

1 Il livello Data Link e le LAN

1.1 Il data link layer

I dispositivi (host, router, server ecc.) che supportano il data link layer sono detti **nodi**. La comunicazione avviene tramite **canali** che connettono nodi adiacenti, questi canali vengono chiamati **link** (via cavo, wireless), un'intera rete LAN viene vista come un unico link. I pacchetti che vengono trasmessi a livello data link si chiamano **frame**.

Richiami esame di reti

A livello IP il pacchetto trasmesso è chiamato **datagramma**, a livello trasporto TCP il pacchetto è chiamato **segmento**.

1.1.1 I servizi forniti

Incapsulamento (framing) Nella maggior parte delle volte la parte contenente i dati del frame contiene il datagramma IP. Il livello data link fornisce come servizio quello di trasferire un datagramma da un nodo ad un altro attraverso i link.

Accesso al link Un protocollo di medium access control (MAC) specifica come il frame deve essere trasmesso sul link.

Trasporto affidabile Un protocollo di trasferimento affidabile garantisce che ogni frame raggiunga la sua destinazione senza errori. Assicura che il trasferimento sia affidabile tra ogni nodo in modo da non dover rimandare tutto il pacchetto TCP ma solo i frame (datagrammi) danneggiati.

Individuazione e correzione degli errori Il nodo mittente fornisce un meccanismo per individuare gli errori, che verranno poi corretti dal destinatario.

Controllo di flusso La velocità con cui il nodo mittente trasmette e il nodo ricevente riesce a ricevere.

Half-duplex e full-duplex Con half-duplex in una comunicazione tra due nodi solo uno alla volta può trasmettere mentre l'altro resta in attesa e viceversa. Con full-duplex entrambi i nodi possono trasmettere e ricevere contemporaneamente.

1.1.2 Implementazioni

Il data link layer deve essere implementato in ogni nodo connesso alla rete. Le funzionalità Ethernet sono integrate nella scheda madre o in un chip Ethernet. Il livello collegamento è implementato su un chip detto network adapter o NIC.

1.2 Individuazione e correzione degli errori

L'individuazione degli errori prevede aggiungere ai dati trasmessi dei bit/byte aggiuntivi che servono per permettere di rilevare o correggere l'errore. Potrebbero presentarsi degli errori sia sulla parte dei dati che sui bit aggiuntivi perché viaggiano su un canale non affidabile.

1.2.1 Controllo di parità

Ad una certa sequenza di bit aggiunto un singolo bit, il bit di parità, che viene aggiunto dal mittente seguendo certe regole che dovranno essere condivise con il ricevente. Se il numero di bit corrotti è pari il test viene passato con successo anche se è presente un errore.

Esempio

Aggiungo un 1 se il numero di 1 è dispari 0 se il numero di 1 è pari. Attraverso le matrici di parità calcolo i bit di parità sia per le righe che per le colonne e confronto se il numero di bit coincide con il bit di parità. Posso correggere l'errore solo se è presente **1 errore** ma segnalare l'errore su **2 bit**.

Foto/CheckParita.png

1.2.2 CRC (Cyclic Redundancy Check)

È uno dei più potenti rilevatori di errori ma non può correggerli.

I codici CRC sono anche detti codici polinomiali in quanto 'è possibile considerare la stringa da inviare come un polinomio i quali coefficienti sono i valori 0 e 1 della stringa di bit. I CRC possono individuare *resto ≤ bit errati consecutivi*.

Svolgimento (1)

D = numero di bit per i dati;

G = sequenza di bit (generatore) che deve soddisfare determinate caratteristiche. In questo caso $r + 1$ bit.

Aggiungo in coda una sequenza di CRC lunga r bit (un bit in meno rispetto al generatore).

$$\langle D, R \rangle = D \cdot 2^r \text{ XOR } R$$

$D \cdot 2^r$ shifto il dato di r bit

$\text{XOR } R$ metto come ultimi bit significativi quelli di R

Il mittente prende gli R bit dei CRC e prende $\langle D, R \rangle$ e lo divide per G , fa il modulo 2, **se il resto della divisione è 0 il test è passato** altrimenti segnalo che è presente un errore.

Posso rilevare un numero di bit errati $< G$. Inoltre posso rilevare un numero di errori dispari.

Svolgimento (2) Come calcolare R

R deve essere tale per cui:

$$D \cdot 2^r \text{ XOR } R = nG$$

AB	A XOR B
00	0
01	1
10	1
11	0

$$R = \text{resto}\left[\frac{D \cdot 2^r}{G}\right]$$

Esempio

$D=101110$ $G=1001$ $r=3$

```
10111000 :1001
1001 XOR  101011
  101
  000 XOR
  1010
  1001 XOR
    110
    000 XOR
    1100
    1001 XOR
      1010
      1001 XOR
        011 = R
```

Alla fine il **mittente** trasmetterà come sequenza 101110**011** con gli ultimi 3 bit uguali a quelli calcolati.

Il **ricevente** prende la sequenza e la divide per G , se il resto è 0 il test è passato altrimenti segnalo la presenza di un errore.

Generatori Esistono dei generatori G standard da 8, 12, 16 o 32 bit prestabiliti.

1.3 Protocolli ad accesso multiplo

Esistono due tipologie di link:

1. **Punto-Punto** = un nodo è collegato direttamente ad un altro attraverso un link (canale) dedicato;
 - eg. collegamento ethernet tra un host e uno switch.
2. **Broadcast** = il link è condiviso, potenzialmente diversi nodi possono trasmettere e/o ricevere contemporaneamente;
 - Eg. prime versioni di ethernet (tipologia a bus), comunicazione wireless.

I protocolli ad accesso multiplo stabiliscono un insieme di regole che ogni nodo deve seguire affinché la comunicazione avvenga con successo. In assenza di protocolli avremmo il problema delle **collisioni**, cioè due o più nodi che trasmettono contemporaneamente.

Consideriamo algoritmi di tipo distribuito che determinano come il canale viene condiviso, ad esempio determinano quando un determinato nodo deve trasmettere oppure cosa succede in caso di collisione. La comunicazione avverrà, sia per trasmettere dati che per regolare la trasmissione stessa, sullo stesso canale.

Protocollo ideale Consideriamo un canale ad accesso condiviso che è in grado di trasmettere ad un rate di R bps (bit per secondo).

Si desidera che:

1. quando un singolo nodo vuole trasmettere, può trasmettere ad una velocità di trasmissione pari ad R ;
2. quando ci sono M nodi che voglio trasmettere contemporaneamente, ogni nodo può trasmettere ad una velocità media di $\frac{R}{M}$;

3. completamente decentralizzato:

- non voglio nodi specializzati che coordinino la trasmissione (single point of failure);
- non voglio la sincronizzazione dei clock dei diversi nodi o degli slot.

4. semplice.

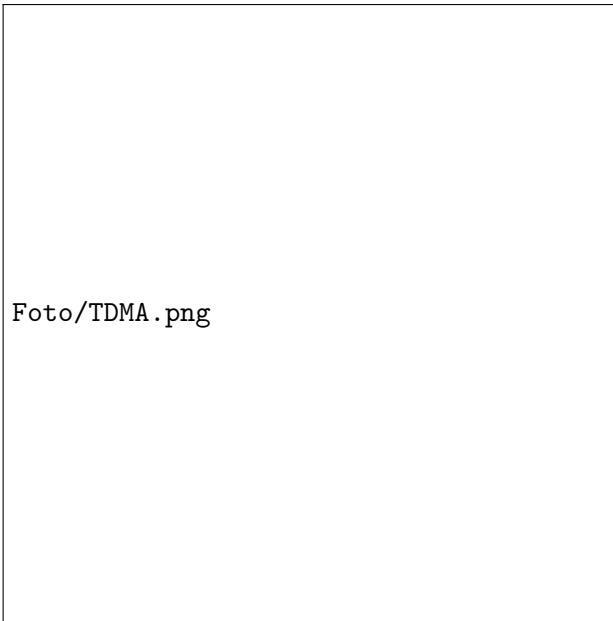
1.3.1 Protocolli a partizionamento del canale

Divido la capacità complessiva del canale in diverse parti che assegno ai diversi nodi.

TDMA Ogni nodo può accedere al canale solo in determinati intervalli di tempo. Ad ogni stazione viene assegnato un intervallo di tempo fisso in cui può trasmettere. Gli slot inutilizzati vengono persi, NON riutilizzati da altri nodi.

Esempio

Il time-frame viene suddiviso in un numero di slot pari al numero di nodi complessivi della rete.



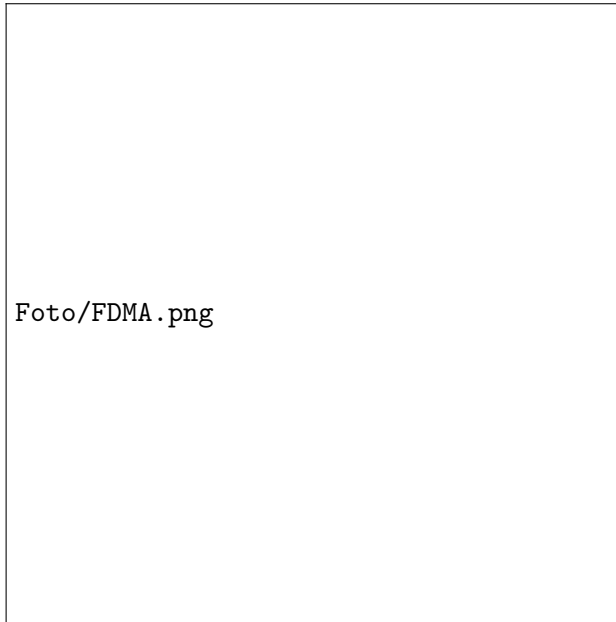
Proprietà ideali:

1. Soddisfatto;
2. Soddisfatto;
3. Non soddisfatto, è necessario mantenere gli stessi clock per sapere quando inizia uno slot;
4. Soddisfatto.

FDMA Viene partizionato il dominio delle frequenze. Ad ogni stazione viene assegnata una frequenza stabilita. Il tempo non utilizzato in frequenza viene sprecato. Posso non usare certe frequenze tra un intervallo ed un altro per evitare le sovrapposizioni di banda.

Esempio

Vengono partizionate le frequenze in base al numero di nodi presenti.



Proprietà ideali:

1. Non soddisfatto;
2. Soddisfatto;
3. Soddisfatto;
4. Soddisfatto;

1.3.2 Protocolli ad accesso casuale

Il canale non è diviso e si permette l'occorrenza di collisioni, nel caso di collisioni devo fare in modo di risolverla. Quando un nodo ha un pacchetto da inviare lo trasmette alla massima velocità, a priori non c'è coordinamento tra i nodi.

Slotted ALOHA Assumiamo che tutti i frame abbiano la stessa dimensione, il tempo è diviso in slot di uguale dimensione necessaria per trasmettere un intero frame. I nodi iniziano a trasmettere solo all'inizio dello slot. I nodi sono sincronizzati. Se 2 o più nodi trasmettono simultaneamente tutti i nodi della rete sono in grado di rilevare la collisione.

Quando un nodo ottiene un nuovo frame da inviare lo trasmette nel successivo slot, dopo aver trasmetto si mette in ascolto, se c'è un picco di energia viene a conoscenza della presenza di una collisione in questo caso ritrasmetto nel successivo slot con una probabilità p .

Esempio

All'inizio tutti i nodi vogliono trasmettere e avviene una collisione. Successivamente, dopo uno slot vuoto, i nodi 1 e 2 ritrasmettono e vanno in collisione. Allo slot successivo solo 2 trasmette ed ha successo. E così via finché tutti non hanno trasmesso.



Proprietà ideali:

1. Soddisfatta;
2. Non soddisfatta;
3. Non soddisfatta, perché è richiesta sincronizzazione;
4. Soddisfatta.

Durante le collisioni vengono sprecati degli slot. Un nodo è in grado di accorgersi subito della collisione ma deve aspettare un nuovo slot per trasmettere.

Efficienza

Un indicatore dell'efficienza è la proporzione di slot che hanno una comunicazione con successo rispetto a tutti gli slot disponibili su un intervallo di tempo lungo.

Supponiamo che N nodi vogliono continuare a trasmettere dei frame ed ognuno trasmette con probabilità p .

- La probabilità che un dato nodo abbia successo nella trasmissione è $= p(1 - p)^{N-1}$
- La probabilità che un qualsiasi nodo abbia successo è $= Np(1 - p)^{N-1}$
- Per la massima efficienza devo trovare p^* che massimizzi : $Np(1 - p)^{N-1}$
- Per molti nodi prendo il limite di $Np^*(1 - p^*)^{N-1}$ quando N tende a infinito.
- Efficienza massima $= \frac{1}{e} = 0.37$

Nella migliore delle ipotesi utilizzerò il canale il 37% del tempo.

CSMA (Carrier Sense Multiple Access) Ascolto prima di trasmettere (*carrier sense*), se il canale è libero allora trasmetto, altrimenti aspetto.

Esempio

Possono capitare delle collisioni anche ascoltando prima di trasmettere sul canale perché il segnale non si propaga immediatamente ovunque (*propagation delay*).

Foto/CSMA.png

CSMA/CD Protocollo utilizzato in ethernet. CSMA con il rilevamento della collisione (*collision detection*). Mentre trasmetto riesco a rilevare che è presente una collisione, appena riesco a farlo interrompo la trasmissione per ridurre lo spreco del canale. Facile da implementare su canali via cavo, difficile in wireless.

Esempio

Per velocizzare il fatto che tutti rilevino la collisione, appena qualcuno la rileva manda un segnale agli altri nodi intorno in modo tale che tutti rilevino la collisione.

Foto/CSMACD.png

Algoritmo di Ethernet

1. Ethernet riceve il datagramma dal livello rete sovrastante e crea il frame;
2. La scheda di rete ascolta il canale:
 - se è libero: inizia la trasmissione del frame;
 - se è occupato: aspetta finché il canale non si libera e poi trasmette.
3. Se sono riuscito a trasmettere l'intero frame senza collisioni ho finito;
4. Se viene rilevata un'altra trasmissione mentre invio, smetto di trasmettere e mando un segnale di *abort* (*jamming signal*) agli altri nodi;
5. Dopo aver abortito devo ritrasmettere e utilizzo un algoritmo di **binary (exponential) backoff**:
 - dopo l' m -esima collisione scelto un numero casuale K nell'intervallo $\{0, 1, 2, \dots, 2^{m-1}\}$. Ethernet aspetta $k \cdot 512$ bit times (tempo necessario per trasmettere un bit) e poi ritorno al punto 2.
 - Più collisioni ci sono maggiore sarà l'attesa per ritrasmettere.

1.3.3 Protocolli a turni

Ogni nodo può trasmettere in maniera dedicata ma solo in determinati turni.

Polling Controllore centralizzato, ciclicamente richiede ad ogni nodo se ha bisogno di trasmettere se si dedica un intervallo di tempo per trasmettere altrimenti passa al successivo.

I contro di questo protocollo sono: l'overhead del polling (pochi byte), latenza perché devo aspettare che il controllore faccia il giro e il single point of failure del controllore.

Questo protocollo è utilizzato in Bluetooth.

Token-passing Invece di avere un master ho un *token*. I nodi sono disposti ad anello, in modo ciclico. Il token passa al successivo nodo che conosce il suo successore. Chi ha il token può trasmettere per un intervallo di tempo fisso, costante e limitato.

I contro di questo protocollo sono: l'overhead del token, latenza per l'attesa del token e single point of failure (ciascun nodo quando ha il nodo diventa un punto di fallimento).

1.4 Switched Local Area Networks

Gli switch utilizzano indirizzi propri al livello collegamento per inoltrare i frame.

1.4.1 Indirizzo MAC

Indirizzi IP Indirizzi definiti a livello di rete associati alla singola interfaccia da 32 bit, utilizzati per fare il forwarding dei pacchetti.

MAC Le interfacce di rete degli host e dei router hanno indirizzi livello collegamento, ma non gli switch. Un indirizzo livello collegamento viene chiamato indirizzo MAC. Gli indirizzi MAC sono di solito lunghi 6 bytes (**48 bit**) e sono espressi in notazione esadecimale. Ogni interfaccia nella LAN ha un **unico indirizzo MAC** e un (localmente) unico indirizzo IP. Ogni interfaccia possiede un unico indirizzo MAC

perché questi sono controllati dalla IEEE (tipicamente si associano i primi 24 bit più significativi sono associati ad uno specifico produttore).

Quando un'interfaccia vuole inviare un frame, inserisce l'indirizzo MAC di destinazione nel frame e lo inoltra sulla LAN. Se un mittente vuole inviare un frame a tutte le interfacce presenti sulla LAN, questo inserisce un indirizzo speciale, l'indirizzo broadcast nel frame.

1.5 Protocollo ARP (Address Resolution Protocol)

Il protocollo ARP, tipicamente implementato su tutti gli host, serve per fare il mapping tra l'indirizzo IP e l'indirizzo MAC corrispondente all'interno di una sotto-rete.

Ciascun nodo ha una **tabella ARP** che contiene:

- Corrispondenza tra indirizzo IP e indirizzo MAC;
- TTL (Time To Live) = intervallo di tempo dopo il quale il mapping non vale più.

Esempio protocollo ARP nella stessa LAN

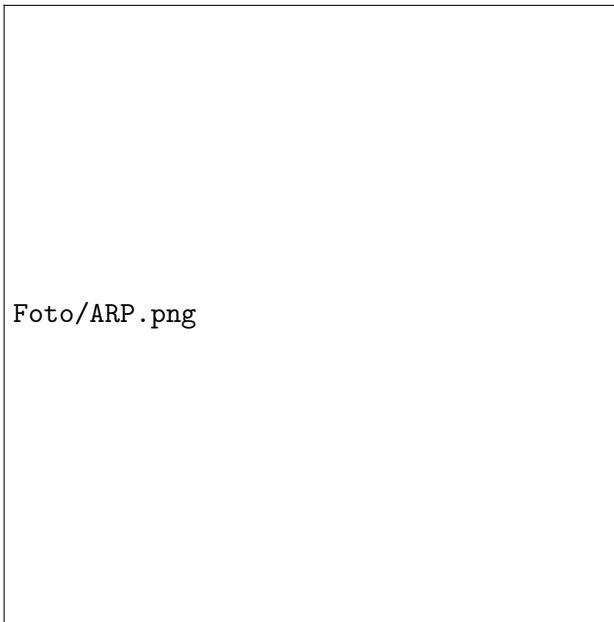
Supponendo che *A* voglia inviare un datagramma a *B* e l'indirizzo MAC di *B* non è presente nella tabella ARP di *A*.

A fa un **broadcast ARP**, sull'intera rete fisica attraverso l'indirizzo MAC *FF-FF-FF-FF-FF-FF*, in cui specifica l'indirizzo IP di *B* e l'indirizzo MAC sorgente. Tutti i nodi ricevono questa richiesta e la processano, solo *B* replicherà in modo diretto, unicast, specificando il suo indirizzo MAC associato al suo indirizzo IP.

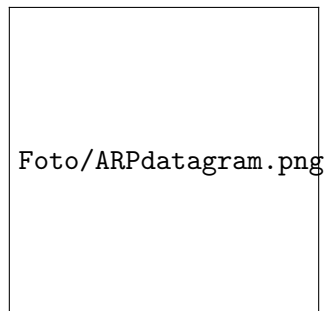
A salva l'informazione nella sua tabella ARP. Questa entry può essere cancellata se finisce il suo TTL oppure rinnovata se viene rinoltrata un'altra richiesta con la stessa associazione IP-ARP.

Esempio protocollo ARP invio in sotto-reti diverse

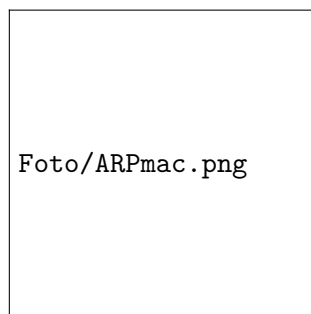
Supponiamo che A debba inviare un datagramma a B . Assumiamo che A conosca l'indirizzo IP di B e che A conosca l'indirizzo IP del primo hop router (attraverso il DHCP) e che conosca anche l'indirizzo MAC dell'interfaccia del router.



- A crea il datagramma IP con sorgente l'IP di A e destinazione l'IP di B ;



- A incapsula il datagramma in un frame con indirizzo MAC sorgente quello di A e **indirizzo di destinazione l'indirizzo MAC dell'interfaccia del router**;



- il frame viene inviato da A al router;
- il router processa il pacchetto, rimuove il datagramma e lo passa a livello IP;
- il router passa all'altra interfaccia il datagramma con IP sorgente e destinazione;
- il router crea un nuovo frame, gli indirizzi IP rimangono gli stessi (A e B) invece gli indirizzi

1.6 Ethernet

La LAN Ethernet originariamente utilizzava un bus coassiale per connettere i nodi, e i frame erano trasmessi in broadcast. Successivamente, il bus è stato rimpiazzato con uno switch, che a differenza dei router opera esclusivamente sul livello collegamento.

Struttura di un frame ethernet

Preambolo	Indirizzo destinazione	Indirizzo sorgente	Tipo	Campo dati	CRC
-----------	------------------------	--------------------	------	------------	-----

- **Preambolo** = utilizzato per sincronizzare mittente e destinatario e per regolare la velocità con cui invierò i bit. È formato da 7 byte con pattern *10101010* seguiti da 1 byte con il pattern *10101011*. Serve anche per capire che dopo questo pattern iniziano le informazioni;
- **Indirizzo destinazione** = indirizzo MAC del destinatario, 6 byte;
- **Indirizzo sorgente** = indirizzo MAC del mittente, 6 byte;
- **Tipo** = indica qual è il protocollo di livello superiore del payload (tipicamente IP), 4 bit;
- **Campo dati** = contiene il datagramma IP, massimo di 1500 byte, minimo 46 byte;
- **CRC** = per verificare la correttezza del frame stesso.

In ethernet non c'è handshake tra mittente e destinatario quindi è un protocollo **connectionless**.

In caso di errore ethernet lo rileva e scarta il frame quindi è un protocollo **unreliable**.

In ethernet il protocollo MAC che si usa è **CSMA/CD con binary backoff**.

Ethernet 802.3 È uno standard che definisce sia aspetti di livello data-link che di livello fisico.

1.7 Switch ethernet

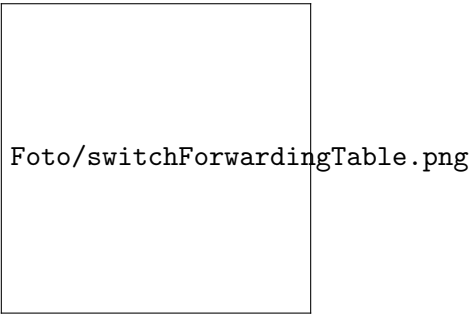
Gli switch ethernet sono dei dispositivi a livello collegamento, hanno un ruolo attivo perché memorizzano e inoltrano i frame ethernet e non solo, riescono ad esaminare gli indirizzi MAC e decidono in modo selettivo se inoltrare il frame e a quale interfaccia (porta), viene inoltrato su un determinato segmento sul quale si usa CSMA/CD per accedervi.

Gli switch sono dispositivi **trasparenti** agli host e utilizzano soluzioni di tipo **plug and play** (nessuna necessità di configurazione) e **self learning**.

Questo dispositivo permette la trasmissione simultanea di più frame, in particolare ogni host ha una connessione diretta e dedicata con lo switch (tipicamente full-duplex), inoltre bufferizzano i pacchetti (no collisioni) e si usa il protocollo ethernet su ogni singola connessione.

Switch forwarding table

Come fanno gli switch a sapere che A' è raggiungibile attraverso l'interfaccia 5?



Ogni switch ha una tabella di inoltra che contiene:

- Indirizzo MAC;
- interfaccia per raggiungere l'host;
- time stamp = time to live.

Le entry di questa tabella sono riempite in modo diverso rispetto alle tabelle di routing. Lo switch impara la corrispondenza tra indirizzo MAC e porta a cui inoltrare il frame in modo dinamico.

Self learning Ipotizziamo che A voglia inoltrare un frame ad A' . A invia il frame sul cavo fino a raggiungere lo switch, in questo momento memorizza che l'host A è arrivato dalla porta 1 e lo aggiunge alla sua tabella.

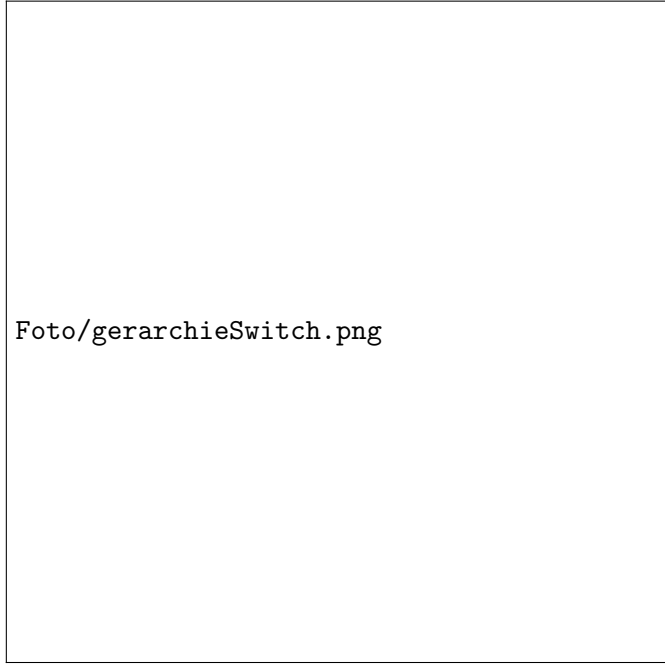
Frame filtering/forwarding Il **filtering** è la funzionalità degli switch che determina se un frame deve essere inoltrato o scartato. Il **forwarding** è la funzionalità che determina le interfacce alle quali il frame va inoltrato.

Quando il frame raggiunge lo switch:

1. memorizza l'indirizzo MAC dell'host mittente;
2. indicizza la tabella dello switch utilizzando l'indirizzo MAC di destinazione;
3. se viene trovata la entry del destinatario
 - allora:
 - se la destinazione è la stessa del segmento da cui il frame è arrivato lo scarta;
 - altrimenti lo inoltra all'interfaccia del destinatario.
 - altrimenti:
 - fa il **flooding**, cioè inoltra il frame su tutte le interfacce eccetto quella del mittente.

1.7.1 Interconnessione tra gli switch

La tipologia a stella non è sufficiente per LAN di dimensioni importanti, allora gli switch possono essere connessi tra loro in modo gerarchico. Durante il flooding l'informazione si propaga fino a raggiungere tutti gli host.



Foto/gerarchieSwitch.png

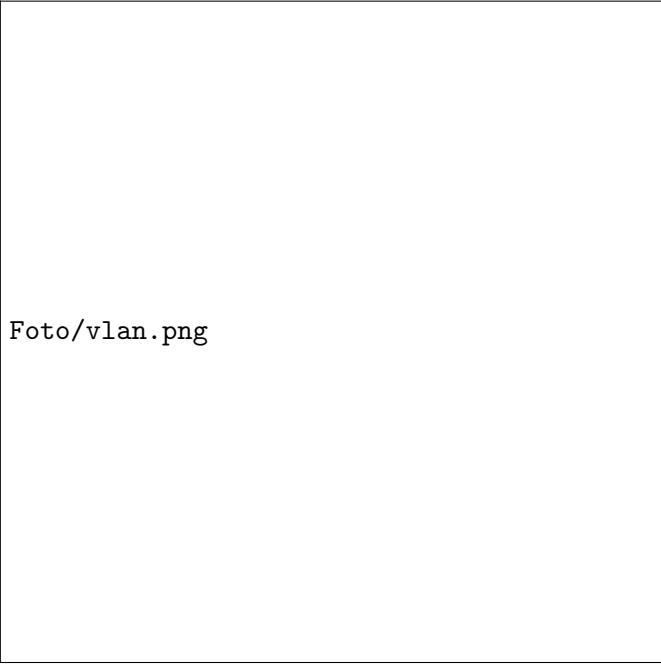
1.8 Switch vs Router

- Entrambi hanno un meccanismo di store-and-forward;
 - *router* = dispositivi di livello rete (esaminano solo le intestazioni a livello di rete);
 - *switch* = dispositivi di livello collegamento (esaminano solo le intestazioni a livello 2).
- Entrambi hanno delle tabelle di inoltro;
 - *router* = calcolata attraverso gli algoritmi di routing e basandosi sugli indirizzi IP;
 - *switch* = calcolata attraverso il self-learning utilizzando il flooding e imparando gli indirizzi MAC man mano.

1.9 VLAN (Virtual LAN)

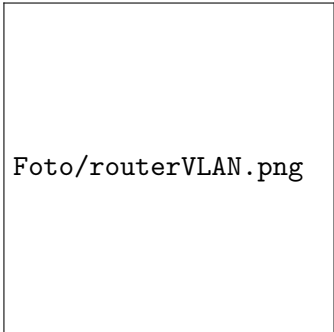
Le reti LAN tradizionali non sono facilmente scalabili o portabili, ma le VLAN sì. Uno switch che supporta VLAN permette di definire LAN virtuali su una singola LAN fisica. Gli host su una VLAN possono comunicare tra loro come se fossero solo loro connessi a quello switch.

VLAN basata sulle porte



Foto/vlan.png

Lo switch può essere configurato per fare in modo che le porte vengano etichettate e dedicate ad una particolare VLAN (solitamente caratterizzate da un colore), le porte in ogni VLAN formano un dominio di broadcast. Per far comunicare dei dispositivi di VLAN diverse devo utilizzare un router esterno.

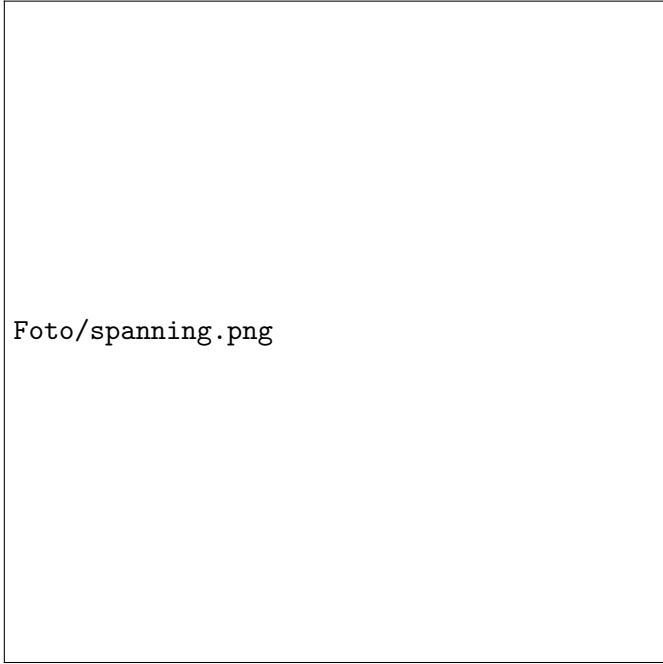


Foto/routerVLAN.png

Vantaggi Posso **configurare in modo dinamico** lo switch per, ad esempio, cambiare la VLAN di un dispositivo senza doverlo spostare fisicamente. Si può anche definire la VLAN basandosi sugli indirizzi MAC degli endpoint piuttosto che per le porte.

1.10 VLAN spanning (trunk port)

In presenza di più switch distinti posso fare in modo che host dello switch secondario possano far parte delle VLAN del principale.



Foto/spanning.png

Utilizzo una particolare **porta trunk** dello switch che permette di interconnettere LAN differenti. (Eg. se si manda un messaggio in broadcast sulla VLAN 1 viene inoltrato anche sulla porta 16, connessa fisicamente all'altro switch, e arriveranno alla porta 1 del secondo switch. Sulla porta trunk passano i frame di una qualunque VLAN, allora per identificare a quale VLAN appartiene il frame si è stabilito un nuovo **protocollo 802.1q** che aggiunge un campo aggiuntivo nell'header del protocollo ethernet, fatto solo dagli switch. Quando uno switch riceve il frame riconosce a quale VLAN deve essere indirizzato il frame e lo inoltra eliminando i dati aggiuntivi e tornando ad un protocollo normale.

Pream.	Dest.	Sorg.	Tipo di protocollo	Informazioni di controllo	Tipo	Dati	CRC
--------	-------	-------	--------------------	---------------------------	------	------	-----

- **Tipo di protocollo** = 2 byte (valore 81-00);
- **Informazioni di controllo** = 12 bit per identificare la VLAN e 3 bit di priorità.

1.11 Networking nei datacenter

1.11.1 Architetture dei data center

I data center non sono solamente connessi a Internet, ma a dei loro network interni che connettono gli host tra loro. Gli host nei data center sono chiamati **blades**, impilati in rack. In cima ad ogni rack c'è uno switch, detto **Top of Rack (TOR)**, che connette gli host nel rack tra di loro e con gli altri switch nel data center. Ogni host nel rack ha un'interfaccia di rete che si connette al proprio TOR, e ogni TOR ha delle porte che possono essere connesse ad altri switch.

I data center supportano due tipi di traffico: il **traffico tra client esterni e host interni** e **traffico tra host interni**. Per gestire il primo, i data center utilizzano dei **border router**, che connettono il data center con Internet.

1.11.2 Load balancing

Le richieste ricevute da un data center sono innanzitutto dirette a un load balancer che si occupa di distribuire le richieste agli host. I grossi data center possiedono molti load balancer, ognuno dedicato a specifiche applicazioni cloud. Un tale load balancer è spesso chiamato "layer-4 switch" in quanto basa

le sue decisioni sul numero di porta e sull'indirizzo IP del pacchetto. Al ricevimento di una richiesta, il load balancer la inoltra a uno degli host che gestisce l'applicazione. Il load balancer funziona anche come NAT, in quanto traduce l'IP pubblico della richiesta nell'IP interno e viceversa.

1.12 Il processamento di una richiesta



Foto/journeyTopology.png

1.12.1 Connessione ad Internet

Si connette e come prima cosa deve sapere qual è il suo indirizzo IP e l'indirizzo IP del first hop router oltre all'indirizzo del DNS server, per ottenere tutte queste informazioni utilizza il **DHCP**.

Fa una richiesta DHCP che viene incapsulata in UDP a sua volta incapsulata in IP e in fine incapsulata in 802.3 Ethernet. Questa richiesta viene inoltrata in broadcast.

DHCP risponde con un ACK specificando l'indirizzo IP del client e le altre informazioni richieste, incapsula tutte queste informazioni e lo inoltra in unicast al nodo.

Il client ora ha un indirizzo IP, sa il nome e l'indirizzo del server DNS e conosce l'indirizzo IP del first-hop router.



Foto/ConnessioneAInternet.png

1.12.2 Richiesta HTTP

Prima di mandare la richiesta HTTP necessita dell'indirizzo IP di *www.google.com* al DNS.

Viene creata una query DNS, incapsulata in UDP a sua volta incapsulata in IP e in fine incapsulata in ethernet. Per mandare il frame al router necessito dell'indirizzo MAC del router quindi utilizzo il protocollo ARP.

ARP fa una richiesta in broadcast, ricevuta dal router che risponde con l'indirizzo MAC dell'interfaccia. Ora il client è a conoscenza dell'indirizzo MAC del router e posso fare la richiesta al DNS.



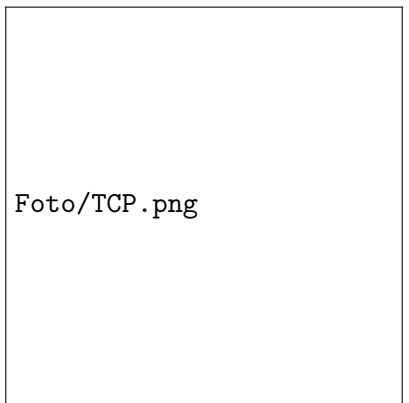
Il datagramma IP contenente la query DNS viene inoltrata, attraverso lo switch LAN, dal client al first hop router.

Poi il datagramma IP viene inoltrato dal *campus network* fino al *comcast network* fino al server DNS.

Il server DNS risponde con l'indirizzo IP di *www.google.com*.



Per la richiesta HTTP devo aprire una connessione TCP e fare il 3-way handshake.




Foto/TCP.png

La richiesta HTTP viene inoltrata attraverso il socket TCP.

Il datagramma IP contenente la richiesta HTTP viene inoltrato fino a *www.google.com*.

Il web server risponde con una HTTP reply (contenente la pagina web).

Il datagramma IP contenente la risposta HTTP viene inoltrato fino a raggiungere il client.



Foto/fine.png