

Progettazione di Reti Informatiche

A.A. 2021/2022

Disclaimer: questi sono gli appunti che ho preso personalmente a lezione e rielaborato, ovviamente possono esserci errori, refusi o altre imprecisioni per le quali mi scuso in anticipo

Buono studio :)

- Azzurra

Packet Tracer (v6.2): software di emulazione del comportamento di vari apparati di rete, i dispositivi trasmettono in maniera virtuale pacchetti in ingresso e uscita secondo il comportamento reale.

Interfaccia - Catalogo degli apparati:

- a) **Router** - ci sono anche dei modelli generici, in cui cambia il tipo di interfacce presenti
- b) **Switch** - come per i router ci sono diversi modelli e dei modelli generici
- c) **Multi-hub:** componente che funziona a livello 1 (fisico), quando ha più porte il segnale ricevuto su una porta viene replicato fisicamente sulle altre
- d) **Wireless devices**
- e) **End devices:** desktop, laptop, server, TV, tablet ecc. Questi differiscono per il tipo di interfaccia di accesso che usano (*wired o wireless*) e per il tipo di applicazioni. Useremo l'host generico (postazione utente dedicata, con applicazioni client) e il server che emula i servizi di rete (HTTP, DNS, ecc.)
- f) **Firewall**
- g) **Altre categorie**

Costruzione di una prima semplice rete

Tipo plug and play: lo switch (come tipicamente è nella realtà) ha una configurazione di partenza che lo rende immediatamente operativo, cosa che non vale per i router.

1. Seleziono lo switch 2960 e lo metto nello spazio di lavoro

*Nota: quella che vediamo è la versione **logica** dello spazio di lavoro che è diversa da quella fisica che invece rappresenta la collocazione fisica degli apparati*

2. Aggiungo un host generico e un server

3. Collego gli apparati con i cavi: essendo switch, PC e server apparati con interfacce ethernet il cavo da usare è copper straight-through

Dopo aver collegato compaiono due luci, inizialmente quella lato switch è arancione perché significa che la porta non è ancora operativa infatti dopo un po' diventano entrambe verdi (rappresentano i led che indicano sui dispositivi fisici il corretto funzionamento).

Interazione coi singoli apparati

1) **PC:**

- a) *Config:* finestra di configurazione dove possiamo ad esempio disabilitare i diversi moduli. Se disabilito ad esempio la porta ethernet, anche dal lato switch i led diventa rosso perché sente che "non c'è niente" dall'altra parte. Quando la riaccendo torna subito verde lato PC, mentre sullo switch rimane arancione perché deve fare "sensing".
- b) *Desktop:* lista delle applicazioni (applicazioni di rete semplificate) che si possono lanciare la cui esecuzione verrà emulata dall'apparato

2) **Switch:** non devo configurare nulla per il momento essendo plug and play

3) **Server:** nella visione fisica posso aggiungere moduli, per farlo si seleziona il modulo corrispondente e lo si trascina nella figura – non si può farlo quando l'apparato è acceso Oltre alla finestra di configurazione anche il server ha un sistema operativo da cui possiamo accedere al terminale, configurare gli indirizzi IP, ecc.

Il server ha in più la finestra dei *servizi* per la configurazione di quelli in esecuzione: HTTP, DHCP, DHCPv6, TFTP, DNS, ecc. di cui alcuni sono attivi come configurazione predefinita, altri no.

Test di connessione.

Dal computer (nella finestra desktop) si avvia il prompt dei comandi e si lancia il comando *ping 192.168.1.2* (*indirizzo del server*), verificando corretto funzionamento: PC e Server si scambiano pacchetti IP. Per verificare il servizio DNS è necessario fare *ping* al nome di dominio (che si può associare nella scheda di configurazione del servizio DNS del server).

ARP table: primo comando ping corrisponde all'invio di un pacchetto ICMP, quindi il protocollo IP fa un confronto tra il proprio indirizzo di rete e quello destinatario. Entrambi gli IP iniziano con 192.168.1, cioè hanno uguali i primi 24 bit e sono sulla stessa *subnet* quindi possono comunicare direttamente. Però per mandare i frame ethernet al server devo sapere l'indirizzo MAC del server quindi prima di mandare il pacchetto IP che trasporta la richiesta di echo il PC ha mandato un *pacchetto ARP* in broadcast sulla rete LAN che contiene una richiesta di risposta per chi nella rete LAN ha IP 192.168.1.2. Il server lo riceve e riconosce il proprio indirizzo IP, quindi risponde comunicando l'indirizzo MAC che l'host salva. Dopo di che l'host manda il pacchetto IP incapsulato nel frame ethernet al server contenente la richiesta di eco (ping).

modo di simulazione – Si può decidere e manipolare l'andamento della simulazione della rete. *ping 192.168.1.2* → il tempo è fermo e nell'area di lavoro accanto al PC sono comparsi due pacchetti (due colori diversi) che rappresentano protocolli diversi cioè ARP e ICMP. L'applicazione ping prende l'argomento, accede a livello IP e confeziona un echo request, che per essere mandato ha bisogno di un indirizzo MAC (perché usa ethernet), quindi viene creato un pacchetto ARP e l'altro viene messo in standby.

Nella finestra *info* su un **pacchetto specifico** nella finestra di simulazione ci viene mostrata la spiegazione di ciò che sta succedendo:

- 1) **ICMP:** in *out-layer* (viene inviato) ci sono i diversi livelli che vengono attraversati e la descrizione dei singoli passi:
 - a. *Layer 3:* ping crea un ICMP Echo Request message e la manda al livello inferiore. Il source IP non è specificato per cui il device lo imposta sul proprio IP, poi vede che il source e il destination sono sulla stessa subnet quindi il pacchetto va mandato direttamente.
 - b. *Layer 2:* ARP guarda nella ARP table e non trova l'IP del destinatario, quindi il processo manda una richiesta ARP per quell'indirizzo e il pacchetto ICMP viene messo in attesa

Nella finestra “Outbound PDU Details” si può visualizzare il contenuto del pacchetto con il pacchetto IP e il pacchetto ICMP e i relativi vari campi.

2) ARP:

- a. *Layer 2:* crea la richiesta ARP per lo specifico IP e la incapsula in un frame ethernet.
Outbound PDU Details: pacchetto Ethernet (dest. MAC: FFFF.FFFF.FFFF - broadcast) e il pacchetto ARP (. MAC: 0000.0000.0000 perché ancora non lo si conosce)

Andando avanti il pacchetto ARP viene mandato allo switch e possiamo vedere passo passo cosa fa al *layer 2*: prima di inoltrarlo lo switch ethernet deve riempire la propria tabella (self-learning) perché sta scoprendo che sulla porta 1 c'è connesso qualcuno che ha l'indirizzo MAC xxxx.xxxx.xxxx.

Poi il pacchetto ARP arriva al server - se ne avessimo avuti più d'uno il pacchetto ARP sarebbe arrivato a tutti – che riceve il frame ethernet (target MAC 0), poi manda la risposta dove il MAC non è più 0 ma il proprio.

Il pacchetto arriva allo switch, poi al computer e nella ARP table adesso c'è una entrata; il pacchetto ICMP ora può essere mandato e infatti nel pacchetto oltre a IP e ICMP c'è anche il frame ethernet.
Andando avanti lo switch lo manda al server (stavolta non in broadcast), poi l'echo reply viene mandato all'host e nel terminale compare la prima riga dell'output del comando *ping*, quando manda la successiva richiesta non viene usato ARP quindi parte direttamente il pacchetto ICMP.

Un **Router** (IP) ha i componenti tipici di un calcolatore (CPU, RAM, memoria, ecc.) che vengono gestiti dal sistema operativo di Cisco (IOS); l'interfaccia del router è orientata per lo svolgimento di funzioni specifiche.

I pacchetti vengono instradati dal router usando come informazione solo l'indirizzo di destinazione presente nel pacchetto (regola base del protocollo IP), questo valore viene confrontato con la struttura dati interna, cioè la tabella di routing attiva in quel momento – quindi popolata usando informazioni da varie sorgenti.

Nota: un router può usare + protocolli di router contemporaneamente (su sottoinsiemi diversi delle sue interfacce)

Tabella di routing: lista di destinazioni note, ciascuna identificata da un indirizzo di rete (0-32 bit per IPv4) e a ciascuna è associata una interfaccia di inoltro, se un pacchetto fa match con più righe (prefissi di lunghezza diversi) la regola è che viene scelta la riga col prefisso più lungo (*longest prefix match*).

Possiamo in ogni momento “chiedere” al router quale tabella di routing stia usando, vedere cioè lo stato interno in quel dato momento.

Anatomia di un router – CISCO 1841



1. **Porte fast ethernet:** *fe0/0 e fe0/1*

2. **Slot memoria:** normalmente il router tiene il sistema operativo nella memoria removibile, questo per poterlo gestire senza dover rimuovere l'apparato dal rack

3. **Slot 1:** modulo aggiuntivo con quattro porte ethernet di livello 2 che corrispondono alle porte di uno switch montato internamente al router. Queste non hanno quindi indirizzi IP come le porte fast ethernet, perché appartengono allo switch che è “virtualmente” collegato al router

4. **Slot 0:** modulo aggiuntivo con due porte seriali

5. Porta console: utilizzata per l'input/output con PC con porta seriale e dà accesso a una interfaccia a linea di comando. Quando il router viene acceso per la prima volta non ha un comportamento predefinito quindi non può essere configurato da remoto; è utile anche in seguito in caso di guasti che possono isolare il router dalla rete rendendolo non accessibile da remoto.

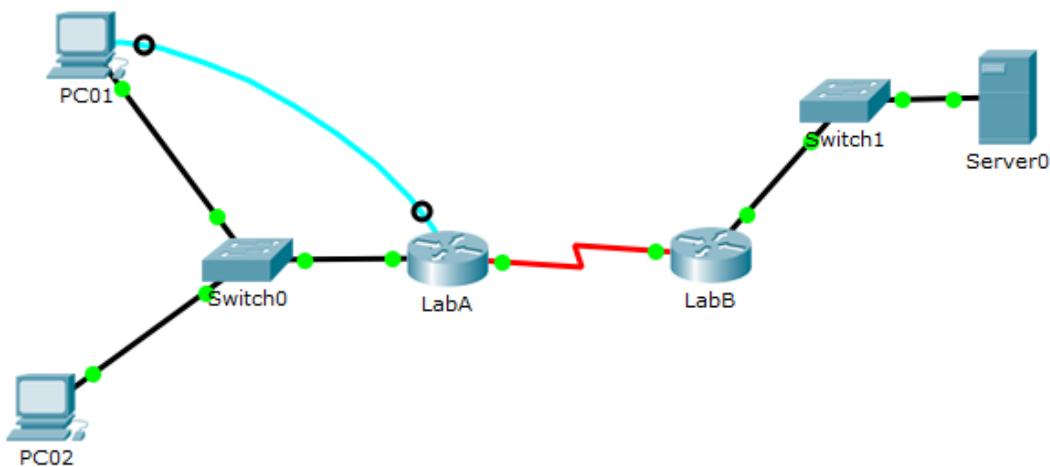
[Packet Tracer]

Usiamo un Router 1841 a cui si aggiungono i moduli HWIC-2T e HWIC-4ESW)ì negli slot come in figura sopra.

Per interfacciarsi con il router si usa un computer, si collega alla porta seriale (RS 232) il cavo che va al alla porta di console del router e si avvia l'applicazione terminale. Questa chiede l'uso della porta seriale al sistema operativo e l'output iniziale che vediamo è l'output del router che ha fatto il bootstrap cioè ha trovato l'immagine dell'OS e l'ha caricata in memoria.

Dal PC stiamo facendo input-output di comandi con il router attraverso il prompt di comando.

Esempio 1:



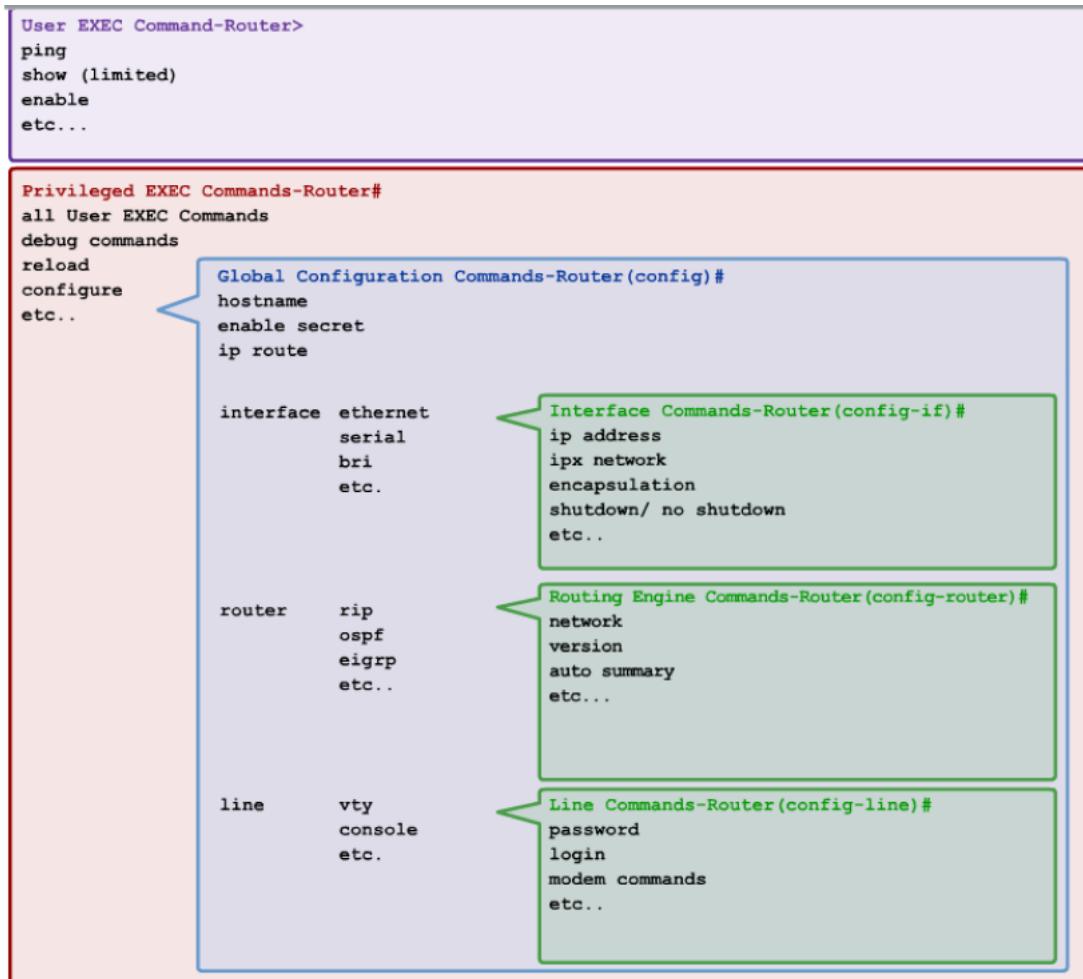
Rete già pronta e configurata: due router collegati in serie tramite le due rispettive porte seriali, ciascuno dei due ha poi connessa una rete locale (switch): in quella LabA ci sono due PC e nella rete di LabB è connesso un server. Il PC01 è collegato tramite la porta seriale con la porta di console del router LabA, quindi possiamo aprire l'emulatore di terminale (viene richiesta una password perché il router è configurato in accesso controllato). Ora abbiamo accesso al router R1 (quello di LabA) e possiamo eseguirci dei comandi.

Nota: per praticità useremo la finestra CLI del router che ci mostra ciò che vedremmo se ci collegassimo all'apparato con il cavo di console, con questa finestra lo possiamo vedere anche se nella rete non abbiamo messo un PC collegato alla porta di console. (infatti lo possiamo fare anche per il router LabB).

Output tipo %xxxxx: arrivano in maniera asincrona dal router per notificare qualcosa a chi è connesso con l'interfaccia, in questo caso avremo:

```
%LINK-5-CHANGED: Interface Serial2/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up
```

Struttura interna dell'interfaccia a linea di comando:



Inizialmente siamo in **modo utente** (user exec - >) dove possiamo eseguire una serie di comandi considerati per l'utente qualunque.

Per diventare **utente privilegiato** (#) dobbiamo inviare uno specifico comando e poi inserire una password, in questo modo si ha accesso a informazioni e comandi sensibili e possiamo modificare il comportamento del router entrando nei suoi modi di configurazione.

Modo di configurazione globale: funzionalità del router a livello globale

Modi di configurazione specifici: modo di configurazione di una specifica interfaccia, modo di configurazione del protocollo di routing, ecc.

Nota: ogni volta che cambia il modo cambia il prompt e cambia l'elenco dei comandi disponibili.

CLI router LabA:

```
R1> ping [indirizzo IP server]
// successo: il router riceve pacchetti destinati a quel server e sa come recapitarli
R1> ?                                // lista dei comandi disponibili
R1> show [parametro]
R1> show ?                            // elenco dei parametri possibili
R1> show arp                           // mostra il contenuto della tabella ARP interna al router
R1 > show clock                         // ora configurata sul router
R1 > enable                            // passare al modo privilegiato
R1# disable                            // uscire dal modo privilegiato
R1# configure terminal                 // entrare nel modo di configurazione
R1(config)# interface fa0/0
// entrare nel modo di configurazione dell'interfaccia fast ethernet 0/0
R1(config-if)# end                   // esco dalla configurazione e torno in modo privilegiato
R1# clock set 12:08:00 7 MARCH 2022    // cambiare l'ora sul router
```

```
// cambiare nome del router
R1# configure terminal
R1(configure)# hostname LabA
LabA(configure)#

```

```
// mostrare la tabella di route del protocollo IP
LabA# show ip route
// mostrare informazioni sulle interfacce: IP, stato, protocollo di linea
LabA# show ip interface brief"
```

Nota: il protocollo di linea può essere down anche se l'interfaccia è up, questo accade se i due apparati vogliono usare protocolli diversi e quindi non si "sentono".

Accesso remoto all'interfaccia CLI -> da PC si usa SSH (*telnet* non ha cifratura quindi usato solo su packet tracer per praticità).

PC> telnet [IP]

Che IP devo usare? Serve l'IP di una interfaccia che sia on, ma fast ethernet o seriale? È indifferente, perché se al router a cui vogliamo accedere abbiamo dato quegli indirizzi IP significa che quel nodo con quell'indirizzo IP è raggiungibile attraverso quella rete.

Se usassi l'IP dell'*interfaccia seriale*: l'host prepara il pacchetto IP, il livello sotto vede che il prefisso non è lo stesso quindi lo deve mandare al router perché è il gateway, quindi manda il pacchetto con quell'IP e con l'indirizzo MAC dell'interfaccia *FastEthernet* di *LabA* (come host non so che il router è anche il destinatario finale). Il Router riceve il pacchetto, vede che il MAC è quello del suo di fast ethernet e capisce che è per lui, quindi spacchetta, guarda l'indirizzo IP destinatario e vede che è l'indirizzo di una delle sue interfacce.

Comandi di configurazione del router – password

Appena acceso il router non ha configurazione né controllo di accesso quindi entriamo in modalità utente e poi in modalità privilegiata senza bisogno di inserire password.

La configurazione della password di accesso per l'utente viene vista, dal router, come configurazione della porta che si usa per scambiare dati col router.

```
Router>enable
Router#configure terminal
Router(config)#line console 0          //modo di configurazione della linea console 0
Router(config-line)#password [password]
Router(config-line)#login             // ora verrà richiesta la password per CLI

Router#configure terminal
Router(config)#enable secret [password] // configurazione psw per lvl privilegiato
```

Nota: in generale in IOS per annullare un comando di usa: no [comando da annullare]

Questo per quanto riguarda la linea di console, ma se adesso provassi con telnet verrà chiesta la password per il modo privilegiato ma non per entrare perché la password l'ho impostata per la linea di console.
 La linea da configurare per l'accesso da remoto è la linea vty (sul router ci sono 16 linee quindi 16 terminali virtuali distinti che si possono usare in parallelo).

```
Router#configure terminal
Router(config)#line vty 0-15      // i comandi verranno applicati a tutte le linee
Router(config-line)#password [password]
Router(config-line)#login
```

Configurazione Interfacce

Per accedere alle interfacce si usa il comando *interface [identificatore]* (si possono specificare solo le interfacce fisicamente presenti).

```
Router(config) #interface type          // type: fast ethernet, gigabit ethernet
Router(config) #interface type slot/port // interfacce su moduli negli slot
Router(config) #interface type slot/subslot/port
```

Le interfacce possono trovarsi in due stati cioè **shutdown** e **non shutdown**, vengono di default tenute shutdown per fare in modo che vengano usate solo dopo aver configurato il router; anche se l'interfaccia è connessa rimane in stato inattivo finché esplicitamente non viene attivata.

```
Router(config-if) #shutdown
Router(config-if) #no shutdown
```

Mezzi fisici di collegamento:

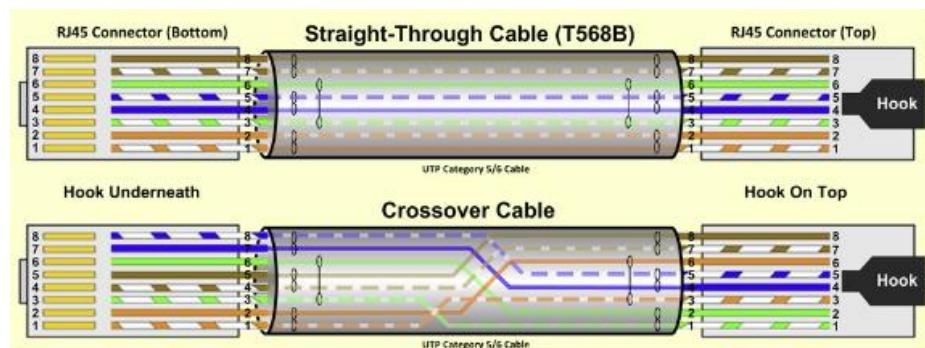
- segnale elettrico -> cavi di rame tipicamente accoppiati (twisted pair)
- segnale ottico -> fibra
- onde elettromagnetiche -> interfacce wireless

Copper Twisted Pair

Quando a lvl 2 si usa il protocollo Ethernet (IEEE 802.3) ci possono essere differenze a livello fisico: i cambiamenti a livello fisico sono invisibili ai livelli superiore quindi con Ethernet si possono usare cavi coassiali, twisted pair, fibra ottica, ecc.

- a) **UTP:** unshielded twisted (copper) pair
- b) **STP:** shielded twisted pair. Coppie inguiniate all'interno di una lamina per aumentare la bontà del canale in termini di larghezza di banda.

Cavi Straight-Through e Crossover



Quando si collegano ad esempio un computer e uno switch si può collegare una qualsiasi delle due estremità a uno dei due apparati, in realtà nelle porte si ha una coppia di pin usata per la trasmissione (1 e 2) e una per la ricezione (3 e 6).

Se 1 e 2 servono per la trasmissione da una parte, all'altra estremità del collegamento dovranno invece essere usati per la ricezione, per questo gli switch hanno un hardware interno che **inverte le due coppie**.

In generale se le due estremità usano **regole diverse** si utilizza un cavo di tipo **straight-through**, se invece le due invece usano la **stessa regola** si usa un cavo di tipo **crossover** che inverte le due coppie e fa diventare quella che è trasmissione da una parte ricezione dall'altra parte e viceversa.

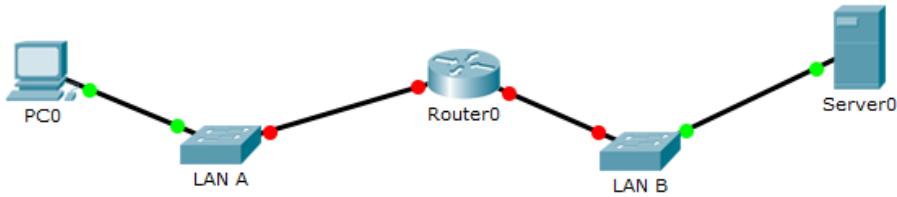
*Al giorno d'oggi gli switch hanno internamente un hardware che si **adatta** a quello che "sente" dall'altra parte, inizialmente assume che dall'altra parte ci sia un host (quindi cavo straight), ma se rileva che non c'è comunicazione allora commuta le coppie.*

L'interfaccia ethernet può usare anche il cavo in fibra ottica; l'hardware è diverso, perché il segnale viene generato da un led e l'informazione è codificata col segnale luminoso:

- Modo singolo: fascio luminoso focalizzato che viaggia linearmente senza essere riflesso
- Modo multiplo: il fascio luminoso può essere riflesso e quindi seguire diversi "percorsi"

Con ethernet se devo collegare più router in uno stesso comprensorio privato le distanze potrebbero essere tali da necessitare l'uso di fibra piuttosto che cablaggio in rame ($> 100m$), ma dal punto di vista del router sono **sempre interfacce ethernet** e ci si aspetta di comunicare come normalmente comunica su una LAN ethernet.

[Packet Tracer] Configurazione interfacce ethernet



- Il Router utilizza FastEthernet0/0 per LAN A e FastEthernet0/1 per LAN B.
- I led dal lato switch rimangono rossi perché le due interfacce del router sono in shutdown finché non le attivo.
- Usiamo il blocco di indirizzi 192.168.10.0 per la LAN A e 192.168.20.0 per la LAN B

1. Configurazione degli indirizzi IP delle interfacce del router

```

Router>enable
Router#configure terminal
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip address 192.168.10.1 255.255.255.0
Router(config-if)#no shutdown
  
```

Appena si dà il comando di accensione il led diventa verde dalla parte del router (LAN A) e arancione per lo switch che rileva poi automaticamente la presenza del router

```
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
```

Il primo messaggio notifica il cambiamento dello stato dell'interfaccia, mentre il secondo specifica che è un link ethernet operativo perché dall'altra parte c'è a sua volta qualcuno che "parla ethernet"

2. Dopo aver configurato anche l'altra interfaccia verifico con il comando show ip interface brief

```

FastEthernet0/0 192.168.10.1 YES manual up up
FastEthernet0/1 192.168.20.1 YES manual up up
  
```

3. Assegno gli IP al PC e al server (192.168.10.100, 192.168.20.100) mettendo come gateway l'IP delle relative interfacce del router (rispettivamente 192.168.10.1, 192.168.20.1)

Facendo il ping dal PC al server verifico che il router funziona correttamente anche se non c'è il protocollo di routing, perché sta facendo routing tra due cose a cui è direttamente connesso – come si può vedere nella tabella col comando show ip route

Il router funziona anche se non c'è il protocollo di routing perché sta facendo routing tra due cose a cui è direttamente connesso; infatti, se faccio ip route show vedo che il router ha già due entrate

```

192.168.10.0/24 connessa direttamente con FastEthernet 0/0
192.168.20.0/24 connessa direttamente con FastEthernet 0/1
  
```

Gestione della configurazione

I comandi cambiano lo **stato interno del router**: assegnando un IP all’interfaccia l’informazione di stato è mantenuta dal router e associata all’interfaccia; attraverso l’interfaccia a linea di comando i comandi di configurazione si aggiungono a quelli precedenti, accumulandosi per stabilire lo stato interno del router. Se riavvio il router lo stato viene perso perché si trova nella RAM; l’insieme di questi comandi viene conservato all’interno di un file (`running-config`) che posso visualizzare con `show running-config`.

Il file mostra una storia “ricostruita” dei comandi eseguiti (vengono riordinati secondo una struttura interna ad IOS).

Si trova infatti:

```
interface fastEthernet0/0
    ip address 192.168.10.1 255.255.255.0
```

nota: manca ‘no shutdown’ perché eseguendo quel comando ho eliminato la riga che conteneva il comando `shutdown` da questo file

Bootstrap: quando il router viene avviato cerca - secondo una sequenza predefinita di posti - il file `startup-config` che si trova nella memoria non volatile (o nella rete) che viene copiato nella RAM ed eseguito in `running-config`.

Con il comando `copy running-config startup-config` il `running-config` viene copiato nello `startup-config`: riavviando il router questo si ritroverà nello stato in cui l’ho lasciato, perché nel bootstrap verranno fatte le operazioni di configurazione che abbiamo fatto e che erano salvate in `running-config`.
La persistenza della configurazione di un router viene gestita attraverso la separazione dei due file di configurazione uno dei quali sta in memoria persistente.

Il file `startup-config` può essere copiato in diverse destinazioni visibili con `copy startup-config ?`

- **Memoria flash (disco)**: `startup-config flash: [/invio] [nome file copia]`
 - Con `dir flash` viene mostrato il contenuto della memoria (c’è anche il .bin del SO).
- **Running-config**: l’operazione è un **merge** dei due file, se si vuole “resettare” il `running-config` è necessario riavviare il router
- **TFTP** (FTP con UDP): viene eseguito il backup di `startup-config` sul server (controllare che TFTP sia in funzione sul server).
 - Verrà quindi chiesto l’IP del server e poi il nome del file copia, possiamo verificare l’operazione esaminando il servizio TFTP sul server.
 - Il file può essere scaricato su una destinazione locale, come un altro router connesso in rete.

Routing

Senza il routing il router può inoltrare pacchetti solo alle reti con cui è direttamente connesso: questi viaggiano dalla sorgente attraverso il router e arrivano a destinazione facendo un solo *hop*, perché il router non ha altre informazioni se non per le reti a cui è direttamente connesso. La **tavella di routing** viene popolata attraverso il **routing**.

Per trasferire dati da una applicazione a un'altra i router devono saper fare l'**instradamento** dei pacchetti; la funzione può essere realizzata in diversi modi, ad esempio il protocollo IP si basa unicamente sull'indirizzo di destinazione e solo leggendo quello il router decide cosa fare del pacchetto.

- Piano dati: dove il router identifica il prossimo ricevitore del pacchetto (che può essere il destinatario finale o un altro router), lavora quindi sui dati utente, cioè i pacchetti IP originati da una applicazione.
- Piano di controllo: dove il router esegue le operazioni utili a riempire la tabella di routing acquisendo le informazioni o dalla configurazione del router stesso o dinamicamente dagli altri router.
 - L'approccio **distribuito** viene seguito anche dal protocollo IP: non avendo un router centrale si evita di avere un *single point of failure*, è ancora molto usato nelle reti aziendali

Tabella di routing

```
RE#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
      * - candidate default, U - per-user static route, o - ODR
      P - periodic downloaded static route

Gateway of last resort is not set

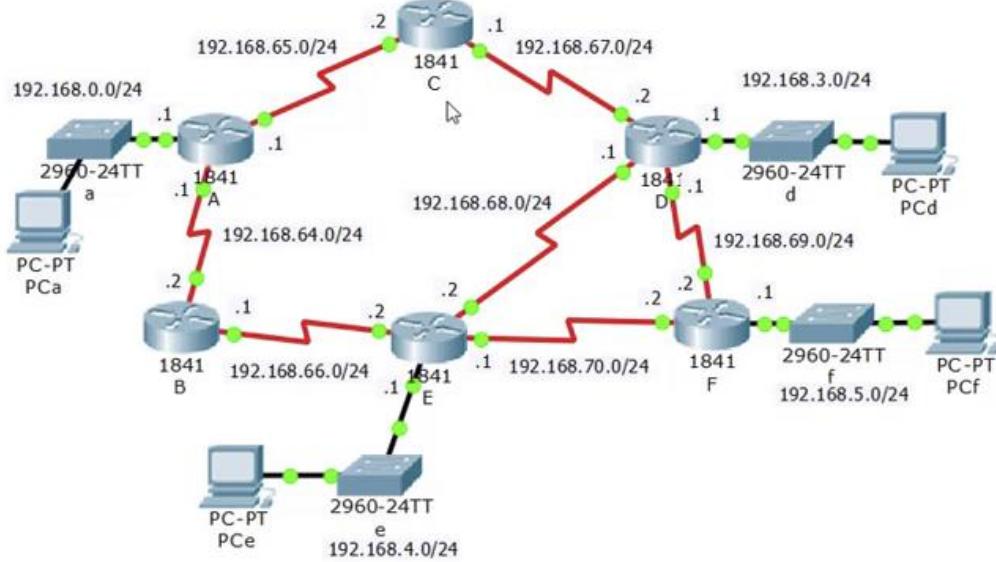
R 192.168.0.0/24 [120/2] via 192.168.66.1, 00:00:03, Serial0/0/1
R 192.168.3.0/24 [120/1] via 192.168.68.1, 00:00:02, Serial0/1/0
C 192.168.4.0/24 is directly connected, FastEthernet0/0
R 192.168.5.0/24 [120/1] via 192.168.70.2, 00:00:03, Serial0/0/0
R 192.168.64.0/24 [120/1] via 192.168.66.1, 00:00:03, Serial0/0/1
R 192.168.65.0/24 [120/2] via 192.168.68.1, 00:00:02, Serial0/1/0
                           [120/2] via 192.168.66.1, 00:00:03, Serial0/0/1
C 192.168.66.0/24 is directly connected, Serial0/0/1
R 192.168.67.0/24 [120/1] via 192.168.68.1, 00:00:02, Serial0/1/0
C 192.168.68.0/24 is directly connected, Serial0/1/0
R 192.168.69.0/24 [120/1] via 192.168.70.2, 00:00:03, Serial0/0/0
                           [120/1] via 192.168.68.1, 00:00:02, Serial0/1/0
C 192.168.70.0/24 is directly connected, Serial0/0/0
```

Nella colonna delle destinazioni ci sono indirizzi più corti di un IP, se un prefisso è lungo quanto un IP (32 bit) la rete è fatta da un solo host, se è lungo 31 bit la rete è composta da due host (casi particolari); solitamente la lunghezza è ≤ 30 bit.

La regola è quella di trovare nella tabella di routing la riga che ha il prefisso più lungo corrispondente ai primi bit dell'indirizzo di destinazione del pacchetto.

Il contenuto della tabella di routing è responsabilità del protocollo di routing: se la tabella è riempita correttamente, ma un pacchetto non viene instradato correttamente allora il problema è nell'algoritmo del protocollo IP.

[Packet Tracer]



- Sullo schema si riportano gli indirizzi di rete assegnati alle singole reti, ad esempio alla rete LAN dove si trova il PC E è stato assegnato 192.168.40.0/24.

Nota: non è vietato usare prefissi più corti anche se la rete è composta da due soli host (B ed E in figura), semplicemente vengono “sprecati” degli indirizzi IP

- I numeri “.1, .2” rappresentano l’indirizzo dell’interfaccia sulla relativa subnet, per esempio, B ha come indirizzo 192.168.66.1 (prefisso di rete + indirizzo dell’host), mentre E ha 192.168.66.2.

Tabella di routing (E) - RE# show ip route

| | | | | | |
|-----|----------------|---------|--|----------|-------------|
| R | 192.168.0.0/24 | [120/2] | via 192.168.66.1 | 00:00:19 | Serial0/1/0 |
| R | 192.168.3.0/24 | [120/1] | via 192.168.68.1 | 00:00:12 | Serial0/1/0 |
| C | 192.168.4.0/24 | | is directly connected, FastEthernet0/0 | | |
| R | 192.168.5.0/24 | [120/1] | via 192.168.70.2 | 00:00:10 | Serial0/0/0 |
| ... | | | ... | ... | ... |

(riga 2): se un pacchetto arriva sul router E e ha come destinazione la sottorete 192.168.3.0 (LAN D) il router inoltra il pacchetto attraverso 192.168.68.1 (router D) usando l’interfaccia seriale 0/1/0.

Questo non è l’unico percorso possibile, ma è quello più breve quindi l’unico a essere mostrato nella tabella; se cancellassi il link che connette router E e router D, la tabella di routing verrebbe modificata automaticamente includendo il più breve percorso alternativo

Le lettere nella prima colonna indicano da dove ha origine l’informazione che si trova in quella riga.

- Connected (C): il router sa di poter raggiungere quella rete di destinazione perché è direttamente connesso a una interfaccia con indirizzo IP con lo stesso prefisso del destinatario
- RIP (R): il router ha trovato questa informazione tramite il protocollo RIP
- EIGP (D): protocollo di routing proprietario di Cisco
- Static (S): informazione statica (non la scopriamo attraverso uno scambio dinamico) nel file di configurazione: si fornisce manualmente al router una informazione di instradamento

Percorsi multipli

```
R 192.168.65.0/24 [120/2] via 192.168.66.1, 00:00:09, serial0/0/1
[120/2] via 192.168.68.1, 00:00:12, serial0/1/0
```

Se E riceve un pacchetto con destinazione 192.168.65.1 (router A della sottorete) per raggiungere la rete ha due alternative, via 66.1 (B) o via 68.1 (D), vengono mostrate entrambe perché i due percorsi hanno la **stessa distanza (costo)**. RIP usa come metrica il numero di hop e per raggiungere quel link (192.168.65.0) devo entrambi i casi farne 2: (B-A) e (D-C).

ECMP (equal cost multiple paths): funzionalità tipicamente implementata su tutti i router che acquisiscono la capacità di instradare pacchetti su più percorsi se questi hanno lo stesso costo.

I pacchetti verso una destinazione con più percorsi disponibili verranno mandati in maniera alternativa su un percorso e sull'altro (**load balancing**).

Nota: di solito questo viene fatto fino a un numero massimo di percorsi disponibili – tipicamente 4.

Metriche e costi

Il costo dei percorsi è misurato attraverso una metrica diversa in **base al protocollo di routing**. Per il protocollo RIP la metrica è il **numero di hop**, cioè il numero di router che devo attraversare. Per altri protocolli di routing il costo può essere calcolato con criteri diversi ad esempio assegnando ad ogni link un costo secondo delle regole impostate dal gestore (*Cisco mette il costo come l'inverso della capacità del link*).

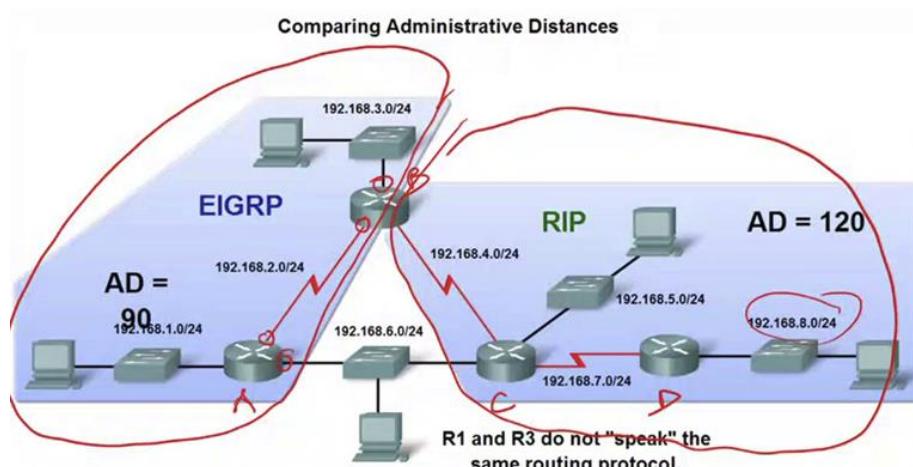
EIGRP usa una funzione per il calcolo del costo che dipende dalla capacità del link, ma anche da misure di latenza e affidabilità.

Nella tabella il costo di ogni percorso è indicato dal secondo numero nelle parentesi quadre

```
R 192.168.0.0/24 [120/2]
```

In questo caso il numero di router che mancano per raggiungere la destinazione (LAN A) a partire da E è **2**: bisogna attraversare i router B e A

Distanza Amministrativa



Nella rete abbiamo 4 router: A e B con le rispettive interfacce utilizzano il protocollo di routing **EIGRP** e B, C e D usano il protocollo **RIP**. Questi due processi sono indipendenti: infatti sul router B le due interfacce sono “partizionate” in due sottoinsiemi distinti e il router utilizza per questi due, due protocolli di routing separati.

È come se B avesse 2 router virtuali separati: uno con solo interfacce che lavorano con EIGRP e l'altro con interfacce che lavorano con RIP, i due router virtuali si parlano ma non si scambiano informazioni di routing.

B sa come raggiungere le destinazioni in entrambi i domini, ma gli altri router no, se voglio che le informazioni vengano copiate da un dominio all'altro devo dirlo manualmente al router “*nel dominio RIP vai a trasferire anche le informazioni che hai imparato dal dominio EIGRP*”

La logica è quella di voler **separare i dominii di routing**.

Guardiamo la **rete 192.168.6.0/24**: a questo link sono collegati A e C che sono in domini separati: ciascuno indipendentemente sa come raggiungere questa rete (direttamente connessa) e lo comunicano al router B con il relativo costo. Essendo due dominii diversi **i costi dei due percorsi non sono confrontabili tra loro**.

Administrative distance: valore associato al modo in cui l'informazione è stata appresa.

EIGRP ha AD = 90, RIP ha AD = 120, quindi qualunque sia il costo calcolato dai due protocolli per i relativi percorsi EIGRP è più affidabile di RIP e quindi B dovrà usare l'informazione ottenuta tramite EIGRP.

Il **costo** di un percorso è quindi composto: **[AD/costo]**

Se ci sono più percorsi alternativi si sceglie quello con AD minore, a parità di AD si sceglie/scelgono quello/i con costo minore

| Route Source | Administrative Distance |
|---------------------|-------------------------|
| Connected | 0 |
| Static | 1 |
| EIGRP summary route | 5 |
| External BGP | 20 |
| Internal EIGRP | 90 |
| IGRP | 100 |
| OSPF | 110 |
| IS-IS | 115 |
| RIP | 120 |
| External EIGRP | 170 |
| Internal BGP | 200 |

L'informazione “statica” è stata ottenuta dall'amministratore di rete quindi considerata affidabile, vincerà sempre contro ogni altra sorgente di tipo dinamico; si può anche forzare il router a fare anche un percorso più lungo

Configurazione route statica

```
router(config)# ip route network-address subnet-mask {ip-address | exit interface } [distance]*
```

Il next hop si specifica mettendo un indirizzo IP o indicando qual è l'interfaccia di uscita dove inoltrare il pacchetto

(*) la distanza amministrativa è un parametro opzionale: installando una rotta statica posso impostare una distanza amministrativa diversa da quella predefinita (1), magari perché venga usata quando il router non sa come raggiungere la destinazione (come backup). Oppure si potrebbe evitare di avere il load balancing impostando una AD maggiore per un percorso più costoso che però rimane disponibile in caso di backup.

Esempio: RE(config)# ip route 192.168.3.0 255.255.255.0 192.168.66.1
 (Sul router E per andare ad A passo per B)

Nella tabella di routing l'entrata precedente per quella destinazione (ottenuta tramite RIP) è stata sostituita da quella appena inserita staticamente, perché l'informazione statica ha AD minore.

L'informazione fornita dal protocollo RIP relativa quella entrata non è stata scartata e lo si può verificare annullando il comando di `ip route` appena inserito e visualizzando la tabella di routing, perché il router gestisce le **diverse informazioni di routing tramite strutture dati separate**.

Nei router CISCO si usano i **Routing Information Base** (nel piano di controllo): ci sono database separati per ogni possibile sorgente di informazione (*database RIP, file di configurazione statica, database OSPF, ecc.*). Queste strutture sono indipendenti e aggiornate periodicamente e continuamente il router (tramite un processo interno) prende l'informazione migliore per ogni percorso e lo mette nella **forwarding information base** (struttura che lavora nel piano dati).

Il processo è quindi **dinamico**: se una informazione sparisce dal routing information base (ad esempio cancello una informazione dal file delle rotte statiche) allora sulla forwarding information base si rimette quella che c'era prima. Il router non si è "dimenticato" del percorso che gli ha indicato RIP, semplicemente non è più il modo migliore; quindi, non compare nella forwarding information base.

- Il next hop può essere quello di un router a cui questo router è direttamente connesso oppure no, nel caso il router utilizza la tabella in modo ricorsivo
- IP gestisce il flusso solo in una direzione e l'indirizzo sorgente è irrilevante al fine del routing; a livello ethernet (2) il sorgente è fondamentale, mentre a livello IP (3) non viene considerato.
 Il router quando riceve la risposta ad un pacchetto inoltrato in precedenza non può agire basandosi su quanto fatto per la richiesta: il **routing IP è unidirezionale e dipende solo dall'indirizzo di destinazione**.
L'esistenza di un percorso per un pacchetto IP in una direzione non implica che esista il percorso nella direzione opposta.

Se il router non trova la destinazione nella tabella di routing non sapendo a chi mandare il pacchetto il router può solo buttarlo, ma - in base alla configurazione del router – si può poi mandare al sorgente un pacchetto ICMP che ha come codice *destination unreachable*. Il router informa il mittente che non sa come raggiungere la destinazione, inserendo nel pacchetto ICMP l'header del pacchetto originale (*specifica quindi qual è la destinazione che non può raggiungere*).

Quando un pacchetto IP viaggia attraverso un router cambiano solo due campi:

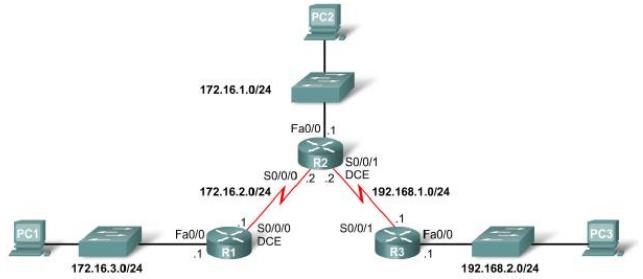
- TTL
- checksum – ricalcolato come conseguenza del decremento del TTL

Non cambiano mai indirizzo sorgente e destinatario, perché si riferisce alla destinazione finale; le trasmissioni intermedie avvengono a livello 2 (ethernet) non a livello IP.

Comando debug: per attivare l'output dei messaggi di debug per particolari eventi, ad esempio con il comando R1# debug ip routing si attivano i messaggi di debug che hanno a che fare con la tabella di routing in modo che ogni modifica di questa venga notificata.

Recursive route lookup

```
R1# show ip route
S 172.16.1.0 [1/0] via 172.16.2.2
C 172.16.2.0 is directly connected, Serial0/0/0
C 172.16.2.0 is directly connected, FastEthernet0/0
S 192.168.1.0/24 [1/0] via 172.16.2.2
S 192.168.2.0/24 [1/0] via 172.16.2.2
```



Se R1 riceve un pacchetto destinato alla sottorete 192.168.2.0/24 deve sapere qual è l'interfaccia di uscita da utilizzare per instradare il pacchetto, ma nella tabella è riportato un altro indirizzo IP, quindi viene fatto un **lookup ricorsivo nella tabella di routing**.

- 1) R1 riceve un pacchetto con destinatario un indirizzo della sottorete 192.168.2.0/24
- 2) Cerca l'indirizzo nella tabella di routing e trova che, per raggiungere la sottorete, deve inoltrare il pacchetto all'indirizzo IP 172.16.2.2
- 3) Cerca quest'ultimo nella tabella di routing e trova che, per raggiungerlo, deve instradare il pacchetto sulla porta serial0/0/0

=> l'algoritmo di lookup è stato quindi eseguito due volte

- Nella realtà il processo viene **ottimizzato**: nella struttura dati vengono risolte inizialmente tutte le ricorsioni in modo da avere subito per ogni riga l'interfaccia d'uscita.
- Se si cambia l'indirizzo all'interfaccia è necessario cambiare anche la riga statica che contiene il suo indirizzo IP (in questo caso la riga di 192.168.2.0/24)

Configurazione: R1(config)# ip route 192.168.2.0 255.255.255.0 [IP next hop]:

l'IP del next hop può essere un indirizzo IP qualunque e non deve necessariamente essere quello di una interfaccia a cui il router è connesso perché tanto verrà risolto ricorsivamente finché non si trova la riga con l'interfaccia di uscita. Ci sono casi in cui è utile specificare un IP “più lontano”

Esempio: rete con dei router di confine che hanno dentro una sottorete di router. Ogni pacchetto che transita in questa rete entra tramite uno dei router di frontiera ed esce tramite un altro router di frontiera: si potrebbe quindi impostare che, per una certa rete di destinazione, su un certo router di frontiera il next hop sarà un altro specifico router di frontiera. Viene fissato un passaggio intermedio nel percorso che non è necessariamente a distanza 1.

Opzioni di configurazione di una route statica: invece che specificare come next hop l'indirizzo IP si può specificare il nome dell'interfaccia, in questo caso nella tabella l'entrata corrispondente sarà **directly connected**

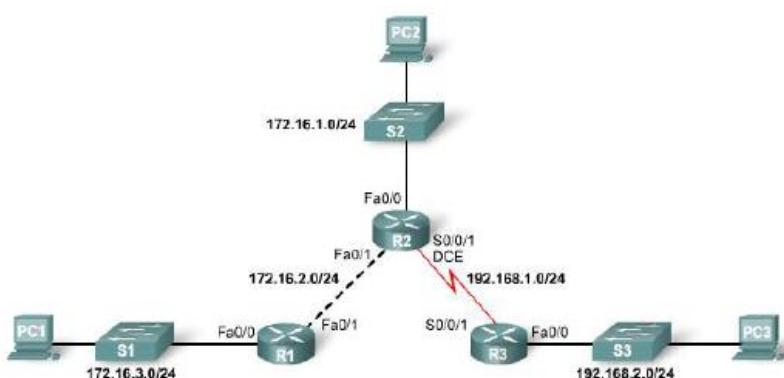
```
R1(config)# ip route 192.168.2.0 255.255.255.0 ?
// Le possibili opzioni per specificare la route sono:
A.B.C.D          //indirizzo IP
Ethernet
FastEthernet
GigabitEthernet
Loopback          // fa tornare il pacchetto a sé stesso
Null              // scartare i pacchetti per la destinazione
Vlan              // se il router ha uno switch devo specificare la VLAN di uscita,
```

Ci sono casi in cui specificare la rotta in un modo o nell'altro può fare la differenza:

- Se specifico l'interfaccia di uscita il router ha l'istruzione di prendere il pacchetto e trasmetterlo su quella interfaccia, quindi se questa fosse down il router non avrebbe più alcuna informazione su come trattare il pacchetto poiché rimuoverebbe dalla tabella ogni riga che prevede esplicitamente come uscita quell'interfaccia.
- Se specifico l'IP di un altro router, invece, non è detto che tra i due router ci sia un unico percorso quindi **l'indirizzo IP continua ad essere raggiungibile** anche se non lo è più tramite l'interfaccia: la **risoluzione ricorsiva di quell'indirizzo IP** porterà a una interfaccia diversa.

Esempio: R1 e R2 connessi tramite fast Ethernet.

```
R1(config)# ip route 192.168.2.0 255.255.255.0 fastEthernet 0/1
```



Se arriva un pacchetto a R1 con destinazione 192.168.2.0 il router non potrà inoltrarlo.

Viene fatto il lookup con la tabella di routing e si trova la entry che dice di mandare il pacchetto sulla porta fastEthernet 0/1.

Si scende quindi al *livello 2* dove il pacchetto IP va incapsulato all'interno del frame ethernet e poi trasmesso. R1, però **non ha l'indirizzo di livello 2 (MAC) del destinatario** di quel frame.

Quando R1 e R2 erano connessi tramite **porta seriale** invece non era un problema, perché **essendo un link punto-punto non si specifica un indirizzo di destinazione**.

Non posso mettere l'indirizzo broadcast perché R1 non sa quanti sono potenzialmente i destinatari; vede solo il suo switch, al quale potrebbero essere connessi molti altri router.

In questo caso nella route statica oltre a specificare l'interfaccia devo **specificare l'indirizzo IP assegnato all'interfaccia del router che è connesso a quel link**; stavolta l'IP non serve per fare la ricorsione nella tabella di routing, ma per **cercare l'indirizzo MAC nella tabella ARP**, quindi non si può mettere un IP qualsiasi, deve essere esattamente quello assegnato all'interfaccia dal *lato* del destinatario

```
R1(config)# ip route 192.168.2.0 255.255.255.0 fastEthernet 0/1 172.168.2.2
```

Quando usare routing statico:

- Forzare una rotta indipendentemente dai costi può essere utile per ragioni di sicurezza
- (Caso particolare): la rete ha un solo punto di ingresso/uscita e quindi sapere come è fatta internamente (cioè “partecipare” al routing dinamico) per chi sta fuori è totalmente irrilevante, poiché qualsiasi sia la destinazione dovrà passare per forza per questo punto.

Address planning

Classless IPv4 addressing: adottato per risolvere il problema della scarsità degli indirizzi IP, l'indirizzo è determinato dalla maschera di sottorete.

CIDR (Classless InterDomain Routing):

- la porzione di indirizzo dedicata alla sottorete è di dimensione arbitraria (**VLSM: variable lenght subnet mask**)
- formato indirizzi: **a.b.c.d/x**, dove x è il numero di bit della porzione di indirizzo dedicata alla subnet (*prima la classe era identificata dai primi x bit*)

Si eseguono operazioni su **blocchi di indirizzi** (insieme di indirizzi che corrispondono a un certo **prefisso**): supporre di avere a disposizione un prefisso significa supporre di avere a disposizione tutti gli indirizzi del blocco. Il blocco può essere ulteriormente suddiviso (**sub-netting** o sub-sub-netting).

Il Routing dà la possibilità di fare **supernetting**, cioè **aggregazione di prefissi**: se si hanno più prefissi di lunghezza x, ma hanno a comune i primi x-1 bit allora posso riferirli tutti con un prefisso di lunghezza x-1. **L'ottimizzazione** sta nella riduzione di entrate nella tabella di routing.

Assegnamento diretto – IANA (Internet Assigned Numbers Authority)

Per combattere il problema della scarsità degli IP si è adottata la soluzione del rinunciare alla univocità degli indirizzi: l'IANA mette da parte tre blocchi di indirizzi (dalle classi A,B e C) come **indirizzi IP privati**:

- L' uso è senza restrizioni nelle reti private
- È necessaria la traduzione per comunicare con la rete Internet

Esistono poi **blocchi di indirizzi speciali** che possono essere usati solo per lo scopo specifico a loro riservato:

1. **Tutti '1' nella parte di indirizzo dell'host:** indirizzo broadcast sul link
2. **"This" network - 0.0.0.0/8:** refer to source hosts on "this" network
3. **Loopback - 127.0.0.0/8:** l'host manda traffico a sé stesso, utile per fare operazioni che coinvolgono uno stack IP senza usare una interfaccia specifica, ad esempio per testare che l'host abbia uno stack IP funzionante.
4. **Link-local - 169.254.0.0/16:** usato da un host per assegnarsi automaticamente un indirizzo IP quando nessun altro modo è possibile. Quando un host, ad esempio, cerca di accedere a una rete, ma non ha e non riesce a ottenere un indirizzo IP, allora prende un indirizzo a caso tra i 2^{16} disponibili con quel prefisso. In questo modo si consente a degli host su una stessa rete di comunicare tra loro anche senza un servizio terzo che assegna gli indirizzi.
5. **TEST-NET addresses - 192.0.2.0/24:** usati per gli esempi nella documentazione di RFC. Sono riservati perché altrimenti si perderebbe la generalità dell'esempio, lo stesso se si usano indirizzi privati.

Calcolo dell'indirizzo di sottorete

Si prenda ad esempio la rete 172.16.20.0/25; il prefisso indica dove finisce l'indirizzo di rete e dove inizia quello dell'host

- a) 172. 16. 20. 0/25 - Indirizzo di Rete
 $10101100.00010000.00010100.0\textcolor{blue}{0000000}$
 |----- network -----| - host - |
- b) 172. 16. 20. 126 - Indirizzo di host più alto
 $10101100.00010000.00010100.0\textcolor{blue}{1111111}$
- c) 172. 16. 20. 127 - Indirizzo di Broadcast
 $10101100.00010000.00010100.0\textcolor{blue}{1111110}$

Per un indicare un indirizzo di rete i **bit in più** che devo specificare ma che **non sono significativi sono tutti a 0**: 172.16.20.0/25.

Nota: se scrivo nel file di configurazione 172.16.20.10/25, leggendo il /25 il software fa una operazione di ottimizzazione normalizzando l'indirizzo come 172.16.20.0/25

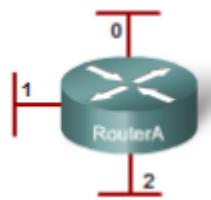
172.16.20.**140**/28 = 10101100.00010000.00010100.1000**1100**: nell'ultimo gruppo di cifre i primi 4 bit identificano la rete e gli ultimi 4 sono **variabili** e sono irrilevante per specificare la rete poiché specificano l'indirizzo di un singolo host.

*A seconda del valore di x alcuni A.B.C.D/x potrebbero non avere senso;
 nel dubbio si esamina la rappresentazione in base 2.*

Pianificazione

Partendo da un blocco di indirizzi a disposizione (es: 192.168.1.0/24) abbiamo una rete con un router e 3 LAN alle quali è connesso; da questo blocco devo ricavare 3 sotto-blocchi (**subnetting**) da assegnare alle 3 LAN che avranno prefisso più lungo rispetto al blocco iniziale.

Quello che non si può fare è prendere indirizzi da 192.168.1.1 a 192.168.1.254 ed assegnarli a caso agli host sulle 3 LAN: queste sono 3 link IP e **ciascun link deve avere il suo indirizzo di rete e gli host su quel link devono avere lo specifico prefisso**



Dividere in blocchi più piccoli significa decidere quanti bit della parte dell'host togliamo all'host ed assegniamo al prefisso di rete, cioè **di quanto si sposta a destra il confine, riducendo la dimensione di ciascun sotto-blocco e aumentando il prefisso**.

Questo il numero sotto-blocchi debba essere sempre una **potenza di 2** (2 -> sposto di 1b, 4 -> sposto di 2b, 8 -> sposto di 3b...).

Avendo qui 3 LAN si divide il blocco di partenza in 4 (potenza del 2 immediatamente successiva); quindi la maschera si allunga di 2 bit (26b), il confine che nella sottorete coincideva con il limite del terzo gruppo di cifre, adesso comprende anche i primi due bit del quarto gruppo (infatti sono a 1 nella maschera).

I blocchi hanno dimensione $2^{(32-(24+2))} = 2^6 - 2$ (riservati) = 62 indirizzi per ogni blocco.

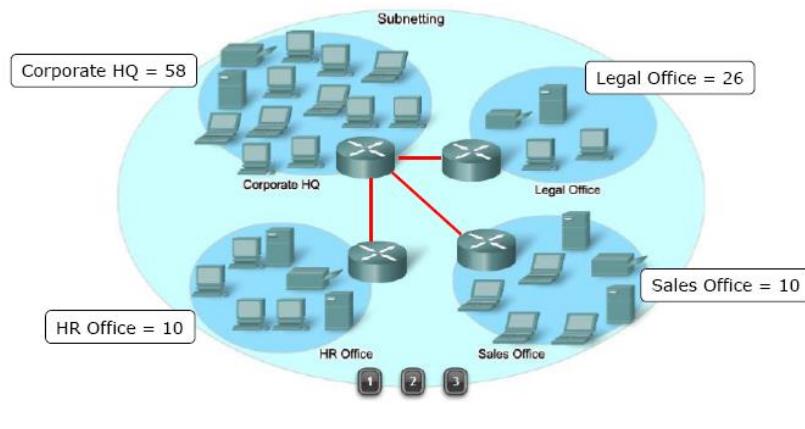
| | | |
|---------------------|--|---|
| - | 192.168.1.0 (/24) 255.255.255.0 | 11000000.10101000.00000001.00000000 11111111.11111111.11111111.00000000 |
| 0 | 192.168.1.0 (/26) 255.255.255.192 | 11000000.10101000.00000001.00000000 11111111.11111111.11111111.11000000 |
| 1 | 192.168.1.64 (/26) 255.255.255.192 | 11000000.10101000.00000001.01000000 11111111.11111111.11111111.11000000 |
| 2 | 192.168.1.128 (/26) 255.255.255.192 | 11000000.10101000.00000001.10000000 11111111.11111111.11111111.11000000 |
| 3 (inutilizzato) | 192.168.1.192 (/26) 255.255.255.192 | 11000000.10101000.00000001.11000000 11111111.11111111.11111111.11000000 |

| Subnet | Network address | Host range | Broadcast address |
|--------|------------------|-------------------------------|-------------------|
| 0 | 192.168.1.0/26 | 192.168.1.1 – 192.168.1.62 | 192.168.1.63 |
| 1 | 192.168.1.64/26 | 192.168.1.65 – 192.168.1.126 | 192.168.1.127 |
| 2 | 192.168.1.128/26 | 192.168.1.129 – 192.168.1.190 | 192.168.1.191 |
| 3 | 192.168.1.128/26 | 192.168.1.193 – 192.168.1.254 | 192.168.1.255 |

Nella tabella si riportano per ogni riga (blocco) un indirizzo di rete, il range di indirizzo IP che possono usare gli host, l'indirizzo di broadcast e la maschera

Esempio: abbiamo una rete privata di cui vengono fornite la topologia e le esigenze.

Il primo passaggio è determinare il prefisso di rete del blocco principale che verrà poi diviso: si deve trovare quindi qual è il blocco più piccolo sufficientemente grande che può indirizzare tutti i link (cioè il prefisso x più lungo)



- a) A questa rete saranno connessi in totale 104 host: non ci basta per trovare il prefisso finché non conosciamo la distribuzione; nella rete ci sono dei router, quindi, non c'è un'unica LAN e abbiamo 4 link IP.
- b) Ci viene fornita la distribuzione degli host: sottorete per sottorete si guarda qual è il prefisso che si può usare, in questo modo si determina la dimensione dei sotto-blocchi:

$$\text{HR Office} = 10 \quad \rightarrow \quad 2^4 (16)$$

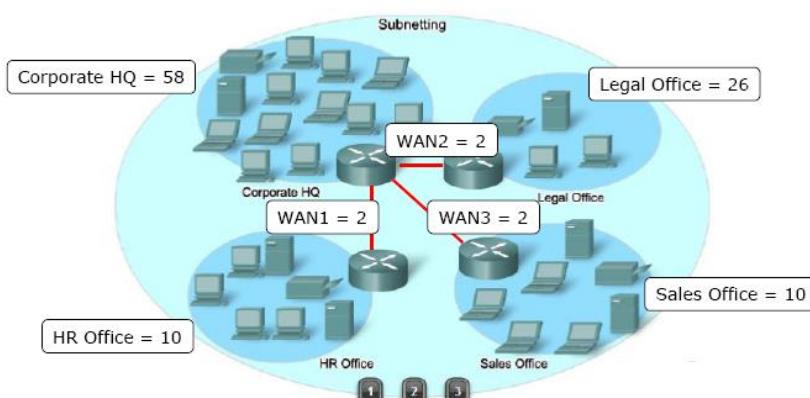
$$\text{Sale Office} = 10 \quad \rightarrow \quad 2^4 (16)$$

$$\text{Legal Office} = 26 \quad \rightarrow \quad 2^5 (32)$$

$$\text{Corporate HQ} = 58 \quad \rightarrow \quad 2^6 (64).$$

Sembrerebbe quindi sufficiente un blocco iniziale da 128 che viene diviso in due da 64; uno assegnato a Corporate HQ e l'altro diviso ulteriormente in due sotto-blocchi da 32. Uno di questi viene assegnato a Legal Office e l'altro diviso ancora una volta in due blocchi da 16 che vengono assegnati a HR Office e Sale Office.

MA per i link IP bisogna considerare anche i collegamenti tra i router: quindi non sono 4, ma 7; altrimenti si assegnano gli indirizzi, ma i router non riescono a comunicare tra loro. Quindi si deve prevedere l'allocazione di 7 indirizzi di rete: i 3 link “aggiuntivi” hanno esigenze contenute, ma comunque non nulle.



Il blocco iniziale da 128 non è sufficiente; serve un blocco di dimensione superiore cioè **256**, cioè con un prefisso a **24 bit** ($256 = 2^{(32-x)} = 2^8$, $x = 24$).

Il procedimento per trovare la dimensione minima del blocco iniziale è:

- 1) Trovare i link che hanno bisogno di un indirizzo di rete
- 2) Allocare i sotto-blocchi

Il blocco di partenza sarà quindi 192.168.15.0/24

Allocazione dei blocchi:

Definiti il numero di link e la dimensione di ciascuno, cioè il numero minimo di indirizzi diversi di cui hanno bisogno, **si parte dalla dimensione più grande** scendendo a quella a piccola; questo garantisce che in fondo avremo usato il blocco di indirizzi più piccolo possibile

Si parte dalla rete più grossa che ha bisogno di 58 host:

- I 126 indirizzi del blocco /24 sono divisi in due blocchi da 128 indirizzi ciascuno con prefisso /25
- Considero il primo sotto-blocco (.0) da /25 che basta per indirizzare 126 host (.0, .127)
- Dividendo questo blocco ottengo i due blocchi /26 sufficienti per 62 host: il primo blocco è .0 con range di indirizzi [.1, .62] e il secondo ha indirizzo .64 e il range è [.65, .126]
- Si divide ancora il primo di questi un due sotto-blocchi /27, e così via

1. Trovo il blocco minimo da usare per 58 host che è il blocco **.0 /26** (sufficiente 62 host)

Visto che uso questo **la sua ripartizione (*a destra*) non è più significativa** perché assegno questo blocco a Corporate HQ, inoltre avendo usato questo primo blocco /26 **la prima metà del blocco /25 non è più a disposizione**.

| /25 (1 subnet bit) 2 subnets 126 hosts | /26 (2 subnet bits) 4 subnets 62 hosts | /27 (3 subnet bits) 8 subnets 30 hosts | /28 (4 subnet bits) 16 subnets 14 hosts | /29 (5 subnet bits) 32 subnets 6 hosts | /30 (6 subnet bits) 64 subnets 2 hosts |
|--|--|--|---|--|--|
| .0 | | | | .0 (.1-.5) | .0 (.1-.2) |
| .4 | | | | .4 (.5-.6) | .4 (.5-.6) |
| .8 | | | | .8 (.9-.14) | .8 (.9-.10) |
| .12 | | | | .16 (.17-.30) | .12 (.13-.14) |
| .16 | | | | .24 (.25-.30) | .16 (.17-.18) |
| .20 | | | | .32 (.33-.38) | .20 (.21-.22) |
| .24 | | | | .40 (.41-.46) | .24 (.25-.26) |
| .28 | | | | .48 (.49-.54) | .28 (.29-.30) |
| .32 | | | | .56 (.57-.62) | .32 (.33-.34) |
| .36 | | | | | .36 (.37-.38) |
| .40 | | | | | .40 (.41-.42) |
| .44 | | | | | .44 (.45-.46) |
| .48 | | | | | .48 (.49-.50) |
| .52 | | | | | .52 (.53-.54) |
| .56 | | | | | .56 (.57-.58) |
| .60 | | | | | .60 (.61-.62) |
| .64 | | | | | .64 (.65-.66) |
| .68 | | | | | .68 (.69-.70) |
| .72 | | | | | .72 (.73-.74) |
| .76 | | | | | .76 (.77-.78) |
| .80 | | | | | .80 (.81-.82) |
| .84 | | | | | .84 (.85-.86) |
| .88 | | | | | .88 (.89-.90) |
| .92 | | | | | .92 (.93-.94) |
| .96 | | | | | .96 (.97-.98) |
| .100 | | | | | .100 (.101-.102) |
| .104 | | | | | .104 (.105-.106) |
| .108 | | | | | .108 (.109-.110) |
| .112 | | | | | .112 (.113-.114) |
| .116 | | | | | .116 (.117-.118) |
| .120 | | | | | .120 (.121-.122) |
| .124 | | | | | .124 (.125-.126) |

Corporate HQ = 58, 192.168.15.0/26

Si procede in ordine decrescente di dimensione, perché altrimenti la suddivisione creerebbe un buco e non verrebbero allocando i range di indirizzi sequenziali; in questo modo invece gli indirizzi non usati sono gli ultimi del range

2. La dimensione successiva è /26 e di quelli rimasti il blocco più piccolo che riesce a contenere 26 host è il blocco **.64 /27** che assegno a Legal Office: a destra non posso più ripartire nulla e il blocco più grande a sinistra non è più disponibile

| /25 (1 subnet bit) 2 subnets 126 hosts | /26 (2 subnet bits) 4 subnets 62 hosts | /27 (3 subnet bits) 8 subnets 30 hosts | /28 (4 subnet bits) 16 subnets 14 hosts | /29 (5 subnet bits) 32 subnets 6 hosts | /30 (6 subnet bits) 64 subnets 2 hosts |
|--|--|--|---|--|--|
| .0 | | | | .0 (.1-.5) | .0 (.1-.2) |
| .4 | | | | .4 (.5-.6) | .4 (.5-.6) |
| .8 | | | | .8 (.9-.14) | .8 (.9-.10) |
| .12 | | | | .16 (.17-.30) | .12 (.13-.14) |
| .16 | | | | .24 (.25-.30) | .16 (.17-.18) |
| .20 | | | | .32 (.33-.38) | .20 (.21-.22) |
| .24 | | | | .40 (.41-.46) | .24 (.25-.26) |
| .28 | | | | .48 (.49-.54) | .28 (.29-.30) |
| .32 | | | | .56 (.57-.62) | .32 (.33-.34) |
| .36 | | | | | .36 (.37-.38) |
| .40 | | | | | .40 (.41-.42) |
| .44 | | | | | .44 (.45-.46) |
| .48 | | | | | .48 (.49-.50) |
| .52 | | | | | .52 (.53-.54) |
| .56 | | | | | .56 (.57-.58) |
| .60 | | | | | .60 (.61-.62) |
| .64 | | | | | .64 (.65-.66) |
| .68 | | | | | .68 (.69-.70) |
| .72 | | | | | .72 (.73-.74) |
| .76 | | | | | .76 (.77-.78) |
| .80 | | | | | .80 (.81-.82) |
| .84 | | | | | .84 (.85-.86) |
| .88 | | | | | .88 (.89-.90) |
| .92 | | | | | .92 (.93-.94) |
| .96 | | | | | .96 (.97-.98) |
| .100 | | | | | .100 (.101-.102) |
| .104 | | | | | .104 (.105-.106) |
| .108 | | | | | .108 (.109-.110) |
| .112 | | | | | .112 (.113-.114) |
| .116 | | | | | .116 (.117-.118) |
| .120 | | | | | .120 (.121-.122) |
| .124 | | | | | .124 (.125-.126) |

Legal Office = 26, 192.168.15.64/27

3. Per le due reti da 10 host uso i blocchi .96 e .112 /28.

| | /25 (1 subnet bit) 2 subnets 126 hosts | /26 (2 subnet bits) 4 subnets 62 hosts | /27 (3 subnet bits) 8 subnets 30 hosts | /28 (4 subnet bits) 16 subnets 14 hosts | /29 (5 subnet bits) 32 subnets 8 hosts | /30 (6 subnet bits) 64 subnets 2 hosts |
|------|--|--|--|---|--|--|
| .0 | | | | | .0 (.1-.6) | .0 (.1-.2) |
| .4 | | | | | .4 (.5-.6) | .4 (.5-.6) |
| .8 | | | | | .8 (.9-.14) | .8 (.9-.10) |
| .12 | | | | | | .12 (.13-.14) |
| .16 | | | | | .16 (.17-.22) | .16 (.17-.18) |
| .20 | | | | | .20 (.21-.22) | .20 (.21-.22) |
| .24 | | | | | .24 (.25-.30) | .24 (.25-.26) |
| .28 | | | | | .28 (.29-.30) | .28 (.29-.30) |
| .32 | | | | | .32 (.33-.38) | .32 (.33-.34) |
| .36 | | | | | .36 (.37-.38) | .36 (.37-.38) |
| .40 | | | | | .40 (.41-.46) | .40 (.41-.42) |
| .44 | | | | | .44 (.45-.46) | .44 (.45-.46) |
| .48 | | | | | .48 (.49-.54) | .48 (.49-.50) |
| .52 | | | | | .52 (.53-.54) | .52 (.53-.54) |
| .56 | | | | | .56 (.57-.58) | .56 (.57-.58) |
| .60 | | | | | .60 (.61-.62) | .60 (.61-.62) |
| .64 | | | | | .64 (.65-.70) | .64 (.65-.66) |
| .68 | | | | | .68 (.69-.70) | .68 (.69-.70) |
| .72 | | | | | .72 (.73-.78) | .72 (.73-.74) |
| .76 | | | | | .76 (.77-.78) | .76 (.77-.78) |
| .80 | | | | | .80 (.81-.86) | .80 (.81-.82) |
| .84 | | | | | .84 (.85-.86) | .84 (.85-.86) |
| .88 | | | | | .88 (.89-.94) | .88 (.89-.90) |
| .92 | | | | | .92 (.93-.94) | .92 (.93-.94) |
| .96 | | | | | .96 (.97-.102) | .96 (.97-.99) |
| .100 | | | | | .100 (.101-.102) | .100 (.101-.102) |
| .104 | | | | | .104 (.105-.110) | .104 (.105-.106) |
| .108 | | | | | .108 (.109-.110) | .108 (.109-.110) |
| .112 | | | | | .112 (.113-.114) | .112 (.113-.114) |
| .116 | | | | | .116 (.117-.118) | .116 (.117-.118) |
| .120 | | | | | .120 (.121-.122) | .120 (.121-.122) |
| .124 | | | | | .124 (.125-.126) | .124 (.125-.126) |

Sales Office = 10, 192.168.15.96/28

HR Office = 10, 192.168.15.112/27

4. Per i 3 link WAN 1, WAN 2 e WAN 3 uso i 3 blocchi .128, .132, .136 /30 da 2 indirizzi ciascuno (4 - 2 riservati); usare questi 3 blocchi mi impedisce di poter usare il blocco .128 /25, il blocco .128 /26, il blocco .128 /27, il blocco .128 /29 e .136 /29

| | | | | | | |
|------|--|--|----------------|--|------------------|------------------|
| .112 | | | .96 (.97-.120) | | .112 (.113-.118) | .112 (.113-.114) |
| .116 | | | | | .116 (.117-.118) | .116 (.117-.118) |
| .120 | | | | | .120 (.121-.122) | .120 (.121-.122) |
| .124 | | | | | .124 (.125-.126) | .124 (.125-.126) |
| .128 | | | | | .128 (.129-.130) | .128 (.129-.130) |
| .132 | | | | | .132 (.133-.134) | .132 (.133-.134) |
| .136 | | | | | .136 (.137-.138) | .136 (.137-.138) |
| .140 | | | | | .140 (.141-.142) | .140 (.141-.142) |
| .144 | | | | | .144 (.145-.150) | .144 (.145-.146) |
| .148 | | | | | .148 (.149-.150) | .148 (.149-.150) |
| .152 | | | | | .152 (.153-.158) | .152 (.153-.154) |
| .156 | | | | | .156 (.157-.158) | .156 (.157-.158) |
| .160 | | | | | .160 (.161-.162) | .160 (.161-.162) |
| .164 | | | | | .164 (.165-.166) | .164 (.165-.166) |
| .168 | | | | | .168 (.169-.170) | .168 (.169-.170) |
| .172 | | | | | .172 (.173-.174) | .172 (.173-.174) |
| .176 | | | | | .176 (.177-.178) | .176 (.177-.178) |
| .180 | | | | | .180 (.181-.182) | .180 (.181-.182) |
| .184 | | | | | .184 (.185-.186) | .184 (.185-.186) |
| .188 | | | | | .188 (.189-.190) | .188 (.189-.190) |
| .192 | | | | | .192 (.193-.194) | .192 (.193-.194) |
| .196 | | | | | .196 (.197-.198) | .196 (.197-.198) |
| .200 | | | | | .200 (.201-.202) | .200 (.201-.202) |
| .204 | | | | | .204 (.205-.206) | .204 (.205-.206) |
| .208 | | | | | .208 (.209-.210) | .208 (.209-.210) |
| .212 | | | | | .212 (.213-.214) | .212 (.213-.214) |
| .216 | | | | | .216 (.217-.218) | .216 (.217-.218) |
| .220 | | | | | .220 (.221-.222) | .220 (.221-.222) |

WAN 1 = 2, 192.168.15.128/30

WAN 2 = 2, 192.168.15.132/30

WAN 3 = 2, 192.168.15.136/30

| Req. addresses | Network prefix | Address range | Broadcast address |
|-------------------|-------------------|---------------|-------------------|
| Corporate HQ (58) | 192.168.15.0/26 | .1-.62 | .63 |
| Legal Office (26) | 192.168.15.64/27 | .65-.94 | .95 |
| Sales Office (10) | 192.168.15.96/28 | .97-.110 | .111 |
| HR Office (10) | 192.168.15.112/28 | .113-.126 | .127 |
| WAN 1 (2) | 192.168.15.128/30 | .129-.130 | .131 |
| WAN 2 (2) | 192.168.15.132/30 | .133-.134 | .135 |
| WAN 3 (2) | 192.168.15.136/30 | .167-.138 | .139 |

Compilazione della tabella

- a) Elencare le reti in ordine decrescente
- b) Scrivere (fuori) il blocco di partenza
- c) Scrivere il blocco .0 e poi calcolare il prefisso usando la potenza di 2 più piccola (y) che comprende il numero di host della sottorete: **$x = 32 - y$**
- d) Scrivere direttamente il primo e l'ultimo indirizzo disponibile per quella riga: **$.1 - .2^y - 2$** .
- e) Scrivere l'indirizzo di broadcast: **$.(2^y - 2) + 1$**
- f) Scrivere la seconda riga **partendo dall'indirizzo successivo a quello di broadcast nella riga precedente**, calcolando opportunamente la lunghezza del prefisso.
- g) Partendo dall'indirizzo della sottorete calcolare il range di indirizzi **$(.ind\ di\ rete + 1) - (.ind\ di\ rete + 2^y - 2)$** , quindi l'indirizzo di broadcast, e così via

(Questo procedimento è valido finché si parte da un blocco iniziale più piccolo di /24, altrimenti anche l'ultimo gruppo di cifre viene coinvolto nel numero di indirizzo).

Route Summarization

Operazione inversa al sub-netting in cui si **raggruppano delle reti** e ci si riferisce ad esse con **un singolo indirizzo che ha il prefisso comune** a tutte; è parte delle funzionalità di un protocollo di routing e viene quindi fatta in automatico.

Si fa in maniera inversa al sub-netting spostando a sinistra il confine per **ridurre la lunghezza del prefisso**; quella che si è ottiene è la **summary route**.

Soltamente il problema è quello di trovare la summary route con **prefisso più lungo** possibile che sia una summary route per le reti su cui dobbiamo fare l'operazione di summarization

Esempio:

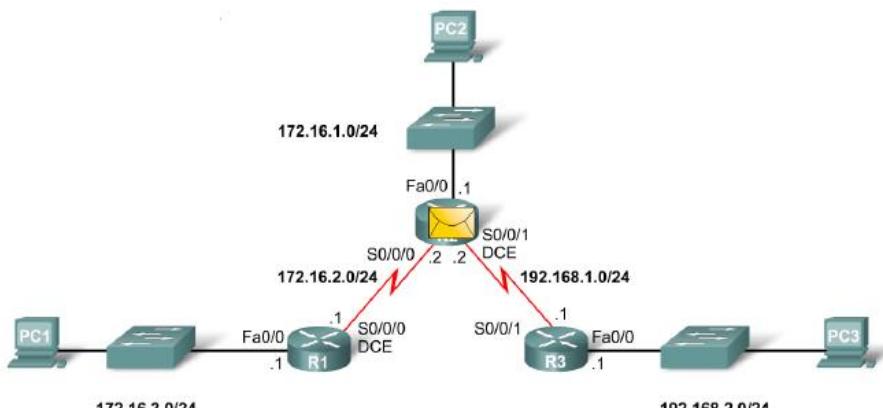
Si hanno tre reti con indirizzi 172.16.1.0/24, 172.16.2.0/24 e 172.16.3.0/24, si vuole trovare un indirizzo di rete tale per cui, se un certo indirizzo IP fa match con uno dei tre indirizzi di rete allora farà sicuramente match con l'indirizzo di summary route (non vale il viceversa).

| | |
|-----------------------|--|
| 172.16.1.0/24 | 10101100.00010000.00000001.00000000 |
| 172.16.2.0/24 | 10101100.00010000.00000010.00000000 |
| 172.16.3.0/24 | 10101100.00010000.00000011.00000000 |
| Summary Route: | |
| 172.16.0.0/22 | 10101100.00010000.00000000.00000000 |

I primi 22 bit del prefisso delle reti sono uguali per tutte e tre; quindi, si considera una **summary route con prefisso a 22 bit**, spostando il confine di due bit a sinistra: togliendo quindi i 2 bit che fanno differenza tra le reti. Ovviamente un indirizzo che fa match con una delle tre reti ha i primi 22 bit uguali a quelli dell'indirizzo summary. **Il viceversa non vale**, se prendo ad esempio 172.16.0.0/24 fa match con la summary route, ma non fa parte del gruppo di tre reti per cui ho fatto la summarization; quando costruisco la summary route potenzialmente aggiungo indirizzi che non fanno parte delle reti che sto raggruppando.

Esempio:

Si vuole configurare una route statica per dire a R3 come raggiungere le altre reti:
172.16.2.0/24,
172.16.1.0/24 e
172.16.3.0/24, ma qualsiasi sia la destinazione, per il router R3 il next hop sarà sempre R2.



Si può eseguire 3 volte il comando `ip route` e indicare le tre reti diverse con next hop sempre R2 oppure si può eseguire il comando una volta sola utilizzando una summary route, cioè **172.16.0.0/22**.

```
R3(config) #ip route 172.16.0.0 255.255.252.0 192.168.1.2
```

Vantaggio – Scalabilità: con una unica entry si fornisce al router il percorso per raggiungere 3 reti diverse e quindi si riduce la dimensione della tabella di routing diminuendo anche il tempo medio di lookup nella tabella. L’implementazione dei protocolli di routing prevede l’esecuzione di route summary per rendere più efficiente sia il protocollo di routing che le tabelle di routing.

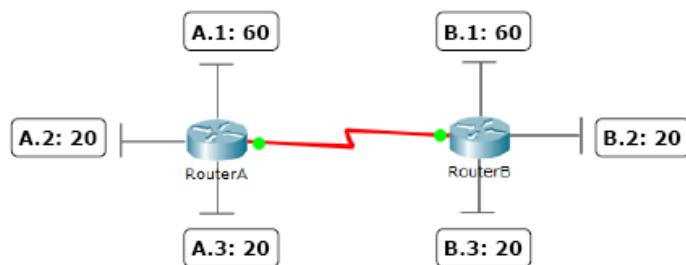
Svantaggio: se PC3 confezionasse un pacchetto diretto a 172.16.0.0 (che non esiste nella rete) questo verrebbe comunque inoltrato su R2, mentre con la prima configurazione (con tre entrate diverse nella tabella) verrebbe scartato perché R3 non troverebbe corrispondenza nella tabella di route.

Con la seconda configurazione, se R2 è configurato correttamente il pacchetto verrebbe scartato su R2, però R3 inoltrerebbe comunque i pacchetti che poi verrebbero scartati: **si sprecano risorse su R3 e sul link tra R3 e R2**, perché lo si impegna per trasmettere un pacchetto che non può raggiungere la destinazione.

Address Planning & Summarization

L’obiettivo **dell’address planning** potrebbe non essere necessariamente quello di usare il blocco più piccolo da ripartire: potremmo avere come obiettivo quello di **favorire la route summarization**.

Esempio:

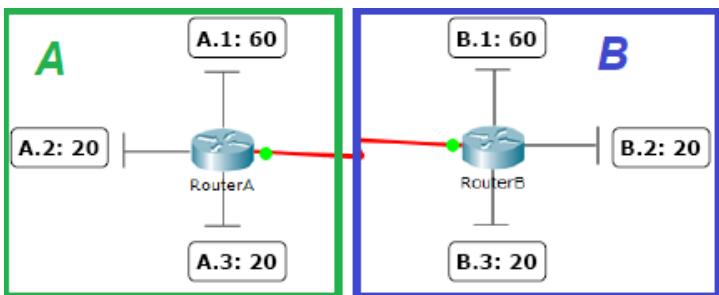


Se l’obiettivo fosse di usare il blocco più piccolo possibile il metodo è quello visto in precedenza da cui si ottiene la seguente tabella, partendo da un blocco di indirizzi /24:

| Req. Addr. | Network prefix | |
|------------|------------------|-------------------------------------|
| A.1 (60) | 192.168.0.0/26 | 11000000.10101000.00000000.00000000 |
| B.1 (60) | 192.168.0.64/26 | 11000000.10101000.00000000.01000000 |
| A.2 (20) | 192.168.0.128/27 | 11000000.10101000.00000000.10000000 |
| A.3 (20) | 192.168.0.160/27 | 11000000.10101000.00000000.10100000 |
| B.2 (20) | 192.168.0.192/27 | 11000000.10101000.00000000.11000000 |
| B.3 (20) | 192.168.0.224/27 | 11000000.10101000.00000000.11100000 |

Se adesso volessi fare la summarization (visto che A per raggiungere B.1, B.2 e B.3 deve sempre passare per B) con questa configurazione sarebbe impossibile: /24 non si può usare perché comprenderebbe anche le reti A.1, A.2 e A.3, mentre /25 esclude la rete B.1 e include A.2 e A.3.

L’alternativa è di **usare un metodo diverso per indirizzare le reti volto favorire la summarization** con una **pianificazione gerarchica**: invece di considerare la rete tutta insieme, dove si intravede la possibilità si va a partizionare la rete di partenza in sottoreti fatte da gruppi di link, in un modo che topologicamente crei le condizioni per fare una summarization, solo dopo si passa alle singole componenti di questa partizione.



Si considera il sottoinsieme A che comprende A.1, A.2 e A.3 e B che contiene B.1 B.2 B.3; per ciascuna si usa un blocco separato che sarà **naturalmente la summary route per quella sottorete**.

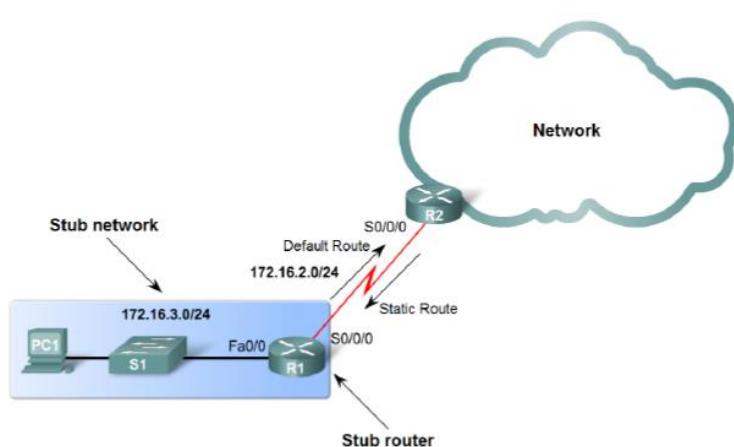
Quindi si assegnano gli indirizzi prima a tutte le reti di A, poi a quelle di B in maniera separata: in ciascuno poi adotto la tecnica per costruire la summary route più piccola. Si ottiene una numerazione diversa dalla precedente che complessivamente non impegna un blocco più grande: l'insieme A e l'insieme B usano un blocco /25, quindi si ha sempre come blocco di partenza il /24.

| | Req. Addr. | Network prefix | |
|------------------|---------------|------------------|-------------------------------------|
| 192.168.0.0/25 | A.1 (60) | 192.168.0.0/26 | 11000000.10101000.00000000.00000000 |
| | A.2 (20) | 192.168.0.64/26 | 11000000.10101000.00000000.01000000 |
| | A.3 (20) | 192.168.0.96/27 | 11000000.10101000.00000000.01100000 |
| 192.168.0.128/25 | B.1 (60) | 192.168.0.128/26 | 11000000.10101000.00000000.10000000 |
| | B.2 (20) | 192.168.0.192/27 | 11000000.10101000.00000000.11000000 |
| | B.3 (20) | 192.168.0.224/27 | 11000000.10101000.00000000.11100000 |

Default Static Route: route statica **0.0.0.0/0**, che ha *prefisso di lunghezza 0*.

Essendo il prefisso più corto possibile la route verrà scelta solo se non viene trovato un match nella tabella (si cerca il longest prefix, quindi qualsiasi altro match ha la priorità sulla route di default); il next hop di questa entry è usato come default.

Esempio:



Stub network: rete con un unico punto di ingresso/uscita verso il resto del mondo – in questo caso R1. R1 ha tutte le informazioni per raggiungere le destinazioni della stub network e riceverà anche tutti i pacchetti destinati a reti “oltre” R2 (esterno).

Su R1 è inutile avere singole entry per ciascuna destinazione potenzialmente raggiungibile: **qualsiasi sia la rete esterna il next hop è sempre R2** quindi è conveniente avere una default route che inoltra su R2.

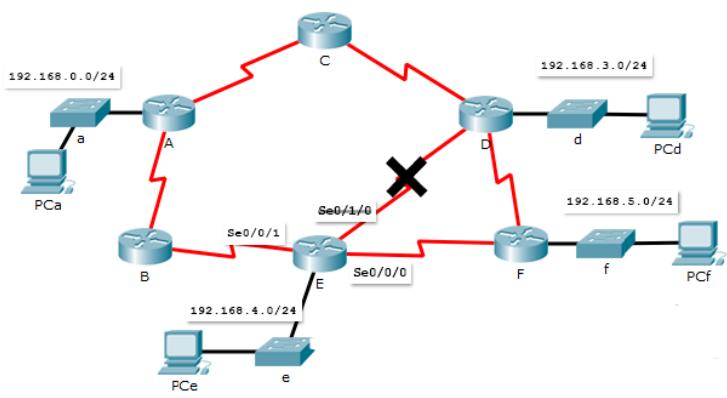
In questo caso conviene nonostante l'overhead che si ha mandando pacchetti “inutili” da R1, perché altrimenti andrebbero specificate tutte le possibili destinazioni esterne, aumentando notevolmente le dimensioni della tabella di routing di R1.

La configurazione è uguale a quella di una route statica qualsiasi:

```
Router(config)#ip route 0.0.0.0 0.0.0.0 [exit-interface | ip-address]
```

Routing Dinamico

Il limite intrinseco del routing statico è rappresentato dalla impossibilità di gestire ogni singolo guasto in maniera reliable: **le informazioni che i router hanno sono limitate alle reti a cui sono direttamente connessi.** Se ad esempio si guasta i link tra A e B, il router E ha nella sua tabella la riga per la destinazione 192.168.0.0/24 con next hop B, che dal suo punto di vista è ancora valido. B può ancora raggiungere A, ma deve passare per E, quindi gli rimanda indietro i pacchetti.



Il problema è che con il routing statico *E non può sapere che si è verificato un guasto sul link tra A e B*

L'RFC di un protocollo serve per una definizione standard; nei singoli apparati che lo

implementano, alcuni aspetti che non inficiano la interoperabilità potrebbero essere diversi tra di loro. Apparati di produttori diversi devono comunque essere in grado di capirsi e scambiarsi informazioni e costruiscono insieme la funzione realizzata da quel protocollo.

1. Routing Centralizzato

Soluzione semplice: presuppone che ci sia un'entità che raccoglie le informazioni sulla topologia della rete, decide qual è l'azione che deve compiere ciascun apparato e lo configura di conseguenza.

2. Routing Isolato

Ciascun router compila la propria tabella di routing senza scambiare informazioni né con un server centrale, né con altri router della rete, la decisione su come inoltrare i pacchetti per una certa destinazione viene presa da un router in maniera del tutto autonoma.

L'apparato riceve i pacchetti e usa l'indirizzo sorgente per decidere che attraverso quel link può raggiungere quella destinazione: usa le informazioni che riceve nel **piano dati** (pacchetti degli utenti) per costruire le tabelle di forwarding nel **piano di controllo**

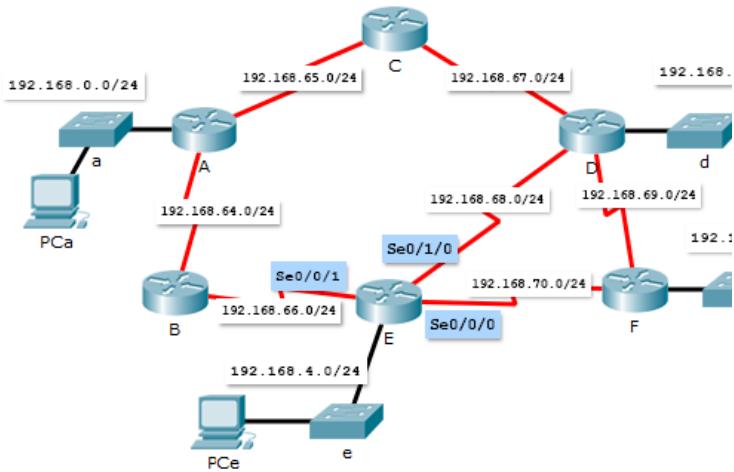
Ethernet basata su apparati di switching usa il routing isolato, ma ad esempio non si potrebbe utilizzare in una rete IP, perché implicherebbe avere un unico percorso per una certa destinazione, altrimenti c'è il rischio che i pacchetti girino all'infinito, ma i percorsi chiusi servono per l'affidabilità con IP.

3. Routing Distribuito

I router usano le connessioni con cui scambiano dati anche nel piano di controllo per **scambiarsi informazioni per costruire le tabelle** di routing: in questo caso il piano di controllo è separato dal piano dati, cioè le informazioni utili per costruire la tabella di routing sono ottenute solo da altri router, senza lo scambio di pacchetti utente.

Algoritmo Distance Vector

L'algoritmo indica qual è l' informazione i router si devono scambiare per compilare le tabelle.



La rete è in uno stato in cui tutti hanno costruito la tabella di routing e sanno quindi come raggiungere le varie destinazioni e con che costo.

Funzionamento (router B)

- B costruisce il **distance vector (DV)** partendo dalla sua tabella di routing e lo manda ad E
- Salta la riga relativa a 192.168.66.0 perché è la rete che usa per comunicare con E, quindi essendo a distanza 0 per entrambi è un'informazione inutile da inviare
- Per ogni destinazione B prende la distanza dalla tabella di routing, la incrementa di 1 per considerare sé stesso nel percorso e la manda a E
- I DV sono diversi in base al router a cui sono destinati
- E riceve quindi il DV da B con i costi già calcolati
- Dopo aver ricevuto tutte le informazioni dai vicini fa un **merge**, per ogni destinazione
 - Se comunicata da un solo vicino allora c'è un solo modo per raggiungere quella destinazione, quindi viene inserita direttamente nella tabella con il costo e il next hop (chi ha mandato l'informazione).
 - Se ci sono più informazioni per una destinazione inserisce nella tabella quella che costa meno e indica come next hop chi l'ha inviata

Il protocollo prevede che continuamente i router comunicano le informazioni ai vicini che a loro volta fanno la stessa cosa; partendo da una situazione in cui ogni router conosce solo le reti a cui è direttamente connesso l'algoritmo prevede che **dopo un certo numero di scambi** si arriva allo **stato stabile**, cioè una situazione in cui ogni router ha una tabella di routing completa e quando riceve un DV il contenuto non cambia.

Con il routing dinamico se la topologia della rete cambia, ad esempio si rompe il link tra E e D, il router E (che lo vede a livello hardware) non potendo più usare D come next hop considera il DV di D non più valido e quindi modifica la tabella di routing togliendo le righe ottenute dal DV di D, per poi propagare l'informazione ai vicini.

I DV ricevuti vengono conservati in un **database interno del router**, in questo modo se il percorso non è più disponibile (ma era nella tabella) posso riprendere l'informazione precedente, se presente.

Protocollo RIP

Dato l'algoritmo chi specifica il protocollo deve occuparsi di definire come operativamente viene implementato

- La metrica è l'**hop count** (numero di router che il pacchetto deve attraversare per raggiungere la sua destinazione finale)
- Una rete direttamente connessa ha un hop count = 0
- Il protocollo ha un limite di 15 hop: la **distanza 16 viene considerata infinita** e quindi la destinazione non raggiungibile
- Il protocollo ha due **versioni**:
 - RIP v1: meccanismo di indirizzamento classful
 - RIP v2: estende RIP v1 per supportare il VLSM
- I DV sono trasferiti attraverso uno o più messaggi (la lunghezza del DV non è prevedibile a priori) chiamati **routing updates** encapsulati in pacchetti UDP e trasmessi alla porta 520 ogni 30s.
- Ci sono casi in cui si vorrebbe poter ritardare la comunicazione, avendo questa un costo: l'aspetto ingegneristico determina il tempo, è necessario analizzare le prestazioni del protocollo e le capacità della rete

Nota: la scelta dell'hop count come metrica non è specificata nell'RFC

Formato pacchetto RIPv2

Trasmesso usando UDP encapsulato in un pacchetto IP multicast: con indirizzo IP riservato 224.0.0.9 messo come indirizzo di destinazione, è composto da un header più un certo numero di entries del DV.

- **Header (32b):**
 - **Req/Resp** (8b): serve ad esempio quando un router si riavvia a seguito di un guasto, ma gli altri sono già allo stato stabile, perché non sempre una rete viene completamente avviata da zero. Il disequilibrio potrebbe portare a un tempo di convergenza lungo, per cui quando il router fa partire il RIP manda una richiesta ai suoi vicini per far scattare l'invio immediato di un DV.
 - **Versione protocollo** (8b)
 - **Riservato**
- **DV:**
 - **Address family** (16b) | **Route Tag** (16b)
 - **Indirizzo IP** (32b)
 - **Subnet mask** (32b)
 - **Next hop** (32b): normalmente è 0 per indicare che il next hop è il mittente, in implementazioni più complesse può servire per indicare un next hop diverso
 - **Metric** (32b)

Il formato di **RIPv1** è simile, ma il campo subnet mask è ignorato (0) e l'indirizzo IP deve essere classful

Tabella di routing con RIP:

R 172.16.0.0/16 [120/3] via 172.17.1.1, 00:00:27, Serial0

R: informazione ottenuta tramite RIP

172.16.0.0: dest (*)

172.17.1.1: next hop (*)

3: distanza (numero di hop) (*)

120 = AD associata a RIP

Serial0: soluzione della ricorsione, ha già risolto 172.17.1.1 nella tabella di routing

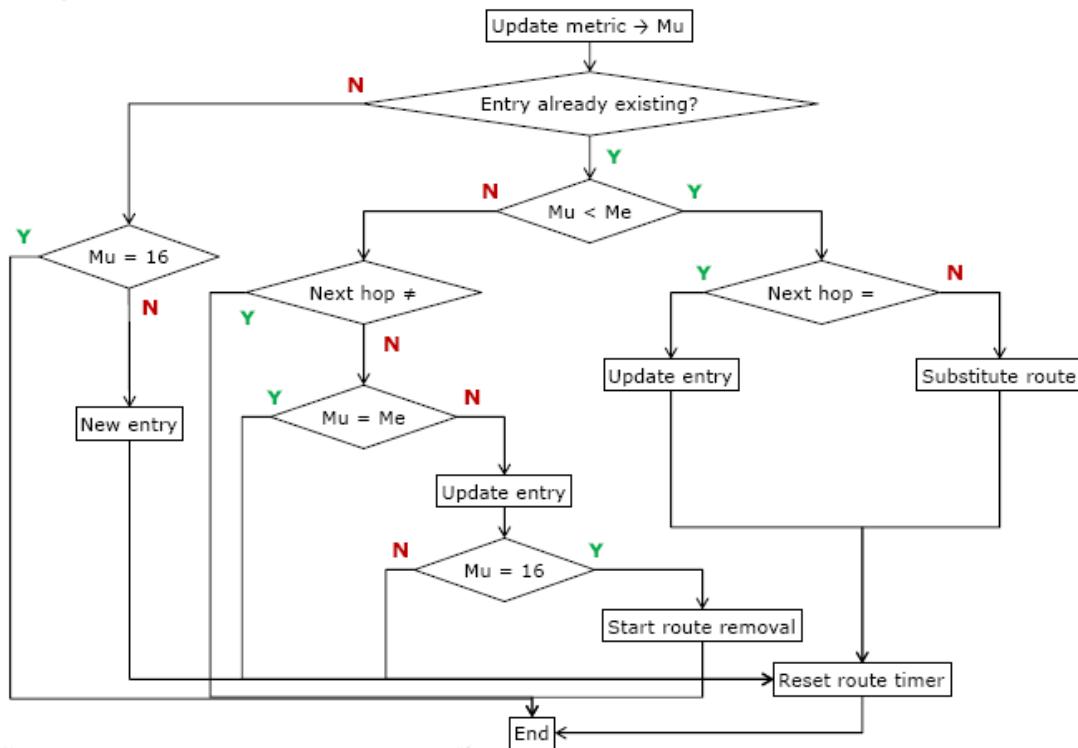
00:00:27: **route time** – tempo trascorso da quando è stata ricevuta l'informazione utilizzata per creare la entry nella tabella

(*) *Informazioni che leggo direttamente nel distance vector*

Nota: perché dovrei voler ricevere un RV con distanza 16, perché il vicino dovrebbe dirmi che una destinazione non è raggiungibile? Per comunicare esplicitamente che una destinazione non è più raggiungibile (ad esempio a seguito di un guasto). Il vicino si accorge che una destinazione con cui è direttamente connesso non è più raggiungibile e lo comunica impostando il costo a 16.

Dobbiamo **prevedere un modo per comunicare la non raggiungibilità di una destinazione**

Routing update calculation (merging)



Mu: metrica aggiornata

Me: metrica esistente

C'è già una entry per quella destinazione?

N. Mu = 16?

Y: informazione inutile

=> fine

N: il vicino ha comunicato che c'è una nuova destinazione da raggiungere con distanza Mu

=> si inserisce la nuova entry e si azzerà il timer

Y. Mu < Me?

Y: il vicino ha comunicato un percorso più corto per la destinazione.

Il next hop è lo stesso della entry già presente?

Y: il vicino ha scoperto un percorso più breve

=> si aggiorna l'entrata con il nuovo valore della metrica e si resetta il timer

N: si sceglie questa come entry

=> si sostituisce l'entrata con questa e si resetta il timer

N: stessa distanza o distanza maggiore

Il next hop è diverso da quello della entry già presente?

Y: va bene la entry già presente => fine

N: Mu = Me?

Y: stato stabile, il vicino ha fatto l'update => reset timer

N: Mu > Me, lo stesso vicino che aveva dato la distanza ora ne manda una maggiore

=> aggiornamento entrata

Mu = 16?

N: Il percorso si è solo allungato => fine

Y: Il vicino comunica che la destinazione non è più raggiungibile

=> si avvia la procedura di rimozione

Procedura di rimozione: la rotta non viene tolta direttamente, perché il fatto che quella destinazione non è più raggiungibile è un **transitorio**. Ad esempio, un link non è o acceso o spento, ma prima di arrivare a non funzionare attraversa degli stati, oppure se si guasta l'interfaccia sulla porta questa inizia a perdere un tot di pacchetti, ma non tutti.

Per gestire queste situazioni transitorie si prende un **approccio conservativo**: l'evento porterebbe a cambiare subito lo stato, ma **ritardiamo quel cambio di stato** perché potrebbe avvenire prima un altro evento che riporta allo stato di prima.

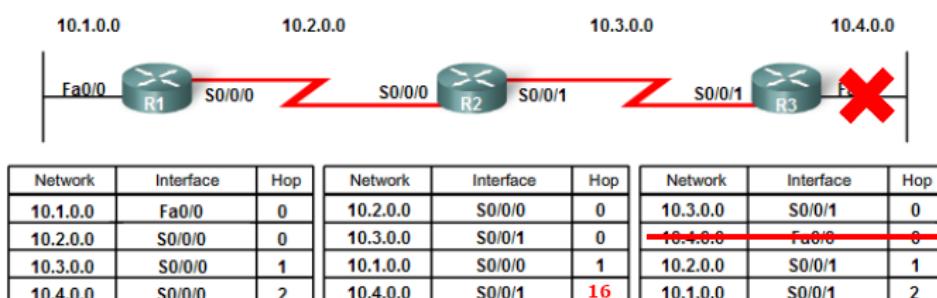
Si vogliono evitare situazioni di instabilità dove il contenuto della tabella di routing cambia continuamente quindi piuttosto si aspetta, se dopo questo tempo (**route timer**) non cambia nulla allora si mette il costo 16: non va tolta, perché si deve segnalare a tutti che quella rotta non è più raggiungibile (*se non si segnala eventualmente se ne accorgeranno ma si perde del tempo inutilmente*)

RIP timers:

- **Update timer (30s)**: intervallo tra l'invio dei routing updates
- **Route timer (180s)**: massimo tempo di validità di una entry, si tollera la perdita di 5 update (180/30).
- **Holddown timer (120s, IOS: 180s)**:
 - scatta quando una entry è segnata come non raggiungibile
 - scatta il route timer
 - il vicino manda un update con costo 16
 - durante questo tempo si propaga l'informazione che questa destinazione non è più raggiungibile, ma nella tabella di routing continua ad esserci questa entry – male che vada un pacchetto prima o poi viene dropato –. È preferibile pagare il costo di inoltrare un pacchetto “a vuoto”, per proteggersi nel caso in cui viene detto che la destinazione non è raggiungibile ma in realtà non è vero

Trigger update (opzionale): update scatenato da un evento. Se una rete a cui il router è direttamente connesso non è più raggiungibile non aspetta l'update timer e mando **subito** un nuovo update

Problema del counting to infinity



- 10.4.0.0 si rompe quindi R3 toglie l'entrata corrispondente dalla tabella
- Supponendo attivo il trigger update – manda l'informazione a R2 che la segna come non raggiungibile (costo 16), ma la continua a tenere nella tabella di routing.
- Prima che R2 mandi l'update a R1 per dire che 10.4.0.0. non è raggiungibile, R1 manda il suo update a R2 e dicendogli che 10.4.0.0 è a distanza $2 + 1 = 3$.

- R2 penserà di aver scoperto un nuovo modo per raggiungere 10.4.0.0, quindi il costo diventa 3 e il next hop diventa R1; R2 non ha modo di sapere che per raggiungere quella destinazione il pacchetto ripasserà per R2.
- Al prossimo update R2 dirà a R1 che 10.4.0.0 è a distanza 4, quindi R1 penserà che la distanza sia aumentata e aggiornerà la tabella
- E così via fino a che il costo non raggiunge 16 e la destinazione viene considerata non raggiungibile
→ il protocollo funziona **ma si perde molto tempo**

Soluzione: split horizon

Quando un router comunica al vicino il DV oltre alle entry relative alle reti a cui è connesso anche il router destinatario, rimuove anche le entry che prevedono quel vicino come next hop.

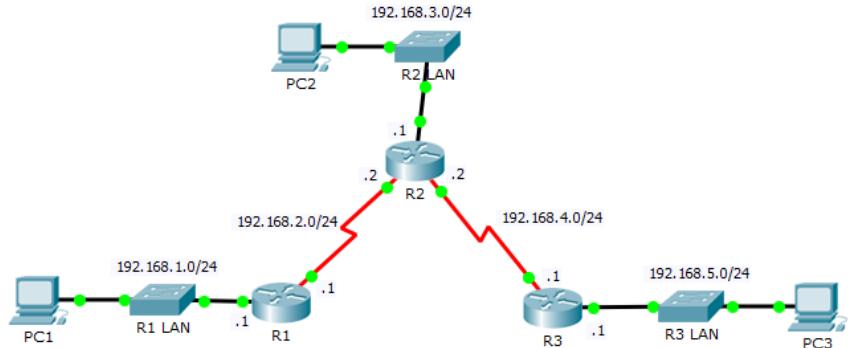
R1 per raggiungere 10.4.0.0 ha come next hop è R2, allora è inutile dire a R2 che 10.4.0.0 si raggiunge con distanza 3 perché è stato lui a comunicarlo.

Split horizon with poison reverse: un router non solo evita di far pensare al vicino che una certa destinazione sia raggiungibile quando questi è il next hop, ma gli comunica esplicitamente che tramite sé stesso quella destinazione non è raggiungibile.

Nota: sugli apparati che prenderemo in esame sono abilitati trigger updates e split horizon, queste sono comunque impostazioni controllabili.

Packet tracer

La versione di RIP predefinita usata da Cisco è RIPv1 (il pacchetto IP viene mandato in broadcast), quindi usa indirizzi classful.



Configurazione RIP

```
R(config)#router rip // modo di configurazione di RIP
R(config-router)#network [indirizzo-ip]
```

Il comando network ha due effetti:

- a. Il router guarda tutte le sue interfacce e controlla se l'indirizzo fa match con quello scritto nel comando, se sì su quella interfaccia comincia a scambiare pacchetti RIP, quindi, manda update e li riceve.
 - b. Negli update il router includerà la rete specificata; nella realtà potrei non volere annunciare questa rete, *ad esempio quando si ha un router che usa due dominii di routing separati, che se da un lato non usa RIP non vuole annunciare quella rete tramite RIP*
- Eseguo network per ciascuna interfaccia del router R1


```
R1(config-router)#network 192.168.1.0 //essendo classful non c'è la maschera
R1(config-router)#network 192.168.2.0
```
 - Eseguo network su R2


```
R2(config-router)#network 192.168.2.0
R2(config-router)#network 192.168.3.0 (*)
R2(config-router)#network 192.168.4.0
```
 - Eseguo network su R3


```
R3(config-router)#network 192.168.4.0
R3(config-router)#network 192.168.5.0
```

(*) *in RIPv1 se mettessi ad esempio 192.168.3.38 il comando verrebbe eseguito senza errori, ma nel file running-config ci sarebbe network 192.168.3.0 perché l'indirizzo è di classe C; il router fa questa conversione implicitamente*

- Nel modo di simulazione posso esaminare i routing updates:

| | | |
|----------------------|----------|---------------------|
| CMD: 0x2 | VER: 0x1 | 0000 0000 0000 0000 |
| ADDR FAMILY: 0x2 | | 0000 0000 0000 0000 |
| NETWORK: 192.168.2.0 | | |
| 0000 0000 0000 0000 | | |
| NEXT HOP: 0.0.0.0 | | |
| METRIC: 0x1 | | |

da R1 a R2 il messaggio contiene solo la entry 192.168.1.0, essendo destinato a R2 il router ha tolto l'informazione relativa a 192.168.2.0 perché connessa direttamente a R2

| | | |
|----------------------|----------|---------------------|
| CMD: 0x2 | VER: 0x1 | 0000 0000 0000 0000 |
| ADDR FAMILY: 0x2 | | 0000 0000 0000 0000 |
| NETWORK: 192.168.3.0 | | |
| 0000 0000 0000 0000 | | |
| NEXT HOP: 0.0.0.0 | | |
| METRIC: 0x1 | | |
| ADDR FAMILY: 0x2 | | 0000 0000 0000 0000 |
| NETWORK: 192.168.4.0 | | |
| 0000 0000 0000 0000 | | |
| NEXT HOP: 0.0.0.0 | | |
| METRIC: 0x1 | | |
| ADDR FAMILY: 0x2 | | 0000 0000 0000 0000 |
| NETWORK: 192.168.5.0 | | |
| 0000 0000 0000 0000 | | |
| NEXT HOP: 0.0.0.0 | | |
| METRIC: 0x2 | | |

da R2 a R1 non c'è l'informazione su 192.168.2.0 e nemmeno quella 192.168.1.0 per via dello split horizon, perché per arrivare a 192.168.1.0 passerebbe per R1.

Nota: essendo pacchetti broadcast il router li manda a tutti

Comando show ip protocol: mostra informazioni sul protocollo

Routing Protocol is "rip" – protocollo in uso

Sending updates every 30 seconds, next due in 11 seconds – intervallo tra un update e l'altro

Invalid after 180 seconds, hold down 180, flushed after 240 – valori timer

(...)

Redistributing: rip – i router mantengono separati i dominii di routing: le entrate nella tabella di routing vengono incluse nei routing update solo specificato

Default version control: send version 1, receive any version – può ricevere anche pacchetti v2
(...)

Automatic network summarization is in effect – il router fa anche la summary

Maximum path: 4 – numero di informazioni di pari costo mantenute in tabella

Routing for Networks: – reti che annuncia ai vicini

192.168.1.0

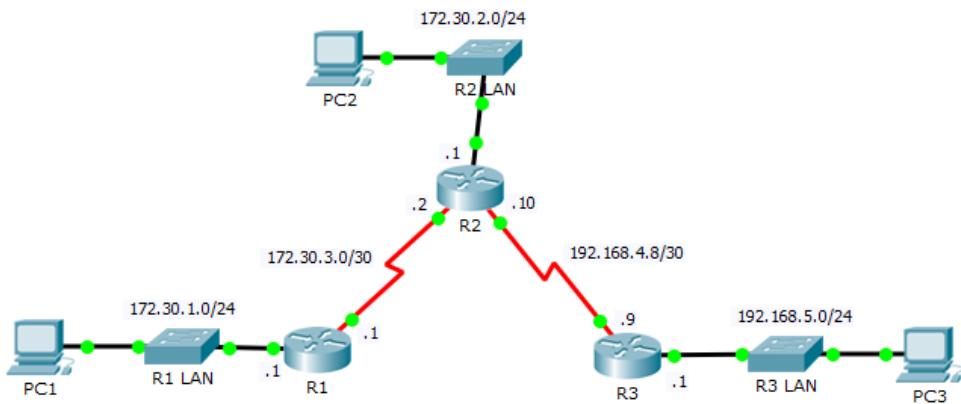
192.168.2.0

(...)

Comando debug ip rip: attiva le notifiche per eventi relativi al protocollo RIP (invio e ricezione di update)

Comando show ip rip database: mostra la struttura dati interna al router (nella RAM) che mantiene gli update ricevuti (per RIP non è molto significativo su packet tracer, per OSPF è più articolato).

RIPv2



Limiti del RIPv1: in questa rete abbiamo degli indirizzi non corretti per il formato classful

- 172.30.1.0/24, 172.30.3.0/30, 172.30.2.0/24 (devono essere /16, classe B)
- 192.168.4.8/30 (deve essere /24, classe C)

Se si utilizza RIPv1:

1. R1(config-router)#network 172.30.0.0

Qualsiasi siano le ultime due cifre dell'indirizzo questo viene normalizzato alla sua classe.

Basta un solo comando, perché l'interfaccia 172.30.1.1 e anche 172.30.3.1 fanno match; quindi, il protocollo è attivo sia sulla porta seriale che sulla LAN.

2. R1#show ip protocols

Non compare nessun vicino, perché R2 non sta ancora usando RIP e l'informazione che R1 invia è solo quella di 172.30.0.0

3. R2(config-router)#network 172.30.0.0

R2(config-router)#network 192.168.4.0

4. R3(config-router)#network 192.178.4.0

R3(config-router)#network 192.178.5.0

5. R3#show ip route

```
R    172.30.0.0/16 [120/1] via 192.168.4.10, 00:00:07, Serial0/0/1
      192.168.4.0/30 is subnetted, 1 subnets
C      192.168.4.8 is directly connected, Serial0/0/1
C      192.168.5.0/24 is directly connected, FastEthernet0/0
```

Ha una sola entrata R con 172.30.0.0/16 a distanza 1 via 192.168.4.0, tutte le reti non direttamente connesse fanno match con questa entrata e raggiunte tramite R2

→ **summary corretto**

6. R1#show ip route

```
172.30.0.0/16 is variably subnetted, 2 subnets, 2 masks
C      172.30.1.0/24 is directly connected, FastEthernet0/0
C      172.30.3.0/30 is directly connected, Serial0/0/0
R      192.168.4.0/24 [120/1] via 172.30.3.2, 00:00:06, Serial0/0/0
R      192.168.5.0/24 [120/2] via 172.30.3.2, 00:00:06, Serial0/0/0
```

R2 non gli manda 172.30.2.0 perché viene normalizzata a 172.30.0.0 e l'informazione verrebbe trasmessa su un link che ha un indirizzo con lo stesso prefisso (172.30.3.0)

→ **inconsistenza: non tutti i router sanno come mandare tutti i pacchetti**

Cambio di versione di RIP:

- R1(config-router)#version 2

Ora R1 elabora **solo pacchetti v2** e quindi ignora gli update di R2 (che è v1), per cui le due entry della tabella di routing dopo un po' (180s) scadono:

```
R      192.168.4.0/24 is possibly down, routing via 172.30.3.2, Serial0/0/0
R      192.168.5.0/24 is possibly down, routing via 172.30.3.2, Serial0/0/0
```

quindi anche se le vede down continua a inoltrare pacchetti per quelle destinazioni (siamo in una situazione in cui è comunque corretto), poi dopo 4 minuti le righe verranno eliminate

Nota: per tornare alla v1 il comando è no version 2: facendo version 1 si impone che debbano essere scambiati solo pacchetti v1

- R2(config-router)#version 2
- R3(config-router)#version 2

- Ora la tabella di routing di R1 comprende anche la rete 172.30.2.0: R2 non ragiona più con indirizzi classful e quindi annuncia anche 172.30.2.0/24, inoltre le righe segnalate come possibly down sono state rimandate da R2 ripristinandone la validità, però ha la destinazione è 192.168.4.0 e non 192.168.4.8
- Nella tabella di R2 ci sono le 5 destinazioni così come sono configurate
- Nella tabella R3 è ancora tutto “aggregato”:


```
R      172.30.0.0/16 [120/1] via 192.168.4.10, 00:00:05, Serial0/0/1
```
- R2# show ip protocols: R2 fa routing per le reti 172.30.0.0 e 192.168.4.0, il motivo per cui R1 vede 192.168.4.0 invece di 192.168.4.8 e per cui R3 vede 30.0.0 invece delle tre sottoreti è dovuto al fatto che **il protocollo RIP vede R2 come un boundary router**.

R2 è visto come un boundary router perché **separa due classi diverse di indirizzi** avendo una interfaccia di classe B (172.30.3.0) e una di classe C (198.168.4.8), quindi fa un **summary automatico** per risparmiare entry nei DV e nelle tabelle di routing (infatti **automatic network summarization is in effect**).

- Per disabilitare il summary automatico si usa: R2(config-router)#no auto-summary

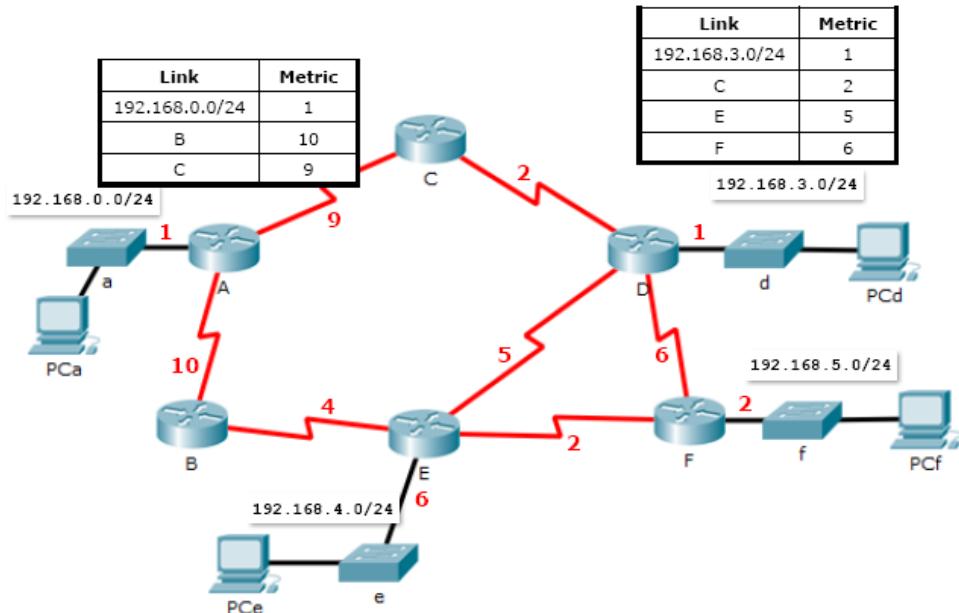
Ora nella tabella di R3: ora compaiono tutte e 4 le reti 172.30, c'è ancora la entry di 172.30.0.0/16 che anche se ha un prefisso più corto e quindi non verrà scelta rispetto alle altre entrate, può essere un problema se arrivano pacchetti destinati a 172.30.5.5, non presente nella rete.

In ogni caso la entry 172.30.0.0/16 non viene aggiornata quindi prima o poi verrà tolta dalla tabella.

Algoritmo di routing link state

- Ciascun router scambia con tutti gli altri router informazioni sulla topologia della ‘parte’ di rete che conosce
- Ciascun router collabora in modo che questa informazione raggiunga **tutti gli altri router della rete** attraverso il **flooding**
- Concettualmente alla **fine tutti condividono lo stesso insieme di informazioni** e attraverso questo ciascun router **indipendentemente** è in grado di ricostruire l’intera topologia della rete e sulla base di questa può calcolare indipendentemente qual è il percorso migliore per ogni destinazione
- A differenza del DV non solo conosce il next hop, ma anche **tutti gli altri hop**: conosce, quindi, **tutto il percorso del pacchetto** (a differenza del DV, dove questa mancanza portava al problema del counting to infinity)
- Essendo molto probabile che ci siano percorsi chiusi se un router inoltra tutto ciò che riceve senza fare una selezione ci saranno pacchetti che girano all’infinito: ci deve essere quindi un **flooding selettivo**
- **Il costo è associato alla trasmissione sul link**
- Prevede routing gerarchico: opera su un dominio diviso in aree, l’algoritmo viene usato solo all’interno di ciascuna area mentre tra le aree diverse si usa distance vector – per favorire la scalabilità del protocollo in reti molto grandi

Esempio:



Router A:

- sa di essere connesso al link 192.168.0.0/24 a distanza 1
- sa che è un **vicino** di B a distanza **10**
- sa che è un vicino di C a distanza 9

Analogamente la tabella riporta le informazioni note a Router D riguardo ai suoi collegamenti e alla distanza di questi.

A un certo punto ogni altro router della rete conoscerà sia le informazioni note ad A, sia quelle note a D.

Selective flooding: l'informazione viene propagata solo la prima volta che la si riceve.

Esempio:

- A manda l'informazione a C e B
- B la manda a E
- C la manda a D
- E la manda a D

A D arriva due volte la stessa informazione (quella mandata dal router A): la prima volta deve propagarla, ma poi non deve più farlo, altrimenti il pacchetto girerebbe all'infinito nella rete.

Un router deve poter capire se questa informazione è nuova o meno:

- se sì la propaga
- se no deve capire se è un aggiornamento dell'informazione o solo una copia e in quest'ultimo caso deve ignorarla.

LSP database

Quando ogni router avrà collezionato l'informazione generata da tutti gli altri router, avrà una tabella contenente le informazioni (come avranno tutti gli altri): **tutti i router interni a un'area dovranno avere lo stesso identico database LSP**, con un'entrata per ciascun nodo contenente l'informazione che tale nodo ha mandato.

Il database 'diventa' la topologia dell'intera rete, quindi ogni router conosce la topologia della rete e può calcolare il percorso minimo tra sé e ogni destinazione (albero dei cammini minimi) e quindi costruirsi la propria tabella di routing.

| Sender | Node/Metric |
|--------|--------------------------------|
| A | B/10 C/9 {192.168.0.0/24}/1 |
| B | A/10 E/4 |
| C | A/9 D/2 |
| D | C/2 E/5 F/6 {192.168.3.0/24}/1 |
| E | B/4 D/5 F/2 {192.168.4.0/24}/6 |
| F | D/6 E/2 {192.168.5.0/24}/2 |

Per ogni destinazione guarda qual è il **percorso di costo minimo**; quindi, inserisce il next hop nella tabella e inserisce come **costo** la **somma dei costi dei vari link** che portano alla destinazione.

→ Ogni router calcola l'albero dei cammini minimi con sé stesso come radice.

Per l'algoritmo di Dijkstra vale la **proprietà di consistenza**: ogni nodo calcola il proprio albero dei cammini minimi e gli **alberi tra loro sono consistenti**, quindi non c'è il rischio che si creino percorsi chiusi

Link State vs Distance Vector

- Link State ha maggiore stabilità e migliore complessità ($O(n \log n)$).
- L'algoritmo di Dijkstra deve essere eseguito sulla cpu dei router ed è un algoritmo cpu intensive
- **LSP reagisce molto rapidamente ai cambi di topologia**: i router interessati dal cambio modificano l'informazione locale che generano, quello che serve è che questa informazione (e solo questa) venga di nuovo condivisa con tutti in modo che tutti aggiornino la topologia della rete e ricalcolino i percorsi
- Con DV la computazione è distribuita, in OSPF i **dati sono distribuiti e la computazione è individuale**
- Le informazioni non devono essere scambiate frequentemente: i LSP possono essere refreshati con periodi nell'ordine delle decine di minuti

⇒ OSPF (link state) è decisamente migliore di RIP (distance vector) e per questo viene adottato nelle reti di piccola-media grandezza

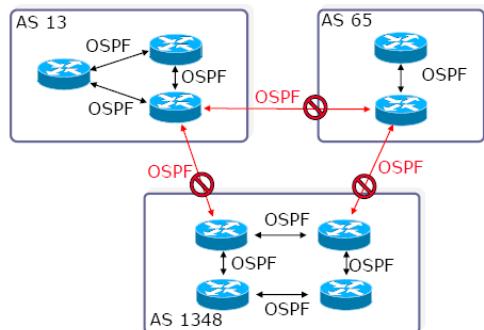
Protocollo OSPF (Open Shortest Path First)

- OSPF v2 (indirizzi classless)
- OSPF v3 (IPv6)

Un dominio di router OSPF - router connessi direttamente che condividono la stessa istanza di OSPF - è detto **Autonomous System** e la gestione della rete è autonoma rispetto all'esterno.

All'interno della rete ogni router deve sapere come raggiungere gli altri (tra AS diversi si usa il protocollo BGP)

- In Internet, ogni AS è identificato da un numero assegnato dall'IANA



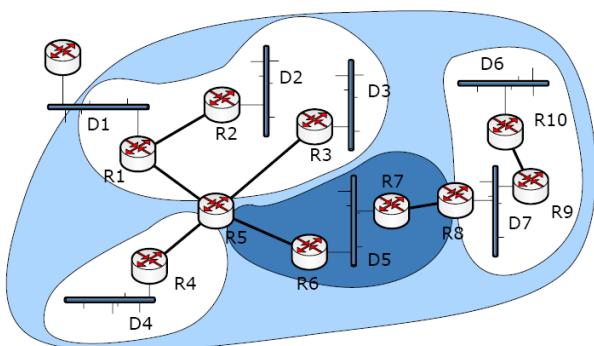
Dentro a un AS ogni router è univocamente identificato da un **router id** a 32bit

Un AS può essere **partizionato in più aree** per motivi di scalabilità: si riducono le operazioni che venivano fatte su tutto l'AS su una area più piccola.

Un' **area** è un sottoinsieme di link e interfacce di router direttamente connesso: da un punto di un'area deve essere possibile raggiungere ogni altro punto della stessa area.

- Ogni area ha un **identificatore a 32 bit**
- a. **Router interno:** tutte le sue interfacce sono nella stessa area
- b. **Area border router:** ha interfacce in almeno due aree
 - Tutti gli area border router devono avere almeno una interfaccia nella stessa area (backbone area – id predefinito 0.0.0.0)
- c. **AS boundary router:** ha almeno una interfaccia collegata a un altro AS

Esempio



AS partizionato in 4 aree connesse (da ogni punto dell'area si può raggiungere qualsiasi altro punto dell'area)

- R2, R3, R4, R6, R7, R9, R10: router interni
- R5, R8: area border router
- R1: AS boundary router

Tutti gli ABR devono avere un'interfaccia nella stessa area detta area di backbone:

Supponiamo di avere un pacchetto proveniente da D4 con destinazione D6, c'è un singolo percorso in termini di aree per andare da D4 a D6.

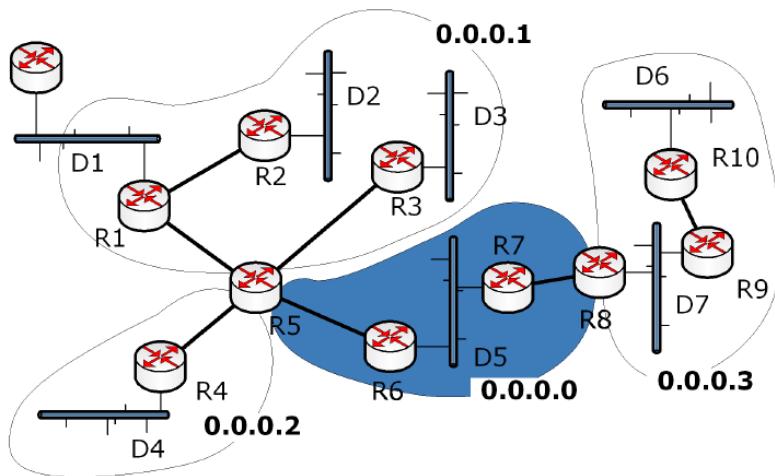
Supponiamo di aggiungere un router generico ABR che non rispetta il vincolo, collegato solo a D3 e D6: si hanno due percorsi per andare da D4 a D6.

Avere **più percorsi a livello di aree** significa che nella partizione esiste un percorso chiuso che attraversa più aree, l'esistenza di percorsi chiusi è un **punto critico per il protocollo di routing**: abbiamo solo spostato il problema a livello di aree, dove ci ritroveremmo comunque il problema del **count-to-infinity**.

Imponendo questo vincolo **non possono esserci percorsi a livello di aree** e lo scambio di informazioni tra aree diverse risulta più semplice e permette di usare l'algoritmo distance vector (link state all'interno delle singole aree)

Inserire quindi un router tra due aree, ma non connesso alla backbone non è un errore; quel router non parteciperà allo scambio tra aree diverse semplicemente farà routing di pacchetti da un'area all'altra, ma non è un ABR.

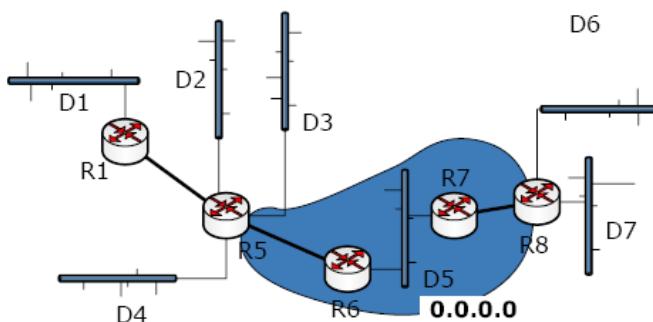
Vista della rete da un router generico



Ogni router ha una vista della topologia delle reti semplificata, ma **equivalente dal punto di vista dei percorsi migliori** e quindi ha la stessa tabella di routing che avrebbe se potesse vedere tutta la rete.

Un router conosce esattamente la topologia propria area, mentre le reti di altre aree vengono viste come se fossero direttamente connesse agli ABR della prima.

Esempio: R6 ha una conoscenza esatta della topologia della sua area

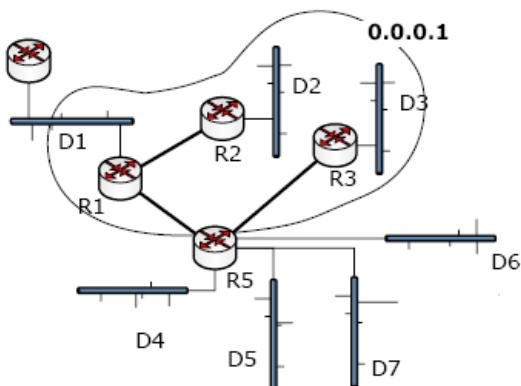


- Le reti dall'area 3 le vede come direttamente connesse a R8 (ABR connesso a 0 e 3)
- Le reti delle aree 1 e 2 le vede come direttamente connesse a R5 (ABR tra 0, 1, 2)
- R6 non sa cosa si trova tra la rete esterna alla sua area e il relativo ABR, conosce solo la distanza a cui questa si trova.

Le viste semplificate sono equivalenti dal punto di vista del routing; un router riceve un pacchetto con una certa destinazione che può essere all'interno o all'esterno dell'area.

- **Interno:** conosce la topologia dell'area e sa come raggiungere la destinazione (*Dijkstra*)
- **Esterno:** la destinazione è raggiungibile tramite un ABR che gli ha comunicato che sa come raggiungere la destinazione. Per il router raggiungere l'ABR equivale ad aver raggiunto la destinazione; quindi, **il percorso migliore per raggiungere la destinazione è proprio il percorso migliore per raggiungere l'ABR**, che il router conosce perché conosce la propria area.

Esempio: Al router R1 arriva un pacchetto da D1 per D6.



- R1 vede D6 come connessa direttamente a R5 quindi calcola il percorso migliore per raggiungere questo router.
- Quando il pacchetto arriva a R5, questo è un router dell'area 1, ma quando deve decidere cosa farne R5 conosce la topologia dell'area 0 e 2
- R5 vede D6 direttamente connesso a R8
- R8 avendo una interfaccia nell'area 3 inoltra il pacchetto a D6 col percorso di costo minimo.

Per eseguire OSPF il lavoro di un router interno è diverso da quello di un ABR, il primo deve solo sapere come è fatta la sua area e ha quindi meno informazioni rispetto al protocollo RIP, il secondo deve anche passare informazioni tra le varie aree.

Problema: se avessi un router non connesso all'area di backbone tra D3 e D6, questi sarebbe parte del percorso più breve $D1 - R1 - R5 - R3 - ABR^* - D6$, ma dato che non partecipa allo scambio tra aree, R1 non sa che può usarlo nel calcolo del percorso, per cui usa il percorso $R1 - R5 - R7 - R8 - R9 - R10 - D6$ che non è ottimo.

Limiti della ripartizione in aree

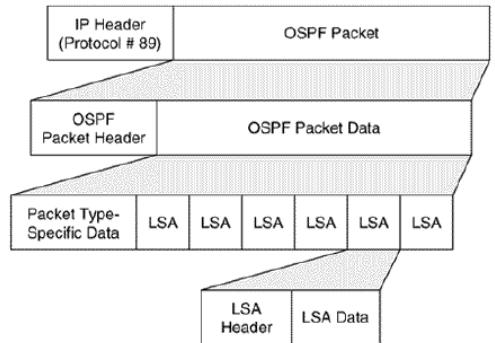
1. Diametro massimo in un'area: 6 hop
2. Numero di router per area: 30 ~ 100
3. Un ABR non può essere connesso a più di due aree (esclusa la backbone): più sono le aree e più complesse sono le operazioni che deve svolgere

OSPF Link State Packets

Il protocollo definisce il formato delle informazioni che i router si scambiano e le procedure che determinano quando scambiarsene e come interpretare le informazioni.

OSPF usa 5 tipi di pacchetti diversi, tutti sono trasferiti come payload di pacchetti IP (non si hanno garanzie sulla ricezione).

Ogni pacchetto contiene un header uguale per tutti e una parte dati che dipende dal tipo



Header (6 parole da 4B)

- [1B] Versione OSPF
- **[1B] Tipo pacchetto**
- [2B] Lunghezza: può cambiare a seconda del tipo
- **[4B] Router ID:** id del router che invia il pacchetto
- **[4B] Area ID:** id dell'area a cui appartiene il pacchetto, se il mittente è un ABR metterà l'id dell'area in cui vuole mandare il pacchetto
- [2B] Checksum: per la error detection
- [2B + 8B] Authdata e AuthType: per le funzionalità di autenticazione del router mittente

| Version | Type | Length |
|-----------|----------|--------|
| Router ID | | |
| Area ID | | |
| Checksum | AuthType | |
| AuthData | | |
| AuthData | | |

Adiacenza

I pacchetti OSPF sono scambiati tra router vicini che prima, però, devono **diventare adiacenti** (*come in TCP i due partecipanti devono aprire la connessione prima di potersi scambiare dati*); perché due router si scambino pacchetti non basta che siano connessi tra loro. A differenza di RIP che ‘accetta’ ogni pacchetto in arrivo, OSPF **processa solo i pacchetti provenienti da router che ha promosso come adiacenti**, quindi prima di elaborarli viene controllato il campo *router id*.

L'adiacenza è il risultato di una operazione preliminare (**dynamic neighbor discovery**) prevista da OSPF, se questa fallisce allora anche se i router sono connessi non si scambieranno informazioni e il link tra i due non esisterà nel *piano di controllo*.

La relazione di adiacenza è una relazione a due a due e quindi una volta stabilita, la **comunicazione OSPF è di tipo punto-punto**.

Quando due diventano adiacenti non aspettano che arrivino tutte le informazioni da tutti i router e in questa fase iniziale **sincronizzano tra di loro i propri database**. Tra i due router viene eletto un master che inizia a comunicare il contenuto del suo database OSPF, l'altro guarda cosa gli manca e lo richiede oppure se vede che all'altro manca qualcosa glielo manda.

I router adiacenti **immediatamente** e senza aspettare il flooding acquisiscono informazioni relative alla topologia della rete, questo può essere utile se un router si spegne e perde la tabella di routing, perché a fine bootstrap avrà eletto i router adiacenti e ricevuto le informazioni contenute nei loro database.

Tipi di pacchetti:

Tipo 1 - Hello Packet

Usato per la *dynamic neighbor discovery*

Tipo 2 – Database Description

Usato per la sincronizzazione dei database tra router adiacenti

Tipo 3 – Link State Request

Trasporta i **Link State Advertisement (LSA)**, cioè le informazioni generate da un router e che vuole siano propagate in tutta l'area

Tipo 4 – Link State Update

Contiene più LSA insieme, ciascuno dei quali poi viene processato indipendentemente (per il *selective flooding*)

Tipo 5 - Link state Acknowledgment

serve per rendere *reliable* lo scambio di LSA (essendo tutto sul protocollo IP non ci sono garanzie)

Protocol Operation:

1. Il router forma le adiacenze con i router vicini
2. Il router manda un pacchetto di tipo Hello su un indirizzo broadcast dedicato a OSPF (224.0.0.5)
3. Parte la fase di sincronizzazione dei database; alla fine di questa fase attraverso una serie di scambi di pacchetti controllata da un master avremo due copie identiche dei database.
4. Solo a questo punto il router si considera **completamente adiacente (l'adiacenza, quindi, implica che i due router abbiano lo stesso database)**

Il costo di un percorso si calcola associando un costo a ciascun link, più precisamente, il costo è **associato all'interfaccia**.

Un router comunica che c'è un costo x verso un altro router, perché c'è un costo x associato all'interfaccia che il primo router usa per trasmettere al secondo; essendo la comunicazione a due vie è possibile che il costo per andare da *RA* a *RB* possa essere diverso da quello per andare da *RB* a *RA*

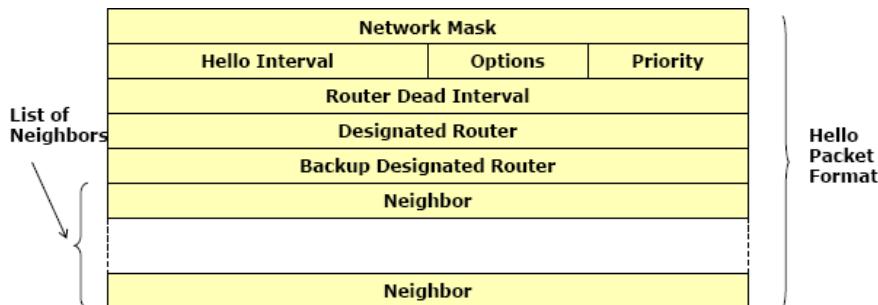
Hello Packet (tipo 1)

Pacchetto generato da un router e trasmesso su tutte le interfacce su cui il router è stato configurato per inviare pacchetti OSPF

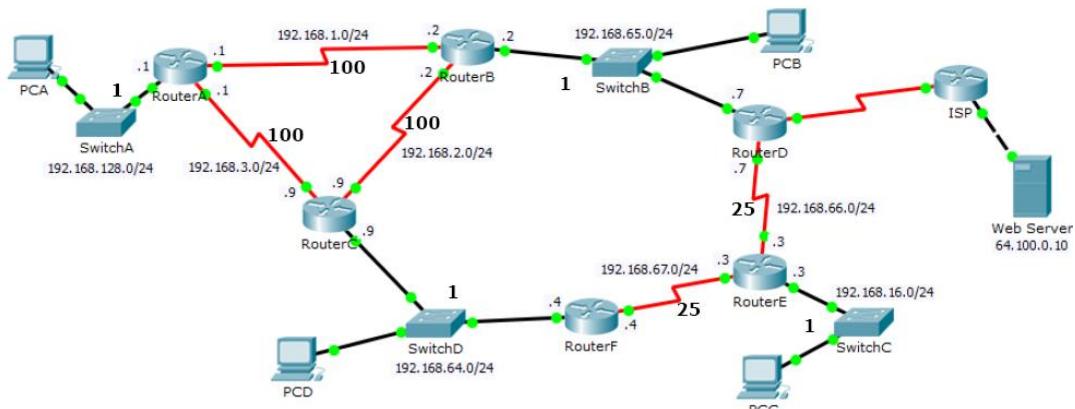
Un router **adiacente** è un router che può essere un **next hop**, quindi deve essere possibile l'inoltro dei pacchetti, per cui si controllano i campi area id, maschera e timer che devono essere uguali

1. **Maschera** utilizzata sull'interfaccia, se la maschera di rete è diversa i due router non possono diventare adiacenti
2. **Hello Interval:** timeout per l'invio periodico di pacchetti Hello in modo da mantenere viva la relazione di adiacenza (default: 10s)

3. **Router dead interval:** Dopo l'invio di un pacchetto Hello, se entro questo tempo non si riceve risposta si assume che ci sia stato un cambio nella topologia della rete e di conseguenza nei percorsi (default: 40s)
4. **List of neighbors:** quando un router A elenca un router B tra i suoi neighbors e B elenca A tra i propri, parte tra A e B la fase di sincronizzazione dei database al fine di diventare adiacenti



Esempio (area singola):



- Per semplicità non c'è distinzione sul costo nelle direzioni
- La trasmissione su FastEthernet ha costo 1
- D è un AS boundary router, perché ISP è un router esterno
- Gli id dei router sono n.n.n.n

Pacchetto inviato da Router C su serial0/0/0 (verso Router A)

| | | |
|-----------------------------------|------------|----------------|
| VERSION: 2 | TYPE: 1 | PKT LENGTH: 48 |
| ROUTER ID: 9.9.9.9 | | |
| AREA ID: 0.0.0.0 | | |
| CHECKSUM | AUTH TYPE | |
| AUTHENTICATION DATA | | |
| NETWORK MASK: 255.255.255.0 | | |
| HELLO INTERVAL: 10 | OPTIONS: 0 | RP: 0 |
| ROUTER DEAD INTERVAL: 40 | | |
| DESIGNATED ROUTER: 0.0.0.0 | | |
| BACKUP DESIGNATED ROUTER: 0.0.0.0 | | |
| NEIGHBOR: 1.1.1.1 | | |

- Tipo 1: pacchetto Hello
- Id router: 9.9.9.9 (router C)
- Area ID: 0.0.0.0 (singola area)
- Maschera: /24
- Hello Interval: 10 secondi
- Router Dead Interval: 40 secondi
- Elenco dei vicini: 1.1.1.1 (router A)

RP e options sono campi dedicati al meccanismo speciale di OSPF per gestire router connessi su un link broadcast

Pacchetto inviato da Router C su FastEthernet0/0 (verso Router F)

| | | | | |
|--|------------|----------------|--|--|
| VERSION: 2 | TYPE: 1 | PKT LENGTH: 48 | | |
| ROUTER ID: 9.9.9.9 | | | | |
| AREA ID: 0.0.0.0 | | | | |
| CHECKSUM | AUTH TYPE | | | |
| AUTHENTICATION DATA | | | | |
| NETWORK MASK: 255.255.255.0 | | | | |
| HELLO INTERVAL: 10 | OPTIONS: 0 | RP: 1 | | |
| ROUTER DEAD INTERVAL: 40 | | | | |
| DESIGNATED ROUTER: 192.168.64.9 | | | | |
| BACKUP DESIGNATED ROUTER: 192.168.64.4 | | | | |
| NEIGHBOR: 4.4.4.4 | | | | |

Gli id, la maschera e i timer sono gli stessi.

Come neighbor c'è 4.4.4.4 (router F), questo perché siamo sulla interfaccia ethernet. Se mettessimo un altro router sullo switch allora come neighbor potrei avere anche questo terzo router, e quest'ultimo e F sarebbero neighbors a loro volta

Soft state: se il sistema è distribuito e quindi non si ha la conoscenza totale del sistema, si ha uno stato *soft* nel senso che deve essere continuamente confermato altrimenti non sarà valido, questo accade anche nei protocolli di routing, in RIP con gli update ogni 30s e in OSPF con i pacchetti di Hello ogni 10s per spostare in avanti la scadenza del soft state.

Il soft state di OSPF è l'adiacenza: finché ho conferma che router adiacenti continuano a esserlo significa che i database sono sincronizzati, gli hello packet sono piccoli quindi si possono scambiare più frequentemente, in questo modo tutti conoscono tutto il database grazie alle relazioni di adiacenza a due a due.

Va rinfrescato anche link state update, ma si può farlo molto più lentamente: per default il tempo di scadenza per il contenuto del database OSPF è 1h.

Link State Update Packet (tipo 4)

Contiene uno o più **Link State Advertisements** (trasporta quindi le entrate del database OSPF), è formato da un campo per il numero degli LSA e un 'vettore' di LSA.

Per ognuno c'è un header e la sezione dati.

Link State Advertisement

Pacchetto con header comune e dati che invece dipendono dal **tipo di LSA**.

I tipi sono in totale più di 10, l'aggiunta di nuovi tipi è il meccanismo tramite cui si estende il protocollo a livello di funzionalità.

Link State Header

| Link State Age | Options | LS Type |
|----------------------------|---------|---------|
| Link State Identification | | |
| Advertising Router | | |
| Link State Sequence Number | | |
| Link State Checksum | Length | |

Campi di servizio: checksum, lunghezza parte dati, opzioni

- Link State Age:** rappresenta l'età dell'LSA, 0 è l'istante in cui l'advertising router lo ha generato quel sequence #, man mano cresce (si aggiorna quando si legge nel database) e quando arriva ad 1h e ancora non è stato ricevuto un nuovo LSA con un sequence # più aggiornato lo si butta via

2. LS Type:

| | |
|-------------------------|--|
| 1 Router LSA | Usato dai router per descrivere ciò che ha intorno |
| 2 Network LSA | Gestione dei router connessi a link broadcast |
| 3 ABR Summary LSA | Gestione della trasmissione tra le aree |
| 4 ASBR Summary LSA | Gestione rotte esterne |
| 5 AS External Route LSA | Gestione rotte esterne |

3. **Link State ID:** numero su 32 bit la cui interpretazione dipende dal tipo di LSA.

4. **Advertising router:** router che ha generato l'informazione.

5. **Link State Sequence Number:** indica se l'informazione ricevuta – se già presente nel database – è un refresh oppure una copia di quella già posseduta

Un LSA è definito univocamente da tipo, id, advertising router, che è la chiave del database

Link State Data

| Rtype | 0 | Number of Links |
|-----------|------|-----------------|
| Link ID | | |
| Link Data | | |
| Type | #TOS | TOS 0 Metric |
| TOS | 0 | Metric |
| | | |
| TOS | 0 | Metric |

1. **Rtype:** flag che danno informazioni sul router (es: router ABR, router ASBR,...)

2. **Number of Links**

3. Per ogni link:

- *Tipo* (stub network: 3, point-to-point: 1)
- Link id
- Link data (informazioni relative al link)
- Costo

Link id e Link data dipendono dal tipo:

- **Tipo 3:** id = indirizzo
data = prefisso
- **Tipo 1:** id = router id
dati = indirizzo interfaccia con cui
comunica con il router

È importante avere un meccanismo esplicito per comunicare un cambio di topologia; se un nodo si accorge che qualcosa è scaduto lo deve notificare agli altri evitando di aspettare la scadenza per risparmiare tempo evitando che gli altri nodi usino informazioni obsolete che porterebbero comunque alla perdita di pacchetti.

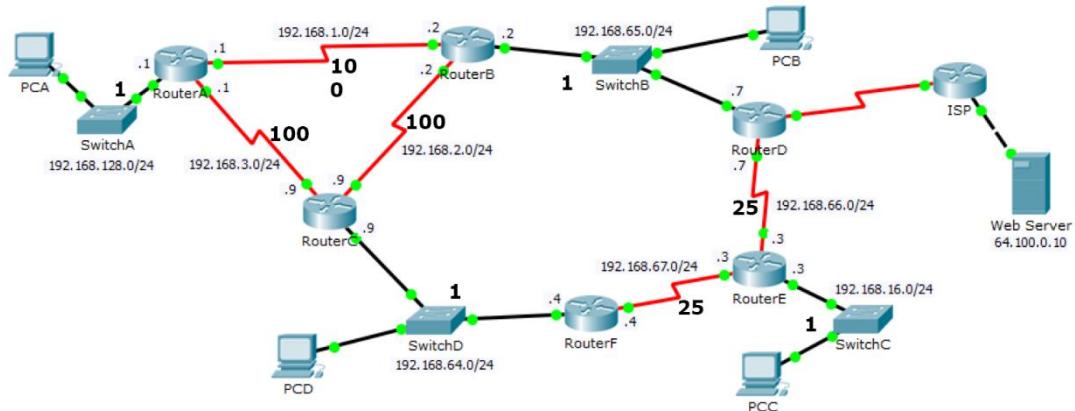
Il meccanismo di segnalazione esplicita della scadenza viene realizzato col campo **Link State Age**: se l'informazione non è più valida va comunicato **subito** agli altri, quindi si confeziona un LSA che contiene quella informazione con il sequence number aumentato (chi la riceve vede che questa è aggiornata) e **1h** nel campo Link State Age.

Chi riceve l'LSA lo trova nel database, vede che il sequence number è maggiore e butta via quello vecchio, poi guarda l'età e vede che è scaduto (**1h**), quindi lo toglie dal database.

LS type 1 (router links LSA)

I router li usano per comunicare a quali link sono connessi; **ogni router ne genera uno per area** (nell'area x comunica i link dell'area x, nell'area y i link dell'area y) e lo manda ai router adiacenti - se raggiunge un ABR questo farà il flooding solo nell'area di cui fa parte il router che ha generato l'LSA.

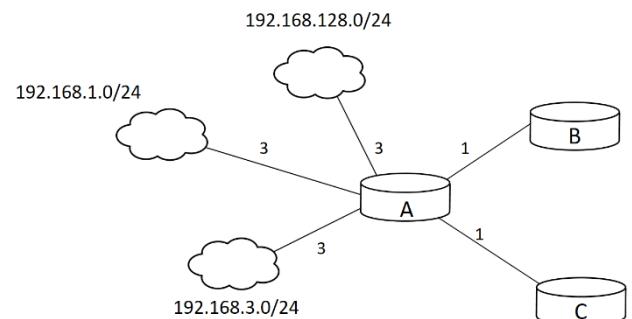
Esempio:



Il router A deve comunicare ai vicini quali reti di destinazione può raggiungere (192.168.128.0/24, 192.168.1.0/24 e 192.168.3.0/24), ma questa informazione non basta per ricostruire la topologia della rete. Mancano infatti le **informazioni sui router adiacenti**, perché un router può usare un **next hop** solo se questo è adiacente a lui, non basta che sia un **neighbor**; se A dicesse solo che è collegato a quelle reti, poi B dicesse che è collegato anche lui a 192.168.1.0, si capirebbe che questi sono collegati, ma non si potrebbe sapere se sono adiacenti o meno.

Oltre alle reti di destinazione A comunica in modo distinto quali sono i **router a cui è adiacente** (tramite il loro id) in modo da comunicare che nel calcolo del percorso un terzo router può usare il passaggio tra A e B.

La topologia che andiamo a ricostruire è fatta dal grafo dei router e per ciascuno sono rappresentate le reti che si possono raggiungere come destinazione finale.



Link State Header

- **Tipo 1:** singola area
- **Advertising router:** 1.1.1.1 (router A)

| | | |
|-----------------------------|------------|------------|
| LSA AGE: 0 | OPTIONS: 0 | LS TYPE: 1 |
| LINK STATE ID: 1.1.1.1 | | |
| ADVERTISING ROUTER: 1.1.1.1 | | |
| LS SEQUENCE NUM: 0X80000006 | | |
| CHECKSUM: 27683 | | LENGTH: 84 |

Link State Data

- Link Count: 5 (tre link + router B e C)
- I primi due si riferiscono ai router (tipo 1: punto-punto).
 - Link ID: 9.9.9.9 (router C) e 2.2.2.2 (router B)
 - Link Data: indirizzi delle interfacce con cui A manda i pacchetti a C e B; utili a questi due quando compilano le proprie tabelle di routing

| | | |
|---------|---|------------------------|
| V+E+B | 0 | LINK COUNT: 5 |
| | | LINK ID: 9.9.9.9 |
| | | LINK DATA: 192.168.3.1 |
| TYPE: 1 | 0 | METRIC: 100 |
| | | LINK ID: 2.2.2.2 |
| | | LINK DATA: 192.168.1.1 |
| TYPE: 1 | 0 | METRIC: 100 |

- Gli altri tre si riferiscono ai link (tipo 3: stub network)
 - Link ID: indirizzo rete
 - Link Data: maschera di sottorete

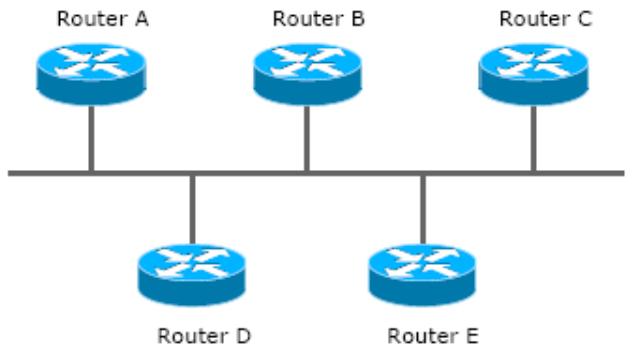
Il pacchetto completo rappresenta la situazione dello schema precedente di A e delle sue connessioni; il router la comunica ai suoi vicini che la manderanno ai vicini, finché tutti i router dell'AS non avranno ottenuto l'informazione.

| | | |
|---------------------------------|---|--------------------|
| LINK ID: 192.168.128.0 | | |
| LINK DATA: 255.255.255.0 | | |
| TYPE: 3 | 0 | METRIC: 1 |
| LINK ID: 192.168.1.0 | | |
| LINK DATA: 255.255.255.0 | | |
| TYPE: 3 | 0 | METRIC: 100 |
| LINK ID: 192.168.3.0 | | |
| LINK DATA: 255.255.255.0 | | |
| TYPE: 3 | 0 | METRIC: 100 |

OSPF – Reti Broadcast

Quando i router sono connessi tra loro con questo tipo di comunicazione di livello 2, OSPF ha problemi relativi a due aspetti:

1. Gestione di scambio di informazioni del protocollo (piano di controllo)
2. Capacità di calcolare percorsi migliori (impatta sul piano dati)



Ogni volta che A manda un pacchetto lo ricevono B, C, D, E ed è quello che succede ad esempio con ethernet, una situazione abbastanza comune come quando, ad esempio, si collegano le reti diversi edifici.

1. Aspetto di controllo

L'**adiacenza** è una proprietà che riguarda i router **a 2 a 2**, in questa situazione il router A anche se usa lo stesso mezzo di trasmissione vede B, C, D, E in maniera indipendente e quando manda il pacchetto di hello tutti lo ricevono ed tutti elencheranno A tra i propri vicini, lo stesso farà a sua volta A mettendo nell'elenco B, C, D, E

Da quel momento in poi la procedura che porta all'adiacenza è **1 a 1**, quindi A fa partire **4 operazione separate di sincronizzazione dei database** con i 4 router e lo stesso fanno poi questi ultimi.

Avendo 5 router che devono sincronizzarsi tra di loro a 2 a 2, man mano che il numero di router aumenta il numero di operazioni aumenta con n^2 quindi **la rete non è scalabile**.

2. Complessità computazionale

Ciascun router comunica nei suoi LSA di tipo 1 quali next hop può usare e quando il router costruisce il grafo anche se questi sono connessi con un solo punto con una unica LAN, si ottiene un **grafo totalmente connesso** e quindi si genera un numero maggiore di link, andando ad **aggiungere una difficoltà fittizia al calcolo dei cammini minimi** (perché nella realtà usano un solo link per comunicare).

Soluzione

Il protocollo, invece che far fare le operazioni a due a due (quindi evitare che siano tutti collegati tra loro), prevede che uno dei router della rete venga scelto come **designated router (DR)** e invece di sincronizzarsi tutti con tutti, **si sincronizzano tutti col master e viceversa**.

In questo modo i nodi sono tutti sincronizzati, ma si torna a una situazione con complessità lineare invece che quadratica, le operazioni di sincronizzazione concorrenti diventano infatti $n-1$.

Se il database del DR è completamente sincronizzato con quelli degli altri router, allora anche il database dei router è sincronizzato tra di loro.

- Nella fase iniziale i router eleggono il designated router, quelli che non sono il designated router si sincronizzano con lui, questo invece si sincronizza con tutti
- Quando si devono calcolare i percorsi, invece che considerare tutti connessi con tutti si considerano **tutti connessi col designated router e viceversa**.
- Se un router non DR riceve un LSA aggiornerà solo il DR e il DR aggiornerà poi tutti gli altri

Nota: questo vale anche se sulla rete ci sono solo due route (se ce n'è uno solo invece è una stub network)

Problema del single point of failure: se il DR cade gli altri router non si sincronizzano più tra di loro, quindi si considerano tutti disconnessi.

Nella rete si sceglie anche un **backup designated router**; i router fanno la sincronizzazione con DR e anche con **BDR**, in questo modo se DR cade il BDR diventa immediatamente il suo sostituto.

Backup DR non deve fare il selective flooding, quello lo fa solo il DR

Elezione del Designated Router (e del Backup Designated Router)

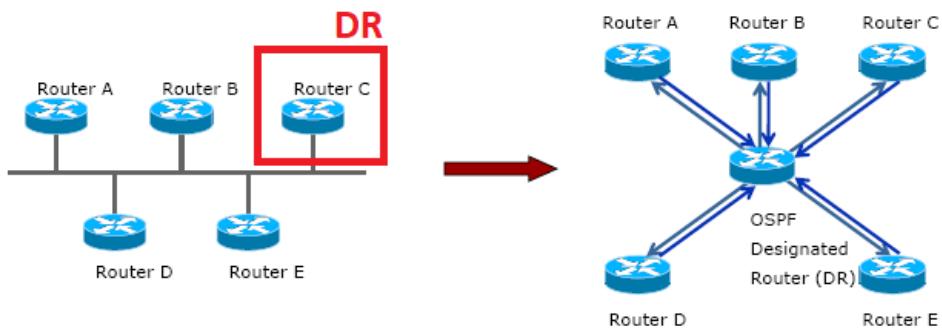
Nella fase iniziale – quando i router formano le adiacenze con i router vicini prima della sincronizzazione dei database si eleggono DR e DRB tramite due campi del pacchetto di hello (**Designated Router** e **Backup Designated Router**) che contengono gli **indirizzi IP** delle interfacce dei router sul link broadcast.

Il campo **priority (PR)** del pacchetto di hello viene usato dai router quando si presentano, per comunicare con quale priorità nell'elezione del DR sono stati configurati (solitamente in base alla potenza dell'hardware, ma può comunque essere modificato) - a parità di priorità si sceglie il router con l'IP più grande.

La prima volta che un router manda un pacchetto di hello mette sé stesso o 0 come DR, quando poi riceve i vari hello guarda quale router ha priorità maggiore e quindi metterà il suo indirizzo IP nel campo **Designated Router** dei suoi pacchetti; dopo un certo tempo tutti i router avranno lo stesso indirizzo di DR e BDR.

Piano dati

A livello di comunicazione nel piano dati si usa **una rappresentazione equivalente (virtuale)** della rete dal punto di vista del calcolo dei percorsi ottimi in cui il router DR (*nell'esempio Router C*) compare due volte, sia come DR che come router 'normale'.



Se arriva un pacchetto al *Router A* che deve essere inoltrato ad *E* per la destinazione finale, nella rappresentazione virtuale della connettività ci sarà il percorso *A – DR – E*.

Dato che **virtualmente un pacchetto attraversa sempre il designated router** il costo da *A* a *DR* sarà uguale al costo dell'interfaccia del router *A* (e così tutti gli altri), mentre il costo da *DR* a ogni altro router viene messo uguale a 0.

In questo modo se mando un pacchetto da *A* ad *E* attraverso il *DR*, ma tra *DR* ed *E* il costo è 0, il costo totale è 1 come nella situazione in cui *A* mandava direttamente sul link broadcast

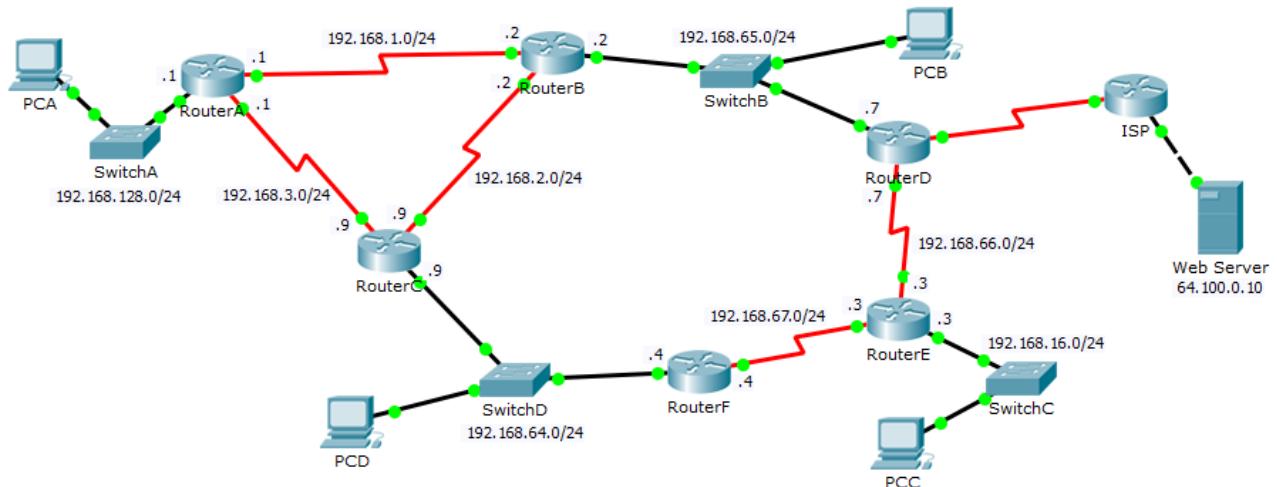
Il calcolo del percorso di costo minimo è equivalente, ma Dijkstra lavora su un numero di percorsi minore

Il *router A* conosce l'intero percorso verso una specifica destinazione, essendo consapevole della presenza del router virtuale DR, quindi quando va a riempire la tabella di routing per metterà come next hop l'ip del router che nel percorso è successivo al DR.

LS type 2 (Network links LSA)

Per descrivere una rete con un unico link broadcast i router che non sono DR manderanno LSA di tipo punto-punto per comunicare che sono connessi al router DR, mentre il **DR userà un LSA di tipo 2 per comunicare chi sono tutti i suoi vicini**

Esempio



Il Router C manderà il suo Link State Update Packet che contiene 5 LS di tipo 1 (Router Link LSA):

- 2 stub network 192.168.2.0/24 e 192.168.3.0/24 (link tipo 3)
- 2 connessioni punto-punto con i router A e B (link tipo 1)
- 1 **transit network** (link tipo 2)
 - Link id: *indirizzo IP del designated router*
 - *Link data: indirizzo interfaccia (in questo caso coincidono perché è C il DR)*

Il router C essendo il **designated router** manda anche un LS di tipo 2 (Network Link LSA):

- Link state ID: indirizzo IP della propria interfaccia connessa alla rete
- Advertising router: ID di C (è generato dal DR)
- Maska di rete
- Router connessi (*attached router*): 4.4.4.4 (F) e 9.9.9.9 (C)

| | | |
|------------------------------------|------------|------------|
| LSA AGE: 0 | OPTIONS: 0 | LS TYPE: 2 |
| LINK STATE ID: 192.168.64.9 | | |
| ADVERTISING ROUTER: 9.9.9.9 | | |
| LS SEQUENCE NUM: 0X80000001 | | |
| CHECKSUM: 45610 | LENGTH: 32 | |
| NETWORK MASK: 255.255.255.0 | | |
| ATTACHED ROUTER: 4.4.4.4 | | |
| ATTACHED ROUTER: 9.9.9.9 | | |

LS type 5 (AS external link LSA)

L'AS della rete di esempio ha un confine dopo il router *D*, con il router *ISP* che non fa parte dell'AS, quindi *D* è un **ASBR** che riceverà pacchetti dall'esterno verso la rete interna e viceversa.

Quando *ISP* deve mandare un pacchetto con una destinazione che ritiene sia interna all'AS, lo manda a router *D* che ha tutte le destinazioni dell'AS e sa quale next hop deve usare, quindi l'ASBR inizia l'inoltro all'interno dell'AS. Se invece un host interno all'AS manda un pacchetto al server 64.100.0.10, il pacchetto deve seguire un percorso verso il router *D* (ASBR) in modo che questi lo mandi su *ISP* e trovi la destinazione finale.

Gli ASBR scambiano informazioni con l'esterno e apprendono informazioni sulle destinazioni, quindi devono **propagare all'interno dell'AS destinazioni che si trovano all'esterno**, compresa una eventuale default route. Lo fanno tramite gli **LS di tipo 5** (AS external link).

I router interni devono calcolare il percorso ottimo solo fino all'ASBR, poi il pacchetto esce dall'AS quindi il compito di OSPF è finito.

Header:

- Link State Identification: indirizzo rete esterna
- Advertising router: Id dell' ASBR

Dati:

- Maschera: completa l'indirizzo della rete esterna
- Metrica: costo del percorso esterno

Per una destinazione esterna il percorso avrà una **parte interna** (per raggiungere l' ASBR) e una **parte esterna**, di quest'ultima non si sa chi la gestisce né si conosce la metrica.

La metrica che si comunica con l'LS di tipo 5 quindi non sempre è consistente con i costi che si usano all'interno dell'AS, si specifica con il bit **E**:

- **E = 1:** il costo va interpretato come un numero maggiore di qualunque percorso all'interno dell'AS. Essendo una metrica esterna e non consistente, non può essere sommata al costo del percorso interno.
- **E = 0:** si può interpretare il costo come se fosse uno qualsiasi all'interno dell'AS e quindi si può calcolare il costo totale del percorso come costo del percorso interno + costo del percorso esterno

| | | | | | |
|-----------------------------------|------------|-------------------|----------|--|--|
| Link State Age | | Options | 5 | | |
| Link State Identification | | | | | |
| Advertising Router | | | | | |
| Link State Sequence Number | | | | | |
| Link State Checksum | | Length | | | |
| Network Mask | | | | | |
| E | 0 | Metric | | | |
| Forwarding Address | | | | | |
| Route Tag | | | | | |
| E | TOS | TOS Metric | | | |
| Forwarding Address | | | | | |
| Route Tag | | | | | |
| | | | | | |

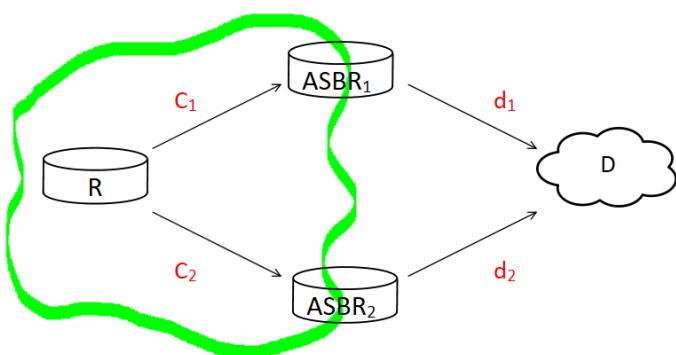
Esempio

| | | |
|-----------------------------|------------|------------|
| LSA AGE: 44 | OPTIONS: 0 | LS TYPE: 5 |
| LINK STATE ID: 0.0.0.0 | | |
| ADVERTISING ROUTER: 7.7.7.7 | | |
| LS SEQUENCE NUM: 0X80000001 | | |
| CHECKSUM: 19052 | LENGTH: 36 | |
| NETWORK MASK: 0.0.0.0 | | |
| E: 1 | 0 | METRIC: 1 |
| FORWARDING ADDRESS: 0.0.0.0 | | |
| ROUTE TAG | | |

D genera un *LS di tipo 5* per la default route: qualsiasi pacchetto destinato fuori dell'AS deve essere inoltrato a lui.

La metrica è 1, ma essendo E = 1 il costo non è sommabile al costo del percorso interno all'AS.

Perché interessa il costo complessivo?



Caso in cui ci sono più ASBR che raggiungono una stessa destinazione esterna D con due costi differenti. Il router interno R avrà un percorso interno ottimo per raggiungere ASBR₁ (con costo c_1) e uno per raggiungere ASBR₂ (con costo c_2).

Se i due ASBR sono in grado di determinare un costo verso la destinazione esterna consistente con le metriche interne ($E = 0$), allora R può calcolare il costo complessivo dei due percorsi per la destinazione D

1. tramite ASBR₁, costo = $c_1 + d_1$
2. tramite ASBR₂, costo $c_2 + d_2$

Se $E = 1$, i costi d_1 e d_2 non sono congruenti con quelli interni all'AS quindi R può solo prendere la decisione sulla base dei costi interni (c_1 e c_2), quindi usa almeno il percorso più breve per uscire dall'AS

[Packet Tracer]

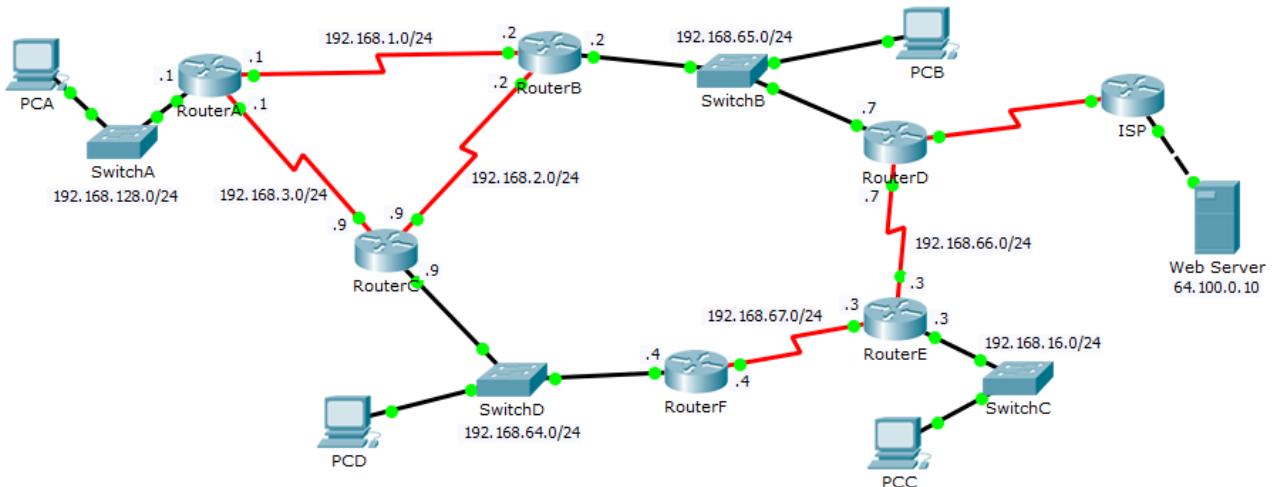


Tabella di routing di A:

```
RouterA# show ip route
(...)
0 192.168.2.0/24 [110/200] via 192.168.1.2, 00:01:36, Serial0/0/0
                  [110/200] via 192.168.3.9, 00:01:36, Serial0/0/1
(...)
C 192.168.128.0/24 is directly connected, FastEthernet0/0
0*E2 0.0.0.0/0 [110/1] via 192.168.1.2, 00:00:56, Serial0/0/0
```

- Ci sono le reti direttamente connesse e tutte le altre destinazioni previste nell'AS
- Per la 2.0 ci sono due alternative che hanno costo uguale, cioè A – C e A – B :
 - Costo = 100 (da A a B/C) + 100 (da B/C alla dest)
 - OSPF ha distanza amministrativa 110
- Per ogni riga c'è l'età associata agli LSA da cui è stata tratta l'informazione; i pacchetti di hello si scambiano ogni 10s ma se non ci sono modifiche gli LSA vengono scambiati con tempi più lenti
- L'ultima entry è la default route appresa tramite OSPF (*E2 indica che il bit E è settato*).

Per qualsiasi router l'elenco delle destinazioni della tabella di routing è lo stesso (con costi e next hop variabili); all'interno dell'AS tutti conoscono tutte le destinazioni

RouterA# show ip ospf database: mostra la lista degli LSA presente nel database OSPF del router. A prescindere dal router su cui eseguiamo il comando il contenuto sarà identico

| Router Link States (Area 0) | | | | | |
|--------------------------------|------------|-----|------------|----------|------------|
| Link ID | ADV Router | Age | Seq# | Checksum | Link count |
| 1.1.1.1 | 1.1.1.1 | 761 | 0x80000006 | 0x008a29 | 5 |
| 3.3.3.3 | 3.3.3.3 | 761 | 0x80000006 | 0x009f9a | 5 |
| (...) | | | | | |
| Net Link States (Area 0) | | | | | |
| Link ID | ADV Router | Age | Seq# | Checksum | |
| 192.168.64.9 | 9.9.9.9 | 727 | 0x80000001 | 0x006c43 | |
| 192.168.65.7 | 7.7.7.7 | 727 | 0x80000001 | 0x004e39 | |
| Type-5 AS External Link States | | | | | |
| Link ID | ADV Router | Age | Seq# | Checksum | Tag |
| 0.0.0.0 | 7.7.7.7 | 770 | 0x80000001 | 0x004a6c | 1 |

RouterA#show ip ospf database router: mostra il contenuto di ciascun LS di **tipo 1**, con l'header comune e poi la parte dati per ogni link

```
LS age: 1163
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 1.1.1.1
Advertising Router: 1.1.1.1
LS Seq Number: 80000006
Checksum: 0x8a29
Length: 84
Number of Links: 5

Link connected to: a Stub Network
(Link ID) Network/subnet number: 192.168.128.0
(Link Data) Network Mask: 255.255.255.0
(...)
```

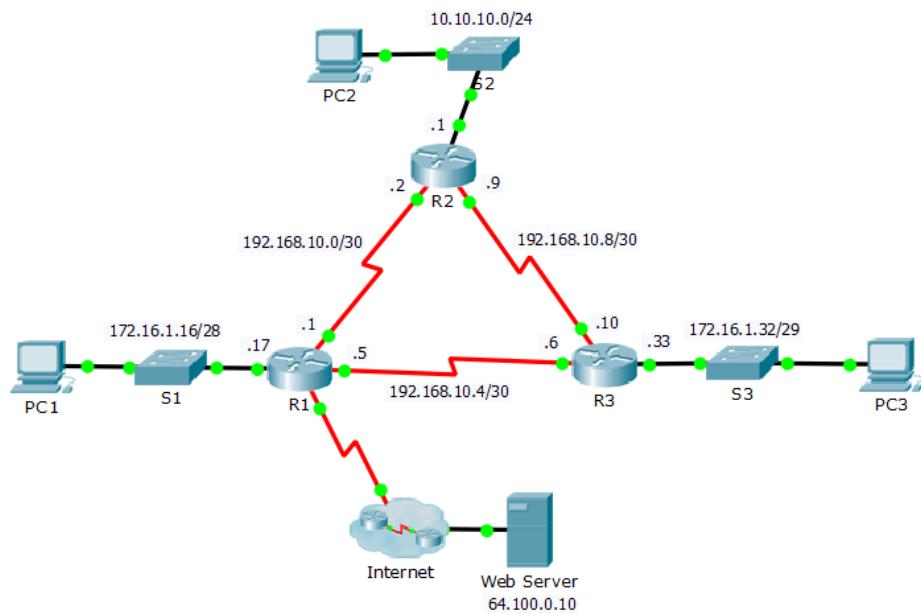
RouterA# show ip ospf database network: mostra il contenuto di ciascun LS di **tipo 2** generato da ciascun designated router

```
Routing Bit Set on this LSA
LS age: 1499
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 192.168.64.9 (address of Designated Router)
Advertising Router: 9.9.9.9
LS Seq Number: 80000001
Checksum: 0x6c43
Length: 32
Network Mask: /24
    Attached Router: 4.4.4.4
    Attached Router: 9.9.9.9
(...)
```

RouterA# show ip ospf neighbor: mostra l'elenco dei vicini per il router

| Neighbor | ID | Pri | State | Dead Time | Address | Interface |
|----------|----|-----|---------|-----------|-------------|-------------|
| 2.2.2.2 | | 0 | FULL/ - | 00:00:38 | 192.168.1.2 | Serial0/0/0 |
| 9.9.9.9 | | 0 | FULL/ - | 00:00:32 | 192.168.3.9 | Serial0/0/1 |

Configurazione OSPF singola area



```
R1(config)#router ospf 1
```

- Il comando router inizializza la configurazione del protocollo di routing
- R1 non manda ancora pacchetti perché non è stato ancora specificato quali informazioni deve scambiare e su che interfacce

Nota: si usa un process id, perché sullo stesso router potrebbero esserci più istanze di ospf in esecuzione contemporaneamente su un sottoinsieme delle interfacce

Assegnamento router id

Quando un router manda un pacchetto di *hello* si presenta con un id, **questo non può essere cambiato finché non si riavvia il processo di routing**; se venisse cambiato mentre è attivo il routing, i router che lo hanno già registrato come vicino vedrebbero un nuovo router – eventualmente il “vecchio” verrebbe eliminato, ma si sprecherebbe molto tempo.

1. Assegnamento automatico (default)

Al router viene assegnato un id uguale all’indirizzo IP più grande tra gli indirizzi delle interfacce di quel router

2. Assegnamento manuale

```
R1(config-router)#router-id 1.1.1.1
```

Vantaggio: si può dare un nome simbolico che permette di identificare velocemente il router nella rete

Svantaggio: avere come id l’indirizzo di una delle interfacce del router implica avere un modo per comunicare con quel router, che è una cosa buona per la gestione della rete

3. Interfacce di loopback

IP ha un blocco riservato per le **interfacce di loopback**: si assegna a un host un indirizzo IP che non è connesso a una interfaccia fisica, ma permette comunque di raggiungere il router, si può quindi definire una **interfaccia virtuale** che termina sul router stesso.

```
R2(config) # interface loopback [id interfaccia]
```

Il comando crea un’interfaccia virtuale che “*esiste*” per ogni layer superiore al 2 e in particolare posso assegnarle un indirizzo IP che sarà l’indirizzo della **rete di loopback** a cui l’interfaccia è connessa (rete sulla quale ci potrà essere collegata solo quell’ interfaccia – non è una “*rete virtuale*”).

Solitamente si crea una interfaccia di loopback su ogni router, le si assegna un IP e poi si fa in modo che il router propaghi quell’indirizzo con il protocollo di routing; è come se si creasse una **rete parallela dove gli unici host indirizzati sono i router**.

Queste sono reti separate: ciascun router è una rete (**host network**) e quindi assegno indirizzi con **maschera /32** (tipicamente si usano indirizzi privati)

```
R2(config-if) #ip address 172.16.0.2 255.255.255.255
```

Quando viene determinato automaticamente l’id del router vengono considerate prima le interfacce di loopback, in questo modo anche se l’indirizzo non dovesse essere il più grande verrà comunque preferito a quelli delle interfacce normali.

Avvio dello scambio di informazioni

```
R1(config-router)# network [indirizzo interfaccia] [wild card mask] [area id]
```

- **wild card mask**: complemento a 1 della subnet mask
- **area id**: nel pacchetto di hello deve essere specificata l’area nella quale viene scambiato il pacchetto, non potendolo configurare staticamente si indica nel comando network

Il comando impone che:

- La rete venga annunciata agli altri router
- Sull’interfaccia vengano scambiati pacchetti

La **sintassi alternativa** prevede che si specifichi l’indirizzo IP dell’interfaccia con wildcard mask a 0; dovrà essere eseguito un comando network per ogni interfaccia del router

R1:

```
R1(config-router)# network 172.16.1.16 0.0.0.15 area 0           //verso stub network
R1(config-router)# network 192.168.10.0 0.0.0.3 area 0           //verso R2
R1(config-router)# network 192.168.10.4 0.0.0.3 area 0           //verso R3
```

R2:

```
R2(config-router)# network 192.168.10.2 0.0.0.0 area 0           //sintassi alternativa
%OSPF(...): Process 1, Nbr 1.1.1.1 on Serial0/0/0 from LOADING to FULL, Loading Done:
R2 ha iniziato a scambiare pacchetti di hello con R1 e sono diventati completamente adiacenti
R2(config-router)# network 192.168.10.9 0.0.0.0 area 0
R2(config-router)# network 192.168.10.1 0.0.0.0 area 0
```

R3:

```
R3(config-router)# network 192.168.10.10 0.0.0.0 area 0
R3(config-router)# network 192.168.10.6 0.0.0.0 area 0
R3(config-router)# network 172.16.1.33 0.0.0.0 area 0
```

Tabella dei vicini di R3

R3#show ip ospf neighbor

| Neighbor ID | Pri | State | Dead Time | Address | Interface |
|-------------|-----|---------|-----------|--------------|-------------|
| 1.1.1.1 | 0 | FULL/ - | 00:00:35 | 192.168.10.5 | Serial0/0/0 |
| 2.2.2.2 | 0 | FULL/ - | 00:00:35 | 192.168.10.9 | Serial0/0/1 |

- I vicini sono R1 e R2, entrambi FULL
- Priorità: non significativa
- Dead time: settato a 40s alla ricezione del pacchetto di hello, poi inizia a decrementare, se arriva a 0 il vicino sarà considerato morto
- Address: indirizzo IP dell'interfaccia del vicino con la quale R3 sta comunicando
- Interface: interfaccia di R3 con la quale comunica col vicino

Interfacce passive: come con RIP possiamo usare il comando `passive-interface` per evitare che i router mandino pacchetti di hello sulle interfacce connesse alle stub network.

Informazioni sul protocollo

1. R1# show ip protocols

- Routing Protocol is "ospf 1"
- Router ID 1.1.1.1
- Number of areas in this router is 1. 1 normal 0 stub 0 nssa: le aree non backbone possono essere normali, stub (non ci sono ASBR) o not so stubby, questa distinzione è utile per fare ottimizzazioni – il tipo di area va specificato
- Maximum path: 4
- Routing for Networks:

| |
|-----------------------------|
| 172.16.1.16 0.0.0.15 area 0 |
| 192.168.10.0 0.0.0.3 area 0 |
| 192.168.10.4 0.0.0.3 area 0 |
- Passive Interface(s):

| |
|-----------------|
| FastEthernet0/0 |
|-----------------|
- Routing Information Sources:

| Gateway | Distance | Last Update |
|---------|----------|-------------|
| 1.1.1.1 | 110 | 00:14:22 |
| 2.2.2.2 | 110 | 00:14:42 |
| 3.3.3.3 | 110 | 00:14:22 |

Distance: (default is 110)

2. R1#show ip ospf

- **Supports only single TOS(TOS0) routes:** In questa implementazione non c'è il supporto per più classi di traffico, quindi tutti gli LSA contengono solo il costo per la classe 0 (best effort)
- **SPF schedule delay 5 secs, Hold time between two SPFs 10 secs:** Quando al router arriva una informazione e decide di rieseguire l'algoritmo di Dijkstra per vedere se cambia la tabella di routing non lo fa subito, ma aspetta 5s in caso arrivassero altri aggiornamenti. Per 5s, quindi, usa le informazioni vecchie in modo da ridurre l'overhead dovuto all'esecuzione dell'algoritmo. Tra una esecuzione e l'altra deve passare un Hold Time di 10s
- **Area BACKBONE(0):** Statistiche sull'area
- **SPF algorithm executed 9 times:** Numero di esecuzioni dell'algoritmo di Dijkstra

3. R1#show ip ospf interface serial 0/0/0

- **Internet address is 192.168.10.1/30, Area 0:** Indirizzo della rete a cui l'interfaccia è connessa e area in cui si trova
- **Process ID 1, Router ID 1.1.1.1, Network Type POINT-TO-POINT, Cost: 64**: ID del processo e del router, tipo di rete (tipo 2), costo
- **Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5:** Valori dei timer
- **Hello due in 00:00:08:** Tra quanto è previsto il prossimo pacchetto di hello
- **Neighbor Count is 1 , Adjacent neighbor count is 1:** Numero di vicini

4. R1#show ip ospf database

OSPF Router with ID (1.1.1.1) (Process ID 1)

| Router Link States (Area 0) | | | | | |
|-----------------------------|------------|------|------------|----------|------------|
| Link ID | ADV Router | Age | Seq# | Checksum | Link count |
| 1.1.1.1 | 1.1.1.1 | 1631 | 0x80000006 | 0x00b9ab | 5 |
| 2.2.2.2 | 2.2.2.2 | 1651 | 0x80000005 | 0x000c1d | 4 |
| 3.3.3.3 | 3.3.3.3 | 1631 | 0x80000005 | 0x00d249 | 4 |

Per ora circolano solo LSA di tipo 1 perché R1 non manda niente alla rete esterna, per far diventare R1 un ASBR si imposta la **default route** e poi la si diffonde con OSPF:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial0/1/0
R1(config)#router ospf 1
R1(config-router)# default information originate
```

Adesso R1 nel suo LSA di tipo 1 comunica che è un ASBR, quindi crea un nuovo **LSA di tipo 5** con la rete di destinazione e lo manda ai suoi vicini.

Nella tabelle di routing di R2 e R3 si trova la default route **0*E2**: rota di default ottenuta tramite OSPF con costo congruente a quello interno all'AS.

Nei database OSPF si troverà quindi anche l'**LSA di tipo 5 (External Link State)** inviato da R1

Configurazione dei costi

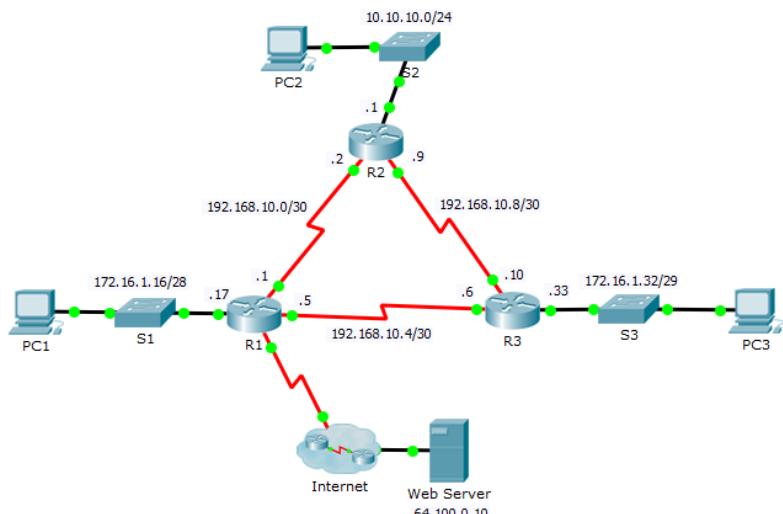
Costi default di IOS

Il costo riflette la capacità di trasmissione sull'interfaccia in maniera inversamente proporzionale: maggiore è la capacità minore è il costo.

$$\text{costo} = \text{reference-bandwidth}/\text{interface-bandwidth}$$

- **reference-bandwidth:** fattore di normalizzazione, default 100Mbs
- **interface-bandwidth:** dipende dalla configurazione dell'interfaccia

| Interface Type | 100/bw in Mbps | Cost |
|---------------------|----------------|------|
| 10 Gigabit Ethernet | 100/10,000 | 1 |
| Gigabit Ethernet | 100/1,000 | 1 |
| Fast Ethernet | 100/100 | 1 |
| Ethernet | 100/10 | 10 |
| E1 | 100/2.048 | 48 |
| T1 | 100/1.544 | 64 |
| 128 kbps | 100/0.128 | 781 |
| 64 kbps | 100/0.064 | 1562 |



Nella tabella di routing di R1 per la rotta 10.10.10.0/24 il costo è **65**: si va sull'interfaccia seriale (**64**) verso R2 e poi sull'interfaccia fast ethernet (**1**).

R1# show interface s0/0/0: informazioni relative al lvl 2 tra cui **BW** che è **1544 Kbit** (valore nominale)

R1#show ip route 10.10.10.0: mostra il protocollo con cui è stata scoperta la rotta, qual è il costo amministrativo e il costo totale, se è intra-area o no, quando è stata ricevuta l'ultima informazione, ecc

Modifica del costo di una rotta

Può essere utile in alcuni casi “forzare” il traffico a passare su un determinato percorso piuttosto che un altro, a prescindere da quale sia il migliore

1. Modo **implicito**: cambio la BW dell'interfaccia in modo che cambi il risultato della divisione

```
R1(config)#interface s0/0/0
R1(config-if)#bandwidth [valore in Kb]
```

Il router quindi deve generare un novo LSA per aggiornare il costo, dopo 5 sec viene poi riseguito l'algoritmo di Dijkstra ed eventualmente modificata la tabella di routing

2. Modo **esplicito**: senza modificare la BW dell'interfaccia

```
R1(config-if)#ip ospf cost [valore]
```

La modifica del costo sarà, però, **solo in una direzione**, perché per un altro router lo stesso link costa ancora quanto prima; non è quindi **simmetrica**

Configurazione con link broadcast

DR: router con priorità maggiore

BDR: router con seconda priorità maggiore

DROTHER: router che non è né DR né BDR

Priorità di default: 1

- Modificare la priorità di un router:

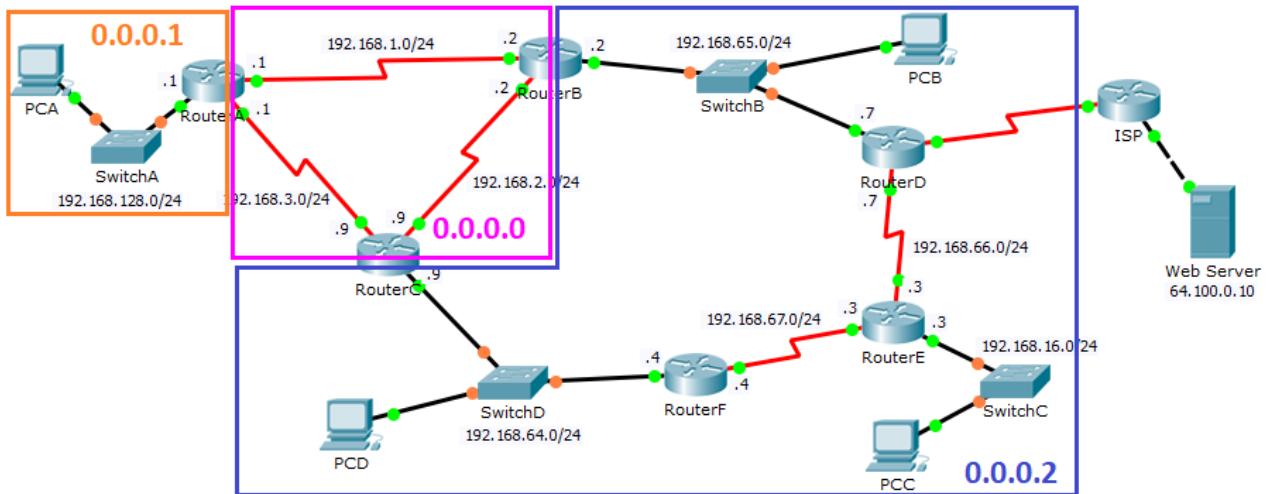
```
R(config-if)#ip ospf priority {0-255}
```

Il processo di elezione è dinamico quindi ha un **transitorio**; nella sua esecuzione si **privilegia la stabilità** rispetto al risultato atteso.

Se ad esempio un router ha priorità più alta, ma altri router si connettono sul link e si mettono d'accordo prima che questo "arrivi", il DR non cambia e cambierà solo se viene riseguito il processo di elezione (viene persa l'adiacenza).

Si favorisce quindi la stabilità anche se non sarebbe esattamente il risultato atteso; se si vuole essere sicuri del risultato occorre configurare prima DR e BDR e poi attivare in seguito le interfacce degli altri router.

OSPF multi-area



Per i router interni non cambia nulla riguardo a quello che comunicano rispetto al caso a singola area, cambia solo quello che fanno i router che sono adesso ABR.

Per i router interni i database diventano più piccoli, per gli ABR i database invece si “moltiplicano” perché avranno una sezione del database separata per ogni area.

Un router ABR (ad esempio C) si comporta in ciascuna area come un qualunque router interno, limitatamente all’interfaccia che si trova in quell’area. Genera due LSA di tipo 1:

1. Uno lo manda nell’area 0
 - a. 2 link tipo 3 (stub network)
 - b. 2 link tipo 1 (router A e B)
 2. Uno lo manda nell’area 2
 - a. 1 link di tipo 2 (transito)
- Il router fa quello che fanno i router interni, ma come se fosse composto due router diversi perché opera separatamente sulle due aree; conosce nel dettaglio sia la topologia dell’area 0 che la topologia dell’area 2.

ABR Summary link LSA (tipo 3)

- Descrivono le rotte **inter-area**, uno per ogni destinazione
- Gli ABR informano i router di una area delle **destinazioni che si trovano nelle altre aree che questi conoscono** (“*questa destinazione la raggiungi a costo x tramite me*”)
 - *C genera un LSA per una destinazione nell’area 2, quindi lo trasmetterà nell’area 0*
 - *Se la rete di destinazione viene appresa da C tramite l’area 0, questi invierà l’LSA nell’area 2*
- **Il flooding si limita all’area in cui quell’LSA viene generato**: se C manda un LSA nell’area 2 questo gira ed arriva fino a B. B non deve rimandare l’informazione perché le altre sue interfacce sono nell’area 0
- Ogni ABR è connesso alla backbone e a una o più aree, in ciascuna di queste altre aree riversa le info che riceve nella backbone e nella backbone riversa le informazioni che riceve dalle altre aree

Formato

- Header
 - a. Link state id: indirizzo della rete
 - b. Advertising Router = id dell' ABR
- Data
 - Maschera di rete
 - Costo **aggregato**: il costo è **calcolato dall'advertising router** ed è il costo complessivo da lui alla destinazione

| | | |
|-----------------------------------|---------------|--------|
| Link State Age | Options | 3 or 4 |
| Link State Identification | | |
| Advertising Router | | |
| Link State Sequence Number | | |
| Link State Checksum | Length | |
| Network Mask | | |
| 0 | Metric | |
| TOS | Metric | |
| | | |

Esempio:**Area 0**

- C conosce la topologia dell'area 2 (scoperta con LSA di tipo 1)
- Per raggiungere 192.168.16.0/24 (area 2) il costo è 27
- Essendo un ABR deve informare quelli delle altre aree (0), quindi costruisce un LSA di tipo 3
- Quando A riceve questa informazione calcola il costo per la destinazione 192.168.16.0/24 e lo mette nella propria tabella di routing:
costo dest = costo minimo da A a C + costo da C a dest
- Anche B genera un LSA per la stessa destinazione, quindi A riceve l'informazione e la compara con quella ricevuta da C per scegliere il percorso con costo minore e lo mette nella tabella di routing. A sa che esistono più percorsi per raggiungere la destinazione è se quello nella tabella dovesse cadere utilizzerò l'altro

| | | |
|-----------------------------|------------|------------|
| LSA AGE: 0 | OPTIONS: 0 | LS TYPE: 3 |
| LINK STATE ID: 192.168.16.0 | | |
| ADVERTISING ROUTER: 9.9.9.9 | | |
| LS SEQUENCE NUM: 0X80000007 | | |
| CHECKSUM: 47720 | LENGTH: 28 | |
| NETWORK MASK: 255.255.255.0 | | |
| 0 | METRIC: 27 | |

Area 2

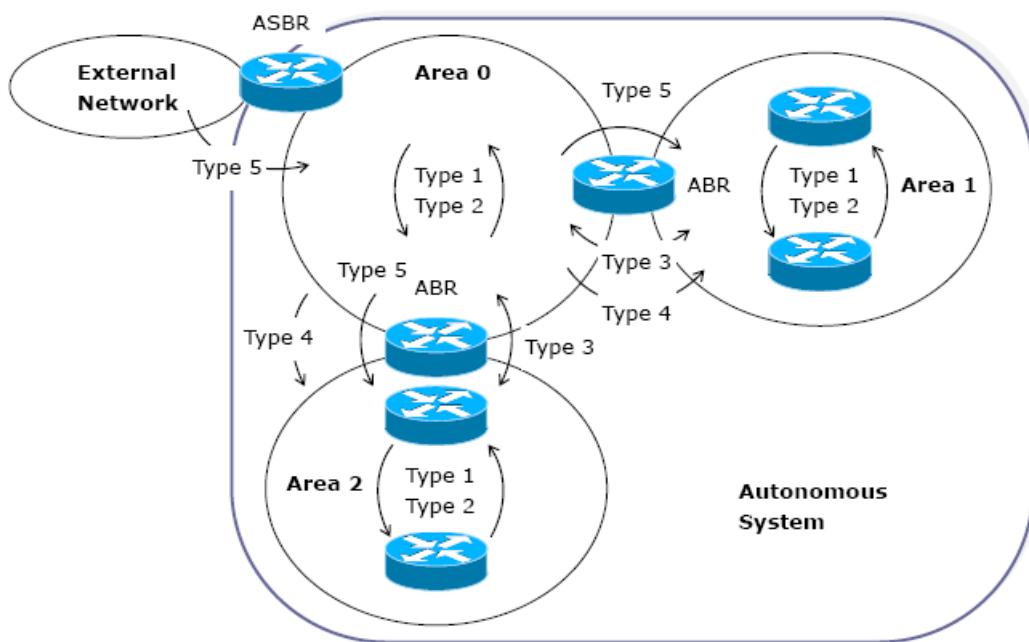
- C informa i router dell'area 2 che sa come raggiungere 192.168.1.0/24 e che il costo è 200
- C viene a sapere da A come raggiungere la destinazione 192.168.128.0/24 quindi manderà un LSA di tipo 3 relativo a questa destinazione nell'area 2

ASBR Summary Link LSA (tipo 4)

- Un **ASBR** genera degli LSA di tipo 5 per comunicare quali rotte esterne conosce e questi **circolano indipendentemente dai confini delle aree**
- Un router fuori dall'area dell'ASBR deve sapere come raggiungere l'ASBR che conosce una determinata rotta esterna che gli è arrivata tramite LSA di tipo 5.
- L'ASBR manderà degli LSA di tipo 1 all'interno dell'area in cui si trova e specificherà che è un ASBR; quando un ABR della stessa area riceve l'informazione manderà un **LSA di tipo 4 alle altre aree**.
- La **struttura dell'LSA di tipo 4** è uguale a quella di tipo 3, ma nel campo Link State Id contiene l'**id del router ASBR** (la maschera in questo caso non è significativa)

Esempio:

- C manda nell'area 0 un LSA di tipo 4 per dire che sa come raggiungere D e la distanza è 51
- Lo stesso farà B che avrà però una distanza diversa
- A riceve l'informazione da B e da C e guarda tramite quale percorso riesce a raggiungere router D con costo minore, questo gli interessa, perché tramite l'LSA di tipo 5 ha saputo che se deve mandare un pacchetto fuori dall'AS deve mandarlo a D.

LSA Selective flooding

Tipo 1 e 2: circolano nell'area in cui sono stati generati

Tipo 3 e 4: generati solo da ABR (informazioni inter-area) circolano nell'area in cui sono stati generati

Tipo 5: circolano senza confini.

[Packet Tracer]

```
RouterA# show ip route
C 192.168.1.0/24 is directly connected, Serial0/0/0
O 192.168.2.0/24 [110/200] via 192.168.1.2, 01:55:42, Serial0/0/0
    [110/200] via 192.168.3.9, 01:55:42, Serial0/0/1
C 192.168.3.0/24 is directly connected, Serial0/0/1
O IA 192.168.16.0/24 [110/127] via 192.168.3.9, 01:55:12, Serial0/0/1
    [110/127] via 192.168.1.2, 01:55:02, Serial0/0/0
O IA 192.168.64.0/24 [110/101] via 192.168.3.9, 01:55:42, Serial0/0/1
(...)
```

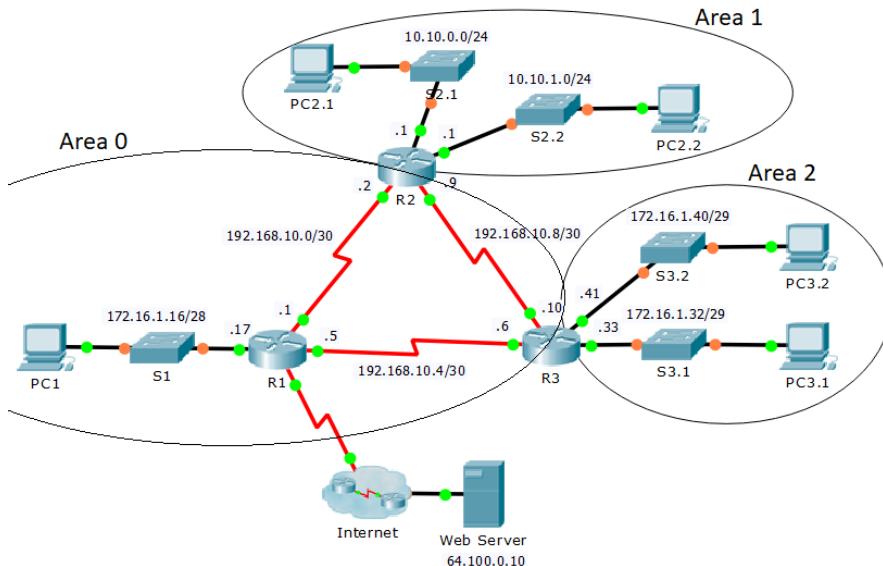
- Le entrate accanto **O IA** indicano che quella è una rotta **inter-area**: l'informazione su come raggiungere la rete è stata appresa sommando il costo per raggiungere l'ABR e il costo comunicato dall'ABR che ha comunicato la destinazione
- L'entrata 192.168.2.0/24 (link tra B e C) è solo “O”, cioè **intra-area**: A la conosce direttamente perché si trova nella sua stessa area

RouterF# show ip ospf database

Il contenuto del database è diverso rispetto al caso della rete a singola area:

- LSA di tipo 1 **solo per i router interni all'area 2**, non ha quindi quelli inviati da A
- LSA di **tipo 3** mandati da C e B in quanto ABR
- LSA di **tipo 4** con cui C e B dicono a F che sanno come raggiungere l'ASBR (D) anche se questo è nella stessa area di F:
 - B nell'area 0 dice che può raggiungere il router D
 - C dall'area 0 scopre che può raggiungere D, non può filtrare questa informazione perché cancellerebbe un possibile percorso alternativo
 - C manda un LSA di tipo 4 per dire che può raggiungere D
 - Lo stesso fa B
 - F sa che può raggiungere D direttamente, ma dagli LSA di tipo 4 scopre che è raggiungibile anche tramite C e B (inter-area); usa questa informazione casomai non potesse più raggiungere D internamente

Configurazione OSPF multi-area



- Si parte dalla configurazione a singola area dove le LAN 2.1, 2.2, 3.1 e 3.2 risultano tutte nell'area 0; per "effettuare" la divisione in aree è necessario annullare i comandi `network` già presenti nel file di configurazione e dare nuovi comandi `network` specificando le aree.

(Sintassi con indirizzo interfaccia e wildcard mask 0)

```
R2(config-router)#no network 10.10.0.1 0.0.0.1 area 0
R2(config-router)#no network 10.10.1.1 0.0.1.1 area 0
R2(config-router)#network 10.10.1.0 0.0.0.1 area 1
R2(config-router)#network 10.10.1.0 0.0.1.1 area 1
```

(Sintassi con indirizzo rete e wildcard mask normale)

```
R3(config-router)#no network 172.16.1.32 0.0.0.7 area 0
R3(config-router)#no network 172.16.1.40 0.0.0.7 area 0
R3(config-router)#network 172.16.1.32 0.0.0.7 area 2
R3(config-router)#network 172.16.1.40 0.0.0.7 area 2
```

- Le interfacce fast ethernet di *R1*, *R2* e *R3* vanno impostate come passive; essendo connesse a stub network non serve propagarvi le informazioni
- Per rendere effettive le modifiche si copia `running-config` in `startup-config` e col comando `reload` si riavviano i router
- R1# show ip protocols*: *R1* è contrassegnato come ASBR, il protocollo l'ha rilevato automaticamente perché *R1* è impostato per diffondere la rotta di default esterna
- Analogamente *R2* e *R3* sono contrassegnati come ABR perché hanno interfacce su più di un'area; per ogni area viene specificato il numero di interfacce, di LSA, di esecuzioni dell'algoritmo SPF, ecc.
- Il comando `show ip ospf border-routers` mostra l'elenco degli ABR e ASBR (escludendo quello su cui viene eseguito il comando) indicando se sono raggiungibili con rotta intra-area (i) o inter-area (I)

```
R1#show ip ospf border-routers
  i 2.2.2.2 [50] via 192.168.10.2, Serial0/0/0, ABR, Area 0, SPF 50
  i 3.3.3.3 [50] via 192.168.10.6, Serial0/0/1, ABR, Area 0, SPF 50
```

```
R2#show ip ospf border-routers
  i 1.1.1.1 [50] via 192.168.10.1, Serial0/0/0, ASBR, Area 0, SPF 50
  i 3.3.3.3 [50] via 192.168.10.10, Serial0/0/1, ABR, Area 0, SPF 50
```

Database OSPF

1. Router Link States (**Area 0**) – Type 1

- a. LSA generato da *R1*: 5 link
 - i. 2 router adiacenti
 - ii. 3 reti di destinazione (172.16.1.16/28, 192.168.10.0/30, 192.168.10.4/30)
- b. LSA generato da *R2*: 4 link
 - i. 2 router adiacenti
 - ii. 2 reti di destinazione (192.168.10.0/30, 192.168.10.8/30)
- c. LSA generato da *R3*: 4 link
 - i. 2 router adiacenti
 - ii. 2 reti di destinazione (192.168.10.4/30, 192.168.10.8/30)

2. Summary Net Link States (**Area 0**) – Type 3

- a. 2 LSA generati da *R2* per comunicare come raggiungere le destinazioni nell'area 1 (10.10.0.0 e 10.10.1.0)
- b. 2 LSA generati da *R3* per comunicare come raggiungere le destinazioni nell'area 2 (172.16.1.32 e 172.16.1.40)

3. Router Link States (**Area 1**) – Type 1

- a. LSA generato da *R2*: 2 link (10.10.0.0/24 e 10.10.1.0/24), anche se non viene diffuso nell'area 1 perché le interfacce fast ethernet sono passive, viene comunque generato e memorizzato nel database

4. Summary Net Link States (**Area 1**) – Type 3

- a. 6 LSA generati da *R2* che manderebbe nell'area 1 (se le interfacce non fossero passive) per informare i router dell'area 1 delle destinazioni presenti nelle aree 0 e 2 e raggiungibili tramite *R2* (le stub network connesse a *R1* e *R3* e i 3 link seriali)

5. Summary ASB Link States (**Area 1**) – Type 4

- a. Generato da *R2* per comunicare che sa come raggiungere l'ASBR (*R1*); questo servirebbe a chi si trova nell'area 1 per sapere come arrivare all'ASBR in modo da inviare pacchetti destinati all'esterno dell'AS
- b. Nell'area 0 non c'è l'LSA di tipo 4 perché i router conoscono direttamente l'ASBR: negli LSA di tipo 1 i **flag di options** identificano il router che ha generato l'LSA come ABR e/o ASBR – quindi *R2* e *R3* sanno che *R1* è un ASBR.

6. AS External Link States – Type 5

- a. Generato da *R1* per comunicare ai router che può inoltrare pacchetti verso l'esterno dell'AS (viene diffuso in tutto l'AS a prescindere dalle aree)

Si possono esaminare gli LSA nel dettaglio con il comando `show ip ospf database summary`

Route Summarization

Nell'area 1 ci sono le due reti **10.10.0.0/24** e **10.10.1.0/24** che potrebbero essere **riassunte** con un indirizzo **10.10.0.0/23**, analogamente si può fare per le reti dell'area 2 con l'indirizzo **172.16.1.32/28**.

Può essere utile per ridurre il numero degli LSA di tipo 3 scambiati dagli ABR; ad esempio, *R2* ne manderebbe uno solo per l'area 1 invece che 2, lo stesso vale per l'ASBR che può fare un **summary** delle **rotte esterne** per ridurre il numero di LSA di tipo 5

A differenza del protocollo RIP la **route summarization** non è il comportamento predefinito, ma va attivato con il comando:

```
R(config-router)#area [area-id] range [prefisso riassuntivo con maschera]
```

- R3(config-router)#area 2 range 172.16.1.32 255.255.255.240:
R3 farà un summary verso l'area 0 mettendo insieme gli indirizzi delle due LAN per mandare un solo LSA di tipo 3 per le reti 172.16.1.40/29 e 172.16.1.32/29
- Il router guarda gli LSA di tipo 3 che dovrebbe mandare nell'area 0 dove per ciascuno annuncia una rete di destinazione, se questa rete fa match con il prefisso smette di mandare quell'LSA, quindi manda un solo LSA di tipo 3 con quel prefisso di destinazione.
- Nel caso in cui le reti riassunte dal prefisso abbiano costi diversi per essere raggiunte, si mette nell'LSA il **costo minore** tra quelli presenti
 - Con la **summarization** si guadagna in termini di overhead, ma si perdono informazioni riguardo ai costi
- Nel database ospf ci sarà quindi un solo LSA di tipo 3 generato da *R3* con rete di destinazione 172.16.1.32 e maschera /28, analogamente nelle tabelle di routing di *R1* e *R2* ci sarà una sola entrata e non più due per le LAN connesse a *R3*

Stub areas

Un'area diversa dalla backbone è un'**area di transito** (arrivano pacchetti da un'altra area e destinati a una terza area) **solo se ha un ASBR** al suo interno.

Una **stub area** può essere ottimizzata:

- se il pacchetto ha sorgente e destinazione dentro l'area lo gestiscono i router interni
- se il pacchetto ha sorgente o destinazione fuori dall'area
 - **sorgente esterna:** quando i pacchetti arrivano agli ABR questo sa come inoltrarli a destinazione
 - **destinazione esterna:** il pacchetto deve passare per forza dall'ABR che diventa una specie di gateway dell'area. Si può **ottimizzare** dicendo all'ABR di smettere di propagare gli LSA di tipo 3 e tipo 4 e mandare invece un unico LSA di tipo 3 (*caso estremo di una summary*) per una default route; per ogni destinazione non interna all'area il pacchetto deve essere inviato all'ABR
 - ✖ Aumento overhead: pacchetti per destinazioni inesistenti vengono comunque inoltrati all'ABR prima di essere gettati
 - ✓ Si riduce il numero di LSA di tipo 3,4 e 5 che devono essere propagatiAnche questo va configurato esplicitamente.

SWITCH

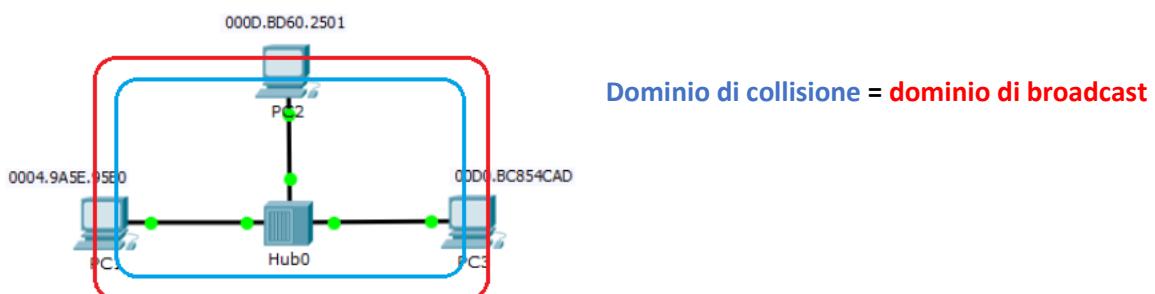
LAN Switching

Una rete locale viene sempre vista dal **livello 3** come uno spazio in cui la comunicazione tra i vari host è diretta - non c'è bisogno di un router per farli comunicare - quindi gli assegna un solo indirizzo di rete.

Il passaggio importante a livello di infrastruttura (**livello 2**) è stato quello da apparati in cui il **dominio di collisione** (*livello 1 – fisico*) coincideva col **dominio di broadcast** (*livello 2 – data link*), ad una realizzazione in cui il **dominio di collisione diventa indipendente** dal dominio di broadcast.

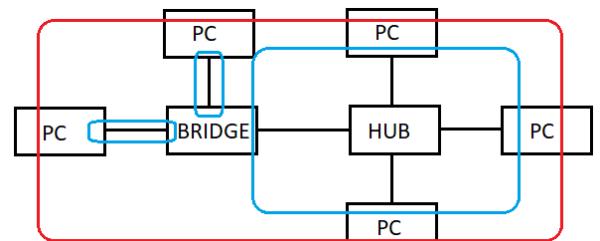
Questo passaggio è anche legato all'evoluzione del mezzo fisico usato per trasmettere trame ethernet che in origine era il **cavo coassiale**, “condiviso” per definizione: il segnale trasmesso sul mezzo viene ricevuto su tutte le interfacce connesse.

1. LAN con **hub**: dispositivi che ritrasmettono i bit ricevuti su una interfaccia su tutte le altre, in modo da estendere il range di trasmissione. Il dominio di collisione comprende tutta la rete ed è allineato al dominio di broadcast, perché un frame inviato su un mezzo fisico viene ricevuto da tutti gli altri host.

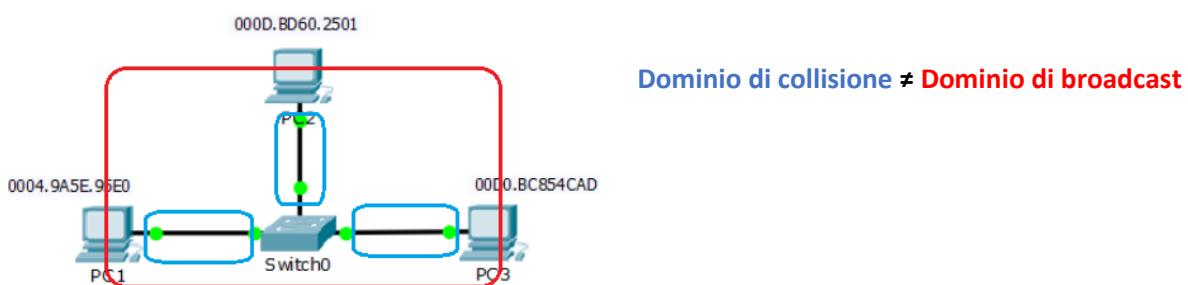


2. LAN con **bridge** (dispositivi intermedi): funzionano come switch, cioè da una interfacci all'altra non trasmettono più a livello fisico i singoli bit, ma ricevono intere trame e le ritrasmettono in maniera selettiva.

Iniziano a fare una separazione tra domini di collisione e domini di broadcast a livello intermedio; su ciascuna interfaccia c'è un dominio di collisione mentre tra interfacce i domini di collisione sono separati. Il bridge **distingue tre dominii di collisione**, ma continua a realizzare un **unico dominio di broadcast**, quindi per il livello IP non è cambiato niente e i PC devono continuare a usare CSMA/CD.



3. **Switched ethernet**: per definizione **tutti i link sono punto-punto**. La comunicazione a livello fisico è sempre punto-punto e il dominio di broadcast è stato realizzato dagli switch, ricostruito virtualmente su una collezione di dominii di collisione separati.

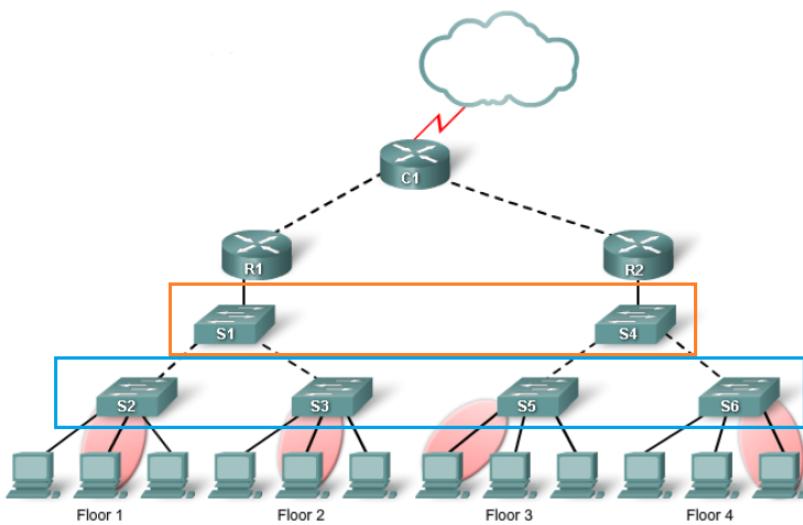


Gli switch fanno la commutazione di frame (*routing a livello 2*), ma nel farlo assumono che esista un **percorso unico all'interno della rete per raggiungere qualsiasi destinazione**. Con questa ipotesi il routing diventa più semplice: gli switch sono **plug and play** e scoprono da soli le destinazioni dopo aver cercato ed eliminato eventuali percorsi chiusi. La complessità, a differenza dal routing IP, è nascosta perché gestita automaticamente dai protocolli.

Con gli hub i pacchetti con indirizzo **unicast** vengono comunque consegnati a tutti gli host, mentre con gli switch vengono consegnati ai singoli destinatari; il **dominio di collisione viene praticamente annullato** perché **ridotto a un collegamento punto-punto**.

I PC non devono più usare CSMA/CD (quindi viene eliminato il relativo overhead) per comunicare con gli switch, la comunicazione è full duplex (2 link separati bidirezionali – su ciascun link c'è un unico trasmettitore).

A livello pratico gli switch sono **differenti in base ai volumi** che devono gestire, ad esempio:



- **switch di accesso:** connessi direttamente agli host, hanno più porte e devono fare controlli di accesso per la sicurezza
- **switch “superiori”:** hanno meno porte, ma sono più veloci poiché devono smaltire più traffico

MAC address table

Gli switch fanno routing a livello 2, hanno quindi una tabella di forwarding per scoprire a quale porta trasmettere i frame in arrivo.

Ogni entrata contiene:

- Indirizzo MAC di destinazione (48 bit) – *tre gruppi da 4 cifre esadecimale*
- Id della porta da utilizzare per raggiungere la destinazione
- Id della VLAN alla quale appartiene l'indirizzo (nel caso più semplice ce n'è solo una in tutta la rete)
- Tipo: indica se l'informazione è stata acquisita staticamente o dinamicamente
- Spanning Tree Protocol (STP): indica se la porta si può usare o meno, perché vengono tagliati i percorsi chiusi
- Aging: le informazioni devono essere rinfrescate altrimenti dopo un po' vengono cancellate.

Switched Ethernet non prevede protocolli di routing a livello 2, ma assume che, non essendoci percorsi chiusi nella rete, ci sia sempre un solo percorso possibile per ogni coppia sorgente-destinazione.

Quando lo switch riceve un frame:

1. **Forwarding:** guarda nella MAC address table se conosce la porta da usare per quella destinazione, se non la conosce fa **flooding** su tutte le porte tranne quella su cui ha appena ricevuto il frame.
 - a. Il flooding è possibile grazie all'assenza di percorsi chiusi, se questi infatti fossero presenti il flooding rischierebbe di intasare la rete di pacchetti che circolerebbero all'infinito
2. **Learning:** guarda l'indirizzo sorgente e “*capisce*” che per raggiungere quell'indirizzo dovrà usare la porta su cui ha ricevuto il frame.
 - a. Controlla se è già presente nella tabella, altrimenti la aggiorna: le tabelle degli indirizzi MAC vengono **popolate andando a guardare gli indirizzi sorgente dei frame stessi**.
 - b. Ogni pacchetto ricevuto da una certa interfaccia con un certo indirizzo è una indicazione senza ambiguità che la porta su cui viene ricevuto è quella da usare per mandare frame a quella interfaccia.

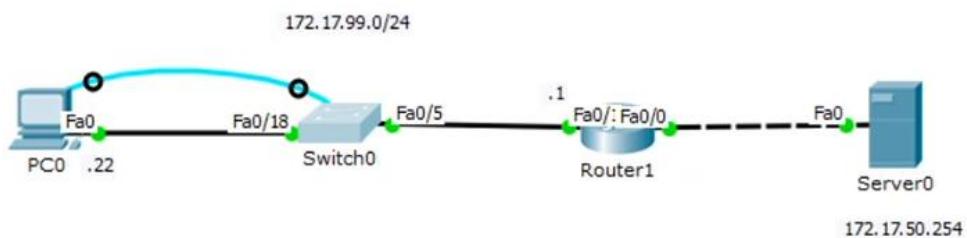
Configurazione

Come i router, gli switch sono macchine con un sistema operativo (nei dispositivi CISCO è sempre IOS):

- Il set dei comandi è diverso, visto che alcuni comandi hanno senso su switch ma non su router e viceversa
- La parte relativa ai contesti, ai modi di configurazione, ecc. è uguale a quanto visto per i router.
- Per la gestione del singolo dispositivo vale quanto visto per i router: si usa l'interfaccia a linea di comando a cui si accede tramite una porta di console e un cavo seriale.

Fisicamente gli switch sono diversi dai router principalmente per il numero di porte che è nell'ordine delle decine di unità.

Rete d'esempio:



Nella rete sono stati configurati gli indirizzi delle interfacce del PC, del router e del server, ma per lo switch non servono configurazioni: essendo un dispositivo **plug and play**, può subito iniziare a scambiare frame.

Su uno switch non ha senso assegnare indirizzi IP alle porte:

- Nei router le porte operano a **livello 3**; quando arriva un frame il router controlla se l'indirizzo MAC del frame **coincide con quello della sua interfaccia**, se si “scarta” il frame e prende il pacchetto IP per gestirlo.
- Le **porte dello switch operano a livello 2**: quando arriva un frame lo switch non controlla se l'indirizzo MAC coincide con quello della porta, perché solitamente riceve frame che vanno inoltrati agli host. Inoltre, non avrebbe senso avere un IP diverso per ogni porta dello switch perché non si deve fare routing da una porta all'altra.
- Assegneremo quindi degli indirizzi IP agli switch, non alle singole porte ma al dispositivo

Si accede all'interfaccia a linea di comando dello switch tramite la porta di console, usando il terminale da PC0:

- Messaggi di controllo: lo switch si è accorto che sulle porte 18 e 5 c'è connesso qualcosa tramite i protocolli a livello fisico usati per negoziare i parametri della comunicazione
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/18, changed state to up
%LINEPROTO-5-UPDOWN: line protocol on Interface FastEthernet0/5, changed state to up
 - **Sw1# show mac-address-table**

| Vlan | Mac Address | Type | Ports |
|------|----------------|---------|-------|
| 1 | 0003.e4ea.0b02 | DYNAMIC | Fa0/5 |
- Il router ha trasmesso un frame sulla porta *fa0/1*, lo switch lo ha ricevuto sulla porta *fa0/5*, ha guardato il sorgente e l'ha segnato nella tabella come destinazione raggiungibile tramite quella porta

- Sw1(config)# interface fa0/5
- Sw1(config-if)# duplex [auto | full | half]: il comportamento predefinito è auto, lo switch si mette d'accordo con chi sta dall'altra parte tramite protocolli standard di ethernet che operano a livello fisico.
- Sw1(config-if)# speed: si può controllare la velocità della porta
- Sw1(config-if)# mdix: MDI (media dependant interface) è il meccanismo che tra le altre cose configura su quale coppia di collegamenti trasmettere e su quale ricevere. In modalità automatica qualsiasi cavo si usa e chiunque ci sia dall'altra, parte lo switch si autoregola per permettere la comunicazione.
- Sw1# show interface fa0/5: mostra lo stato della porta fastEthernet0/5:
 - È indicato l'indirizzo MAC della porta - 00d0.ffc4.4805
 - La velocità nominale - BW 100000 Kbit
 - Il valore di velocità configurato - Full-duplex, 100Mb/s
- Se faccio ping da PC0 a router nella tabella di indirizzi MAC dello switch comparirà anche l'indirizzo della porta di PC0 (00d0.baed.1acb); gli host hanno comunicato tra loro senza dover fare niente sullo switch e, contemporaneamente, lo switch ha aggiornato la propria tabella degli indirizzi MAC.
- Anche nello switch ci sono i file *running-config* e *startup-config*, il primo si trova sempre in RAM e il secondo in RAM non volatile.
- Sw1(config-line) #transport input [all | none | ssh | telnet]: definisce i protocolli ammessi, il valore predefinito è *all*

Assegnamento IP allo switch

Si assegna l'indirizzo IP all'apparato come se fosse un host e l'indirizzo verrà usato per scambiare pacchetti con l'apparato a livello 3; nello switch c'è una **switch virtual interface (SVI)** **associata alla VLAN 1** (predefinita), quindi l'indirizzo IP lo assegno all'interfaccia associata alla VLAN1.

- Sw(config)# interface vlan1
- Sw(config-if)# ip address 172.17.99.2 255.255.255.0; è come se ci fosse un' interfaccia interna allo switch a cui è stato assegnato l'indirizzo IP e che va attivata col comando no shutdown
- Si assegna allo switch anche un gateway (stiamo ragionando come se fosse un host):

Sw1(config)# ip default-gateway 172.17.99.1

Adesso lo switch è un oggetto raggiungibile tramite IP, ad esempio da PC si può accedere all'interfaccia a linea di comando facendo telnet 172.17.99.2

Avere un indirizzo IP permette allo switch di poter fare operazioni come copiare sul server il file *startup-config*; TFTP usa UDP e UDP usa IP, quindi se lo switch non avesse IP non potrebbe comunicare col server

Configurazione di sicurezza

Gli aspetti di sicurezza sono importanti per gli switch di accesso, cioè quelli ai quali vengono connessi gli host, sugli switch di backbone queste funzionalità non sono necessarie perché tutte le sue porte sono connesse ad altri dispositivi *affidabili*, quindi non serve fare controlli che riducono l'efficienza dell'apparato.

Attacchi MAC address flooding

La memoria dove si trova la tabella degli indirizzi MAC ha dimensione limitata; la dimensione massima della tabella (numero massimo di indirizzi memorizzabili contemporaneamente) è una **proprietà hardware**. Uno switch deve poter memorizzare almeno un numero di indirizzi pari agli indirizzi MAC delle **interfacce connesse alla stessa LAN**; questo valore dipende dal numero di host connessi (solitamente 200-300).

Scegliendo ad esempio *8100 righe* per la tabella MAC dello switch ho la garanzia che, se tutti gli host si comportano correttamente, questa non verrà mai saturata; al massimo possono esserci dei picchi quando, ad esempio, si connettono molti nuovi utenti a un access point e non si è ancora cancellato quelli vecchi, per questo si prende un numero abbastanza maggiore del numero di host previsti.

Se un utente malintenzionato si collega allo switch e **generare frame a frequenza elevata cambiando ogni volta l'indirizzo MAC sorgente**, la tabella MAC viene **saturata**.

Quando la tabella è saturata e continuano ad arrivare frame con indirizzi MAC nuovi, lo switch entra in modo operativo **"fail open"**: per gestire queste situazioni di emergenza viene **"degradato"** ad un hub quindi **smette di fare learning e fa flooding di tutti i frame che riceve su tutte le altre porte**.

L'utente malintenzionato potrebbe quindi **forzare la modalità fail open** per ricevere **tutto il traffico scambiato nella rete**.

Soluzione: imporre un **limite di indirizzi MAC diversi** che lo switch può vedere su una singola porta, dopo aver memorizzato x indirizzi sulla stessa porta se ne vede altri non li memorizza nella tabella.
(Se scegliersimo di memorizzare un solo indirizzo per ogni porta si potrebbe avere una limitazione, ad esempio nel caso in cui si volesse sostituire un computer guasto sulla rete)

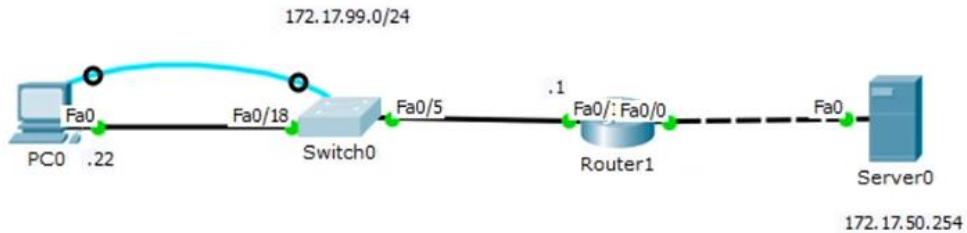
1. Individuazione degli indirizzi MAC (*numero massimo di indirizzi = 1*)

- a. **Statica**: si configurano manualmente le informazioni sull'indirizzo MAC relativo a una determinata porta e si blocca il learning su quest'ultima
 - **Svantaggio**: vanno impostati manualmente gli indirizzi
- b. **Dinamica** (comportamento predefinito): learning dinamico: dove l'indirizzo valido è quello del mittente del primo frame che transita per lo switch
 - **Svantaggio**: la memorizzazione è *volatile*, viene resettata al riavvio del dispositivo
- c. **Sticky** (modalità intermedia): apprendimento dinamico dell'indirizzo che viene memorizzato nel file di configurazione, questo può quindi essere salvato nel file di startup e mantenuto al riavvio
 - È necessario assicurarsi che gli host vengano accesi la prima volta in modo controllato

2. Comportamento in caso di violazione: se arriva un frame con un indirizzo sorgente non configurato per quella porta

- a. **Shutdown** (comportamento predefinito): la porta viene spenta
- b. **Restrict**: la porta non viene spenta ma viene inviato un messaggio di avviso
- c. **Protect**: non vengono presi provvedimenti

[Packet Tracer]



- `Sw(config)#interface fa0/18`
- `Sw(config-if)#switchport mode access`: configura le modalità di switching della porta
- `Sw(config-if)#switchport port-security maximum [1-132]`: imposta il numero massimo di indirizzi MAC diversi ammessi sulla porta (1 è il valore predefinito quando si abilita la port security)
- `Sw(config-if)#switchport port-security mac-address [H.H.H | sticky]`
 - H.H.H specifica l'indirizzo mac
 - sticky configura la memorizzazione dinamica come sticky e quindi memorizza l'indirizzo MAC in running-config
- `Sw1# show port-security interface [interface]`: mostra informazioni sullo stato della port security per quella interfaccia:
 - Port security [enabled | disabled]
 - Port Status
 - Violation mode
 - Maximum MAC Addresses: numero massimo di indirizzi validi per quella porta
 - Total MAC Addresses: numero di indirizzi appresi al momento
 - Configured MAC Addresses
 - Last Source Address

Port security disattivata:

- Nella tabella MAC dello switch ci sono 2 indirizzi: quello di PC0 (00D0.BAED.1ACB) e quello del router.
- Aggiungiamo alla rete PC1 con IP 172.17.99.33, gateway 172.17.99.1 e MAC 0001.423B.820C
- Se ora porta *fa0/18* dello switch connettiamo PC1 invece che PC0 e facciamo ping da PC1 al router, nella tabella degli indirizzi MAC ci saranno i due indirizzi di PC1 e PC0, entrambi sulla porta *fa0/18*.

1. Modalità Statica

```
Sw(config-if)# switchport port-security mac-address 0001.423B.820C
```

- Attiva la port security e imposta l'indirizzo MAC ammesso per quella porta (quello di PC1, su cui lo switch smetterà di fare learning)
- Il comando eseguito compare in running-config, quindi se adesso salvassimo running-config e riavviassimo lo switch l'informazione rimarrebbe memorizzata
- Nella MAC address table l'indirizzo compare con type **STATIC**
- Se colleghiamo PC0 alla stessa porta e proviamo a fare ping al router, lo switch **rileva una violazione e spegne la porta** (port status: secure shutdown)
- Se collegassimo nuovamente PC1 la porta rimane comunque bloccata, perché deve essere sbloccata manualmente con il relativo comando (si comunica che non c'è una violazione)

2. Modalità dinamica:

```
Sw(config-if)#no switchport port-security mac-address 0001.423B.820C
```

- Salviamo in startup-config e riavviamo lo switch
- Sw1# switchport port-security int fastEthernet0/18:
 - Maximum MAC Addresses: 1
 - Configured MAC Addresses: 0
 - Last Source Address: 0000.0000.0000
- Non essendo più in modalità statica con un indirizzo specificato, l'indirizzo ammesso sarà quello del primo host che invia un frame ethernet
- Se facciamo il ping da PC0 al router lo switch registrerà come indirizzo MAC ammesso quello di PC0, perché è stato il primo che ha visto sulla porta *fa0/18*.

L'indirizzo MAC di PC0 non è stato inserito, ma è stato appreso: se adesso riavviassimo lo switch, collegassimo PC1 e facessimo ping al router sarebbe l'indirizzo di PC1 ad essere inserito nella tabella ed essere l'unico indirizzo associato alla porta fastEthernet0/18. Infatti, la configurazione è dinamica e non viene salvata nel file di configurazione, ogni volta che riavviamo lo switch verrà memorizzato l'indirizzo del primo host che manda un frame.

3. Modalità sticky:

```
Sw1(config-if)# switchport port-security mac-address sticky
```

- Facciamo ping da PC1 al router
- Nel file running-config, anche se non l'abbiamo eseguito è comparso il comando
`switchport port-security mac-address sticky 0001.423B.820C`: l'indirizzo MAC è stato rilevato dallo switch autonomamente
- Se salviamo in startup-config e riavviamo lo switch, questo indirizzo MAC rimane salvato
- Se stavolta è PC0 ad usare la rete per primo, lo switch rileva una violazione e manda in secure shutdown la porta

- ✓ Non abbiamo scritto manualmente l'indirizzo MAC di PC1
- ✓ L'indirizzo rimane salvato anche se al “*secondo giro*” è stato PC0 a usare per primo la rete:
 l'amministratore deve assicurarsi che l'host che deve poter comunicare su quella porta **sia il primo ad usare la rete** solo la prima volta

Configurazione di base:

1. Router:

- a. assegnare IP alle interfacce
- b. configurare l'accesso controllato

2. Switch:

- a. assegnare gli IP – se previsto nella rete
- b. configurare l'accesso tramite credenziali
- c. se lo switch è uno switch di accesso
 - i. configurare la port security

Altri tipi di attacchi su switch:

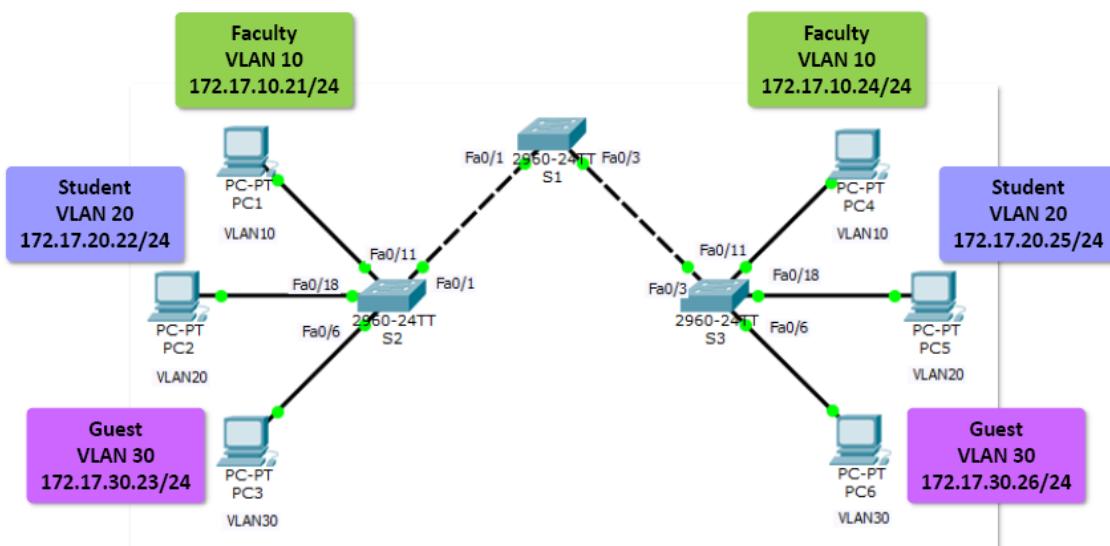
1. **DHCP spoofing:** un host fa partire sulla propria macchina un server DHCP, ogni volta che un host della LAN fa una richiesta DHCP questa raggiungerà anche il server “finto” che dà agli host indirizzo IP di gateway uguale al proprio in modo da poter intercettare tutti i pacchetti che devono uscire dalla rete
 - a. **Protezione:** su ogni porta dello switch di accesso si specifica quali sono quelle su cui pacchetti di risposta DHCP sono ammessi oppure no. Nell'esempio precedente potrei impedire ai pacchetti DHCP di passare sulla porta fa0/18 visto che è quella connessa ad un host.
Per fare questo, però, serve guardare il pacchetto a livello applicativo per vedere se è un pacchetto di risposta DHCP.
2. **DHCP starvation:** un client inizia a chiedere indirizzi IP a raffica per esaurire il pool di indirizzi disponibili sul server
 - a. **Protezione:** si impone un limite al numero di richieste DHCP che possono passare su una singola porta in un certo intervallo di tempo

Virtual LAN

- **Gestione del traffico broadcast:** il traffico broadcast non è una eccezione, ma è costantemente generato – ad esempio da ARP – e diventa un problema di scalabilità man mano che la rete si ingrandisce.
- **Sicurezza:** in una rete tutti possono comunicare con tutti a patto che conoscano l'indirizzo, ma alcune applicazioni in esecuzione su un dato host potrebbero non dover mai comunicare con le applicazioni su altri host e un tentativo di comunicazione potrebbe indicare un problema di sicurezza

La **soluzione** è separare i gruppi di host che non hanno necessità di essere collocati sullo stesso dominio di broadcast, non potendolo fare fisicamente (troppo costoso) lo si fa a livello “virtuale”.

Virtual LAN: meccanismo di astrazione che fa apparire al livello superiore (IP) una rete diversa da quella fisica; si ha una singola infrastruttura fisica con un singolo dominio di broadcast che a livello 3 appare come comporta da tre diverse LAN



Non si può prescindere dall'uso delle VLANs; esiste sempre una VLAN di default (che non si può rimuovere) e inizialmente tutte le porte degli switch appartengono a questa.

Vantaggi:

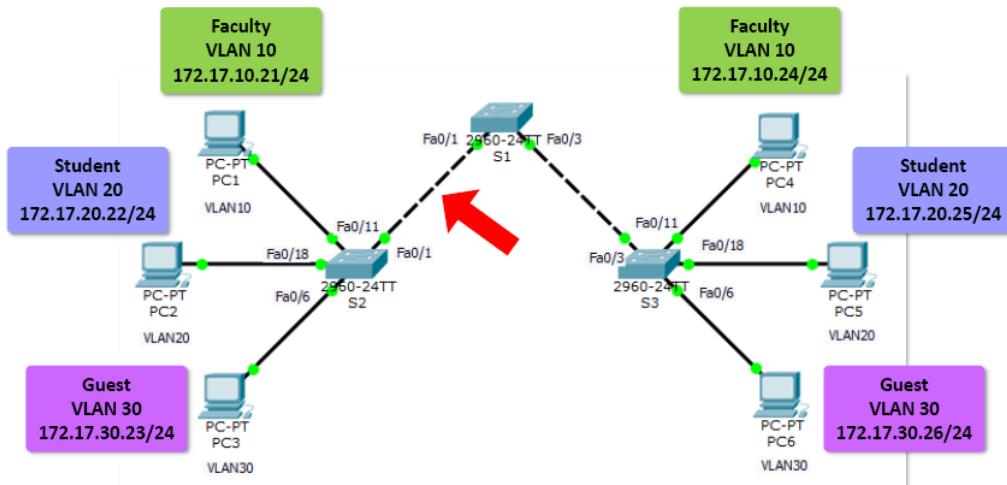
- **Sicurezza**
- **Riduzione costi**
- **Migliori prestazioni:** il traffico broadcast non deve più raggiungere ciascun host della rete, ma solo gli host della stessa VLAN del mittente.
- **Broadcast storm migration:** riduce il problema del broadcast storm

VLAN intra-switch

Lo Switch è la *radice di un albero* e le sue porte venivano raggruppate in diverse VLAN; appartenevano alla stessa VLAN i rami connessi a una determinata porta. Solo in uno switch si potevano separare i domini di broadcast, per i rami valeva l'appartenenza fisica alla rispettiva VLAN

VLAN inter-switch

Un frame potrebbe attraversare più switch e quando arriva a quello finale, questo è in grado di prendere una **decisione VLAN-aware**, cioè che non dipende solo dall'indirizzo di destinazione letto nel frame ma anche dalla **VLAN di appartenenza del mittente**



Esempio:

- PC1 (**VLAN 10**) manda un pacchetto ARP (frame con indirizzo di destinazione broadcast)
- S2 lo riceve sulla porta *Fa0/11* associata alla VLAN 10 (assumiamo che le porte di S2 siano distinte in base alla VLAN). S2 sa che il frame deve essere trasmesso solo sulle porte della stessa VLAN, quindi non trasmette su *Fa0/18* e *Fa0/6*; su *Fa0/1* però c'è un altro switch a valle del quale potrebbero esserci utenti sulla stessa VLAN, quindi manda il frame a S1
- S1 per lo stesso motivo manda il frame a S3
- S3 riceve un frame con indirizzo broadcast, che però deve mandare solo a quegli host che appartengono alla stessa VLAN di PC1; come fa S3 a sapere di quale VLAN faceva parte il mittente?

Soluzione 1 - Frame Filtering: nello switch c'è una tabella dove c'è, per ogni indirizzo MAC, la VLAN di appartenenza. Quando il frame arriva a S3 questi guarda il MAC sorgente e vede a quale VLAN appartiene, quindi manda il pacchetto sulle porte alle quali sono connessi gli host di quella stessa VLAN.

L'appartenenza a una VLAN è basata sugli indirizzi MAC (MAC Address Based)

Soluzione 2 – Frame Tagging: un host viene aggiunto a una VLAN in base a quale porta usa per comunicare. **L'appartenenza a una VLAN è basata sulla porta a cui un host è connesso (Port Based)**

1. Quando S2 riceve un frame da PC1 capisce che è appartenente alla VLAN 10 perché lo riceve sulla porta *Fa0/11* a prescindere dall'indirizzo MAC e lo manda solo sulla porta *Fa0/1*
2. Quando il frame arriva a S3 questo **trova l'informazione sulla VLAN nel frame stesso**

S2 quindi deve scrivere a quale VLAN appartiene il frame all'interno del frame stesso prima di mandarlo:

- a. Incapsulamento Multiplo: utilizzato in passato da Cisco
- b. **Header aggiuntivo:** si usa una versione estesa dello header ethernet che ha un campo per il numero identificativo della VLAN

Frame filtering:

- ✓ Supporto della mobilità: l'appartenenza una VLAN è una proprietà dell'interfaccia, quando un host si sposta all'interno della rete rimane nella VLAN
- ✗ Gestione delle tabelle: ogni switch deve avere una sua tabella

Frame Tagging:

- ✓ Favorisce la scalabilità: la configurazione della VLAN avviene sulla singola porta dello switch , se avessi ad esempio un hub configurando una sola porta avrei assegnato la VLAN a più host
- ✗ No supporto mobilità

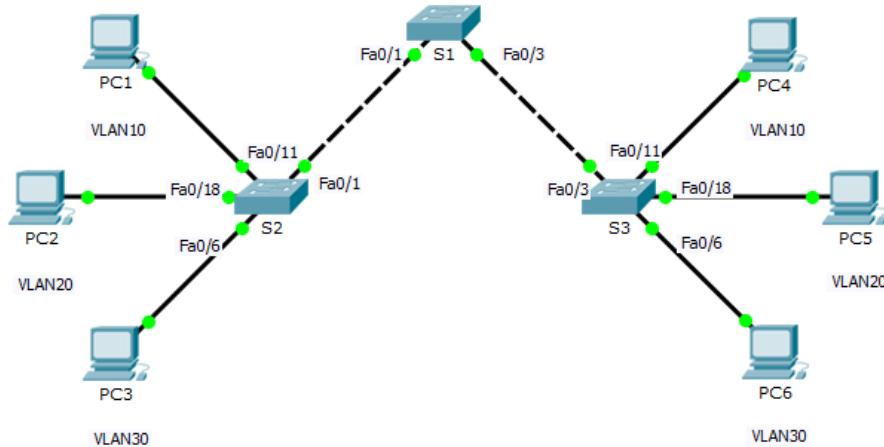
Formato dei pacchetti tagged

| |
|-------------------------|
| Destination address |
| Source Address |
| Length/Type = TPID |
| Tag Control Information |
| Client Length/Type |
| MAC Client Data |
| PAD |
| FCS |

Campi aggiuntivi:

1. **Lenght/Type:** se ha un valore diverso dal campo client lenght/type significa che nel campo successivo c'è il tag
2. **Tag Control Information (16bit):**
 - a. (12b): VLAN di appartenenza, fino a 4093 VLAN (0, 1, 4095 sono riservati)
 - b. (3b): priorità
 - c. (1b): importanza in caso di congestione

[Packet Tracer]

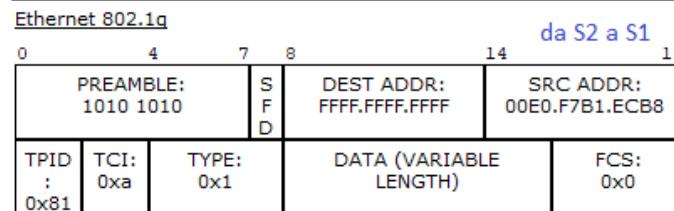
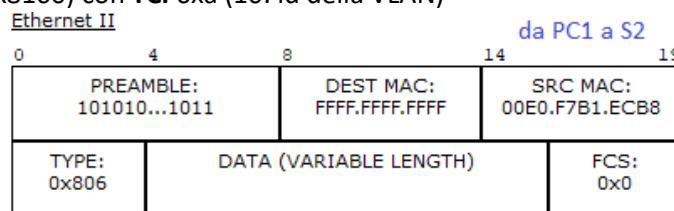


- Come si assegnano indirizzi IP separati a LAN distinte, lo stesso faremo per gli host appartenenti a diverse VLAN perché dal livello superiore devono essere reti diverse con prefissi diversi:

 - PC4 (VLAN 10): 172.17.10.24/24
 - PC5 (VLAN 20): 172.17.20.25/24

A livello IP, PC4 e PC5 non possono comunicare perché non c'è un router e si trovano su reti diverse: le **VLAN sono isolate**

- PC1: ping 172.17.10.24 (PC4)
 - PC1 ha indirizzo 172.17.10.21/24, quindi ha lo stesso prefisso di PC4
 - PC1 vede che PC4 è un host sullo stesso link, quindi manda un pacchetto ARP per scoprire il suo indirizzo MAC:
 - Dest MAC: FFFF.FFFF.FFFF (broadcast)
 - Src MAC: 00E0.F7B1.ECB8 (PC1)
 - PC1 manda il pacchetto su S2
 - S2 trasmette su tutte le porte (tranne *fa0/11*) tranne quelle che sono configurate come appartenenti ad altre VLAN.
 - *fa0/18* è configurata come appartenente alla VLAN 20, *fa0/6* alla VLAN 30
 - *fa0/1* non appartiene a una VLAN particolare, quindi S2 invia il pacchetto
- Da S2 a S1 il frame non è più un frame ethernet normale (type 0x806 – ARP), ma un **frame ethernet 802.1Q** (type 0x8100) con **TCI 0xa** (10: id della VLAN)



- S2 lo manda a S3 e questo guarda le sue porte: *fa0/6* e *fa0/18* non sono della stessa VLAN, quindi manda il pacchetto solo su *fa0/11* perché della VLAN 10 (valore scritto nel frame ethernet).

802.1Q – Device type

- **VLAN-aware** (device conformi allo standard 801.1Q) : gestiscono sia frame con tag che senza
- **VLAN-unaware**: gestiscono solo frame senza tag

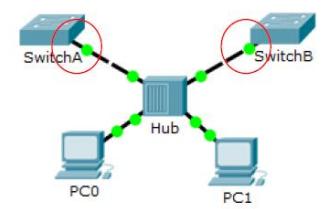
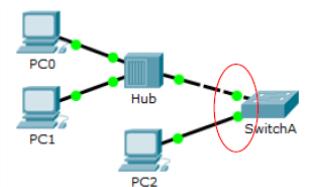
I dispositivi dei due tipi possono coesistere nella stessa rete.

Questo riguarda anche le singole interfacce di rete di un host che possono essere aware o unaware; lo standard non impone che i frame taggati possano essere scambiati solo tra switch.

Nella rete di esempio PC3 può mandare un frame a S2 con dentro un tag - in generale questo sarà utile quando si inseriranno router che possono appartenere a più VLAN contemporaneamente

802.1Q – Port and link types

- **Access port/Access link:** su questi viaggiano frame senza tag, perché **l'appartenenza alla VLAN è determinata dalla porta e dal link stesso**; lo switch sa a che VLAN appartiene il frame in base a dove lo riceve.
- **Trunk port/Trunk link:** i frame devono avere un tag.
È il caso - nell'esempio precedente – della porta fa0/1 di S1: S2 deve aggiunger il tag ai frame che invia su quella porta altrimenti gli switch "dopo" non possono conoscere la VLAN del frame
- **Hybrid port/Hybrid link:** sono permessi sia tagged che untagged frame. Nell'esempio a fianco i frame inviati e ricevuti dagli switch devono avere il tag, ma quelli inviati e ricevuti agli host non hanno bisogno di tag perché il sistema è port-based
 - Se manca il tag sui frame che vanno su trunk port/trunk link, questi vengono considerati appartenenti alla **native VLAN** di quel link



VLAN id ranges:

Cisco IOS distingue due tipi di VLAN:

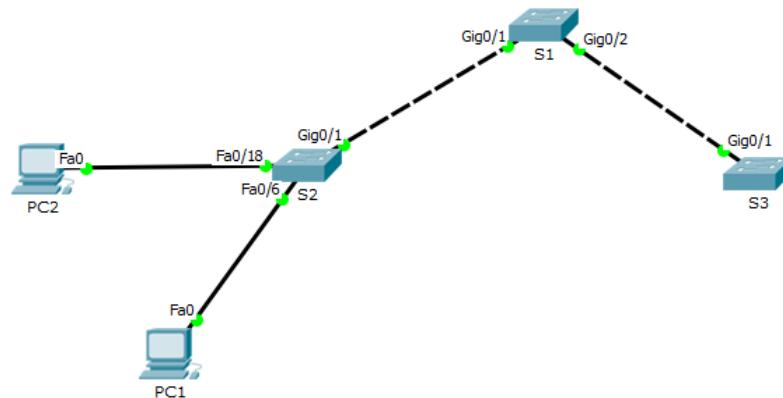
- **Normal Range VLAN:** ID compreso tra 1 e 1005
- Informazioni di configurazione sul file vlan.dat in memoria persistente (a differenza di saved-config che è nella RAM non volatile)
- **Extended Range VLAN:** ID tra 1006 e 4094

Alcune operazioni (che non hanno a che fare con le funzionalità di base) sono possibili solo sulle normal range VLAN.

Special VLAN types

1. **Default VLAN:** già definita quando si accende lo switch, tutte le porte sono assegnate a questa
2. **Native VLAN:** può essere configurata per ogni porta, se la porta è ibrida è la VLAN a cui vengono attribuiti i frame che arrivano senza tag
3. **Management VLAN** (non relativa allo standard ma alla security best practice anche se non si prevede la necessità di usare più VLAN bisognerebbe creare sempre una VLAN separata usata esclusivamente per il management, nessun host eccetto l'amministratore della rete deve appartenere a essa, ma ci appartengono tutti gli switch in quanto apparati. Viene usata per comunicare con gli switch a livello IP e avrà come prefisso di rete quello che usato per gli IP degli switch.

[Packet Tracer]



- **S2#show vlan brief** - Mostra l'elenco delle VLAN definite con id, nome e porte associate, all'inizio c'è solo la VLAN 1 (*default*) e tutte le porte dello switch sono assegnate questa
- **S2(config)#vlan [vlan-id]** - Entra nel modo di configurazione della vlan, se questa non è definita viene creata
- **S2(config-vlan)#name [vlan-name]** - Assegna un nome alla VLAN per facilitarne l'identificazione da parte dell'amministratore

```

S2(config)#vlan 10
S2(config-vlan)#name Faculty
S2(config-vlan)#vlan 20
S2(config-vlan)#name Student
S2(config-vlan)#vlan 30
S2(config-vlan)#name Guest

```

Per assegnare le porte alle VLAN si deve entrare nel *modo di configurazione della porta*, poiché l'appartenenza a una VLAN è una proprietà della porta stessa

Con il comando **interface range x - y** i comandi eseguiti verranno applicati a tutte le porte del range indicato

- **S2(config-if-range)# switchport mode [access|dynamic|trunk]** - Imposta il tipo di porta (per default ibrida)
- **S2(config-if-range)#switchport access vlan [id]** - Assegna le interfacce alla VLAN specificata

```

S2(config)#interface range fa0/18 - fa0/24
S2(config-if-range)#switchport mode access
S2(config-if-range)#switchport access vlan 20
S2(config-if-range)#interface range fa0/1 - fa0/8
S2(config-if-range)#switchport access vlan 30
S2(config-if-range)#interface range fa0/9 - fa0/17
S2(config-if-range)#switchport access vlan 10

```

Le porte che non trunk devono essere assegnate a una certa VLAN, altrimenti non sanno cosa fare con i frame che ricevono visto che lavorano con frame untagged

Nota: se dopo aver assegnato le porte a una VLAN questa venisse eliminata le porte rimarrebbero "appese"

- **S2# show VLAN id [id]**
S2# show VLANS name [name] - Mostrano le informazioni sulla VLAN specificata
- **S2# show interfaces fa0/18 switchport** - Mostra lo stato di questa interfaccia in quanto porta ethernet con diverse informazioni, tra cui l'Administrative mode della porta e la VLAN associata

Quando aggiungo un secondo switch S1 da connettere a S2, le porte su cui comunicano i due devono essere **trunk** perché sopra ci viaggiano frame di VLAN diverse.

```
// Aggiungo una vlan management
S2(config)#vlan 99
S2(config-vlan)#name Management

// Imposto la porta gi0/1 di S2 come trunk e imposto la VLAN nativa 99
S2(config)#int gigabitEthernet0/1
S2(config-if)#switchport mode trunk
S2(config-if)#switchport trunk native vlan 99
```

Se S2 riceverà un frame senza tag sulla porta *gi0/1*, questo verrà automaticamente associato alla VLAN 99. In questo modo i frame della VLAN 99 viaggiano su questa porta senza tag (risparmio spazio).

Se su S1 metto una VLAN nativa diversa, i frame della VLAN 99 che S2 manda senza tag S1 li interpreterebbe come appartenenti ad un'altra VLAN. Bisogna garantire la consistenza

```
S1(config)#int gigabitEthernet0/1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 99
```

Su S1 devo anche definire tutte le VLAN che ho definito su S2; **si devono sempre dichiarare le VLAN anche sugli switch di passaggio.**

(Nota: per comodità su packet tracer si può usare la clonazione dei dispositivi, nella realtà dovrei caricare su un server il file di configurazione di S2, scaricarlo da S1 e poi salvarlo in *startup-config*)

```
S2#show interfaces gigabitEthernet0/1 switchport
  Administrative mode: trunk
  Administrative Trunking Encapsulation: dot1q
  // dt1q = 802.1Q - i tag vengono aggiunti secondo questo standard; va specificato perché Cisco ha anche un metodo di encapsulation proprietario
  Trunking Native Mode VLAN: 99 (Management)
  Trunking VLANs Enabled: ALL
```

Trunking VLANs Enabled: indica quali sono le VLAN per cui lo switch può fare inoltro di frame su quella porta (default ALL). Se si ha una rete con più VLAN si potrebbe voler mettere dei vincoli su quali VLAN possono o non possono passare su un link trunk, magari perché sappiamo che c'è una separazione ed è inutile far proseguire dei frame di una certa VLAN se da quel punto in poi non c'è nessuno che li può ricevere.

```
S2#show interfaces trunk - Mostra informazioni su tutte le interfacce trunk
Port      Mode   Encapsulation        Status       Native vlan
Gig0/1    on     802.1q                trunking    99

// VLAN potenzialmente ammesse
Port Vlans allowed on trunk
Gig0/1 1-1005

// VLAN ammesse e attive
Port Vlans allowed and active in management domain
Gig0/1 1,10,20,30,99

Port Vlans in spanning tree forwarding state and not pruned
Gig0/1 1,10,20,30,99
```

Per rimuovere delle porte dall'elenco delle VLAN allowed su una certa porta:
S2(config-if)#switchport trunk allowed vlan except [vlan id]

Gestione del Trunking

Cisco Dynamic Trunking Protocol (CDTP): gestisce la trunk negotiation per il setup di un link trunk, si può disabilitare con switchport nonegotiate

Trunking mode (switchport mode [mode])

- **Access**: porta di accesso
- **Trunk**: porta trunk
- **Dynamic Auto** (predefinito): la porta non è trunk, ma si può negoziare. Se l'altro chiede di diventare trunk allora la porta diventa trunk
- **Dynamic Desirable**: la porta non è trunk, ma il dispositivo vorrebbe che lo fosse

Essendo dynamic auto il modo predefinito quando collego due switch entrambi possono diventare trunk, ma finché nessuno lo chiede la porta non lo diventa.

Se configuro trunk la porta di uno switch e lo connetto a una porta dell'altro switch (default dynamic auto) avendo forzato trunk da una parte lo diventa anche dall'altra.

| | Dynamic Auto | Dynamic Desirable | Trunk | Access |
|-------------------|--------------|-------------------|-----------------|-----------------|
| Dynamic Auto | Access | Trunk | Trunk | Access |
| Dynamic Desirable | Trunk | Trunk | Trunk | Access |
| Trunk | Trunk | Trunk | Trunk | Not Recommended |
| Access | Access | Access | Not Recommended | Access |

Trubleshooting (errori comuni):

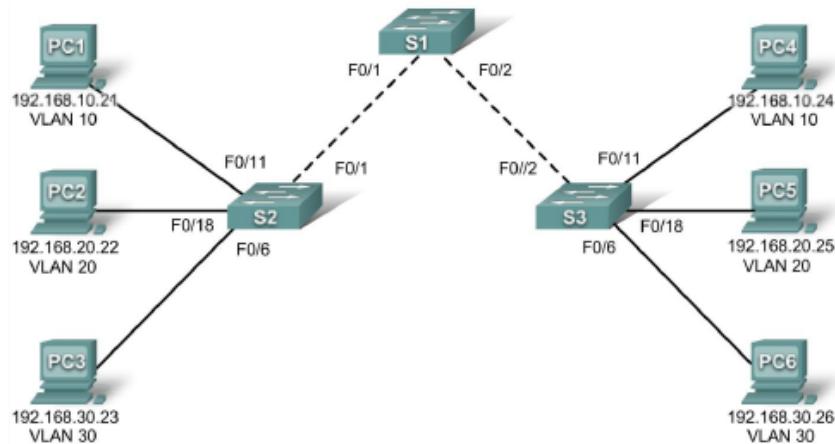
1. Mancanza di consistenza tra indirizzi IP e VLAN
2. LAN Native mismatched
3. Trunk mode mismatched
4. VLAN ammesse sbagliate o mancanti

Automazione della configurazione

Configurazione **dinamica**: gli switch si scambiano informazioni di configurazione relative alle VLAN senza bisogno di ripeterle staticamente su ogni switch.

- *Cisco Virtual Trunking Protocol (VTP)*: protocollo proprietario Cisco
- *Multiple VLAN Registration Protocol (MVRP)*: protocollo standard

Inter-VLAN networking

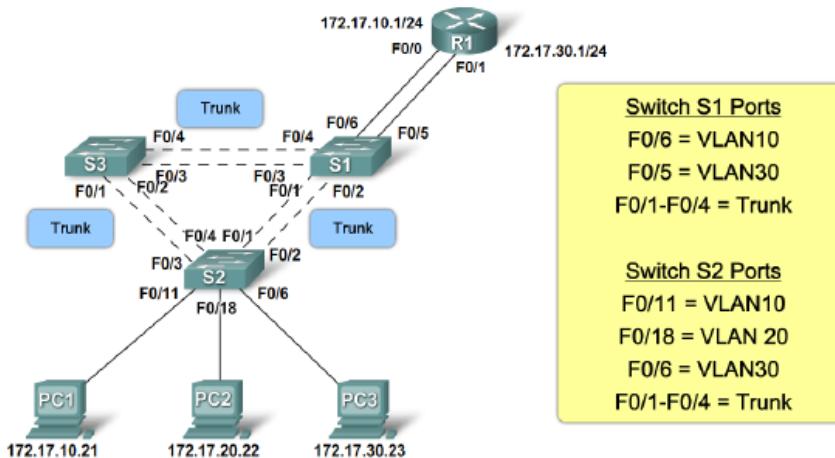


Se PC1 vuole mandare un pacchetto a PC3 (essendo VLAN diverse dal punto di vista di IP gli host sono su reti diverse) la comunicazione non può avvenire a livello 2, ma deve avvenire a **livello 3** tramite un router – la comunicazione diventa un problema di **inter-VLAN routing**.

Soluzione 1: inter-VLAN routing classico

Si collega un router allo switch per gestire le funzionalità di routing lvl 3, collegando una porta ad una porta di accesso della prima VLAN e un'altra porta a una porta di accesso della seconda VLAN.

Esempio: su S1 metto la porta fa0/6 come porta di accesso per VLAN 10 e la porta fa0/5 come porta di accesso per VLAN 30. Collego R1 alla porta fa0/6 di S1 tramite fa0/0 e alla porta fa0/5 con fa0/1



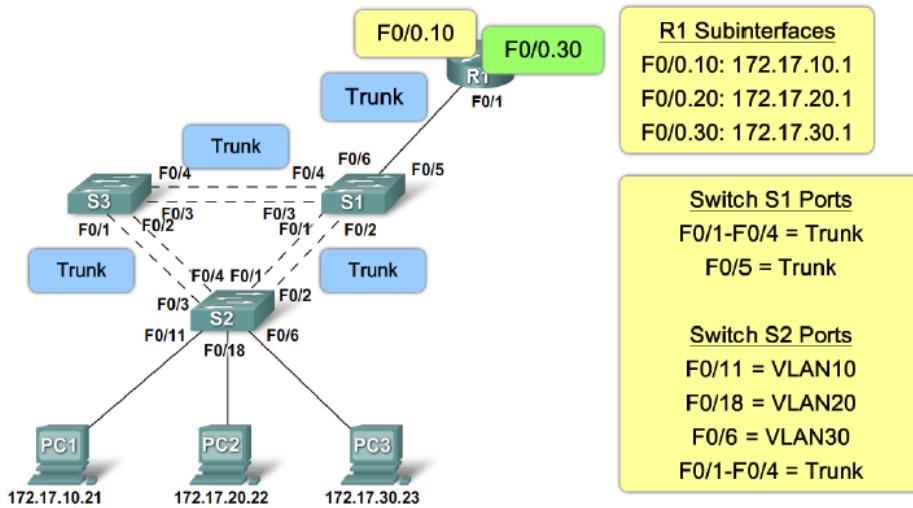
Quando PC1 (172.17.10.21) fa ping per PC3 (172.17.30.23):

1. PC1 invia al suo gateway (S2) un pacchetto IP con source 172.17.10.21 (PC1) e dest 172.17.30.23 (PC3)
2. S2 riceve il frame sulla porta fa0/11 (porta di accesso per VLAN 10), quindi lo inoltra verso S1 aggiungendo il tag (**trunk**)
3. S1 riceve il frame con il tag, legge che il frame è della VLAN 10 e lo inoltra sulla porta di accesso fa0/6
4. R1 riceve il pacchetto su fa0/0, guarda l'indirizzo di destinazione e **consulta la tabella di routing** che indica che l'interfaccia di uscita per raggiungere la rete 172.17.30.1/24 è fa0/1
5. Il pacchetto arriva sulla porta fa0/5 di S1
6. S1 lo inoltra sulla porta fa0/2 di S2, questo lo inoltra su fa0/6 verso PC3

Svantaggio: più VLAN ci sono più porte il router deve avere (**non scalabile**)

Soluzione 2: Router-on-a-stick

Si usa un solo link **trunk** tra lo switch e il router, visto che i frame hanno il tag lo switch può mandare sulla stessa porta sia i frame della VLAN 10 che quelli della VLAN 30 e il **router** deve diventare **VLAN-aware**



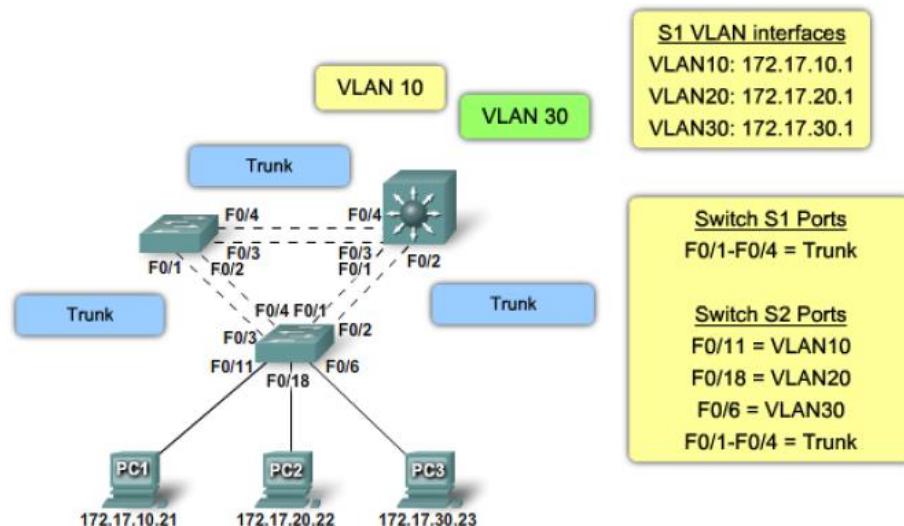
Il router utilizza le **sub-interfaces**: interfacce logiche associate a una interfaccia fisica e associate univocamente a una VLAN (con indirizzi IP diversi tra loro); in questo caso il router avrà 2 sotto-interfacce, una per VLAN 10 e una per VLAN 30.

Quando il router riceve il frame consulta la tabella di routing che specificherà su quale sotto-interfaccia inoltrare il pacchetto; quindi, in base alla VLAN associata alla sotto-interfaccia il router **cambierà il tag** del frame e quando questo arriva su S1 avrà **tag 30** e verrà inoltrato dallo switch verso PC3

Nota: si chiamano sub interfaces, ma stanno “*a fianco*” alle interfacce, se si esegue un comando sull’interfaccia principale non vale anche per le sub-interface. A fianco all’interfaccia fisica si hanno queste sotto-interfaccia associate, ma che logicamente sono separate

Soluzione 3: switch multi-layer

Le funzioni di routing e di switching e quindi la coppia di apparati R1 – S1 può essere **combinata in un unico apparato fisico**: uno **switch multi-layer** ha porte che possono funzionare come porte di livello 2 o di livello 3 ed è quindi è in grado di fare sia switching a livello 2 che routing a livello 3



[Soluzione 1 – Packet Tracer]

Il router ha una interfaccia dedicata per ciascuna VLAN presente

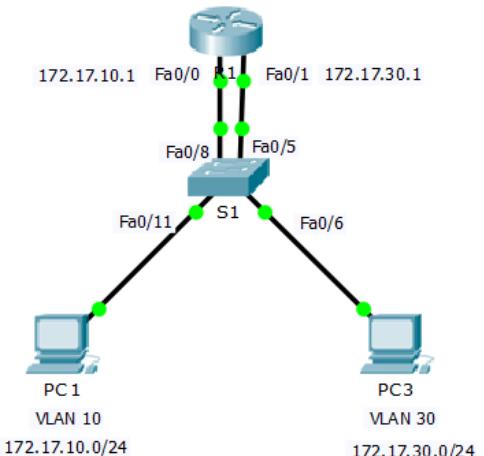
- **S1:** fa0/11 → VLAN 10 → 172.17.10.0/24
- **S1:** fa 0/6 → VLAN 30 → 172.17.30.0/24
- **R1:** fa0/0 (172.17.10.1) → gateway per la VLAN 10
- **R1:** fa0/1 (172.17.30.1) → gateway per la VLAN 30

1. Configurazione switch: definizione VLAN e configurazione porte

```
S1(config)#vlan 10
S1(config)#vlan 30

S1(config)#int fa0/8
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 10
S1(config)#int fa0/11
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 10

S1(config)#int range fa0/5 - fa0/6
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 30
```



2. Configurazione Router: assegnamento indirizzi IP alle interfacce

```
R1(config)#int fa0/0
R1(config-if)#ip addr 172.17.10.1 255.255.255.0
R1(config-if)#no shutdown

R1(config)#int fa0/1
R1(config-if)#ip address 172.17.30.1 255.255.255.0
R1(config-if)#no shutdown
```

3. Configurazione host: assegnamento indirizzi IP e default gateway

PC1: 172.17.10.10, default gateway: 172.17.10.1 (fa 0/0 su R1)
 PC3: 172.17.30.10, default gateway: 172.17.30.1 (fa 0/1 su R1)

Ping da PC1 a PC3

1. PC1 genera il pacchetto ICMP e lo invia a S1
2. S1 riceve il pacchetto sulla porta fa0/11, essendo una porta di accesso lo switch sa che quel frame appartiene alla VLAN 10 e non è taggato. Il pacchetto è un frame con MAC di destinazione 0001.64C8.AE01 (porta fa0/0 di R1). Lo switch inoltra il frame sulla sua porta fa0/8 perché è associata alla VLAN 10.
3. Il pacchetto viene inoltrato a R1 senza tag, perché la porta fa0/8 di S1 è una porta di accesso
4. R1 riceve il pacchetto sulla porta fa0/0, l'indirizzo MAC di destinazione del frame ethernet coincide con quello della porta quindi il frame viene processato a livello 3
 - a. R1 legge l'IP di destinazione (172.17.30.10) e lo cerca nella tabella di routing che indica come interfaccia di inoltro fa0/1

5. *R1 inoltra il pacchetto verso S1, che visto che lo riceve sulla porta fa0/5, capisce che appartiene alla VLAN 30 quindi lo inoltra solo su fa0/6.*

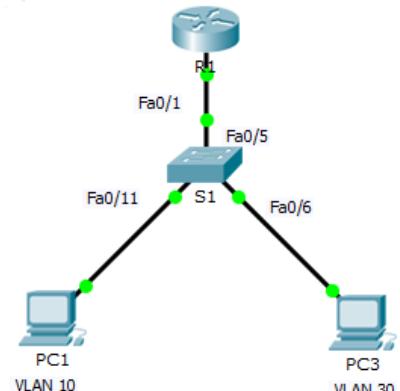
Il pacchetto che *R1* rimanda a *S1* ha **lo stesso payload IP del pacchetto di prima**, solo che prima lo switch *S1* non poteva inoltrarlo direttamente alla su *fa0/6* perché non può prendere decisioni a livello 3, ma solo a livello 2. Quello che cambia nel pacchetto sono gli indirizzi MAC, che sono proprio quelli che lo switch esamina per decidere come inoltrare il frame

[Soluzione 2 – Packet Tracer]

Link trunk tra lo switch e il router: si usa uno stesso link fisico per il trasporto di frame appartenenti a VLAN diverse

Il router ha una sola interfaccia *fa0/1* connessa allo switch, ma deve avere 2 indirizzi IP diversi, quindi è necessario usare le **sotto-interfacce**:

- Ogni sotto-interfaccia deve avere un id univoco del tipo ***id_interfaccia_fisica.id***
- È buona norma usare come id della sotto-interfaccia l'id della VLAN di cui diventa gateway
 - *fa0/1.10* -> 172.17.10.1
 - *fa0/1.30* -> 172.17.30.1



1. Configurazione switch:

- a. *fa0/11: porta di accesso VLAN 10*
- b. *fa0/6: porta di accesso VLAN 30*
- c. *fa0/5: porta trunk, S1 deve considerare che a valle di fa0/5 potrebbero esserci host appartenenti a una delle VLAN definite (di default tutte ammesse), quindi ci deve inoltrare i frame destinati a queste aggiungendo il tag*

Nota: è bene che né la VLAN 10 né la 30 siano native per evitare che frame senza tag vengano erroneamente considerati come appartenenti a una di queste

2. Configurazione router

R1 riceve i frame sulla porta *fa0/1*

- a. se il tag è 10 li deve considerare come ricevuti sulla sotto-interfaccia *fa0/1.10*
- b. se il tag è 30 li deve considerare come ricevuti sulla sotto-interfaccia *fa0/1.30*

Analogamente quando trasmette un frame

- a. tramite la sotto-interfaccia *fa0/1.10* deve aggiungere il tag 10
- b. tramite la sotto-interfaccia *fa0/1.30* deve aggiungere il tag 30

Prima dell'assegnamento dell'indirizzo IP alla sotto-interfaccia è necessario **specificare il protocollo di tagging attivo sul link**; perché la sotto-interfaccia possa gestire correttamente un frame deve poter capire quando tale frame è destinato a lei, cioè deve conoscerne la VLAN. Questo è possibile solo se si usa un protocollo che prevede la presenza del campo tag nel frame ethernet.

```
R1(config)#int fa0/1.10
R1(config-if)#encapsulation dot1Q 10
R1(config-if)#ip address 172.17.10.1 255.255.255.0
R1(config)#int fa0/1.30
R1(config-if)#encapsulation dot1Q 30
R1(config-if)#ip address 172.17.30.1 255.255.255.0
```

Nota: il comando no shutdown deve essere eseguito sull'interfaccia principale per rendere operative anche le sotto interfacce, questo è comunque un caso particolare, perché in generale i comandi eseguiti sull'interfaccia principale NON hanno effetto sulle sotto-interfacce

Ping da PC1 a PC3

1. PC1 crea il pacchetto ICMP, cioè un pacchetto IP con destinazione PC3
 - a. Il mac di destinazione è quello della sotto-interfaccia del router
2. Il pacchetto arriva a S1 tramite *fa0/11* quindi capisce che il frame appartiene alla *VLAN 10*
3. Il frame ethernet in uscita contiene anche il tag **perché la porta fa0/5 è trunk**
4. R1 riceve il frame ed è il tag a comunicargli che è la VLAN 10, associata all' interfaccia *fa0/1.10*. Il pacchetto viene “scartato” ed R1 esamina il pacchetto IP contenuto nel frame
5. R1 cerca la destinazione nella forwarding table e trova la entry che ha come interfaccia di inoltro *fa0/1.30*, quindi incapsula il pacchetto IP in un frame ethernet
6. Il pacchetto di uscita ha come tag 0xe1 (30) perché viene inviato da R1 tramite *fa0/1.30*
7. Il pacchetto torna a S1 sullo stesso link ma con un tag diverso
8. S1 lo inoltra sulla porta *fa0/6* verso PC3

Troubleshooting (errori comuni):

- Errori nella definizione di porte di accesso/porte trunk
- Errori nella connessione delle interfacce del router alle porte dello switch
- Mancata indicazione del protocollo di encapsulation

| Physical Interface vs Sub-interface | |
|--|--------------------------------|
| Una interfaccia per VLAN | Una interfaccia per più VLAN |
| NO bandwitch contention | Bandwidth contention |
| Connessa a porta di accesso | Connesse a porta di tipo trunk |
| Più costosa | Meno costose |
| Configurazione meno complessa | Configurazione più complessa |

Switch Multi-Layer

Switch Virtual Interfaces:

È buona norma definire una VLAN che è riservata alla comunicazione con gli switch; su questa non ci devono essere host, ma solo gli switch stessi in quanto apparati ed eventualmente host che appartenenti all'amministratore di rete.

Per comunicare con uno switch in quanto apparato è necessario assegnare allo switch un indirizzo IP che verrà assegnato a una interfaccia logica interna detta **Switch Virtual Interface (SVI)**. Questa ha un id corrispondente a quello della VLAN di cui fa parte (con `#interface vlan 99` si entra nel modo di configurazione dell'interfaccia identificata dall'id vlan 99).

È come se esistesse, dentro lo switch, una porta virtuale associata alla VLAN 99 (entrano ed escono solo frame appartenenti a questa), i frame che ci passano attraverso sono diretti allo switch e contengono pacchetti di livello 3 che vengono processati dallo switch.

Le **SVI sono interfacce di livello 3**, quindi gli si può assegnare un indirizzo IP; quando nella rete viaggia un pacchetto con l'IP della SVI, se è della VLAN 99, arriverà allo switch che vedrà che l'indirizzo di destinazione corrisponde a quello della SVI, capisce quindi che è indirizzato a sé stesso, quindi esamina il pacchetto IP.

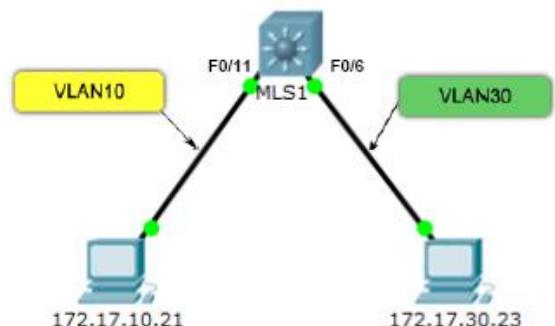
- Switch non multi-layer: si può definire di solito una sola SVI, perché lo scopo è scambiare pacchetti IP con lo switch
- **Switch multi layer**: si può pensare allo switch come composto da uno switch che opera a lvl 2 e un router che opera a lvl 3 (nella realtà queste funzioni sono però mischiate e ottimizzate). Le SVI sono le interfacce logiche del router virtuale che si trova interno allo switch multi-layer.

| | | |
|--|--------------------|---------------|
| Soluzione 1 (inter-VLAN routing classico) | Interfacce fisiche | Router fisico |
| Soluzione 2 (router-on-a-stick) | Interfacce logiche | Router fisico |
| Soluzione 3 (switch multi-layer) | Interfacce logiche | Router logico |

Multi-layer switching

I due host (su VLAN diverse) sono connessi ad uno stesso switch multi-layer: se PC1 vuole comunicare con PC3 manda un pacchetto allo switch che lo elabora prima di tutto come un frame di livello 2.

Lo switch capisce che il frame è indirizzato alla SVI che fa da gateway per PC1, quindi consulta la tabella di routing (essendo multi-layer ha tutto ciò che c'è sia in uno switch che in un router). Lo switch internamente prende una decisione **in base al contenuto del pacchetto IP trasportato dal frame**, quindi capisce che deve trasmettere il frame sulla *fa0/6*.



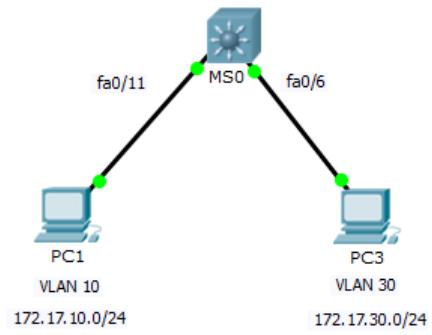
Il frame arriva da PC1 allo switch e dallo switch esce direttamente verso la destinazione: la decisione viene presa dallo switch **usando contemporaneamente il livello 2 e il livello 3** (alla stessa velocità dello switching di livello 2, perché a livello hardware il tutto è ottimizzato)

[Packet Tracer]

Configurazione Switch:

1. Definizione VLAN
2. Configurazione porte:

```
S1(config)#int fa0/11
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 10
S1(config)#int fa0/6
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 30
```



Attualmente *PC1* e *PC3* non possono comunicare tra loro, se anche *PC1* conoscesse il MAC di *PC3* l'invio non andrebbe comunque a buon fine, perché quando il frame raggiunge lo switch su *fa0/11*, vede che il MAC destinatario è su una VLAN diversa, quindi non inoltra il pacchetto su *fa0/6*

Se vogliamo che lo switch prenda decisioni basandosi anche sul contenuto del pacchetto ip dobbiamo **configurare le funzionalità di livello 3**:

1. Attivazione routing:
- S1(config)#ip routing
2. Configurazione interfacce (SVI):
- S1(config)#int vlan 10
S1(config-if)#ip address 172.17.10.1 255.255.255.0
S1(config-if)#int vlan 30
S1(config-if)#ip address 172.17.30.1 255.255.255.0

Ora anche sullo switch possiamo usare **ip route show** per vedere la tabella di routing, dove per le due destinazioni ci sono gli id delle due SVI

```
C      172.17.10.0 is directly connected, Vlan10
C      172.17.30.0 is directly connected, Vlan30
```

Ping da PC1 a PC3:

1. Lo switch riceve il pacchetto mandato da *PC1*, non contiene tag perché è un link di accesso
2. Lo switch lo riceve su *fa0/11* e il MAC destinatario fa match con il MAC della SVI
3. Il payload (pacchetto IP) viene processato sempre dallo switch a livello 3
4. S1 cerca la destinazione nella tabella di forwarding, la porta di uscita è VLAN 30 (la SVI)
5. L'interfaccia VLAN 30 guarda il MAC destinatario, che è quello della fast ethernet di *PC3* e capisce che deve inoltrare il frame su *fa0/6*

Anche se il pacchetto entra ed esce sempre dallo switch, si deve tenere a mente che all'interno del multi-layer switch viene fatto sia switching che routing perché i due host sono su VLAN diverse

Routed port vs Switch Port

Le porte degli switch di accesso (S1 e S2) sono **switched port**, cioè lavorano solo a livello 2.

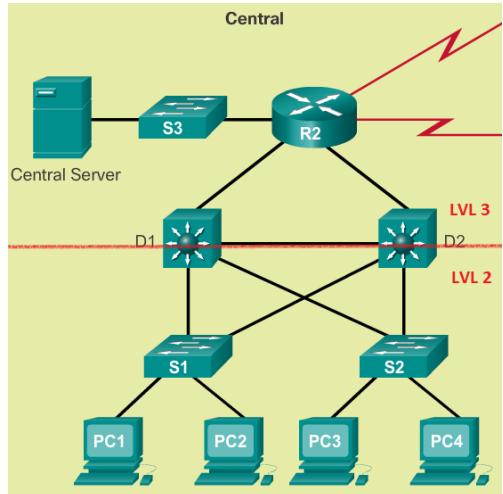
Gli switch del **livello di distribuzione** (D1 e D2) sono multi-layer:

- verso gli switch di accesso le porte sono switched port
- verso il livello 3 sono configurate come **routed port** cioè sono interfacce di livello 3

R2 vede quindi D1 e D2 come se fossero anch'essi dei router e al link che li collega dovrà essere assegnato un indirizzo di rete /30 e un indirizzo IP alla **porta fisica dello switch multi-layer** (non alla SVI)

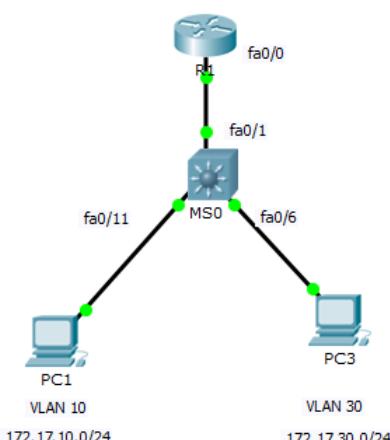
Lo stesso vale anche per la porta con cui D1 è connesso a D2 e viceversa; vale anche tutto ciò che abbiamo visto per i router, in particolare D1 e D2 si scambieranno pacchetti OSPF.

D1 e D2 sono gateway per tutte le VLAN presenti, e da loro in poi i pacchetti sono scambiati usando il livello IP



[Packet Tracer]

Aggiungiamo un router alla rete: collegiamo la porta 1 dello switch multi-layer alla porta fa del router: sarà quindi come se fossero **due router collegati**



```
S1(config)#int fa0/1
S1(config-if)#no switchport
S1(config-if)#ip address 192.168.1.2 255.255.255.252
```

In questo modo la porta fa0/1 diventa una **routed port** con indirizzo 192.168.1.2/30 (esattamente come su un router si attiva con no shutdown)

Configuriamo quindi la porta fa0/0 del router con l'indirizzo 192.168.1.1/30

```
S1# show ip route
C 172.17.10.0 is directly connected, Vlan10
C 172.17.30.0 is directly connected, Vlan30
C 192.168.1.0 is directly connected, FastEthernet0/1
```

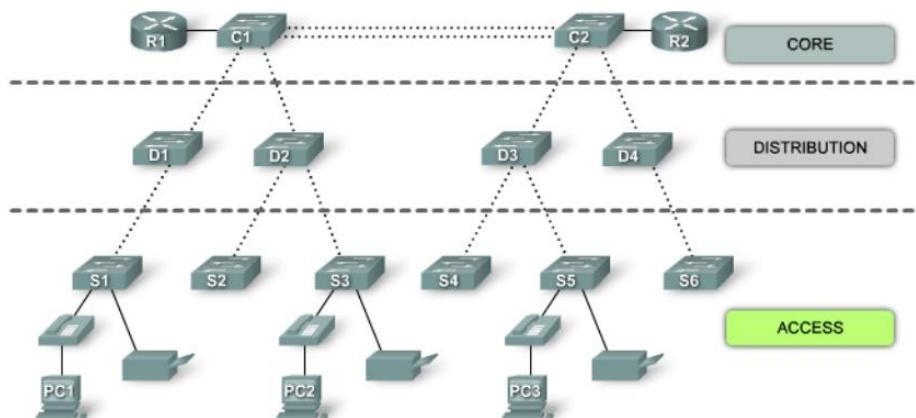
Sullo switch possiamo anche definire una route statica di default in modo che, quando lo riceve un pacchetto che non è destinato ad alcuna delle sue VLAN, lo switch userà la default route per mandarlo al router.

Facendo ping da PC0 all'interfaccia IP del router vediamo che il frame arriva sulla porta fa0/11 di S1, viene gestito a lvl 2 e riconosciuto come indirizzato alla SVI vlan 10 e, da quel momento in poi, gestito a livello 3

Protocollo Spanning Tree

Una rete locale ha tipicamente una topologia gerarchica ad albero, cioè senza percorsi chiusi; l'appartenenza di uno switch a un livello piuttosto che a un altro determina il ruolo dello switch all'interno della rete locale.

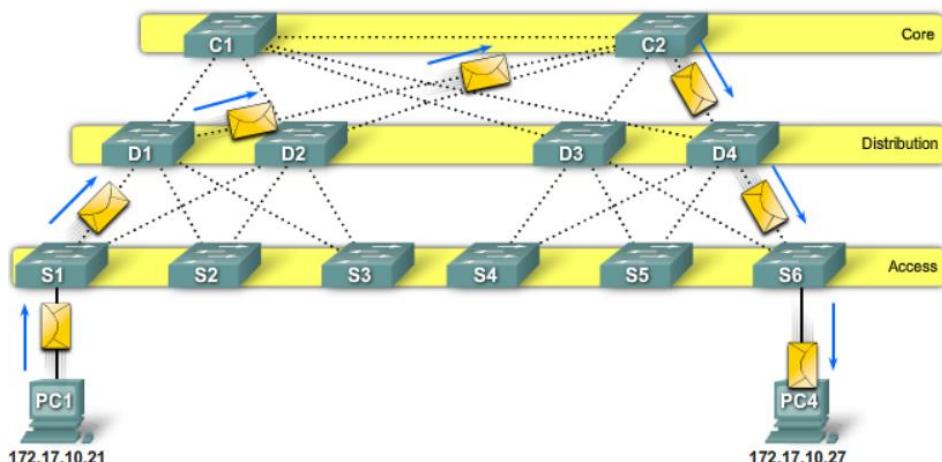
1. Livello di **accesso** (traffico *intra-VLAN*): switch di accesso, devono fare controlli di sicurezza perché sono sulla frontiera della rete, hanno un numero elevato di porte perché le forniscono a ciascun host.
2. Livello di **distribuzione** (traffico *inter-VLAN*) e **core** (traffico *inter-network*): per consegnare frame mandati tra host che si trovano in reti di accesso diverse; si usano switch multi-layer per poter fare routing per gli host che si trovano in VLAN diverse. Hanno meno porte ma devono essere più veloci.



Questo tipo di rete non è capace di resistere ad alcun tipo di guasto: se uno dei link si rompe la rete diventa "divisa", poiché esiste un solo percorso da un punto ad un altro

Per rendere il sistema più affidabile (più robusto ai guasti) si usa un **design ridondante**: il primo passo è costruire un sistema resistente rispetto al singolo guasto che si ottiene duplicando il numero di apparati e di collegamenti.

Al livello di accesso l'affidabilità non è un requisito, specialmente se gli host sono client; diventa più importante per i server perché da questa dipende anche l'accessibilità al servizio.



Togliendo in questa rete un collegamento o uno switch qualunque (eccetto gli switch di accesso) continuerà ad esserci sempre un percorso tra un host e un qualsiasi altro host:

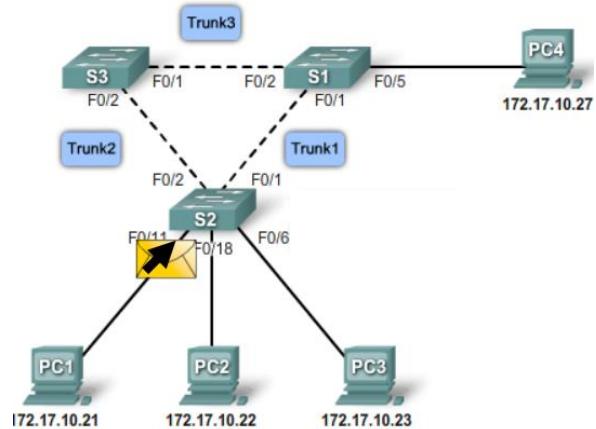
- ogni switch di accesso è collegato a due switch di distribuzione diversi
- ogni switch di distribuzione è collegato a due switch di core diversi
- il numero di switch di core è pari in modo che ognuno abbia una controparte che può prendere il suo posto in caso di guasto

Problemi causati dall'introduzione di ridondanze

1. Frame Unicast Duplicati

PC1 manda un frame a PC4:

- S2 non conosce l'indirizzo MAC di destinazione, quindi, fa flooding sulle porte 1, 2, 6 e 18
- Dalle porte 6 e 18 i frame vanno a due host che hanno MAC diverso e quindi lo buttano via
- Le altre due copie viaggiano sui link Trunk1 e Trunk2 verso S1 e S3
 - S3 legge l'indirizzo destinatario nella tabella MAC e inoltra il frame sulla porta 1
 - S1 legge l'indirizzo destinatario nella tabella MAC e inoltra il frame sulla porta 5
- PC4 riceve il frame da S1
- S1 riceve una nuova copia dello stesso frame (che non riconosce come copia), quindi lo inoltra sulla porta 5



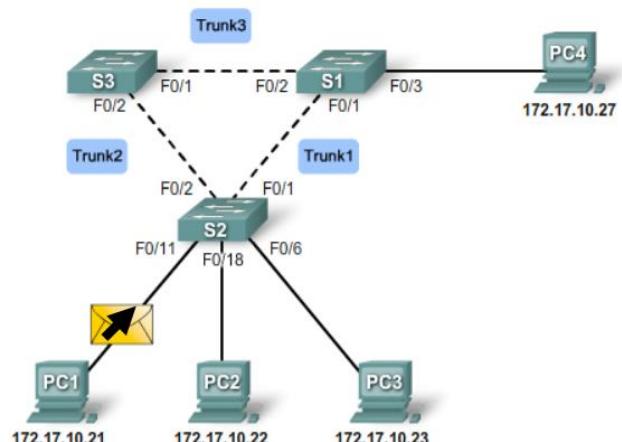
I frame vengono duplicati e le copie arrivano tutte a destinazione (il numero delle copie dipende dal numero di percorsi alternativi nella rete)

2. Broadcast Storms

Il numero di frame che circolano nella rete cresce fino a congestionare la rete

Es slide 9

- PC1 manda un frame in broadcast
- Il frame arriva su S2 che fa flooding
- Quando S1 riceve la copia del frame fa flooding sulla porta 3 verso PC4 e sulla porta 2 verso S3
- Allo stesso tempo S3 fa flooding della copia ricevuta da S2 sulla porta 1 verso S1
- Quando S1 ed S3 ricevono queste nuove copie fanno nuovamente flooding, S1 verso PC4 e S2, S3 verso S2
- S2 riceve due copie che poi rimanda agli host



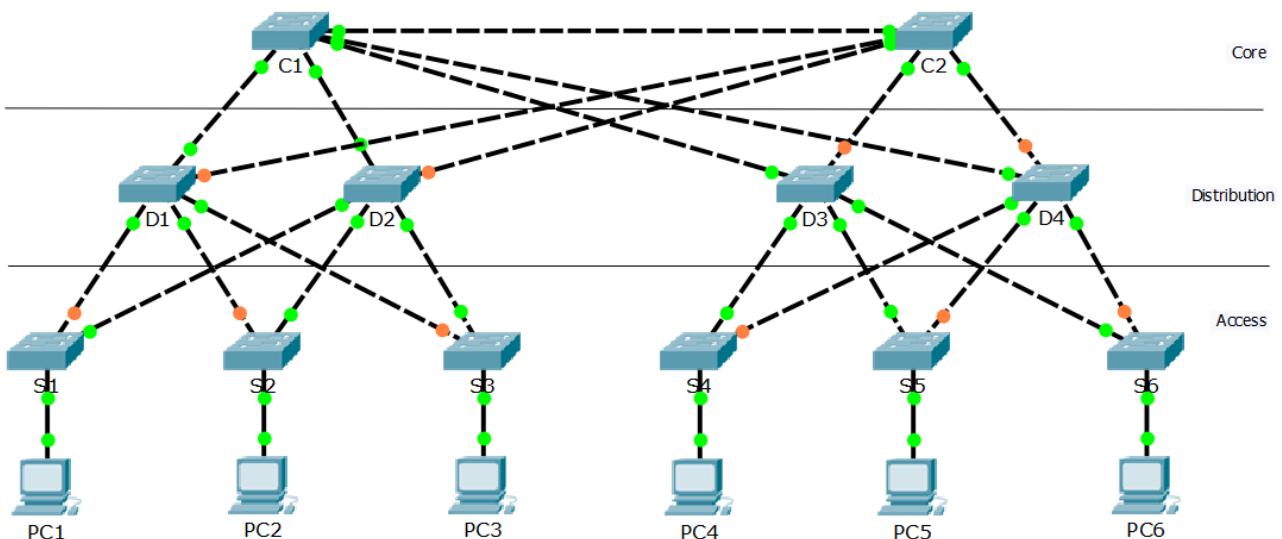
Gli host continuano a ricevere copie del frame iniziale e le altre copie del frame continuano a girare sul percorso chiuso all'infinito, perché non hanno TTL

Protocollo Spanning Tree

L'implicazione del processo di learning è che possiamo avere solo reti senza percorsi chiusi, ma il requisito di affidabilità ci impone di inserire ridondanze; è necessario **cambiare qualcosa nel processo di forwarding**

Se nella rete introduco fisicamente dei percorsi chiusi, chiedo agli switch di disabilitare logicamente il forwarding su determinate porte; in presenza di un guasto questo insieme di porte disabilitate viene riarrangiato in modo da recuperare l'utilizzo di un link per coprire quello non più disponibile.

Gli switch calcolano un **albero di copertura** logico sulla topologia fisica: i link che non appartengono all'albero non vengono utilizzati e **quando si verifica un guasto viene ricalcolato l'albero** in modo da ristabilire un percorso che connetta tutti i nodi



Dopo la fase iniziale le porte degli switch diventano "verdi" perché possono essere utilizzate per il forwarding, alcune però rimangono arancioni poiché vengono disabilitate proprio per evitare i problemi dovuti alle ridondanze; esistono dei percorsi chiusi a livello fisico ma non a livello logico

Per verificare che non ci siano percorsi chiusi si fa un ping da PC1 a PC6 e si guarda in modo simulazione come viaggiano i pacchetti ARP:

1. Il pacchetto raggiunge S1 che lo inoltra verso D2, ma non verso D1 perché la porta corrispondente è in stato di non forwarding
2. D2 lo inoltra sulle sue altre tre porte abilitate verso PC2, PC3, C1
3. C1 lo inoltra verso D1, D2, D3, D4, C2
4. C2 lo inoltra verso D3, D4, D1
5. D1, D3, D4 ricevono il frame da C2 su una porta non forwarding quindi non viene processato
6. D4 manda a S4, S5, S6, ma ricevono il frame su porte non forwarding quindi lo gettano via
7. D3 manda a S4, S5, S6 su porte in stato di forwarding quindi inoltrano il frame ai rispettivi host.

Il protocollo **Spanning Tree** è continuamente in funzione perché è responsabile all'inizio del calcolo dell'albero di copertura, ma anche dopo di rilevare eventuali modifiche alla topologia della rete e in caso ricalcolare l'albero

Esempio: eliminiamo il link tra S1 e D2: potenzialmente abbiamo isolato S1 perché il link verso D1 è disabilitato, il protocollo noterà questa variazione e riconfigurerà la porta di S1 connessa a D1 come attiva (più si sale nella gerarchia più porte cambiano stato).

Funzionamento del protocollo (diviso in 3 fasi che nella realtà sono concorrenti):

1. **Root Bridge election:** gli switch si scambiano pacchetti **bridge protocol** per eleggere una **radice** dell'albero (essere radice non è una proprietà topologica dell'albero)
2. **Root Port selection:** ciascuno switch determina qual è la porta con la quale raggiunge la radice con costo minimo e la attiva. Lo switch root accende tutte le porte (essendo radice non genera percorsi chiusi)
3. **Designated Port selection:** ogni altro switch decide cosa fare con le sue altre porte

1. Root Bridge election

Ogni switch ha un proprio **id** unico su 8B:

- [2B] Bridge priority: si può modificare per controllare chi sarà la radice
- [6B] Indirizzo MAC

Per l'elezione della radice gli switch si scambiano gli id tramite pacchetti **Bridge Protocol Data Unit (BPDU)** e viene scelto come root quello che ha id più piccolo: viene valutata prima la priorità (parte più significativa dell'id) e a parità di questa viene scelto lo switch con indirizzo MAC più piccolo.

Come radice vogliamo che venga scelto un certo switch piuttosto che un altro perché questo avrà tutte le porte attive (per esempio uno switch di accesso non sarebbe adatto).

Inizialmente su ogni switch il root id è il proprio id, quando poi riceve gli id degli altri switch se ne trova uno più piccolo aggiorna il root id; all'inizio, quindi, tutti pensano di essere radici e alla fine del processo avranno tutti "cambiato idea" tranne uno, cioè quello che diventa effettivamente la radice.

Negli switch CISCO il campo priorità è composto da:

- [4b]: priorità
- [12b]: estensione dell'id di sistema

Quando si hanno più VLAN esiste **uno spanning tree per ogni VLAN** che opera sulla topologia logica di quella VLAN, ci sarà quindi **un root switch per ogni VLAN**.

Non basta quindi l'id per identificare il bridge, perché quello switch potrebbe essere radice per una VLAN, ma non per un'altra: si **estende l'id** mettendo dopo la priorità e prima del MAC l'id della VLAN, in questo modo **la priorità rimane significativa all'interno della VLAN**, mentre l'extended system id serve per capire su quale VLAN si sta facendo STP.

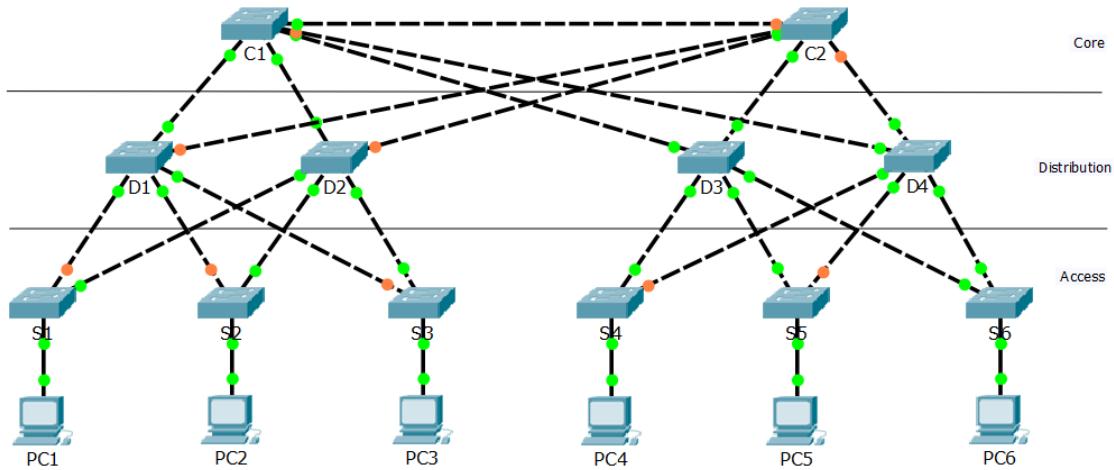
Nota: la priorità cambia per multipli di 4096 perché sono i primi 4b dei 2B del campo priorità.

CISCO usa come default 32768, ad esempio 32769 comprende i bit relativi alla VLAN (in questo caso la 1)

[Packet Tracer]

Modifica della priorità:

```
S1(config)#spanning-tree vlan vlan-id prority value
S1(config)#spanning-tree vlan vlan-id root primary: imposta la priorità in modo che sia la più alta
S1(config)#spanning-tree vlan vlan-id prority root secondary: imposta la priorità in modo che sia la seconda più alta (in caso si guastasse la radice voglio che questo switch sia la nuova radice)
```



S6#show spanning-tree:

```
VLAN0001
Spanning tree enabled protocol ieee
Root ID Priority 32769
Address 0000.0C96.DD0C
This bridge is the root
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
Bridge ID Priority 32769 (priority 32768 sys-id-ext 1)
Address 0000.0C96.DD0C
(...)
Interface      Role      Sts      Cost      Prio.Nbr      Type
Fa0/2          Desg      FWD      19        128.2        P2p
Fa0/3          Desg      FWD      19        128.3        P2p
Fa0/1          Desg      FWD      19        128.1        P2p
```

Oltre alle informazioni su chi sia il Root Bridge, con l'indirizzo MAC e la priorità viene mostrato anche il ruolo e lo stato di ciascuna porta; in questo caso tutte attive, designated e in stato di forwarding (FWD)

D3#show spanning-tree:

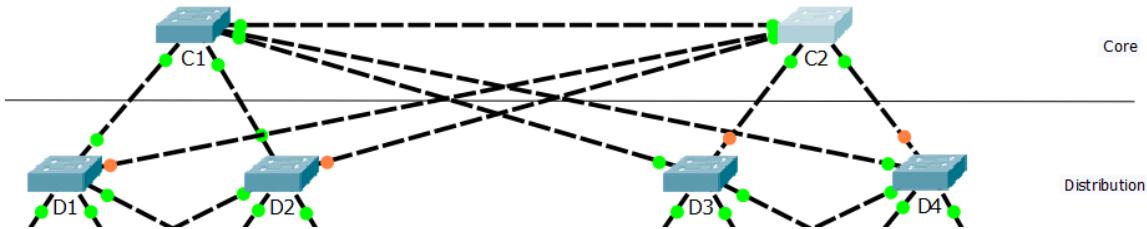
```
(...)
Fa0/2 Desg FWD 19 128.2 P2p
Fa0/4 Desg FWD 19 128.4 P2p
Fa0/1 Desg FWD 19 128.1 P2p
Fa0/3 Desg FWD 19 128.3 P2p
Fa0/5 Root FWD 19 128.5 P2p
```

Il ruolo Root indica che la porta Fa0/5 è quella con cui D3 raggiunge il Root Bridge (S6)

In questa rete gli switch hanno tutti la stessa priorità quindi il root bridge è scelto in base all'indirizzo MAC; questo caso non è ottimale perché S6 è uno switch di accesso su cui idealmente quindi transitano solo frame da/per l'host connesso.

Per garantire un bilanciamento vogliamo che il root sia il più in alto possibile nella gerarchia, quindi andiamo a modificare la priorità dello switch C1

- C1(config)#spanning-tree vlan 1 root primary
- Vediamo che la priorità del root bridge è diventata $24577 = 32769 - 2 * 4096$ (aumentata di due "spazi" per lasciare posto al root secondary)
- C2(config)#spanning-tree vlan 1 root secondary
- L'effetto del comando è nullo sull'operatività della rete finché C1 è operativo, però la priorità di C2 è diventata $28673 = 32769 - 4096$



Al momento C2 non serve a nulla perché tutte le porte connesse a lui sono arancioni: queste non inoltrano i frame utente, ma processano i BPDUs perché se C1 va via e C2 comunica che è il nuovo root gli altri switch lo devono sapere.

In questo caso quindi la ridondanza è passiva: si ha un componente di backup che però al momento non sta facendo niente

2. Root port selection

I BPDUs ricevuti da uno switch contengono diverse informazioni, tra cui l'id del root bridge (**root id**), l'id dello switch che sta trasmettendo il BPDUs (**bridge id**) e il costo con cui si raggiunge la radice (**root path cost**).

Il root path cost viene calcolato usando **distance vector semplificato** (a differenza di IP non ci interessano tutte le distanze ma solo quella verso il root).

Il costo è associato alla porta perché inversamente proporzionale alla velocità e porte diverse hanno velocità diverse, quando poi lo switch propaga il BPDUs ci aggiunge il costo dovuto a lui stesso come nel distance vector standard.

| Field # | Bytes | Field |
|---------|-------|---------------|
| 1-4 | 2 | Protocol ID |
| | 1 | Version |
| | 1 | Message type |
| | 1 | Flags |
| 5-8 | 8 | Root ID |
| | 4 | Cost of path |
| | 8 | Bridge ID |
| | 2 | Port ID |
| 9-12 | 2 | Message age |
| | 2 | Max age |
| | 2 | Hello time |
| | 2 | Forward delay |

Ogni switch riceve BPDUs che identificano la radice da più porte, tra tutte queste seleziona quella che dà il costo minore e quindi la segna come **root port**, questa porta sarà per forza in forwarding e **non ci verranno più trasmessi BPDUs** perché è da quella porta che lo switch si aspetta di riceverli dalla radice.

A parità di costo si sceglie il bridge che ha l'id più piccolo, ma non basta perché si potrebbero ricevere su più porte diverse 2 BPDU provenienti dallo stesso bridge (se ci sono degli hub nella rete), quindi si sceglie la porta che ha l'id più piccolo, questo infatti viene mandato nel campo **port id** del BPDU.

Il **port id** ha a sua volta due campi

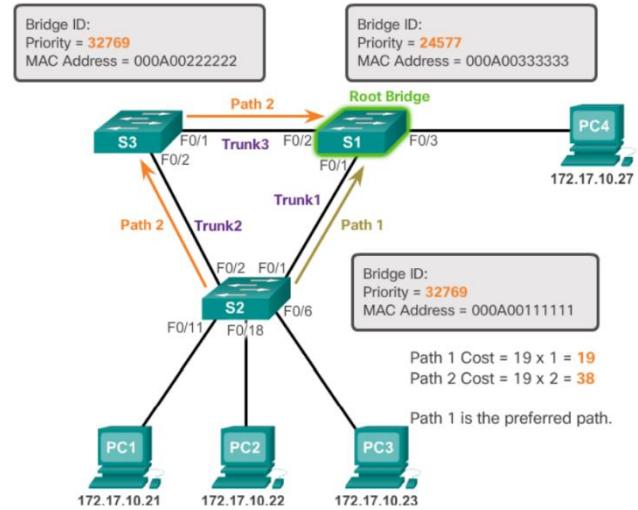
- [1B] **port priority**: usato per controllare la scelta della root port
- [1B] **port number**

I costi delle porte sono inversamente proporzionali alla velocità della porta stessa:

- 10 Gb/s = 2
- 1 Gb/s = 4
- 100 Mb/s = 19
- 10 Mb/s = 100

Esempio:

- S1 è il root bridge (id più piccolo)
- Tutti sapranno poi che è S1 il root bridge
- Quando S2 riceve il BPDU da S1 sulla porta *fa0/1*, “impara” che da quella porta c’è un percorso verso il root bridge
- Il BPDU generato da S1 ha costo 0
- **S2 calcola il costo del percorso verso il root bridge S1** sommando al costo letto nel BPDU il costo della porta su cui lo riceve: $0 + 19 = 19$
- S2 riceve un BPDU su *fa0/2* inoltrato da S3
- S2 “impara” che può raggiungere il root bridge anche dalla porta *fa0/2*
- S2 calcola il costo del percorso sommando al costo letto nel BPDU (19) il costo associato alla porta *fa0/2* (19) = 38
- Essendo il costo di questo percorso maggiore, S2 sceglierà come **root port** *fa0/1*



3. Designated port selection

Cosa fanno gli switch non root delle porte che non sono root port?

- Le porte di accesso vengono messe in forwarding per definizione (perché dall’altra parte c’è un host)
- Per le porte trunk ne viene designata solo una
 - Quando uno switch riceve un BPDU su una porta non root port, significa che quella porta sarà un altro percorso per raggiungere la radice
 - Dopo aver ricevuto i vari BPDU ogni bridge conoscerà la distanza dalla radice di tutti i percorsi alternativi
 - Viene **designata** la porta relativa al percorso **più vicino** (a parità di distanza si usa il bridge id)

Nell’esempio *fa0/2 di S2 e fa0/2 di S3* hanno entrambe costo 19, quindi viene designata la porta di S2 perché ha l’id più piccolo, *fa0/2 di S2* avrà lo stato **forwarding** e *fa0/2 di S3* avrà lo stato **blocked** (non fa forwarding); la prima è quella designated e l’altra è alternative. Una è **designated**, l’altra è **alternative**

Varianti di STP

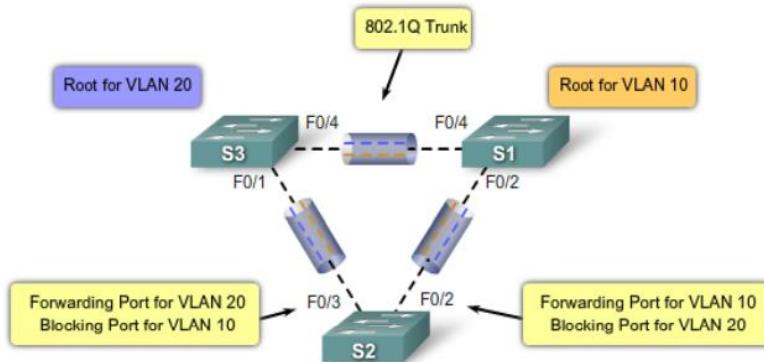
1. **Rapid Spanning Tree (RSTP)**: versione attuale del protocollo (2004), sfrutta il fatto che gli switch riducono il dominio di collisione al singolo link e mette in campo una serie di meccanismi per velocizzare il recupero dopo un guasto
2. **Multiple Spanning Tree (MSTP)**: per reti con più VLAN

Cisco ha una serie di varianti proprietarie del protocollo

1. **Per-VLAN Spanning Tree (PVST)+**: spanning tree per VLAN e usa il trunking 802.1Q
2. **Rapid-PVST+**: RSTP con uno spanning tree per VLAN

PVST+

- **VLAN diverse con stesso root**: l'albero sarà costruito allo stesso modo quindi per entrambe le VLAN viene utilizzato lo stesso gruppo di porte. Si ha quindi un link che è totalmente **inutilizzato**
- **VLAN diverse con root diversi**: sia S1 il root per VLAN 10 e S2 il root per VLAN 20, allora gli alberi vengono costruiti in maniera diversa e **nessuno dei link è a utilizzazione 0**



Per avere un utilizzo più efficiente delle risorse è quindi necessario fare attenzione a scegliere correttamente il root per ogni vlan

DHCP

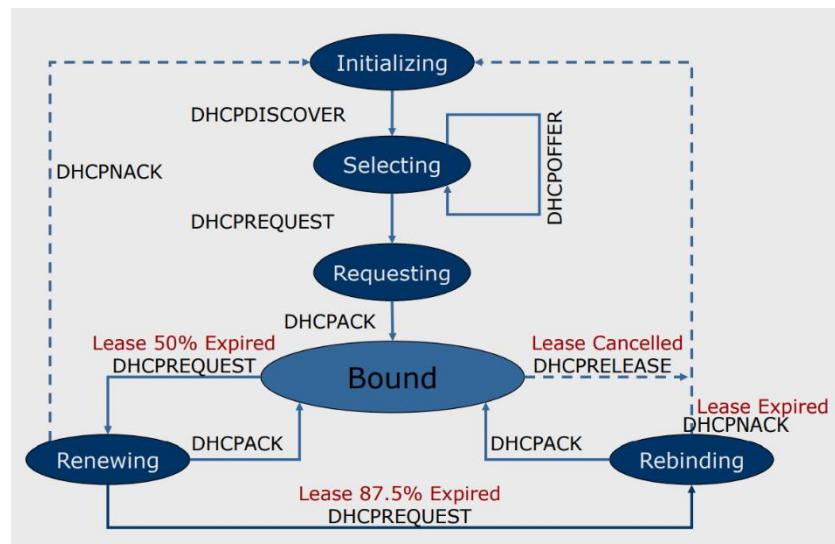
Il protocollo DHCP nasce a partire da **Boot Protocol (BOOTP)** per permettere agli host di scaricare il bootstrap da un server della rete, poiché la configurazione statica degli indirizzi era contenuta nel sistema operativo.

Allocazione dinamica:

- L'indirizzo IP viene allocato a tempo (**leasing**): il protocollo fissa un termine di validità dell'indirizzo oltre il quale l'host non può più usarlo (se non con estensione)
- Il meccanismo di scambio di pacchetti è stato definito da BOOTP: i messaggi DHCP sono messaggi BOOTP, il protocollo usa UDP per il livello di trasporto e IP per il livello network
- La negoziazione avviene anche attraversando più router; se un host si trova su una certa VLAN può comunque ottenere un indirizzo da un server DHCP che si trova su una VLAN differente

Funzionamento generale:

1. L'host manda **DHCPDISCOVER** con sorgente 0
2. Ogni server risponde con una **DHCPOFFER**
3. L'host sceglie il server mandando **DHCPREQUEST**
4. Il server risponde e conferma l'assegnamento con **DHCPACK**
5. A metà del tempo di lease l'host può richiedere una estensione con una **DHCPREQUEST**
6. Se non viene esteso il tempo di lease, all'87.5% viene mandata una nuova richiesta in broadcast
7. Al termine del tempo di lease l'host, se non è stata richiesta l'estensione l'host rilascia l'indirizzo con una **DHCPRELEASE**



Formato pacchetto DHCP

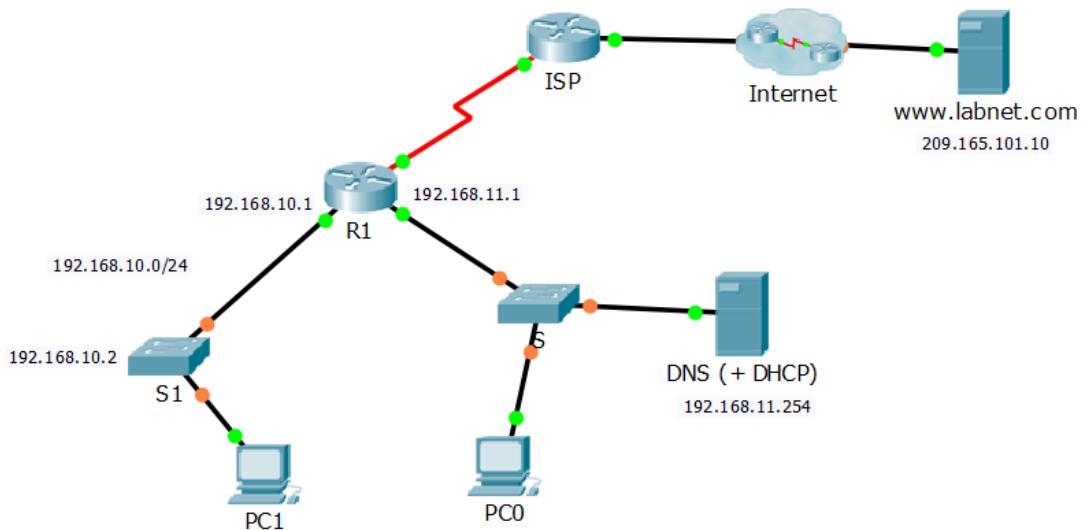
- **OPcode:** indica se è una richiesta o risposta
- **Your IP address:** riempito dal server con l'indirizzo IP che offre all'host
- **Boot file name:** protocollo BOOTP
- **Options:** informazioni su DHCP
 - **TAG:** tipo di informazione
 - **LEN:** lunghezza valore dell'informazione
 - **VALUE:** valore informazione

Il campo TAG ha dei valori noti (1: *subnet mask*, 61: *client identifier*, 54 *server identifier*, 53 **dhcp message**, ecc)

Quando il tipo è **53** nel campo valore è specificato il tipo di richiesta DHCP (**DHCPOFFER** = 2, **DHCPREQ** = 3, **DHCPACK** = 5, ecc.)

| Operation Code | Hardware Type | Hardware Length | Hop Count |
|------------------------------------|---------------------------------|-----------------|-----------|
| Transaction ID | | | |
| Number of seconds | Flag (1 bit) (15 unused bits) | | |
| Client IP address | | | |
| Your IP address | | | |
| Server IP address | | | |
| Gateway IP address | | | |
| Client hardware address (16 bytes) | | | |
| Server name (64 bytes) | | | |
| Boot file name (128 bytes) | | | |
| Options (up to 312 bytes) | | | |
| 4 bytes | | | |

[Packet Tracer]



Tre casi possibili:

1. Server DHCP dedicato nella stessa LAN del client
2. Il router fornisce il servizio DHCP ai client
3. Il server nella LAN 0 viene usato anche dagli host della LAN 1

CASO 1 – Server nella stessa LAN del client

- Sul server si mette in funzione il servizio DNS e si aggiunge a mano il record **Errore. Riferimento a collegamento ipertestuale non valido.**
- Si attiva il servizio DHCP sul server: si configura una **lista di indirizzi** che fanno parte del **pool di allocazione** (per host su lan servono pool diversi per ciascun blocco di indirizzi), con le relative informazioni:
 - Indirizzi disponibili per l’assegnamento: il primo indirizzo è il primo disponibile successivo a quello del server, quindi in questo caso **192.168.11.2/24**
 - Il numero di indirizzi previsti
 - **Gateway:** 192.168.11.1
 - **Server DNS:** 192.168.11.254
- Queste informazioni arriveranno insieme al singolo indirizzo IP come risposta della richiesta DHCP
- Per forzare l’host a fare una richiesta DHCP si imposta la configurazione IP come **dinamica**

Caso 2 – DHCP con Router

- PC1 non ha un server DHCP nella propria LAN, quindi chiediamo al router di fare da server DHCP (può farlo solo per le VLAN a cui è direttamente connesso)
- Sul router il servizio DHCP è sempre in esecuzione anche se non configurato (per disabilitarlo si utilizza il comando `no service dhcp`) le operazioni di configurazione del servizio sono le stesse del caso precedente
- Il router, in base a dove ricevere le richieste, capisce **di che LAN fa parte un certo host** e quindi **sceglie il pool di indirizzi relativo in base ai prefissi**

1. Specificare manualmente gli indirizzi IP da escludere dal pool (il pool è definito da un prefisso di rete, non da un range):

```
R1(config)#ip dhcp excluded-address 192.168.10.1
```

2. Configurare il pool di allocazione (definito da un prefisso di rete, non un range):

```
R1(config)#ip dhcp pool LAN-POOL-A [NOME]
```

```
R1(dhcp-config)#network 192.168.10.0 255.255.255.0
```

3. Configurare il gateway

```
R1(dhcp-config)#default-router 192.168.10.1
```

4. Configurare il DNS server

```
R1(dhcp-config)#dns-server 192.169.11.254
```

Nota: in questo caso il server offre a PC1 un indirizzo 10.2 che appartiene all'interfaccia dello switch e non è stato escluso, comunque prima di adottarlo il client fa una prova mandando un ARP e vede che quell'indirizzo è stato già preso, per cui lo scarta e prende uno di quelli riservati per l'auto assegnamento.

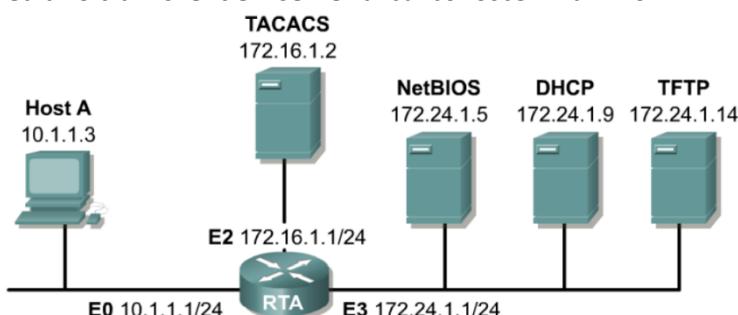
Comandi per lo stato del DHCP:

```
R1#show ip dhcp binding
```

```
R1#show ip dhcp pool
```

CASO 3 – Unico server DHCP per tutta la rete

Relay Agent (meccanismo di BOOTP): sfrutta il campo gateway IP address del pacchetto che indica il **relay gateway** che fa **intermediario tra il client e il server** di cui conosce l'indirizzo IP



1. A manda una *DHCPREQUEST* in broadcast

| Your IP Address | Server IP Address | GI Address | Packet Source MAC Address | Packet Source IP Address | Packet Destination MAC Address | Packet Destination IP Address |
|-----------------|-------------------|------------|---------------------------|--------------------------|--------------------------------|-------------------------------|
| 0.0.0.0 | 0.0.0.0 | 0.0.0.0 | 00E0.1EF2.C441 | 0.0.0.0 | ffff.ffff.ffff (broadcast) | 255.255.255.255 |

2. Quando il router riceve la richiesta, **se è configurato come relay** scrive nel campo **GI address** l'indirizzo IP dell'interfaccia dove ha ricevuto il messaggio

3. Il router manda un **pacchetto unicast al DHCP server**

| Your IP Address | Server IP Address | GI Address | Packet Source MAC Address | Packet Source IP Address | Packet Destination MAC Address | Packet Destination IP Address |
|-----------------|-------------------|-------------|---------------------------|--------------------------|--------------------------------|-------------------------------|
| 0.0.0.0 | 0.0.0.0 | 192.168.1.1 | Interface E0 MAC Address | 192.168.1.1 | MAC Address of DHCP Server | 192.168.2.2 |

4. Il server riceve la richiesta e manda indietro un pacchetto **unicast DHCPOFFER** all'indirizzo scritto in quel campo sorgente della richiesta (**indirizzo del router**)

| Your IP Address | Server IP Address | GI Address | Packet Source MAC Address | Packet Source IP Address | Packet Destination MAC Address | Packet Destination IP Address |
|-----------------|-------------------|-------------|----------------------------|--------------------------|--------------------------------|-------------------------------|
| 192.168.1.2 | 192.168.2.2 | 192.168.1.1 | MAC Address of DHCP Server | 192.168.2.2 | Interface E0 MAC Address | 192.168.1.1 |

5. Il server sceglie l'indirizzo da offrire dopo aver letto il **GI address** in modo da sapere a quale rete deve appartenere l'indirizzo, essendo quello **l'indirizzo dell'interfaccia sul quale il router ha ricevuto la richiesta dal client**
6. Il router manda la **DHCPOFFER** al client, che **riceve la stessa informazione che riceverebbe nel caso in cui non ci fosse il relay agent** (a meno del GI Address); riceve inoltre l'indirizzo del server DHCP che utilizzerà per fare il rinnovo o una eventuale nuova richiesta al server

| Your IP Address | Server IP Address | GI Address | Packet Source MAC Address | Packet Source IP Address | Packet Destination MAC Address | Packet Destination IP Address |
|-----------------|-------------------|-------------|---------------------------|--------------------------|--------------------------------|-------------------------------|
| 192.168.1.2 | 192.168.2.2 | 192.168.1.1 | Interface E1 MAC Address | 192.168.1.1 | ffff.ffff.ffff (broadcast) | 255.255.255.255 |

1. Si aggiunge al server un pool di indirizzi
 - a. Default gateway: 192.168.10.1
 - b. DNS server: 192.168.11.254
 - c. Indirizzo di partenza: 192.168.10.3
 - d. Numero di host massimi: 100
2. Vogliamo che R1 mandi le richieste al server invece che gestirle lui stesso quindi eliminiamo la configurazione precedente:

```
R1(config)#no ip dhcp pool LAN-POOL-A
R1(config)#no dhcp excluded-address 192.168.10.1
R1(config)#no dhcp excluded-address 192.168.10.2
```

Per configurare il router come relay lo dobbiamo **configurare su ogni interfaccia su cui deve fare da relay**:

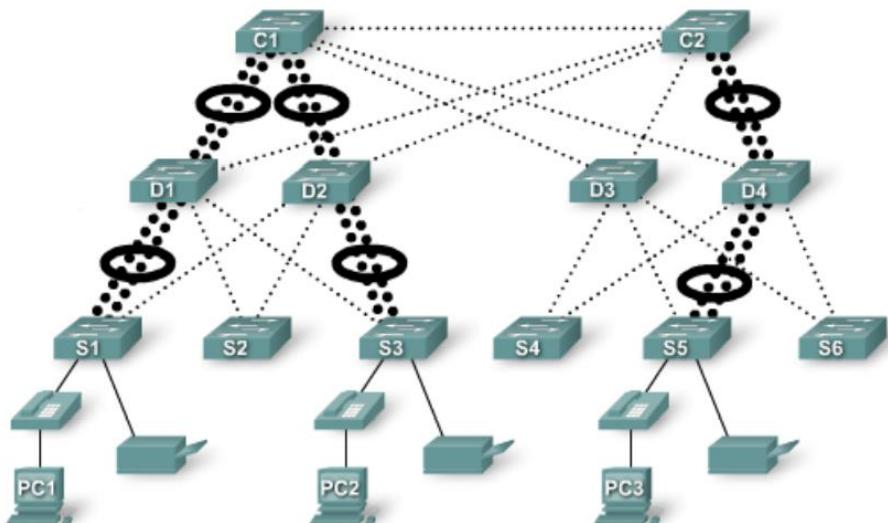
```
R1(config)#int fa0/0
R1(config-if)#ip helper-address 192.168.11.254 [indirizzo server]
```

Adesso quando PC1 manda una richiesta DHCP al router, questi la inoltrerà al server che prenderà un indirizzo da assegnare dal pool definito prima

First Hop Redundancy

Sui link del livello di accesso il traffico riguarda il singolo utente, ai livelli distribuzione e core viaggia il **traffico aggregato** cioè da/per più host, nei link dei livelli in alto si ha una quantità di traffico multipla di quella configurata al livello di accesso:

- Lo switch di accesso può avere porte per il collegamento con il link di distribuzione più veloci, ma la proprietà è legata all'hardware dello switch ed è fissa quindi non si può usare un fattore di scala diverso
- **Link Aggregation:** per poter avere più flessibilità si possono utilizzare più porte per un singolo collegamento tra due switch:



Esempio: collego D1 con C2 con due porte su ogni switch in questo modo se ogni porta è 1Gb/s ottengo un collegamento a 2 Gb/s, se lo facessi solo fisicamente spanning tree rileverebbe il percorso chiuso quindi devo fare in modo che dal punto di vista logico questi appaiano come un singolo collegamento, in questo modo spanning tree vedrà una sola porta logica su D1 e una su C2 con link a 2Gb/s

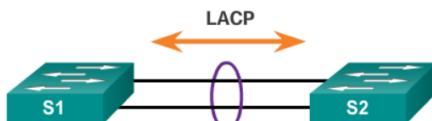
Link Aggregation Control Protocol (LACP)

- Gli switch negoziano scambiando pacchetti LACP: si scambiano i messaggi con dei keep alive per verificare che il link logico sia sempre attivo
- Si può fare su massimo 8 porte
- Se una porta si guasta il collegamento rimane, semplicemente degrada da 2Gb/s a 1Gb/s

Modi

- No:** gli switch ignorano il fatto che sono connessi su più porte (no LACP)
- Attivo:** lo switch prende l'iniziativa per la negoziazione
- Passivo:** lo switch aspetta l'altro per la negoziazione

Se almeno uno di due switch non è attivo quindi non parte la negoziazione



| S1 | S2 | Channel Establishment |
|-------------------|----------------|-----------------------|
| On | On | Yes |
| Active/Passive | Active | Yes |
| On/Active/Passive | Not Configured | No |
| On | Active | No |
| Passive/On | Passive | No |

Configurazione link logici

```
S1(config)#interface range fa0/1-2
S1(config-if-range)#channel-group 1 mode active
Lo switch crea una nuova porta logica port-channel-1
```

D'ora in poi la configurazione viene fatta non solo sulle due singole porte fa0/1 e fa0/2, ma sulla porta virtuale che comprende le due porte fisiche.

Ridondanza sul default gateway

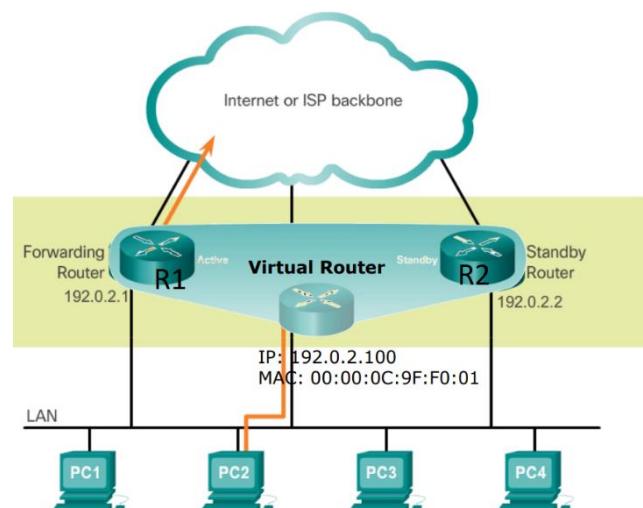
Con una singola VLAN un PC può avere un solo indirizzo come default gateway ed è quindi un single point of failure, possiamo avere però R2 come **gateway di backup**.

Router redundancy: a PC1 non si assegna come gateway né 192.0.2.1 né 192.0.2.2, ma un nuovo indirizzo IP 192.0.2.100 appartenente a un **router virtuale (VR)**

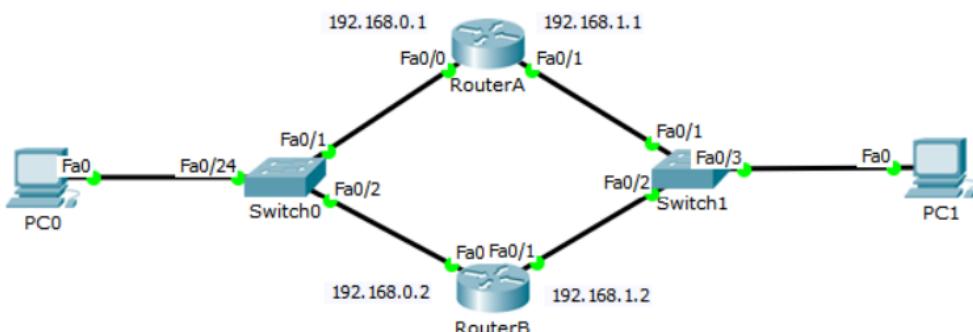
In condizioni normali il R1 impersona il router virtuale usando anche questo indirizzo: quando PC1 manda un pacchetto ARP chiedendo il MAC di **VR** gli risponde il router R1 con un MAC particolare associato a VR.

Quando **R1 cade allora è R2 che impersona il router virtuale** e risponderà alle richieste per 192.0.2.100.

Virtual Router Redundancy Protocol (VRRP): per gestire le interazioni tra R1 e R2, che devono mettersi d'accordo sull'assumere il ruolo di router virtuale, viene eletto un router come principale e l'altro fan da backup



Hot Standby Router Protocol (HSRP) - versione proprietaria CISCO
(configurazione per interfaccia perché riguarda una VLAN):



```
RouterA(config)#interface Fa0/0
RouterA(config-if)#ip address 192.168.0.1 255.255.255.0
RouterA(config-if)#standby 1 ip 192.168.0.254
RouterA(config-if)#standby 1 priority 120
RouterA(config-if)#standby 1 preempt
RouterA(config-if)#no shutdown
```

Lo stesso va poi fatto per il router RouterB; in questo modo avendo cambiato la priorità del router A lo forziamo ad essere il gateway principale e B sarà quello di backup (il valore standard della priorità è 100).

Gateway Load Balancing Protocol (GLBP): versione più avanzata che fa load balancing tra i due router

Wide Area Networks (WAN)

Componente di una rete privata (aziendale) che si estende su un'area geografica relativamente ampia; il privato che vuole realizzare la rete deve rivolgersi a un **provider** che fornisce connettività tra uno e più punti dislocati geograficamente. Il provider ha una propria infrastruttura di rete per mezzo della quale fornisce servizi di connettività a terzi.

Topologie

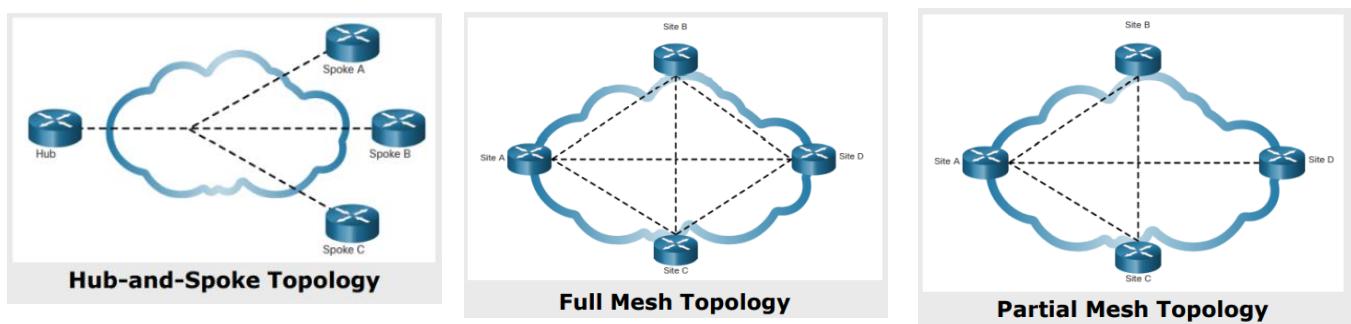
La scelta dipende dal numero di sedi da connettere

▪ Due siti

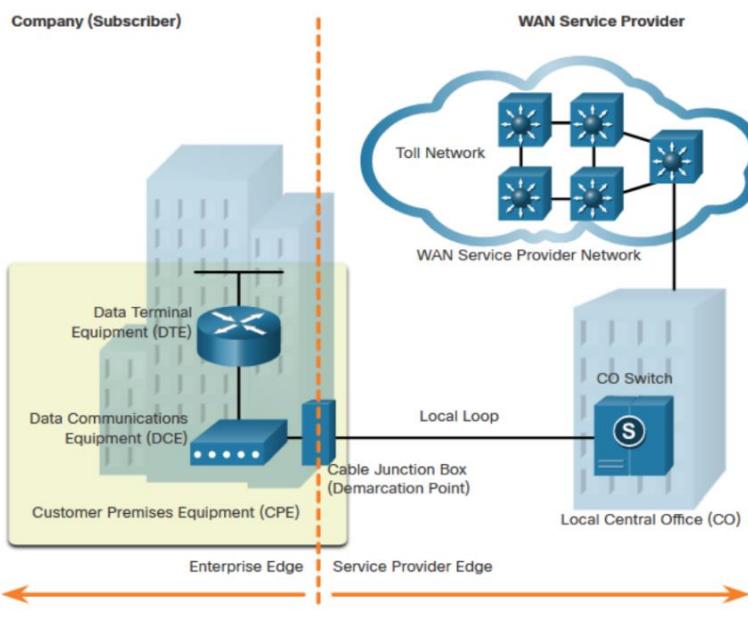
- **Punto-punto:** caso più semplice con comunicazione seriale, il mezzo fisico è la rappresentazione del servizio di connettività fornito dal provider

▪ Più siti

- **Hub and Spoke:** un sito fa da hub per gli altri (detti spoke); qualsiasi comunicazione tra siti diversi non hub deve essere intermediata dall'hub stesso
- **Full mesh:** tutti i siti sono connessi con tutti gli altri
- **Partial mesh:** non tutti sono connessi con tutti, ma è garantita la connettività tra due siti



Terminologia



Punto di demarcazione: punto di confine tra il **Data Terminal Equipment (DTE)** e il confine della sede aziendale, oltre questo inizia la parte di rete sotto responsabilità del provider.

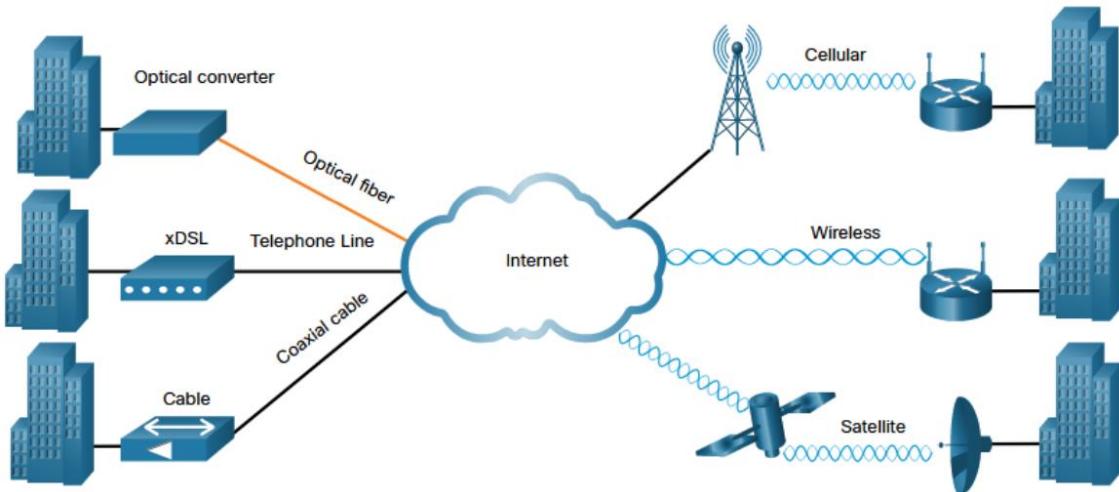
Local Loop: collegamento verso il **Local Central Office**, cioè il punto di accesso locale del fornitore del servizio, all'interno la rete del provider può essere realizzata con diverse tipologie.

Il local loop è un mezzo fisico e da esso dipende il tipo di comunicazione, per questo tra il DTE e il punto di demarcazione c'è il **Data Communications Equipment (DCE)** la cui funzione esatta dipende dal local loop.

Dal lato **local loop** il DCE può essere:

- Modem DSL: come nelle reti domestiche, sfrutta il collegamento per la comunicazione telefonica
- Voiceband model
- Cable Modem: come DSL usa il collegamento per la comunicazione televisiva
- Optical converter: connette un mezzo a fibra ottica a uno a rame, quindi converte segnali ottici in impulsi elettrici, il local loop è a fibra
- CSU/DSU: segnale digitale
- Wireless router: il local loop è wireless

DCE Layer: livello dove viene assorbita la eterogeneità del collegamento verso il provider (wireless, wired, analogico, digitale, ecc.), facendo in modo che dal punto di vista della rete dell'utente **quello che va configurato è sempre un router**



Servizi WAN tradizionali per la realizzazione della connettività

- **Linee dedicate**
 - **Leased line:** si noleggia un collegamento a velocità fissa (tipicamente doppino in rame con trasmissione digitale DSU/CSU) → *T1/E1 e T3/E2*
- **Linee a commutazione (switched)**
 - **Circuit-switched:** (comunicazione telefonica) si crea un circuito dedicato alla comunicazione per tutta la sua durata → *PSTN e ISDN*
 - **Packet-switched:** scambio di pacchetti → *Frame Relay e ATM*

Servizi WAN moderni per la realizzazione della connettività

Le reti dei provider si sono evolute, prima la comunicazione riguardava i livelli 1 o 2 adesso le reti sono basate interamente sul livello 2 o 3

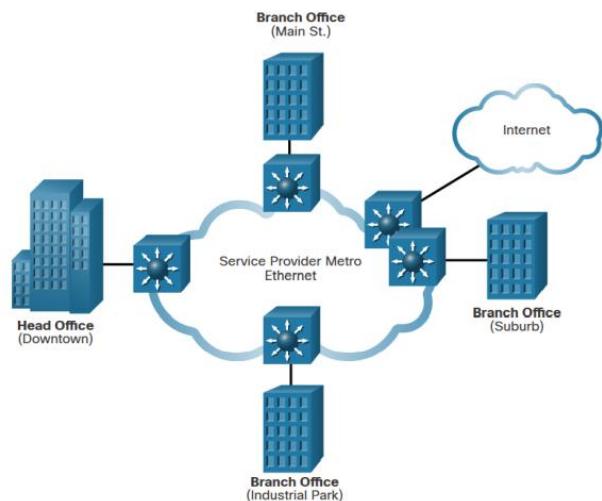
1. **Dedicated broadband:** l'infrastruttura è messa a disposizione del cliente ed è disponibile sempre
 - a. **Dark fiber:** fibra ottica non illuminata noleggiata al cliente
2. **Switched:** solo packet-switched
3. **Internet-based:** si realizza una connessione ad internet e su questa si mette un collegamento privato di tipo virtuale

1. Commutazione di pacchetto

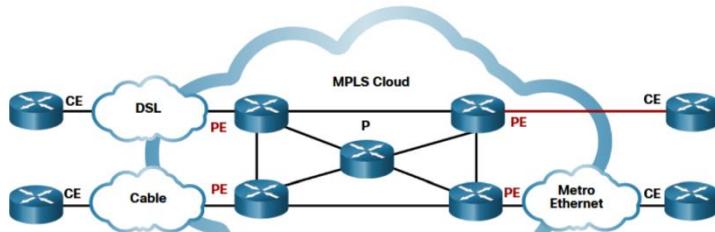
Come cliente a valle vedo solo uno switch (L2VPN) o un router (L3VPN), tutto il resto è configurato dal provider.

Ethernet WAN (L2VPN): servizio di connettività (livello 2) fornito dal provider, al confine di ogni rete l'utente collega uno switch ethernet, questi vengono connessi alla rete del provider e la rete del provider si comporta come se fosse un unico switch virtuale che interconnette gli switch fisici; questi switch si scambiano roba e si vedono connessi.

Realizzo una unica ethernet anche se singoli rami di questa ethernet sono distribuiti (ciascun sottoalbero avrà il suo livello di accesso e distribuzione) e gli switch che connettono al provider fanno parte del livello core (realizzati da tecnologie diverse: metro ethernet, ethernet over MPLS, Virtual Private LAN Service).



MPLS/BGP VPNs (L3VPN): servizio fornito dal provider livello 3, CE è il punto di ingresso-uscita dalla rete dell'azienda, ogni CE router è connesso a un PE (provider edge) router con DSL, fibra, metro ethernet, ecc. La rete del provider si comporta come se fosse una specie di singolo router virtuale; se ad esempio CE in alto a sinistra annuncia con OSPF le rotte, la rete del provider fa in modo che questi annunci arrivino a tutti gli altri router dello stesso cliente.



Ovviamente si gestiscono più clienti contemporaneamente, ma i pacchetti di un cliente arrivano solo nella rete di quel cliente. Il comportamento del provider equivale a quello di una rete dedicata ma solo a livello virtuale.

2. Internet-Based connectivity

Il servizio fornito dal provider non è un servizio VPN, ma un servizio di connessione a internet (quindi fornito da un ISP). Per realizzare i collegamenti geografici privati è l'azienda che deve realizzare una astrazione di rete privata virtuale sopra a questo insieme di collegamenti a internet, tramite dei protocolli specifici. La connessione può essere effettuata tramite:

- **Mezzi wired:** (questo si riferisce comunque a ciò che il cliente vede fino al local loop)
 - Doppino in rame (xDSL)
 - Fibra ottica (FTTH)
- **Mezzi wireless:**
 - Reti cellulari
 - WiFi municipali
 - Satellite

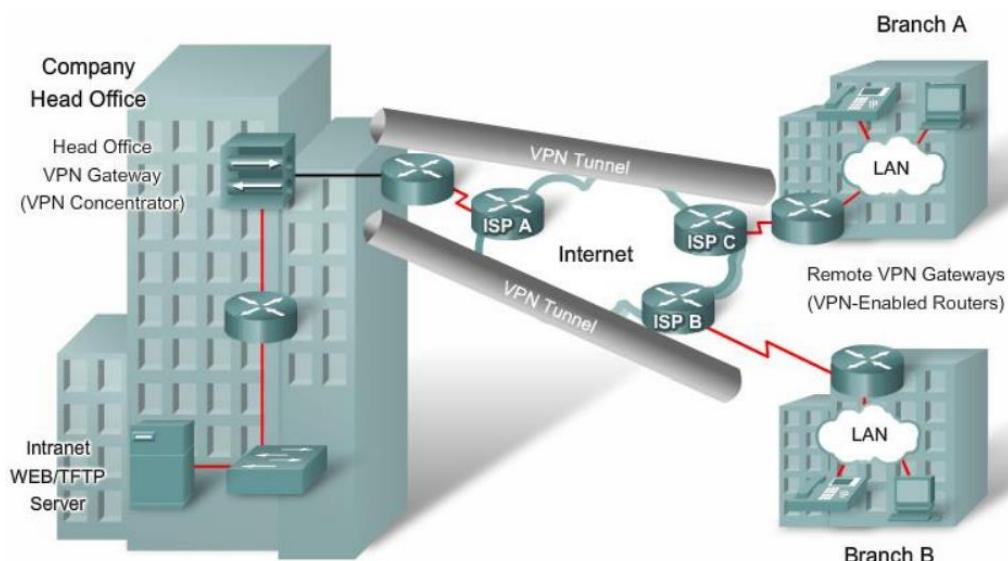
Virtual Private Network (VPN)

Una rete aziendale può voler essere connessa a Internet, ad esempio per scambiare pacchetti diversi, connettendo tutte le reti aziendali a internet viene a mancare la privatezza perché i pacchetti che vanno da una sede all'altra attraversano la rete pubblica.

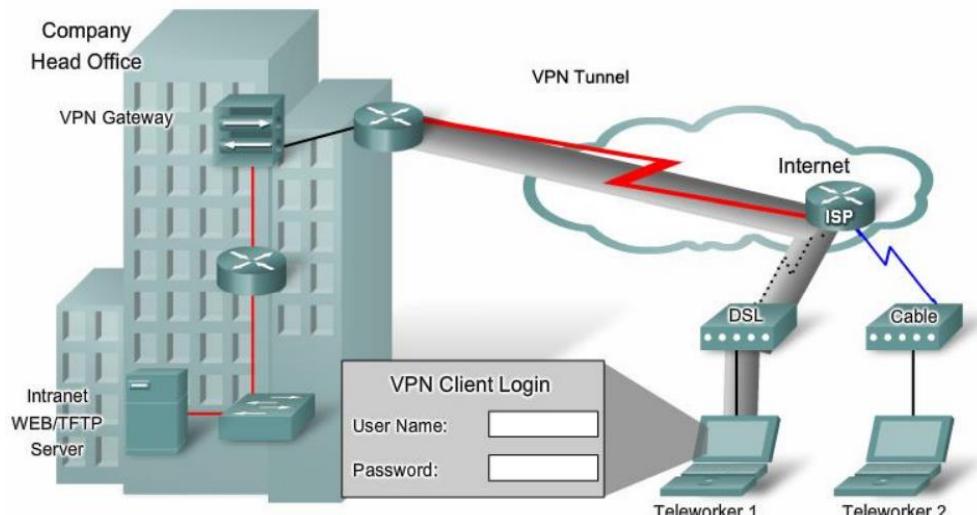
Sulla infrastruttura Internet si realizza una **astrazione di rete privata virtuale** attraverso protocolli **dalla rete aziendale stessa**: gli host fanno in modo che i dati scambiati abbiano determinate caratteristiche

- Costi inferiori rispetto ai servizi di connettività dedicate
- Migliore scalabilità
- Sicurezza
- Flessibilità: ovunque siano gli host, se connessi a Internet possono accedere alla parte privata della rete

1. Site-to-site VPN: vengono realizzati collegamenti virtuali **VPN tunnel** a due a due. Il singolo host è ignaro che sta usando una VPN e vede solo una rete privata con i router delle sedi; anche lo spazio di indirizzamento viene visto come privato



2. Remote access VPN: il singolo host è consapevole dell'uso della VPN perché si collega tramite internet alla rete aziendale usando un software client VPN



Protocollo IPSEC (livello network)

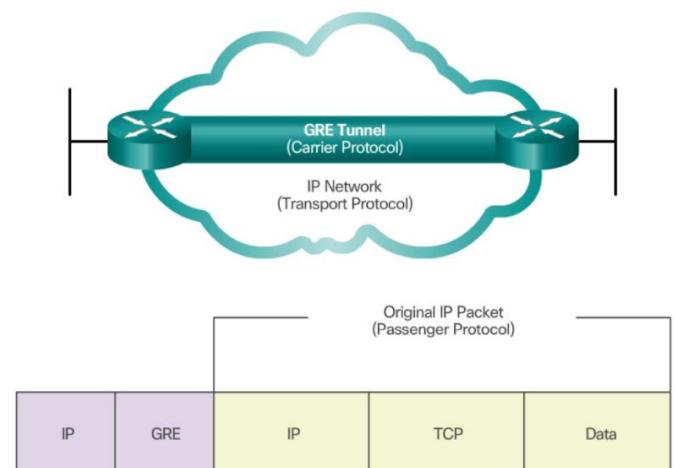
Estensione del protocollo IP che aggiunge degli header ai pacchetti per implementare:

- Confidentialità – contenuto del pacchetto cifrato
- Integrità – il contenuto non può essere alterato
- Autenticazione – conferma dell'identità delle parti che comunicano

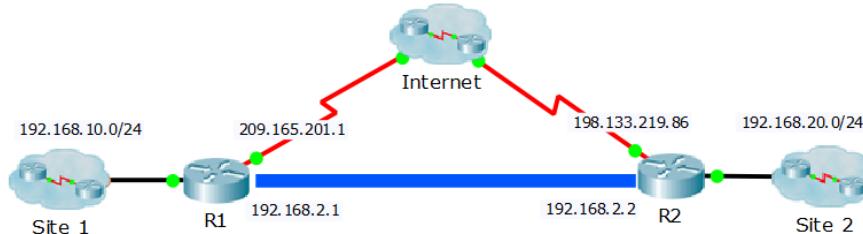
Nella rete privata i pacchetti circolano in formato IP originale, nella rete internet invece quelli IPsec

Il primo header IPsec trasporta il pacchetto **Generic Routing Encapsulation (GRE)** che è il **protocollo di trasporto**: crea la pipe all'interno della quale saranno scambiati i pacchetti della rete privata.

- Il pacchetto GRE è composto da header e payload IP
- Gli indirizzi nel pacchetto GRE sono indirizzi di host all'interno della rete privata
- Tra i campi c'è il protocol type per poter trasportare sia pacchetti IPv4 sia IPv6 e i flags per creare i sottotunnel



[Packet Tracer]



- Azienda con 2 siti: *Site1* e *Site2*
- Ciascuno di questi ha un router di confine che è connesso alla rete internet: *R1* e *R2*
- *R1* è connesso a Internet con l'interfaccia seriale e usa l'ip fornito dal provider 209.165.201.1
- Lo stesso vale per *R2* che ha IP 192.133.219.86

Vogliamo realizzare un **collegamento privato virtuale** tra *R1* e *R2*, a questo link assegniamo degli indirizzi IP distinti che devono far parte del piano di indirizzamento privato dell'azienda: 192.168.2.1 (*R1*) e 192.168.2.2 (*R2*). Quando un host di *Site1* vuole mandare un pacchetto a un host di *Site2*, su *R1* deve esserci l'indirizzo di rete 192.168.20.0/24 con next hop *R2* cioè 192.168.2.2

Si crea una interfaccia virtuale su *R1* che realizza il collegamento con *R2*:

```
R1(config)# interface Tunnel [id]
```

Si indica come i pacchetti viaggiano nel tunnel:

```
R1(config-if)# tunnel mode [gre|ipv6] [transport protocol]
R1(config-if)# ip address 192.168.2.1 255.255.255.252
```

Si indicano le estremità del tunnel:

```
R1(config-if)# tunnel source [interface id]
R1(config-if)# tunnel destination [indirizzo IP R2]
```

Adesso R1 se deve inoltrare qualcosa tramite l'interfaccia Tunnel 0 sa che deve incapsularla con GRE in un pacchetto IP e poi mandarlo sulla interfaccia seriale, sapendo che deve avere come destinazione l'indirizzo specificato

La stessa cosa va fatta anche sul router R2

```
R2(config)#interface tunnel 0
R2(config-if)# tunnel mode gre ip
R2(config-if)# ip 192.168.2.2 255.255.255.252
R2(config-if)# tunnel source serial 0/0/0
R2(config-if)# tunnel destination 209.165.201.1
```

R2# show interface tunnel 0: mostra le informazioni sull'interfaccia virtuale tunnel 0

Facendo ip route show si vede che R2 è direttamente connesso alla sottorete 192.168.2.0 tramite l'interfaccia Tunnel 0; adesso si può fare tutto ciò che si fa in una rete normale considerando R1 e R2 come se fossero direttamente connessi.

R2# ping 192.168.2.1 – ping da R1 a R2 usando gli indirizzi IP privati (attraverso il tunnel)

1. Viene generato il pacchetto IP con destinazione 192.168.2.1
2. R2 guarda la tabella di routing e vede che è direttamente connesso a destinazione tramite l'interfaccia Tunnel 0
3. Viene impostato come indirizzo sorgente quello dell'interfaccia Tunnel 0: 192.168.2.2
4. Il pacchetto viene inoltrato su Tunnel 0
5. Da qui il pacchetto viene incapsulato in GRE:
 - a. Flag: 0
 - b. Protocol type: 0x0800 (IPv4)
6. Il pacchetto GRE viene incapsulato nel pacchetto IP con sorgente 198.133.219.86 e come destinazione 209.165.201.1
7. Quando R1 lo riceve vede che dentro c'è GRE e può ricostruire il pacchetto per l'invio della risposta

Per connettere i due siti nelle reti private Site1 e Site2 dobbiamo configurare il protocollo di routing

```
R1(config)# router ospf1
R1(config-router)# router-id 0.0.0.1
R1(config-router)# network 192.168.10.0 0.0.0.255 area 0
R1(config-router)# network 192.168.2.0 0.0.0.3 area 0
```

Quando si fa partire OSPF su R2 verrà creata un'adiacenza punto-punto sul link virtuale

```
R2(config)# router ospf1
R2(config-router)# router-id 0.0.0.2
R2(config-router)# network 192.168.20.0 0.0.0.255 area 0
R2(config-router)# network 192.168.2.0 0.0.0.3 area 0
```

Adesso R2 inizia a mandare pacchetti di hello sul link virtuale (Tunnel 0) e nella tabella di routing di R2 compare la rotta 0 192.168.10.0/24 [10, 1001] via 192.168.2.2, quindi R2 può raggiungere la rete privata Site1

Access Control Lists

Packet filtering: vengono impostati dei filtri sui router per decidere quale traffico deve essere ammesso o meno, il filtraggio avviene nel piano dati, non nel piano di controllo perché non si filtrano gli annunci dei router e simili. Le regole vengono **applicate a ogni singolo pacchetto** per decidere se farlo proseguire nell'inoltro oppure scartarlo.

Le regole di filtraggio dei router sono definite dalle **Access Control Lists (ACL)**, il filtraggio in questo caso avviene a livello 3 e le regole si basano su:

- IP destinazione e sorgente
- Porta di destinazione e sorgente
- Protocollo

Per ogni regola viene fatta la verifica di matching, quindi in base all'azione associata alla regola (**ALLOW** o **DENY**) il pacchetto viene scartato o inoltrato.

I router solitamente consentono di definire le regole e applicarle sia in ingresso che in uscita: un pacchetto che attraversa un router può essere controllato sia sull'interfaccia di ingresso che su quella di uscita; si può definire al più una ACL per interfaccia, per direzione, per protocollo.

Motivazioni:

1. Eliminare traffico non consentito
2. Eliminando traffico non consentito riducendo il traffico
3. Sicurezza

ACL operation

1. Il pacchetto arriva all'interfaccia
2. Se fa match l'indirizzo di livello 2 dell'header allora il pacchetto viene processato dal router
3. Si verifica se c'è una ACL per l'interfaccia di ingresso
 - a. Si verifica se la prima regola fa match col pacchetto, se sì si procede con l'azione corrispondente e le regole successive vengono ignorate
 - b. Se la regola non fa match si passa alla regola successiva
 - c. Così finché non si trova una regola che fa match o si arriva all'ultima senza trovare un match e quindi viene applicata **l'azione di default (DENY)**
4. Si cerca l'interfaccia di inoltro
5. Si verifica se c'è una ACL per l'interfaccia di uscita e si procede come in ingresso

Quando si mette una ACL è necessario definire **tutti i casi possibili in cui il pacchetto può andare avanti**, altrimenti verrà scartato; ovviamente deve esserci almeno un PERMIT

Cisco IOS ACLs

1. **ACL standard:** regole basate esclusivamente sull'ip sorgente del pacchetto

```
Access-list 10 permit 192.168.30.0 0.0.0.255
```

2. **ACL estese** operazioni di controllo basate su:

- IP sorgente e destinatario
- Porta sorgente e destinatario
- Tipo di protocollo

```
Access-list 130 permit tcp 192.168.30.0 0.0.0.255 any eq 80: sono permessi tutti i pacchetti per 192.168.30.0/24 indirizzati a qualsiasi server web (porta 80)
```

Identificazione

1. **Numbered ACL:** identificate tramite un id che indica anche il tipo
 - a. 1-99 e 1300-1999: standard
 - b. 100-199 e 2000-2699: estese
2. **Named ACL:** identificate tramite nome, il tipo va definito esplicitamente

ACL placement

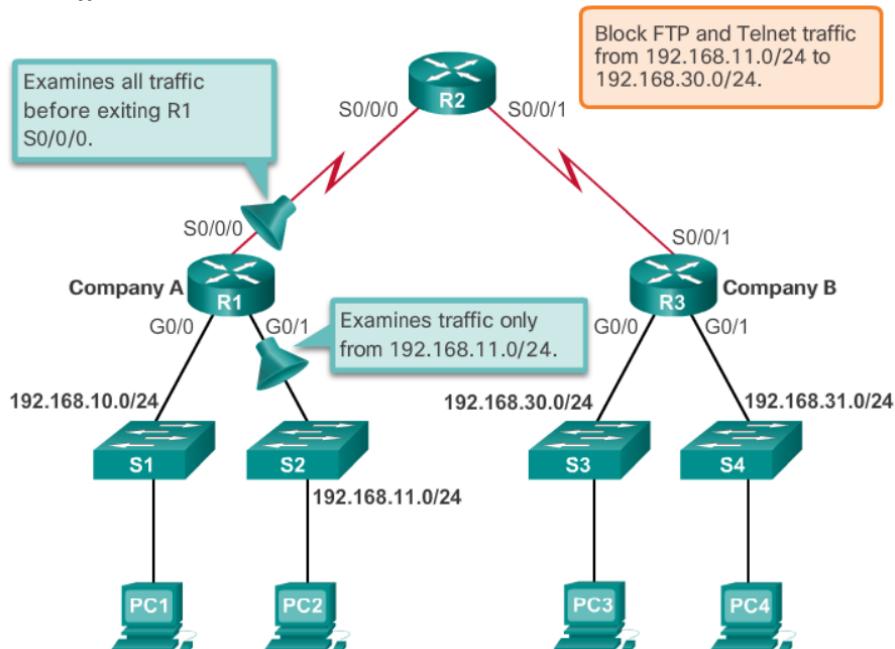
Le **ACL standard** sono solitamente piazzate **il più vicino possibile alla destinazione**

- ACL standard filtra i pacchetti in base all'IP sorgente, quindi se sono vicino alla sorgente blocca la comunicazione di quell'host verso qualsiasi destinazione
- Le ACL standard sono "rigide" quindi se mi avvicinassi troppo alla sorgente potrei bloccare più "comunicazioni" del dovuto

Le **ACL estese** sono solitamente piazzate **il più vicino possibile alla sorgente**

- Essendo regole più complesse cerco di trovare subito pacchetti da scartare in modo da ridurre il traffico di pacchetti inutili

Esempio: bloccare il traffico FTP e telnet dalla rete 192.168.11.0/24 alla rete 192.168.30.0/24



Deve essere bloccato il traffico in uscita dal sorgente con porta di destinazione (o sorgente) 20 o 21, quindi mettiamo la ACL tra S2 e R1

- Si riduce il traffico evitando che R1 invii pacchetti che verranno poi scartati da R2 e R3
- Se invece applicassimo la regola su R2 questa verrebbe applicata anche ai pacchetti originati da 192.168.10.0/24 quindi ad un volume di traffico maggiore causando overhead

ACL Standard

Il matching è basato solo sull'indirizzo sorgente del pacchetto, dopo l'azione si specifica il prefisso della rete sorgente (prefisso a 32b per indicare un singolo host). L'ordine delle regole dell'ACL coincide con l'ordine con cui sono stati eseguiti i comandi.

Sintassi: access-list [id] [PERMIT|DENY] [source] [source wildcard mask]

esempio:

```
access-list 2 deny host 192.168.10.1
access-list 2 permit 192.168.10.0 0.0.0.255
access-list 2 deny 192.168.0.0 0.0.255.255
access-list 2 permit 192.0.0.0 0.255.255.255
```

quando arriva un pacchetto:

1. Se proviene dall'host 192.168.10.1 - DENY
2. Altrimenti, se proviene da 192.168.10.0/24 - PERMIT: va bene se il pacchetto proviene dalla stessa sottorete di quell'host ma non dall'host specifico
3. Altrimenti, se proviene da 192.168.0.0/16 – DENY: se viene dalla sottorete più piccola viene fatto passare dalla regola 2
4. Altrimenti, se proviene da da 192.0.0.0/8 - PERMIT

Mask keyword

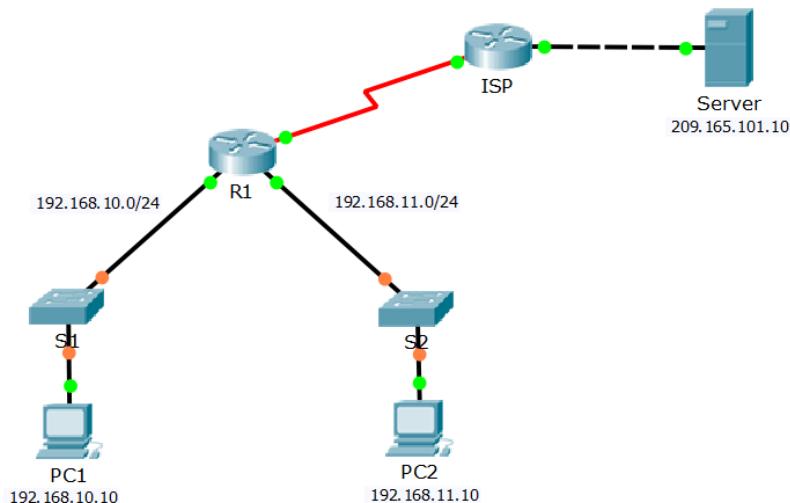
- **any:**
 - access-list 1 permit 0.0.0.0 255.255.255.255
 - access-list 1 permit **any**
- **host:**
 - access-list 1 permit 192.168.10.10 0.0.0.0 è equivalente a
 - access-list 1 permit **host** 192.168.10.10

Per configurare l'ACL si entra nel modo di configurazione della interfaccia dove deve essere applicata:

R(config-if)# ip access-group [access-list-number | access-list-name] [in|out]

*in: il pacchetto sta entrando nel router da quella interfaccia
out: sta uscendo dal router su quella interfaccia*

[Packet Tracer]



show access-list mostra le ACL definite con le relative regole e il numero di volte che sono state applicate

Nota: la prima regola di una access list ha numero di ordine default 10, poi le successive avranno 20, 30, ecc. in modo da permettere l'editing della ACL per inserire una nuova regola tra quelle già presenti specificando il numero di sequenza.

Configuro una ACL sulla interfaccia seriale di R1:

```
R1(config)# access-list 1 permit 192.168.10.0  0.0.0.255
R1(config)# interface serial0/0/0
R1(config-if)# ip access-group 1 out
```

Adesso la ACL è diventata operativa quindi, dopo il routing, quando R1 deve trasmettere sulla seriale 0 applica la access-list 1.

- Il ping da PC1 al server ha successo perché PC1 è nella rete 192.168.10.0/24
- Il ping da PC1 a PC2 ha successo perché l'ACL non viene applicata visto che il pacchetto viene inoltrato da R1 sulla interfaccia fast ethernet e non sulla seriale.
- Il ping da PC2 a server non ha successo, ogni pacchetto viene bloccato da R1 che genera un messaggio ICMP di destination host unreachable e lo invia a PC2

Editing: vogliamo che tutti gli host possano comunicare con l'esterno, eccetto l'host 192.168.11.10, per bloccare l'host non si può aggiungere una regola successiva perché i pacchetti da lui inviati faranno sempre matching con la prima regola quindi si cancella la lista e la si ridefinisce

Configurazione access list con nome

```
R1(config)# ip access list [standard-extended] [nome]
```

A differenza del caso precedente si entra nel modo di configurazione della access list dove poi aggiungiamo le regole con la sintassi vista prima, anche qui poi applichiamo la ACL sulla interfaccia

```
R1(config-std-nacl)# permit 192.168.0.0 0.0.255.255
R1(config-if)# ip access-group [nome] out
```

Per aggiungere una regola con numero di sequenza specifico si mette il numero all'inizio del comando:

```
R1(config-std-nacl)# [seq-num] deny host 192.168.10.10
```

Extended ACL

```
R(config)# access-list access-list-number [deny | permit] protocol source
      [source-wildcard] [operator [port-number | name]] destination
      [destination-wildcard] [operator [port-number | name]] [established]
```

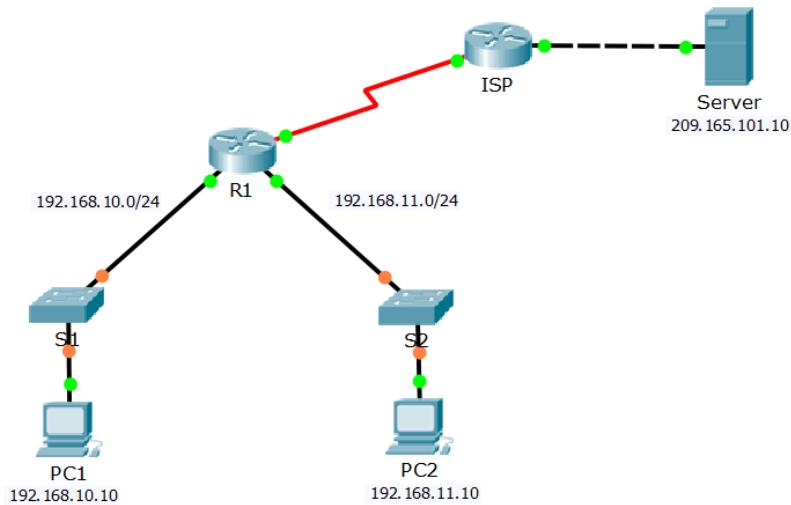
- **Protocol:** ICMP, IP, TCP, UDP
- **Source:** indirizzo IP e wildcard mask
- **Porta** (se TCP/UDP): se non specificato la porta è indifferente
Può essere specificata con keyword per servizi specifici: telnet 23, ftp 21, ftp-data 20, http 80
- **Operator:** ls (less than), gt (greater than), eq (equal), neq (not equal), range

Esempio

```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 23
```

Fai passare pacchetti TCP originati dalla rete 192.168.20.0/24 (la porta è assente quindi qualsiasi va bene), destinato a qualunque indirizzo di destinazione con porta 23 (telnet).

[Packet Tracer]



“Gli unici host che possono accedere a un servizio web esterno alla rete aziendale sono quelli nella sottorete 192.168.10.0/24”

```
R1(config)#ip access-list extended WEB
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any (*) eq www (web)
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any (*) eq 443 (https)
R1(config)# interface serial0/0/0
R1(config-if)# ip access-group WEB
```

Per il resto vale il deny implicito: qualsiasi pacchetto in uscita dalla rete aziendale che non sia mandato da quella sottorete verso una porta per il servizio web viene scartato

() applicando la ACL sulla seriale in uscita di R1 possiamo mettere any perché l’interfaccia è connessa verso la rete esterna, quindi stiamo vietando il traffico verso tutti gli indirizzi esterni*

- Ora se da PC1 si usa il browser ci si può connettere al server
- Il ping da PC1 al server non ha successo perché è un pacchetto ICMP e viene filtrato
- Avendo messo la regola in uscita dalla seriale, i pacchetti che arrivano da internet passano tutti perché non ho alcuna regola in ingresso a R1.
- Il ping da Server a PC2 non ha successo, l'echo request arriva a PC2 è la echo reply di PC2 a essere filtrata da R1

Se il requisito fosse stato quello di far passare sia in ingresso che in uscita solo il traffico per il servizio web esterno dalla rete 192.168.10.0/24 allora sarebbe stato necessario mettere la ACL in entrambe le direzioni per fare in modo che sul link seriale passino solo pacchetti relativi al servizio web destinato all'esterno.

Keyword “established”: il pacchetto fa **matching se è parte di una connessione TCP già esistente**, in questo modo si permette di filtrare tutto il traffico in ingresso alla rete privata ad esclusione dei pacchetti di risposta alle richieste http ammesse dalla ACL.

Mettendo nella ACL in ingresso la regola `permit tcp any 192.168.10.0 0.0.0.255 established` facciamo in modo che vengano fatti passare i pacchetti TCP di una connessione già stabilità che quindi sono necessariamente risposte alle richieste inviate dagli host della rete privata.

Network Address Translation (NAT)

Meccanismo che permette di utilizzare uno stesso indirizzo IP pubblico su più host diversi senza rinunciare alla connettività end-to-end, aumentando la dimensione dello spazio di indirizzamento. L'univocità dell'indirizzo IP non è più necessaria globalmente, ma solo **all'interno del dominio in cui viene utilizzato**.

Il problema si ha quando i pacchetti sono destinati a un host appartenente a un dominio diverso da quello del sorgente perché **gli indirizzi IP sorgente e destinazione sono interpretati correttamente solo in domini diversi**. Quando un pacchetto **attraversa il confine tra due domini** si usa un **meccanismo di traduzione: l'indirizzo che sta uscendo dal suo dominio di validità viene tradotto in un indirizzo valido nel dominio in cui sta entrando**, per riferire sempre lo stesso host (quando un pacchetto attraversa più domini possono essere tradotti entrambi gli indirizzi)

Traditional NAT

Coinvolge tipicamente una stub network dove i pacchetti o entrano e sono destinati a host interni o escono e sono destinati ad host esterni.

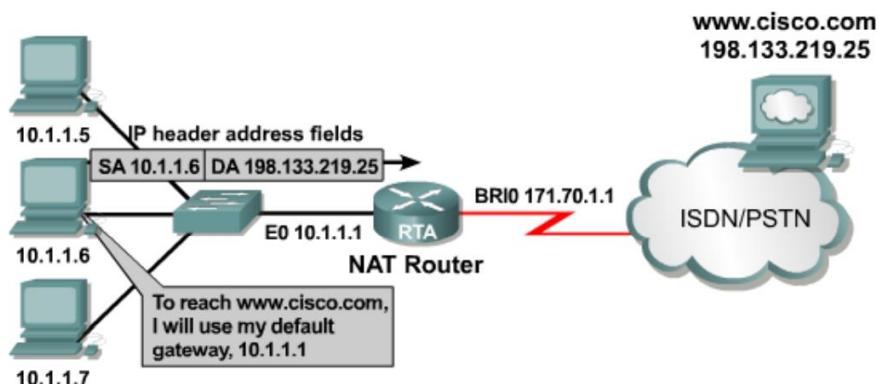
Si può identificare un router che rappresenta il **punto di ingresso/uscita** al confine tra i dominii, su questo avviene la traduzione degli indirizzi.

Quando il pacchetto esce dalla rete interna viene tradotto l'indirizzo sorgente, quando il pacchetto entra nella rete interna viene tradotto l'indirizzo di destinazione

Esistono due tipi di NAT:

- Basic NAT: traduce solo gli indirizzi IP
- NATP: traduce anche le porte

Basic NAT – esempio:



1. L'host della rete *inside* confeziona il pacchetto con indirizzo sorgente 10.1.1.6 e destinatario 198.133.219.25
2. Quando arriva al NAT router questo sostituisce l'**indirizzo sorgente** con 171.70.2.1
 - a. La traduzione avviene tramite la NAT table che mostra le corrispondenze
3. Il pacchetto col nuovo sorgente arriva a destinazione
4. Chi risponde manderà la risposta a 172.70.2.1
5. Quando arriva al NAT router questo sostituisce l'**indirizzo destinatario** con quello che trova nella tabella NAT cioè 10.1.1.6

Ogni host ha due indirizzi diversi in due reti diversi, questi indirizzi sono significativi solo nei corrispondenti dominii

Terminologia

Reti:

- **Inside network:** rete i cui indirizzi sono soggetti a traduzione
- **Outside network:** rete esterna

Indirizzi: la prima parola indica dove si trova l'host, la seconda dove viene usato l'indirizzo

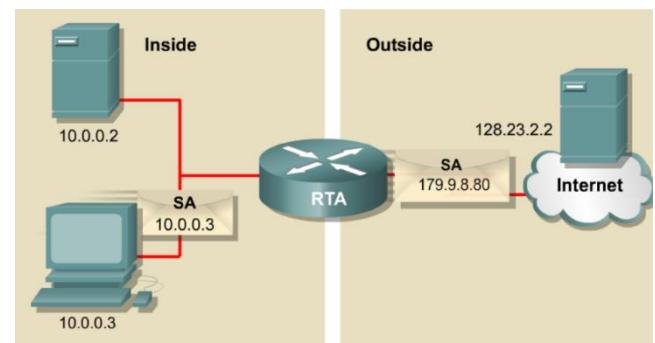
- **Inside local:** host nella rete inside, usato nella rete inside
- **Inside global:** host nella rete inside, usato nella rete outside
- **Outside local:** host nella rete outside, usato nella rete inside
- **Outside global:** host nella rete outside, usato nella rete outside

Esempio

10.0.0.3 = **inside local** – si riferisce all'host nella rete inside ed è valido nella rete inside, se lo uso nella rete esterna non ha alcun significato

179.9.8.80 = **inside global** – si riferisce allo stesso host nella rete inside ma viene usato nella rete esterna

128.23.2.2 = **outside global** – l'host si trova nella rete outside e chi si trova fuori usa questo indirizzo per riferirsi a esso, è anche **outside local**



Nel caso del Traditional NAT gli indirizzi outside global e outside local coincidono, questo perché nella rete outside si continuano ad assegnare IP univoci agli host

Come indirizzi inside local si usano indirizzi appartenenti a blocchi riservati, si ha la certezza che non ci siano collisioni perché questi non possono essere usati nel dominio global

- (A) 10.0.0.0 – 10.255.255.255
- (B) 172.16.0.0 – 172.31.255.255
- (C) 192.168.0.0 – 192.168.255.255

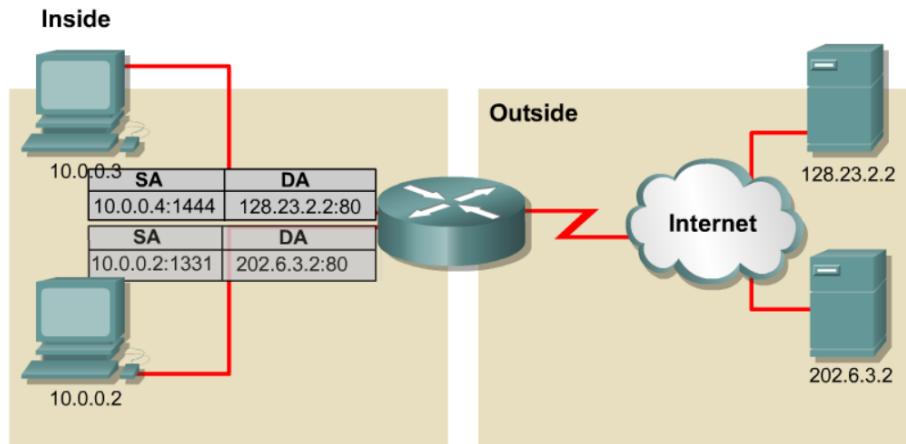
NAT Table: usate per tradurre uno specifico indirizzo inside local nel suo corrispondente inside global, la corrispondenza può essere configurata

- **statisticamente:** si inserisce manualmente nella tabella
- **dinamicamente:** la tabella viene riempita dal router creando le entrate quando servono, prendendo gli indirizzi inside global da un range specificato. Quando un indirizzo inside global non viene usato per certo periodo di tempo viene deallocated; se non ci sono più indirizzi disponibili il pacchetto non può essere tradotto e viene bloccato

Per i pacchetti che invece entrano nella rete (il cui destinatario va tradotto da inside global a inside local) si deve usare per forza la **configurazione statica**

NAPT

Se il pacchetto IP trasporta un pacchetto TCP/UDP si fa **overloading**: si usa lo stesso indirizzo inside global per riferirsi a host interni diversi **differenziando la corrispondenza sulla base del valore della porta**



179.9.8.20 viene usato sia per 10.0.0.2 che per 10.0.0.3 – quando il router riceve un pacchetto in ingresso con l'indirizzo di destinazione 179.9.8.20 guarda la porta per capire a quale dei due host è destinato

Inside Local IP

| |
|---------------|
| 10.0.0.2:1331 |
| 10.0.0.3:1555 |

Inside Global IP

| |
|-----------------|
| 179.9.8.20:1331 |
| 179.808.20:1555 |

Utilizzo del NAT – casi tipici:

- Collegamento di una rete aziendale alla rete internet globale (solitamente configurazione dinamica o ibrida)
- Load balancing
- Reti overlapping: collegamento di due reti dove originariamente sono stati usati gli stessi blocchi di indirizzi IP quindi si hanno due spazi di indirizzamento sovrapposti

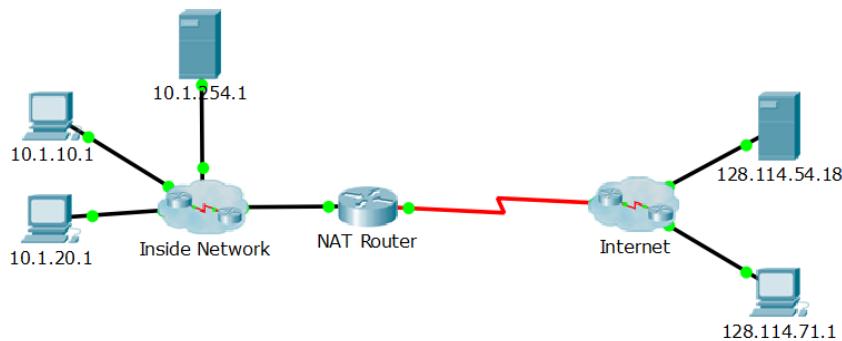
Vantaggi:

- ✓ Riutilizzo degli indirizzi IP
- ✓ Migliore flessibilità nella connessione con l'ISP

Svantaggi:

- ✗ Overhead: tutti i pacchetti devono essere elaborati dal router
- ✗ Creando domini di indirizzamento diversi la struttura della rete dell'altro dominio diventa invisibile perché mascherata dal meccanismo di traduzione
- ✗ Alcuni protocolli di rete prevedono nel funzionamento che parte dell'informazione scambiata dal protocollo trasporti gli indirizzi ip degli host e/o numeri di porta
 - Non essendo consapevole della traduzione, se un host interno deve comunicare il proprio indirizzo a uno fuori gli comunicherà ciò che vede e cioè l'indirizzo inside local che per l'host che si trova fuori non ha significato
 - Perché i protocolli continuano a funzionare è necessario che i router NAT siano in grado di fare una traduzione consistente **anche nel payload del pacchetto**

[Packet Tracer]



- I pacchetti che dalla rete outside viaggiano verso la rete inside hanno indirizzo di destinazione inside global, il router ISP deve essere configurato in modo da inoltrare al NAT Router i pacchetti che hanno un indirizzo appartenente al pool specificato
 - Sul router ISP è meglio specificare l'indirizzo del blocco invece che usare la default route
- Quando il pacchetto arriva al NAT Router **prima avviene la traduzione, poi l'inoltro**: nella tabella di routing non ci sarà l'entrata per l'indirizzo inside global perché viene sempre tradotto prima di consultarla

Nota: quando si definisce un pool si definisce una lista di indirizzi, per semplicità si usa un blocco allineato

1. Traduzione statica

```
Router(config)# ip nat inside source static [ip inside local] [ip inside global]
```

Si indica che per i pacchetti che arrivano dall'inside network si deve tradurre l'indirizzo sorgente con quello specificato. Devo poi dire al router **dov'è il confine della rete inside/outside** in modo che possa capire quando deve fare la traduzione: devo **marcare l'interfaccia del router come inside o outside**

```
Router(config)# interface serial0/0/0
Router(config-if)# ip nat outside
Router(config)# interface fa0/0
Router(config-if)# ip nat inside
```

- show ip nat translation: mostra il contenuto della tabella NAT
- show ip nat statistics: mostra informazioni statiche sulla traduzione

2. Traduzione dinamica

La lista degli indirizzi inside local da tradurre si definisce con una **ACL**, mentre il pool degli indirizzi va definito a parte

```
Router(config)# ip nat pool [nome pool] [primo ip] [ultimo ip] netmask [maschera]
Router(config)# ip access-list standard [nome lista]
Router(config-std-nacl)# permit 10.1.10.0 0.0.0.255
Router(config-std-nacl)# permit 10.1.20.0 0.0.0.255
Router(config)# ip nat inside source list [nomelista] pool [nomepool]
```

La traduzione NON BLOCCA I PACCHETTI!: se un pacchetto non fa match con le regole messe nella ACL non viene bloccato, ma semplicemente **non viene tradotto**.

`clear ip nat translation *:` elimina tutte le entrate della tabella NAT

3. Port Translation

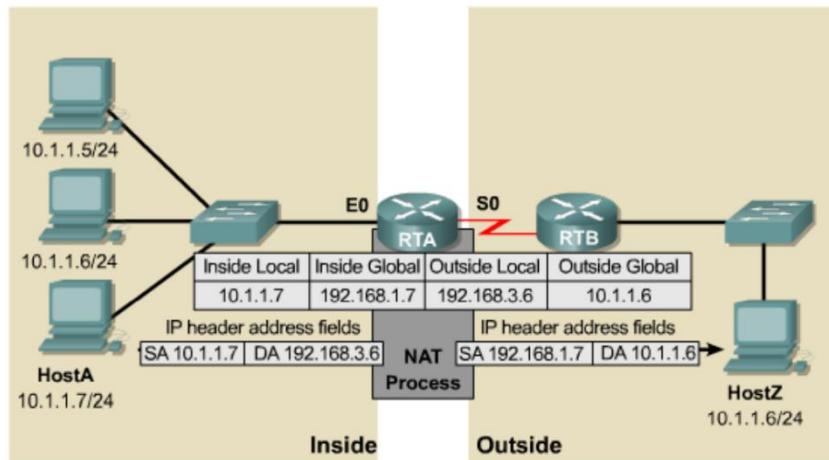
Adesso quando i due host della inside network mandano un pacchetto vengono tradotti i loro indirizzi con due indirizzi inside global diversi presi dal pool.

```
Router(config)#no ip nat inside source list NAT-LIST-INSIDE pool NAT-POOL-A
Router(config)#ip nat inside source list NAT-LIST-INSIDE pool NAT-POOL-A overload
```

In questo modo il router sa che può usare lo stesso indirizzo global più volte sfruttando il numero di porta per distinguere gli host tra di loro.

Se non si vuole usare un pool, ma un singolo indirizzo IP inside global (come nelle reti domestiche) si specifica direttamente l'id dell'interfaccia – non si specifica l'indirizzo stesso perché il provider potrebbe darmi un nuovo indirizzo inside global

Overlapping Networks



Traduzione **bidirezionale**: gli indirizzi outside global e outside local non coincidono, è il caso di reti che vengono progettate individualmente e che poi devono essere unite, ma che hanno usato uno stesso blocco di indirizzi privati. Si sceglie una delle due reti come inside network e una come outside quindi si fa la traduzione come se fossero appunto reti separate.

Esempio:

HostB ha indirizzo 10.1.1.6, ma anche nella rete outside c'è HostZ con stesso indirizzo; l'indirizzo *outside global* di HostZ non può coincidere con quello *outside local*, perché se A vuole comunicare con Z e usa come indirizzo destinatario 10.1.1.6 il pacchetto arriva a HostB.

Quindi nella rete inside devo usare un indirizzo *outside local* per HostZ diverso da 10.1.1.6

Il router fa **due volte** la traduzione: sostituisce il sorgente inside local con l'inside global e anche l'indirizzo di destinazione outside local viene sostituito con quello outside global.

Nelle reti domestiche dove il NAT non è bidirezionale quando apro un sito mando la richesta al DNS che risponde con l'indirizzo IP del server outside global che coincide con l'outside local, nel caso dell'overlapping network invece il router trasforma l'indirizzo mandato dal DNS.