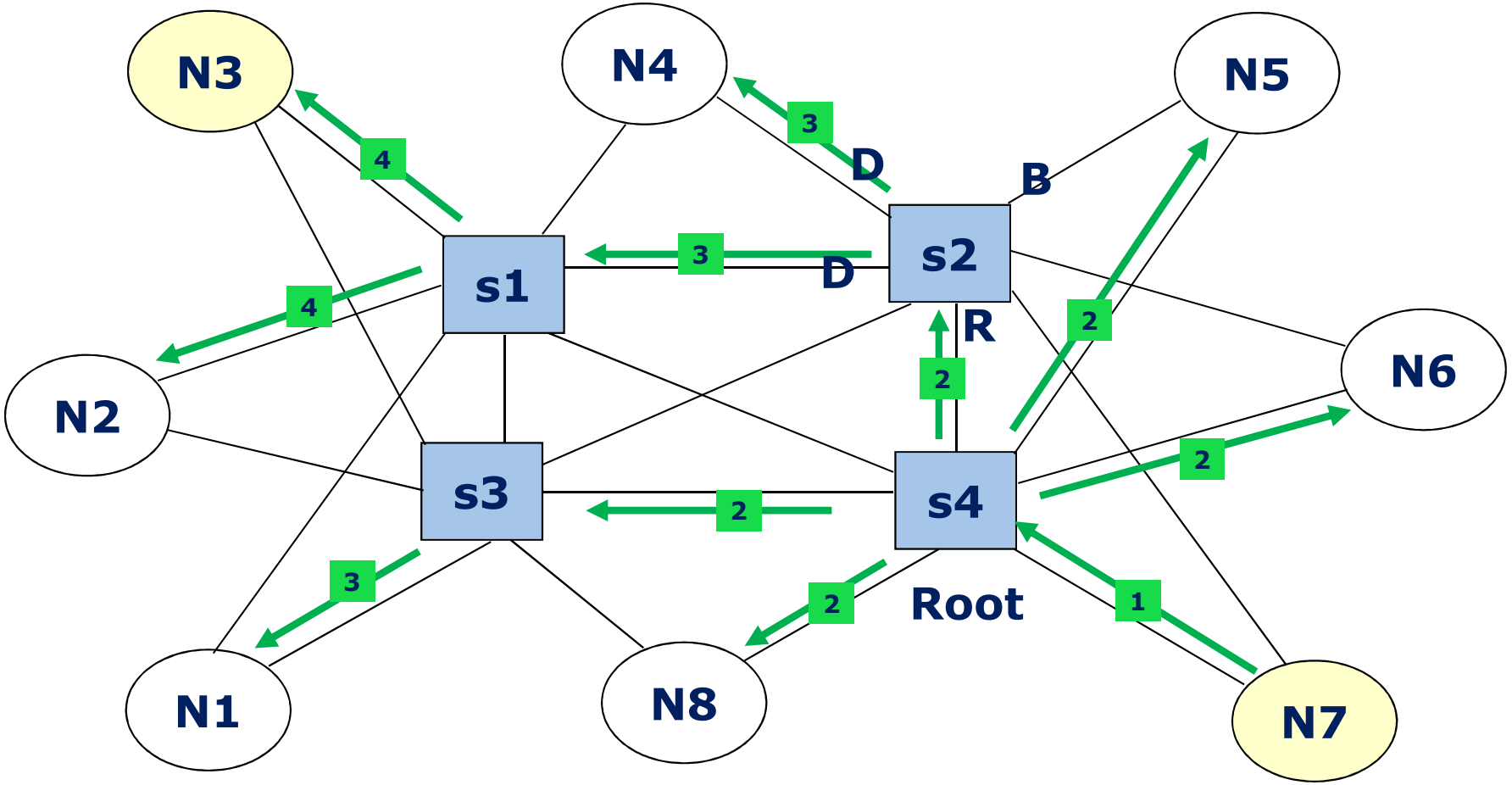
Spanning Tree Protocol in action (2)

- Having elected s4 as the Root Bridge, this is the resulting spanning tree
- Red arrows show the path of a broadcast packet from the root to any possible destinations
- Links not marked with an arrow are not traversed by any packets



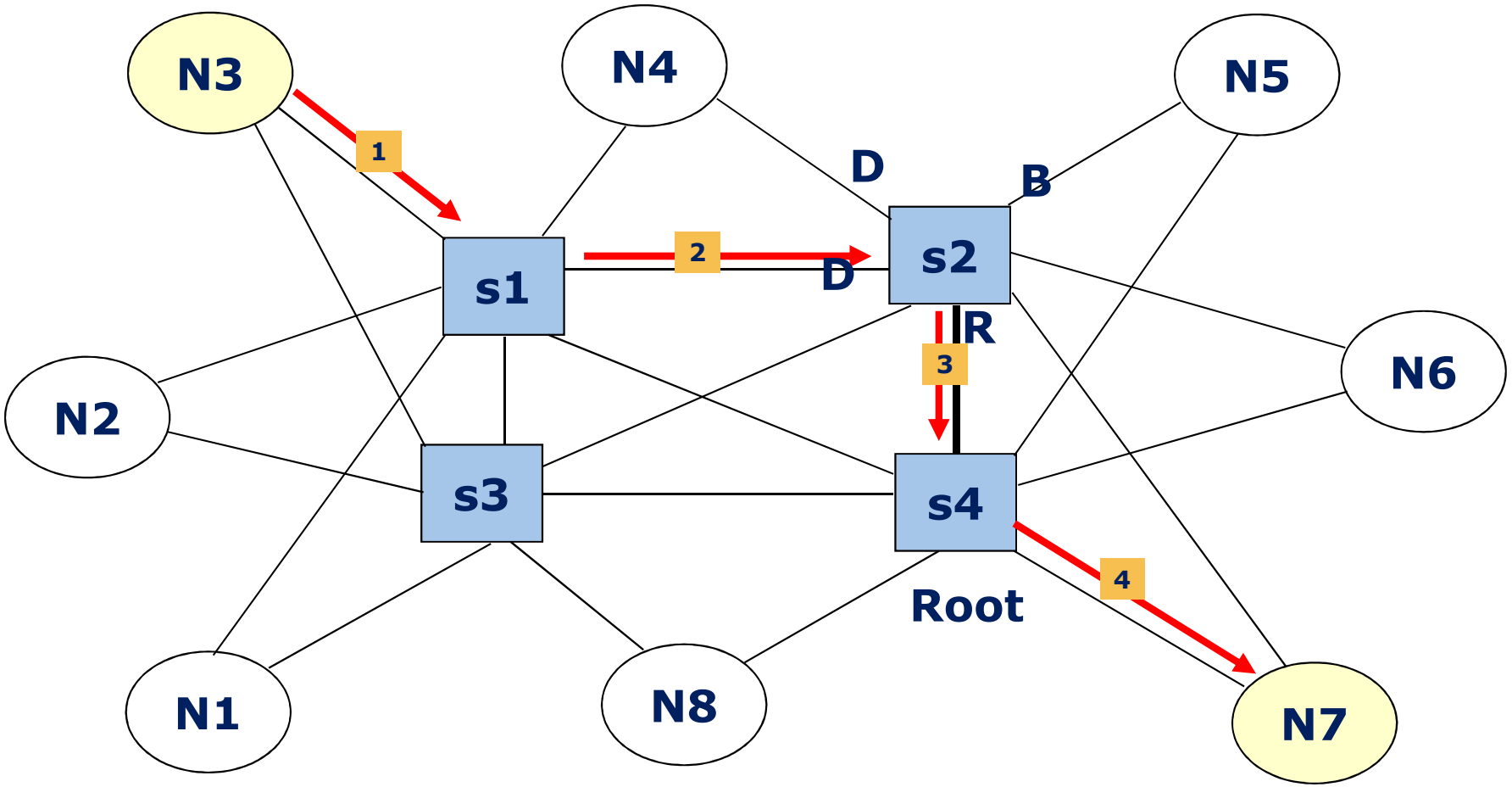
Spanning Tree Protocol in action (3)

- N7 sends in broadcast (i.e. to FF:FF:FF:FF:FF) an ARP request querying for N3's MAC
  - Packet is sent to the root s4 and from the root down the spanning tree towards all destinations
- N3 replies to N7 with its own MAC address
  - switches have learned where N7's MAC is located from the previous transmission



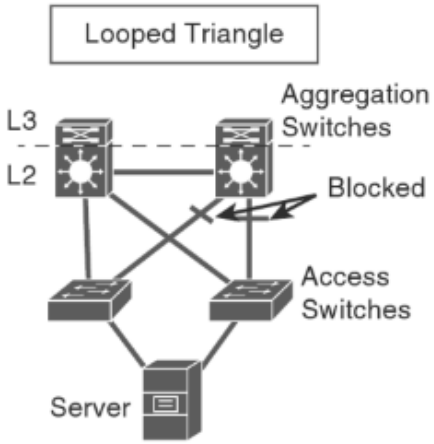
Spanning Tree Protocol in action (4)

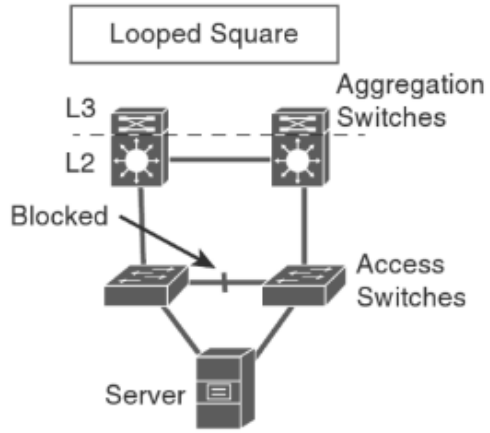
- N7 sends in broadcast (i.e. to FF:FF:FF:FF:FF) an ARP request querying for N3's MAC
  - Packet is sent to the root s4 and from the root down the spanning tree towards all destinations
- N3 replies to N7 with its own MAC address
  - switches have learned where N7's MAC is located from the previous transmission

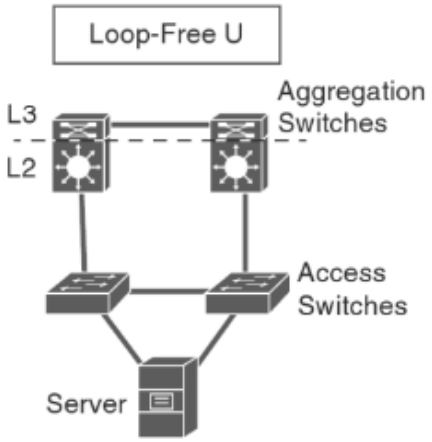


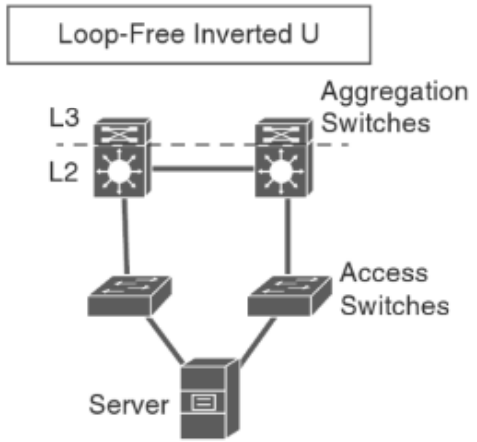
Access-Aggregation connection options

- Looped Triangle topology
  - STP blocks 2 uplinks out of 4
- Looped Square
  - STP blocks 1 horizontal link
  - In case of failure of an uplink, traffic is routed to the adjacent access switch → oversubscription doubles
- Loop-Free U
  - Communication between aggregation switches is L3
  - No loops → no links blocked by STP
- Loop-Free Inverted U





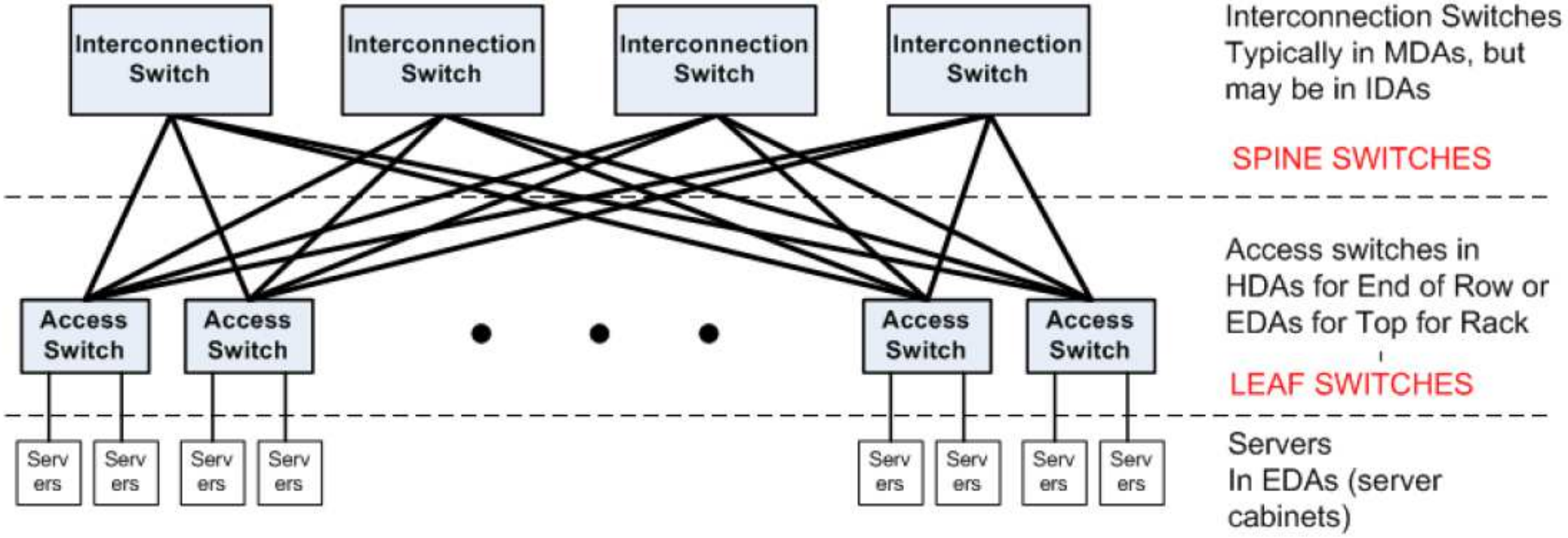




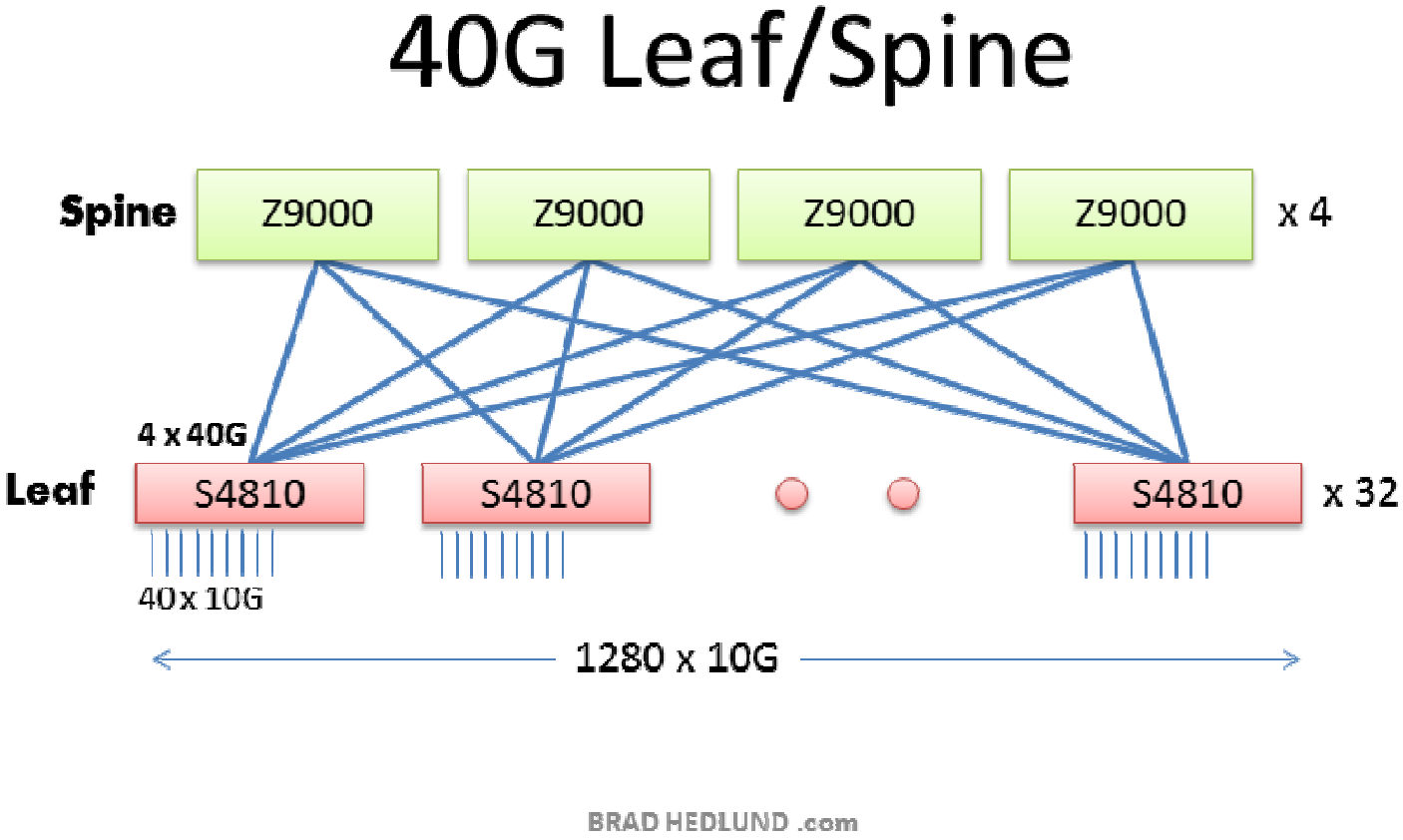
Source: Cloud Computing: automating the virtualized data center. Gustavo A. A. Santana. CISCO Press (2014)

Topologie multi-rooted Leaf-Spine

- Dette anche fat-tree, derivano dalle reti di Clos (*folded Clos*)
- Due livelli, ogni switch *leaf* è collegato a tutti gli switch *spine*
- Uno dei suoi vantaggi è la elasticità:
  - se aumenta il numero di rack, aumenta il numero degli switch leaf
  - se occorre incrementare la capacità della rete, si può aumentare il numero di spine

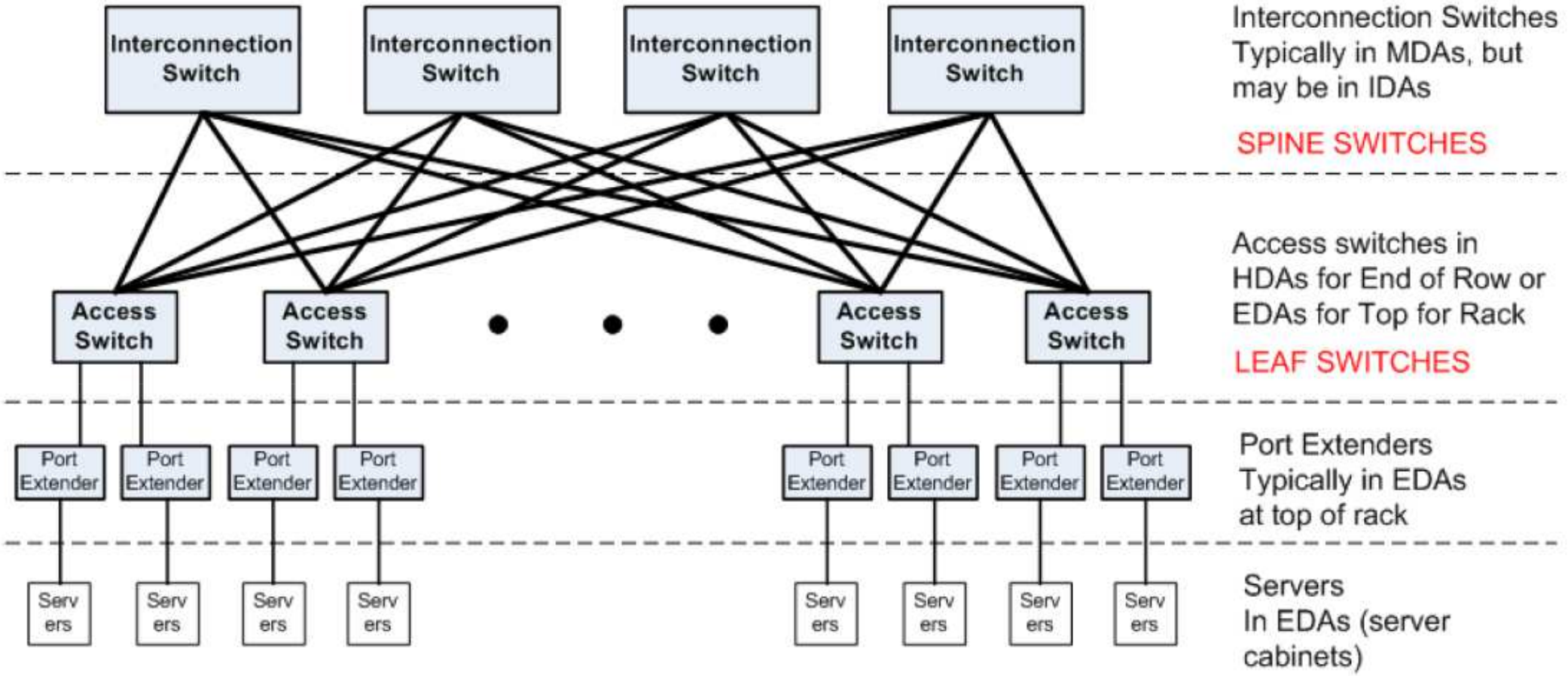


Una rete Leaf-Spine con link 40GbE



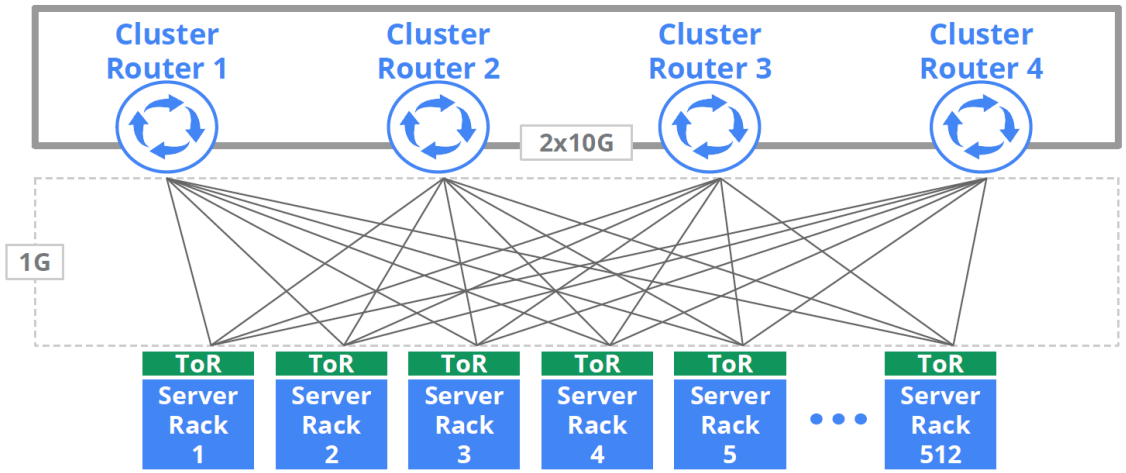
https://s3.amazonaws.com/bradhedlund2/2012/40G-10G-leaf-spine/clos-40G.png

Reti leaf-spine con Port Extender



Architettura di un DC Google nel 2004

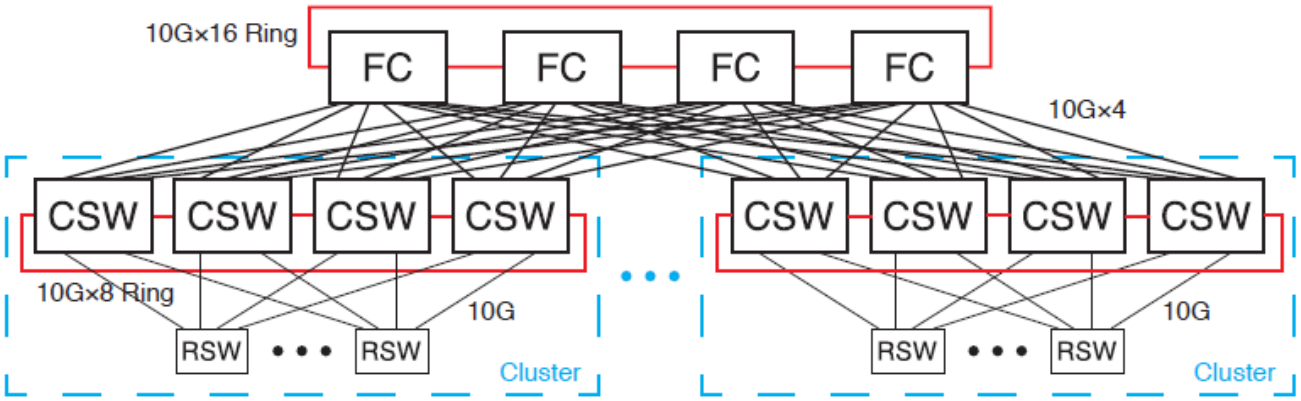
- Switch ToR collegati ad 1 Gb/s ad un livello di aggregazione costituito da 4 router collegati ad anello mediante collegamenti doppi da 10 Gb/s
- Ogni rack comprende 40 server, ciascuno connesso ad 1 Gb/s
- L'intero cluster comprende $512 \cdot 40 \approx 20000$ server
- Banda aggregata del cluster: $4 \cdot 512 \cdot 1$ Gb/s = 2 Tb/s
- Un rack può produrre una banda aggregata di 40 Gb/s ma è collegato al livello gerarchico superiore con una capacità di $4 \cdot 1$ Gb/s = 4 Gb/s
  - Possibilità di congestione se tutti i server di un rack devono scambiare dati con il resto del DC
  - Il traffico deve essere mantenuto il più possibile locale all'interno del rack



Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat.
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network.
SIGCOMM Computer Communications Review, 45, 4 (August 2015), pp. 183-197

Architettura di un DC Facebook nel 2013

- Server collegati mediante link a 10 Gb/s ad uno switch ToR (RSW) in ciascun rack
- Switch RSW collegati mediante 4 uplink a 10 Gb/s ad un *aggregation layer* formato da 4 *cluster switch* (CSW) collegati ad anello
  - Oversubscription: 40 server · 10 Gb/s : 4 uplink · 10 Gb/s = 10 : 1
- Un singolo anello di 4 CSW identifica un cluster (es. da 16 rack)
  - I 4 switch CSW sono collegati ad anello mediante fasci di 8 link da 10 Gb/s
- Switch CSW collegati mediante 4 uplink a 10 Gb/s ad un *core* costituito da 4 switch detti *Fat Cats* (FC) collegati ad anello mediante fasci di 16 link da 10 Gb/s
  - Oversubscription: 16 rack · 10 Gb/s : 4 uplink · 10 Gb/s = 4 : 1

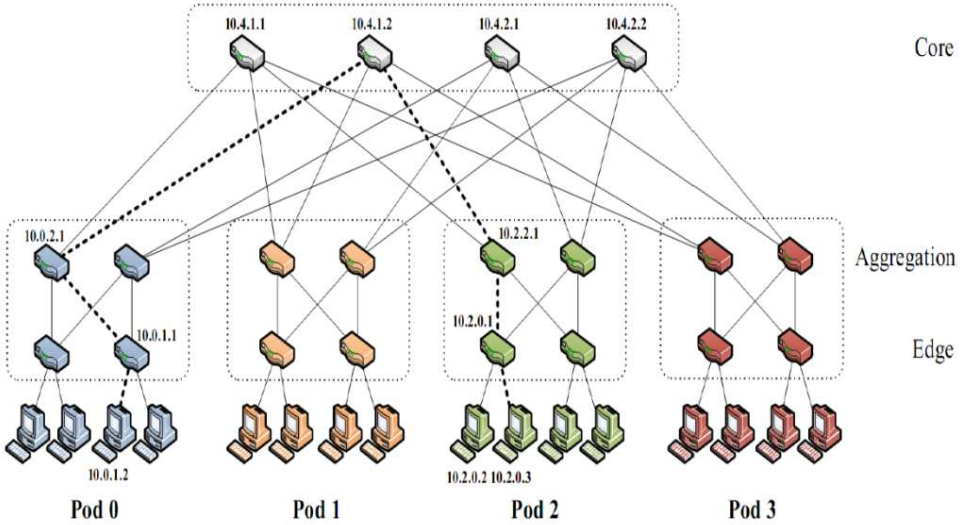


N. Farrington and A. Andreyev. ***Facebook's data center network architecture***. In *Proc. IEEE Optical Interconnects*, May 2013

Datacenter network topology: fat-tree

- Approccio derivato dalle reti di Clos
- Organizzazione gerarchica a 3 livelli
- $k^3/4$ host raggruppati in
  - k *pod* da $(k/2)^2$ host ciascuno
- Caratteristica peculiare:
  - Per ciascuno switch il numero di collegamenti ($k/2$) verso un livello superiore è uguale al numero di collegamenti verso il livello inferiore ($k/2$) → No oversubscription (1:1)
  - switch da k porte a tutti i livelli
- Ciascuno switch edge collega k/2 server a k/2 switch aggregation
- Ciascuno switch aggregation collega k/2 switch edge a k/2 switch core
- $(k/2)^2$ switch core
- Proprietà: ogni livello possiede la stessa banda aggregata

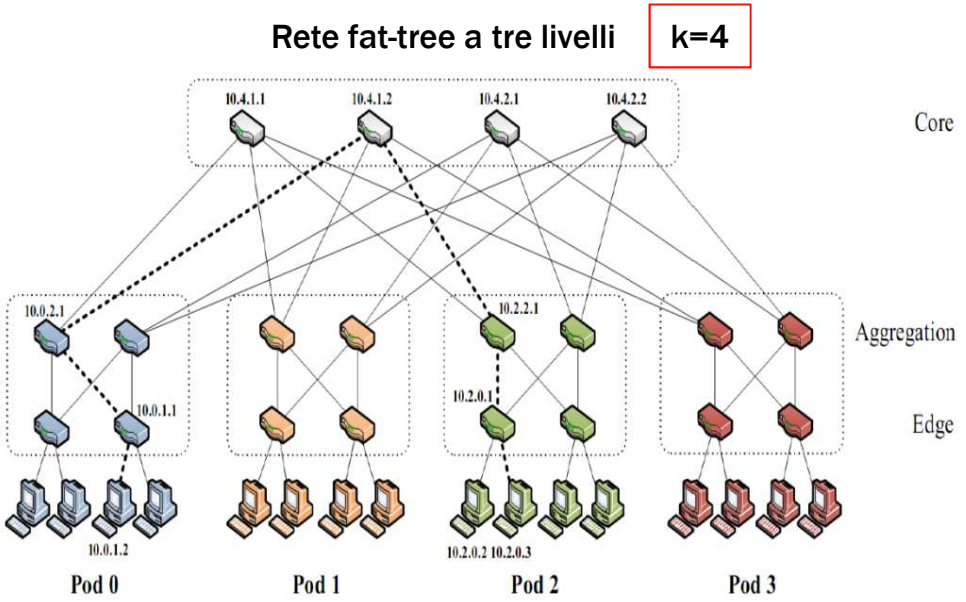
Rete fat-tree a tre livelli **k=4**



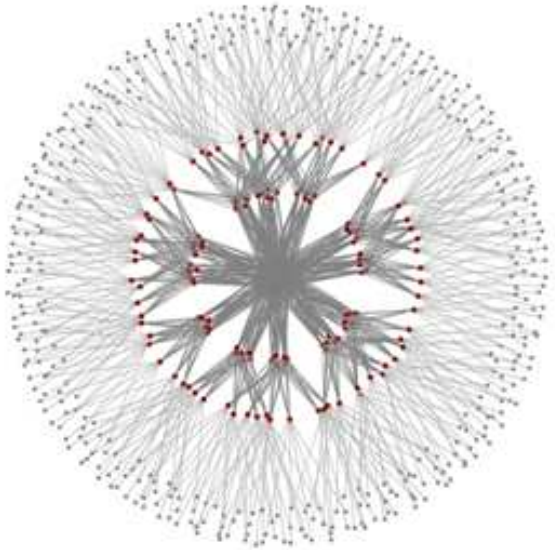
Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat.
A scalable, commodity data center network architecture.
SIGCOMM Computer Communications Review, 38, 4 (August 2008), pp. 63-74

Datacenter network topology: fat-tree

- $k^3/4$ host raggruppati in
  - k *pod* da $(k/2)^2$ host ciascuno
- Ciascuno switch edge collega $k/2$ server a $k/2$ switch aggregation
- Ciascuno switch aggregation collega $k/2$ switch edge a $k/2$ switch core
- $(5/4)\,k^2$ switch, di cui $(k/2)^2$ switch core
- La maggiore capacità dello strato core si ottiene aggregando un elevato numero di link



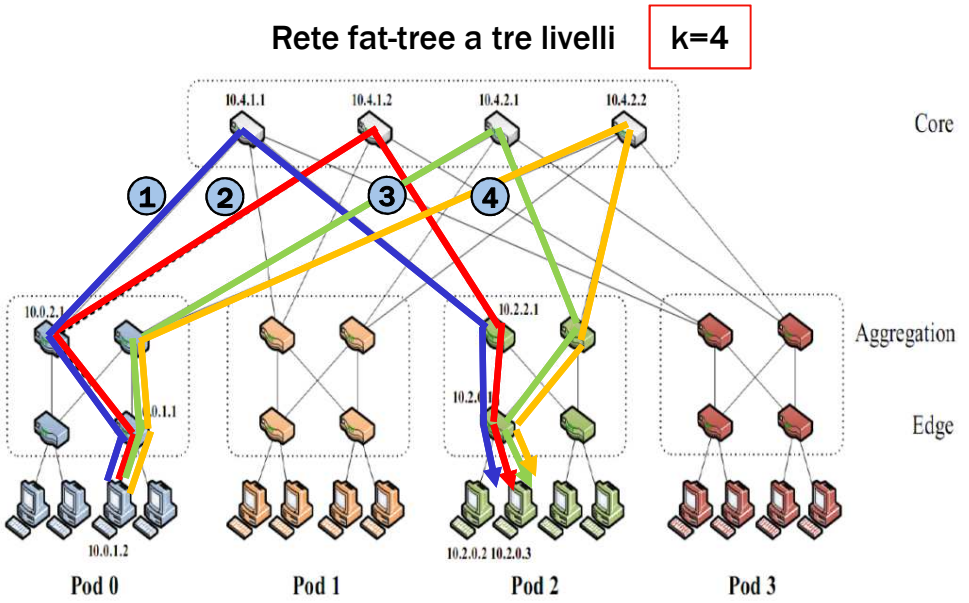
k	# host ($k^3/4$)	# switch core $(k/2)^2$	# switch $(5/4)\,k^2$
4	16	4	20
12	432	36	180
16	1.024	64	320
24	3.456	144	720
32	8.192	256	1.280
48	27.648	576	2.880
96	221.184	2.304	11.520

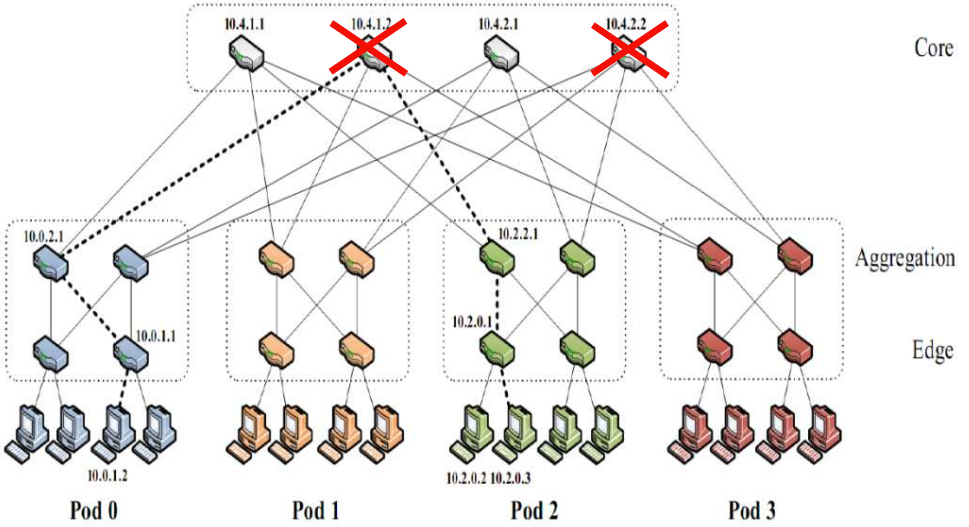


3-level fat-tree
432 servers, 180 switches, k=12

Datacenter network topology: fat-tree

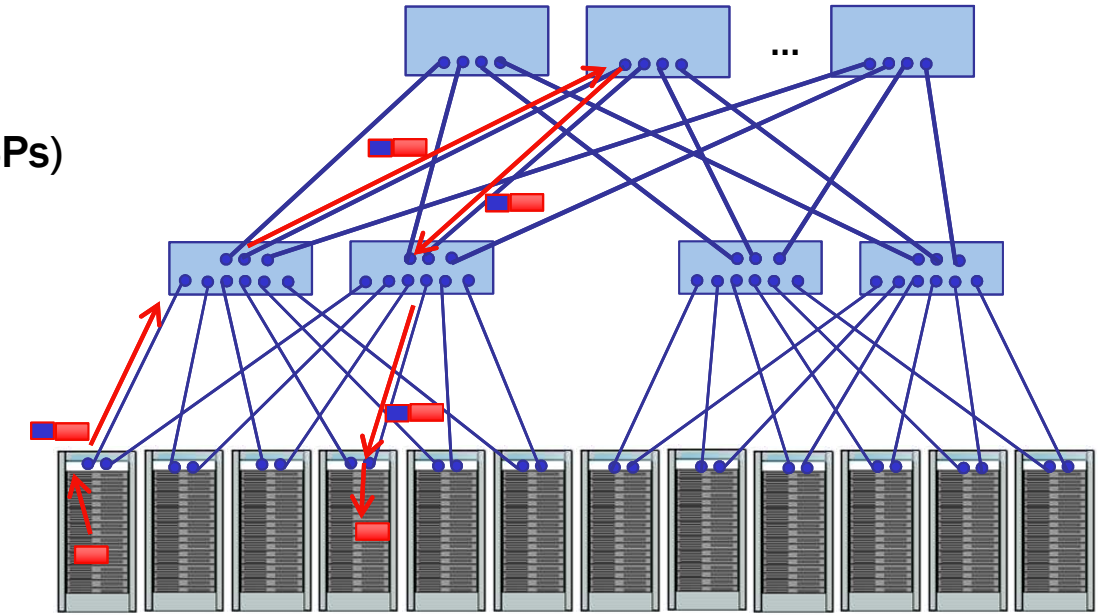
- Rete fat-tree: ridondanza
  - Per la stessa coppia di host esistono k path distinti che li collegano
  - Fissato uno switch core, esiste un solo percorso da quello switch verso ciascuna destinazione
  - Domanda: come utilizzare questi molteplici percorsi ?
- La topologia dell'esempio ha:
  - 16 link di accesso (server-switch)
  - 16 link edge-aggregation
  - 16 link aggregation-core
- Non ci sono colli di bottiglia nei livelli superiori
- E' possibile introdurre un certo livello di oversubscription (ad esempio usando solo 2 switch core)





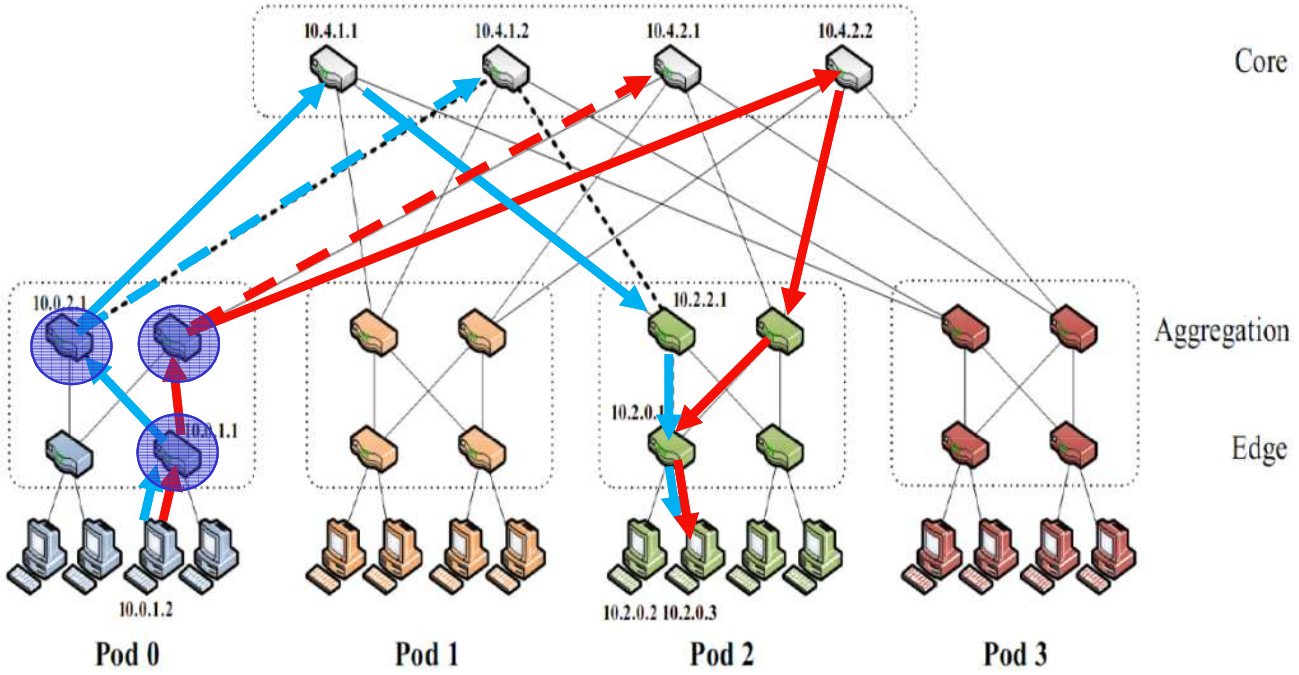
TRILL: Transparent Interconnection of Lots of Links

- A link-state protocol is run in the switches so that they learn the network topology through the exchange of *Link State Packets* (LSPs)
  - Cisco's IS-IS
  - A switch speaking TRILL is called *R-Bridge*
- In order to learn end-hosts' identity, something else (e.g. a directory) is needed
- Once the whole topology is known, multi-path routing is possible
  - Leaf switches encapsulate each packet they receive from hosts with a header bringing the ID of the next-hop R-Bridge in the shortest path to the destination
  - The R-bridge which is closest to the destination decapsulates the packet before delivering it to the destination



Multi-path routing: ECMP

- In a datacenter with multiple possible paths between any (source,destination) couple ECMP allows to randomly spread traffic over alternative paths



At the first edge switch, traffic from 10.0.1.2 to 10.2.0.3 is randomly routed either on the left path or on the right path
Also aggregation switches may randomly choose one among two different paths

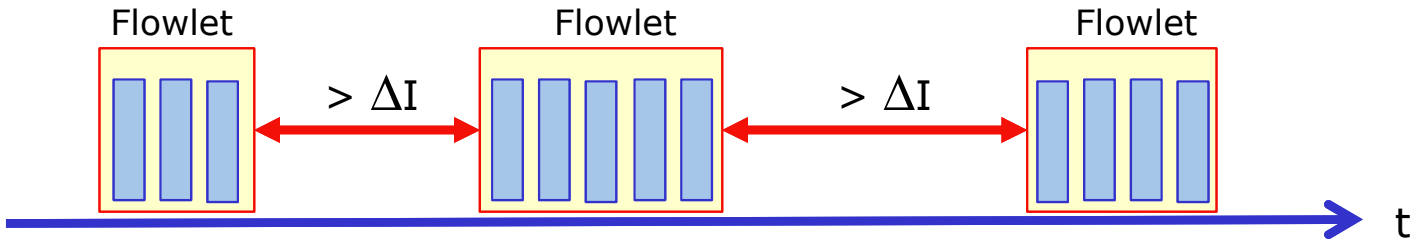
- If upper layer switches have internal queues filled up differently, packets may arrive mis-ordered to destination → TCP performance degrades
- To avoid this problem, packets of the same flow need to be routed on the same path

ECMP and flow hashing

- To avoid misordered delivery of packets belonging to the same flow, ECMP calculates the hash of the packet header to determine the output port at each switch
- In this manner, packets of the same flow, i.e. with same (source, destination), follow the same path and are not misordered
- Works well for a large number of small flows $\rightarrow$ traffic is evenly distributed
- If multiple long-lasting flows are mapped onto the same ports, this technique may lead to an unbalance of traffic flows
- This problem arises because, actually, the concept of flow above is too coarse
- To avoid this problem and achieve a more fine-grained balancing of traffic, randomization may occur at micro-flow or *flowlet* level

ECMP and flowlets

- A *flowlet* is a sequence of consecutive packets whose inter-arrival is smaller than the conservative estimate of latency difference between any two paths within the datacenter network
  - If two flowlets are routed along different paths, no misordered delivery may happen anyway



- Flowlet-based routing first proposed in FLARE in 2007
- Flowlet-to-path mapping is performed by using a hash table whose entries are

  (hash_key, last_seen_time, path_id)

- When a packet arrives, FLARE computes a hash of

  source IP, destination IP, source port, destination port

- and uses this as the key in the hash table

[FLARE] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. *Dynamic load balancing without packet reordering*. ACM *SIGCOMM Comput. Commun. Rev.* 37, 2, pp. 51-62, March 2007

ECMP issues: local decisions

- One issue with ECMP is that it only takes local decisions without any knowledge of further links status
- In this example topology, once the path has been pinned to the core switch, there's no further alternative to a given destination (i.e. only one path)
- If a link fails, ECMP can do nothing to prevent upstream switches to select the path that contains that link, even if an alternative path exists