Politecnico di Torino
Laurea Magistrale in Ingegneria Informatica

appunti di

Protocolli e architetture di routing

Autori principali: Luca Ghio
Docenti: Fulvio Giovanni Ottavio Risso, Mario Baldi
Anno accademico: 2014/2015
Versione: 1.0.0.1
Data: 9 maggio 2016

Ringraziamenti

Speciali ringraziamenti vanno ad Andrea Marcelli per il suo contributo.

Oltre agli autori precedentemente citati, quest'opera può includere contributi da opere correlate su WikiAppunti e su Wikibooks, perciò grazie anche a tutti gli utenti che hanno apportato contributi agli appunti *Protocolli e architetture di routing* e al libro *Protocolli e architetture di instradamento*.

Informazioni su quest'opera

Quest'opera è pubblicata gratuitamente. Puoi scaricare l'ultima versione del documento PDF, insieme al codice sorgente LaTeX, da qui: http://lucaghio.webege.com/redirs/e

Quest'opera non è stata controllata in alcun modo dai professori e quindi potrebbe contenere degli errori. Se ne trovi uno, sei invitato a correggerlo direttamente tu stesso realizzando un commit nel repository Git pubblico o modificando gli appunti *Protocolli e architetture di routing* su WikiAppunti, oppure alternativamente puoi contattare l'autore principale inviando un messaggio di posta elettronica a artghio@tiscali.it.

Licenza

Indice

Parte I

Algoritmi di instradamento

Capitolo 1

Inoltro e instradamento

L'**instradamento** è il processo che determina il percorso "migliore" per un pacchetto e lo invia in uscita verso la destinazione:

- **algoritmo di instradamento**: è responsabile di decidere i percorsi da prendere per i pacchetti in arrivo:
  1. determina le destinazioni raggiungibili da ogni nodo;
  2. calcola i percorsi migliori (secondo certi criteri) in maniera cooperativa con gli altri nodi;
  3. memorizza delle informazioni locali in ogni nodo;
- **algoritmo di inoltro**: è responsabile di prendere il percorso deciso per ogni pacchetto in arrivo:
  1. effettua una ricerca nelle informazioni locali calcolate e memorizzate dall'algoritmo di instradamento;
  2. invia il pacchetto in uscita lungo il percorso migliore.

I protocolli di instradamento si differenziano in due classi:

- **Interior Gateway Protocol** (IGP): comprende i protocolli utilizzati nell'**instradamento intra-dominio** (ad es. RIP, IGRP, OSPF) per propagare le informazioni di instradamento all'interno di un Autonomous System[1];
- **Exterior Gateway Protocol** (EGP): comprende i protocolli utilizzati nell'**instradamento inter-dominio** (ad es. BGP) per propagare le informazioni di instradamento fra Autonomous System (si rimanda alla sezione 6.2).

Secondo il modello OSI, l'instradamento è una funzionalità propria del livello rete, ma può essere implementato a livelli diversi:

- l'instradamento è implementato interamente a livello rete da protocolli come IP, X.25 e OSI/Decnet;
- alcuni degli algoritmi di instradamento sono implementati a livello data-link da protocolli come Frame Relay e ATM e dai bridge nelle LAN commutate.

I router moderni implementano due tabelle:

- **Routing Information Base** (RIB): è la classica tabella di instradamento che elenca tutte le destinazioni raggiungibili nella rete;

[1] Un Autonomous System (AS) è generalmente la rete sotto il controllo di un ISP (si rimanda alla sezione 6.1).

- **Forwarding Information Base** (FIB): è una tabella di instradamento ottimizzata per velocizzare l'inoltro dei pacchetti:
  - hardware dedicato: le TCAM sono in grado di memorizzare bit i cui valori possibili sono 0, 1 e "don't care" ⇒ la netmask è integrata nell'indirizzo di rete stesso: ogni bit nella parte aggregata ha valore "don't care";
  - cache: la FIB include solo gli ultimi indirizzi di destinazione utilizzati;
  - informazioni aggiuntive: porta di uscita, indirizzo MAC di destinazione.

1.1 Algoritmi di inoltro

1.1.1 Instradamento per indirizzo di rete

- Ogni nodo è identificato da un indirizzo di rete.
- Ogni pacchetto contiene l'indirizzo del nodo di destinazione.
- Ogni nodo contiene la lista degli indirizzi delle destinazioni raggiungibili con i relativi next hop.

Quando arriva un pacchetto, il nodo utilizza l'**indirizzo di destinazione** in esso contenuto come "chiave" nella tabella di inoltro per trovare il next hop.

Vantaggio È un algoritmo semplice ed efficiente perché è stateless: l'inoltro di un pacchetto avviene indipendentemente dall'inoltro degli altri pacchetti, cioè il nodo una volta inoltrato un pacchetto se ne dimentica.

Svantaggio Non è possibile la selezione di rotte diverse per la stessa destinazione in base al tipo di traffico per la qualità del servizio.

Adozione I protocolli connectionless (come l'IP) tipicamente utilizzano questo algoritmo di inoltro.

1.1.2 Tecnica dei "percorsi colorati"

- Ogni percorso tra due nodi è identificato da un PathID ("colore").
- Ogni pacchetto contiene un'etichetta che corrisponde al PathID del percorso da seguire.
- Ogni nodo contiene la lista dei PathID con le relative porte di uscita.

Quando arriva un pacchetto, il nodo utilizza l'**etichetta** in esso contenuto come "chiave" nella tabella di inoltro per trovare la porta di uscita.

Vantaggio Sono possibili molti percorsi verso una stessa destinazione ⇒ è possibile scegliere il percorso migliore in base al tipo di traffico per la qualità del servizio.

Svantaggio Il PathID è globale:

- la "colorazione" dei percorsi deve essere coerente su tutti i nodi nella rete;
- scalabilità: il numero di percorsi possibili tra tutte le coppie di nodi nella rete è molto grande ⇒ sono necessari molti bit per codificare ogni PathID, ed è difficile trovare un identificativo che non è ancora stato utilizzato.

1.1.3 Label swapping

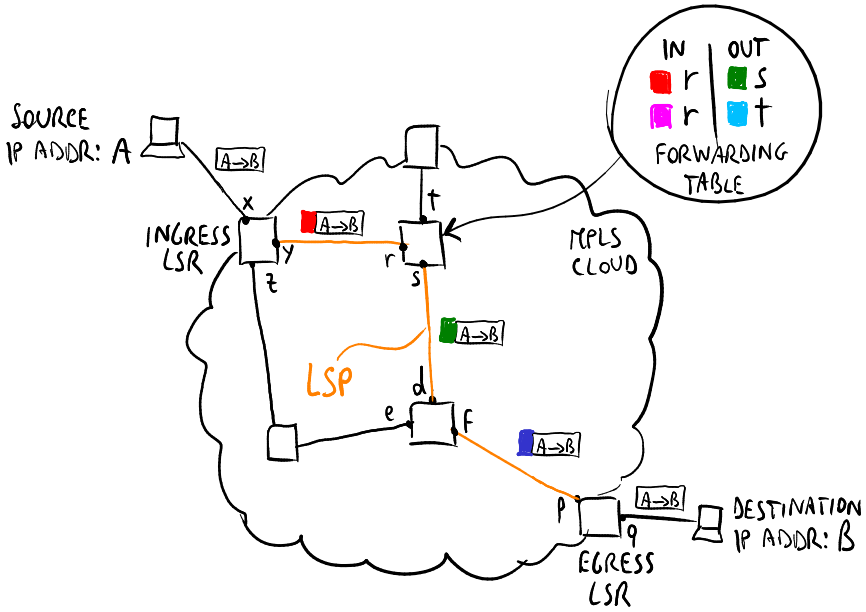


Figura 1.1: Label swapping in una rete MPLS.

La tabella di inoltro di ogni nodo contiene la mappatura tra le etichette delle porte in ingresso e le etichette delle porte in uscita, includendo entry di tipo:
`<porta in ingresso> <etichetta in ingresso> <porta in uscita> <etichetta in uscita>`

Quando arriva un pacchetto, il nodo utilizza l'etichetta in esso contenuto e la porta in ingresso come "chiave" nella tabella di inoltro per trovare la porta di uscita, e rimpiazza l'etichetta corrente del pacchetto con l'etichetta in uscita.

Vantaggi

- scalabilità: il PathID del percorso da seguire non è globale, ma l'etichetta viene decisa localmente nodo per nodo, e deve essere coerente solo tra i nodi alle estremità del link:
  - le etichette sono composte da meno bit perché devono codificare meno percorsi;
  - ogni nodo deve conoscere solo le etichette dei percorsi che passano per esso ⇒ la tabella di inoltro è più piccola;
- efficienza: il label swapping è veloce rispetto ad algoritmi di inoltro come il "longest prefix matching" di IP.

Adozione Il label swapping è utilizzato da:

- tecnologie di rete di derivazione telecomunicazionista (ad es. X.25, Frame Relay, ATM): il label swapping permette la qualità del servizio, una funzionalità considerata importante dal mondo degli operatori telefonici;
- MPLS: nel backbone i percorsi sono in numero abbastanza limitato e piuttosto stabili poiché sono creati non end-to-end ma nella nuvola MPLS, dove la topologia della rete cambia meno di frequente e il traffico è più regolare rispetto ai margini.

Instaurazione del percorso

Quando un host vuole generare e inviare il primo pacchetto verso una destinazione, come chiede l'instaurazione di un nuovo percorso e quale etichetta deve utilizzare?

Instaurazione manuale I percorsi e le relative etichette sono impostati manualmente dall'amministratore della rete.

Svantaggi

- rischio elevato di errori umani nella configurazione;
- nessun re-instradamento automatico in caso di guasti;
- non adatto per reti altamente dinamiche dove gli utenti chiedono frequentemente nuovi percorsi.

Instaurazione su richiesta È richiesta una fase di segnalazione per l'instaurazione del percorso, ovvero per la preparazione delle etichette in ogni nodo, al termine della quale l'host apprende l'etichetta da utilizzare e può inviare il primo pacchetto verso la destinazione.

Vantaggi La qualità del servizio è più semplice:

- è possibile instaurare dei percorsi diversi in base alla sorgente che ne richiede l'instaurazione (ad es. il rettore può disporre di un percorso privilegiato rispetto al ricercatore);
- è possibile includere all'interno del pacchetto di segnalazione un'informazione che specifica quanta larghezza di banda riservare per il percorso.

Svantaggi

- complessità: la segnalazione è ottenuta attraverso un'altra tecnica di inoltro (ad es. l'instradamento per indirizzo di rete) su un circuito dedicato ⇒ la complessità aumenta perché la rete deve ora gestire due tecniche di inoltro differenti;
- scalabilità: se il percorso è lungo e il numero di nodi da attraversare è elevato, la fase di segnalazione può durare troppo a lungo, soprattutto se le sessioni di comunicazione sono abbastanza brevi come nel mondo delle reti.

1.1.4 Source routing

L'host mittente scrive nel pacchetto stesso l'intero percorso che deve seguire per arrivare fino alla destinazione.

Vantaggio I router interni alla rete sono estremamente semplici.

Svantaggio L'host mittente deve conoscere la topologia della rete e deve interagire direttamente con gli altri host per poter calcolare i percorsi ⇒ ciò viola il paradigma secondo cui gli utenti finali devono limitarsi ad inviare i pacchetti e alla rete è affidato il compito di inoltrare i pacchetti verso le destinazioni.

Adozione IPv4 e IPv6 prevedono un'opzione che influenza il percorso dei pacchetti.

1.1.5 Confronto

Instradamento per indirizzo di rete

+ **semplicità**: no instaurazione, no stato
+ **scalabilità (inoltro)**: no stato "per sessione" (stateless)
− **efficienza**: intestazione dei pacchetti grande

– **scalabilità (instradamento)**: tabella di instradamento molto grande

– **affidabilità**: difficile garantire il servizio

– **multipath**: non supporta più percorsi tra due entità

Label swapping

+ **scalabilità (instradamento)**: tabella di instradamento ridotta

+ **efficienza**: intestazione dei pacchetti piccola

+ **garanzia di servizio**: possibilità di garantire il servizio (prenotazione del percorso)

+ **multipath**: più percorsi permessi tra due entità

– **scalabilità (instaurazione)**: elaborazione dei pacchetti per l'instaurazione del percorso (critico con sessioni "brevi")

– **scalabilità (inoltro)**: stato "per sessione" (necessario per la qualità del servizio)

– **complessità**: instaurazione del percorso (processo di instaurazione del percorso, inoltro ad-hoc dei pacchetti per l'instaurazione del percorso)

Source routing

+ **efficienza (router)**: i sistemi intermedi sono estremamente semplici

– **efficienza (sorgenti)**: gli end system devono preoccuparsi di calcolare i percorsi

Capitolo 2

Algoritmi di instradamento[1]

Un **algoritmo di instradamento** è un processo di tipo collaborativo responsabile di decidere, in ogni nodo intermedio, le direzioni che devono essere utilizzate per raggiungere le destinazioni:

1. determina le **destinazioni raggiungibili** da ogni nodo:
   - genera le informazioni sulla raggiungibilità delle reti locali: il router informa i router vicini che la rete locale esiste ed è raggiungibile attraverso di esso;
   - riceve le informazioni sulla raggiungibilità delle reti remote: un router vicino informa il router che la rete remota esiste ed è raggiungibile attraverso di esso;
   - propaga le informazioni sulla raggiungibilità delle reti remote: il router informa gli altri router vicini che la rete remota esiste ed è raggiungibile attraverso di esso;
2. calcola i **percorsi ottimali** (next hop), in maniera cooperativa con gli altri nodi, secondo certi criteri:
   - occorre stabilire una metrica: un percorso può essere il migliore in base a una metrica ma non in base a un'altra metrica;
   - i criteri devono essere coerenti tra tutti i nodi nella rete per evitare cicli, buchi neri, ecc.;
   - l'algoritmo deve operare in maniera automatica per evitare gli errori umani nella configurazione manuale e per favorire la scalabilità;
3. memorizza delle informazioni locali nella **tabella di instradamento** di ogni nodo: un algoritmo di instradamento non necessita di conoscere l'intera topologia della rete, ma è solo interessato a costruire la tabella di instradamento corretta.

Caratteristiche di un algoritmo di instradamento ideale

- semplice da implementare: meno bug, facile da capire, ecc.;
- leggero da eseguire: i router devono occupare meno risorse possibili per l'esecuzione dell'algoritmo poiché dispongono di CPU e memoria limitati;
- ottimale: i percorsi calcolati devono essere ottimali secondo le metriche scelte;
- stabile: deve passare a un altro percorso solo quando c'è un cambiamento di topologia o di costo per evitare il **route flapping**, cioè il cambiamento frequente delle rotte preferite con conseguente eccesso di periodi di transitorio;
- equo: non deve favorire alcun nodo o percorso particolare;

[1] Gli algoritmi di instradamento presentati nel seguito assumono di operare su una rete basata sull'instradamento per indirizzo di rete.

- robusto: deve essere capace di adattarsi automaticamente ai cambiamenti di topologia o di costo:
  - **rilevamento dei guasti**: non deve affidarsi a componenti esterni per rilevare un guasto (ad es. un guasto non può essere rilevato a livello fisico se avviene al di là di un hub);
  - **autostabilizzazione**: a fronte di variazioni nella rete deve convergere a una soluzione senza alcun intervento esterno (ad es. configurazione manuale esplicita);
  - **robustezza bizantina**: deve riconoscere e isolare un nodo vicino che sta inviando delle informazioni fasulle, a causa di un guasto o di un attacco malevolo.
    Internet non implementa la robustezza bizantina, ma si basa sulla fiducia ⇒ i guasti e i comportamenti malevoli richiedono l'intervento umano.

Classificazione degli algoritmi di instradamento

- **algoritmi non adattativi** (statici): prendono le decisioni indipendentemente da come è fatta la rete (sezione 2.4):
  - instradamento statico (o fixed directory routing)
  - random walk
  - flooding, flooding selettivo
- **algoritmi adattativi** (dinamici): apprendono informazioni sulla rete per prendere meglio le decisioni (sezione 2.5):
  - instradamento centralizzato
  - instradamento isolato: hot potato, backward learning
  - instradamento distribuito: Distance Vector, Link State
- **algoritmi gerarchici**: permettono agli algoritmi di instradamento di scalare su infrastrutture ampie (capitolo 5).

2.1 Metrica

Una **metrica** è la misura di quanto buono è un percorso, ricavata dalla trasformazione di una grandezza fisica (ad es. distanza, velocità di trasmissione), o di una combinazione di esse, in forma numerica (**costo**), al fine di scegliere il percorso a costo inferiore come il percorso migliore.

Non esiste una metrica migliore per tutti i tipi di traffico: ad esempio la larghezza di banda è adatta per il traffico di trasferimento file, mentre il ritardo di trasmissione è adatto per il traffico in tempo reale. La scelta della metrica può essere determinata dal campo "Type of Service" (TOS) del pacchetto IP.

Problemi

- (non-)ottimizzazione: il compito primario dei router è inoltrare il traffico degli utenti, non passare il tempo a calcolare percorsi ⇒ conviene prediligere soluzioni che, pur non essendo pienamente ottimizzate, non compromettono la funzionalità primaria della rete e non manifestano dei problemi percepibili dall'utente finale:
  - complessità: più criteri vengono combinati, più l'algoritmo diventa complesso e richiede risorse computazionali in fase di esecuzione;
  - stabilità: una metrica basata sulla banda disponibile sul link è troppo instabile, poiché dipende dal carico istantaneo di traffico che è molto variabile nel tempo, e può portare al route flapping;

- inconsistenza: le metriche adottate dai nodi nella rete devono essere coerenti (per ogni pacchetto) per evitare il rischio di cicli, ovvero il "rimbalzo" di un pacchetto tra due router che utilizzano metriche differenti in conflitto.

2.2 Transitori

Gli algoritmi di instradamento moderni sono sempre "attivi": si scambiano dei messaggi di servizio continuamente per rilevare i guasti autonomamente. Tuttavia, non cambiano la tabella di instradamento a meno che non viene rilevato un **cambiamento di stato**:

- cambiamenti di topologia: guasto di un link, aggiunta di una nuova destinazione;
- cambiamenti di costo: ad esempio un link a 100 Mbps sale a 1 Gbps.

I cambiamenti di stato danno origine a **fasi di transitorio**: non è possibile aggiornare allo stesso tempo tutti i nodi di un sistema distribuito, poiché una variazione è propagata nella rete a velocità finita ⇒ durante il transitorio, le informazioni di stato nella rete possono non essere coerenti: alcuni nodi hanno già le informazioni nuove (ad es. il router che rileva il guasto), mentre altri hanno ancora le informazioni vecchie.

Non tutti i cambiamenti di stato hanno lo stesso impatto sul traffico dati:

- cambiamenti di stato positivi: l'effetto del transitorio è limitato perché la rete può solo funzionare temporaneamente in una condizione sub-ottimale:
  - un percorso ottiene un costo migliore: alcuni pacchetti possono continuare a seguire il percorso vecchio ora divenuto meno conveniente;
  - viene aggiunta una nuova destinazione: la destinazione può apparire irraggiungibile a causa di buchi neri lungo il percorso verso di essa;
- cambiamenti di stato negativi: l'effetto del transitorio si manifesta più gravemente all'utente perché interferisce anche con il traffico che non dovrebbe essere influenzato dal guasto:
  - avviene il guasto di un link: non tutti i router hanno appreso che il percorso vecchio non è più disponibile ⇒ il pacchetto può cominciare a "rimbalzare" avanti e indietro saturando il link alternativo (routing loop);
  - un percorso peggiora il proprio costo: non tutti i router hanno appreso che il percorso vecchio non è più conveniente ⇒ analogo al caso di guasto (routing loop).

In generale, due problemi comuni affliggono gli algoritmi di instradamento durante il transitorio: i buchi neri e i routing loop.

2.2.1 Buchi neri

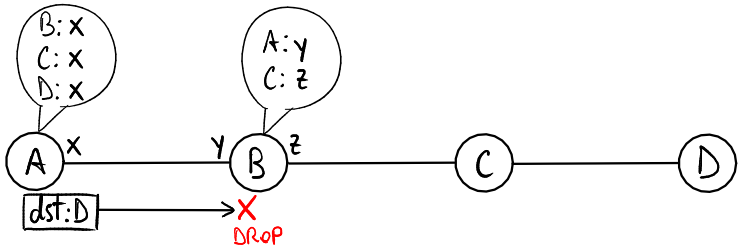


Figura 2.1: Buco nero.

Si dice **buco nero** un router che, pur esistendo almeno un percorso attraverso cui potrebbe raggiungere una certa destinazione, non ne conosce (ancora) nessuno.

Effetto L'effetto sul traffico dati è limitato ai pacchetti diretti verso la destinazione interessata, che vengono scartati fino a quando il nodo non aggiorna la tabella di instradamento e acquisisce le informazioni su come raggiungerla. Il traffico diretto ad altre destinazioni non è assolutamente influenzato dal buco nero.

2.2.2 Routing loop

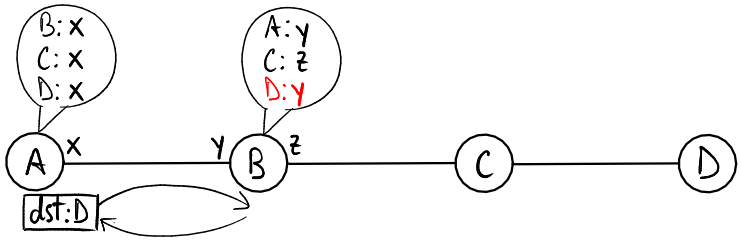


Figura 2.2: Routing loop.

Si dice **routing loop** un percorso ciclico dal punto di vista dell'instradamento: un router invia un pacchetto su un link ma, a causa di un'inconsistenza nelle tabelle di instradamento, il router all'altra estremità del link lo rispedisce indietro.

Effetto I pacchetti diretti verso la destinazione interessata cominciano a "rimbalzare" avanti e indietro (**bouncing effect**) ⇒ il link viene saturato ⇒ viene influenzato anche il traffico diretto ad altre destinazioni che passa per quel link.

2.2.3 Rotta di backup

La **rotta di backup** è un concetto principalmente utilizzato nelle reti telefoniche basate su una organizzazione gerarchica: ogni centrale è collegata a una centrale di livello superiore con una rotta primaria, e a un'altra centrale di livello superiore con una rotta di backup ⇒ se avviene un guasto nella rotta primaria, la rotta di backup è pronta a entrare in funzione senza periodi di transitorio.

Una rete dati è invece basata su una topologia magliata, dove i router sono interconnessi in vario modo ⇒ è impossibile prevedere tutti i possibili guasti della rete per predisporre i percorsi di backup, ma quando avviene un guasto è preferibile un algoritmo di instradamento che calcoli automaticamente sul momento un percorso alternativo (anche se la fase di calcolo richiede un periodo di transitorio).

La rotta di backup nelle reti moderne può avere ancora delle applicazioni:

- un'azienda collegata a Internet tramite ADSL può mantenere la connettività quando la linea ADSL cade passando a una rotta di backup tramite tecnologia HiperLAN (senza fili);
- il backbone Internet è ormai in mano alle compagnie telefoniche, che l'hanno modellato secondo i criteri delle reti telefoniche ⇒ l'organizzazione è abbastanza gerarchica da consentire la predisposizione di rotte di backup.

2.3 Multipath routing

Con l'instradamento per indirizzo di rete, tutti i pacchetti verso una destinazione passano per lo stesso percorso, anche se sono disponibili dei percorsi alternativi ⇒ sarebbe preferibile far passare una parte del traffico su un percorso alternativo, anche se di costo superiore, per non saturare il percorso scelto dall'algoritmo.

Il **multipath routing**, anche noto come "load sharing", è una funzione di traffic engineering che mira a distribuire il traffico verso la stessa destinazione su più percorsi (quando disponibili),

consentendo più entry per ogni destinazione nella tabella di instradamento, per un uso più efficiente delle risorse di rete.

2.3.1 Multipath routing a costi differenziati

Un percorso alternativo viene usato solo se ha un costo c_{sec} non troppo maggiore del costo c_{prim} del percorso primario a costo minimo:

$$\text{dati } K \geq 1,\ c_{\text{sec}} \geq c_{\text{prim}} :\ \text{sec usato} \Leftrightarrow c_{\text{sec}} \leq K \cdot c_{\text{prim}}$$

Il traffico viene distribuito in modo inversamente proporzionale al costo delle rotte. Ad esempio in questo caso:

$$\begin{cases} c_{\text{prim}} = 10 \\ c_{\text{sec}} = 20 \end{cases} \Rightarrow \begin{cases} \text{prim} = 66\% \text{ traffico} \\ \text{sec} = 33\% \text{ traffico} \end{cases}$$

il router può decidere di mandare il pacchetto con il 66% di probabilità lungo il percorso primario e con il 33% di probabilità lungo il percorso secondario.

Problema Dato un pacchetto, ogni router decide autonomamente su quale percorso inoltrarlo ⇒ decisioni incoerenti tra un router e l'altro possono far entrare il pacchetto in un routing loop, e siccome l'inoltro è di solito basato sulla sessione quel pacchetto non uscirà mai dal ciclo:

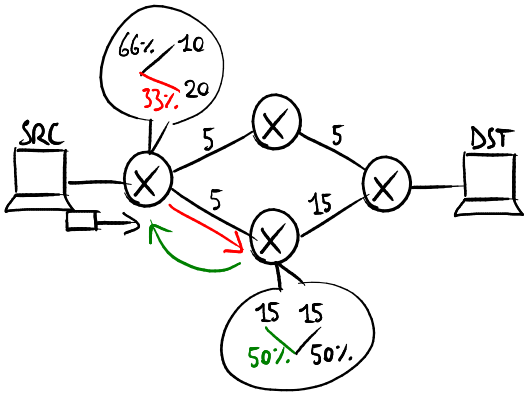


Figura 2.3: Routing loop causato dal multipath routing a costi differenziati.

2.3.2 Multipath routing a costi equivalenti

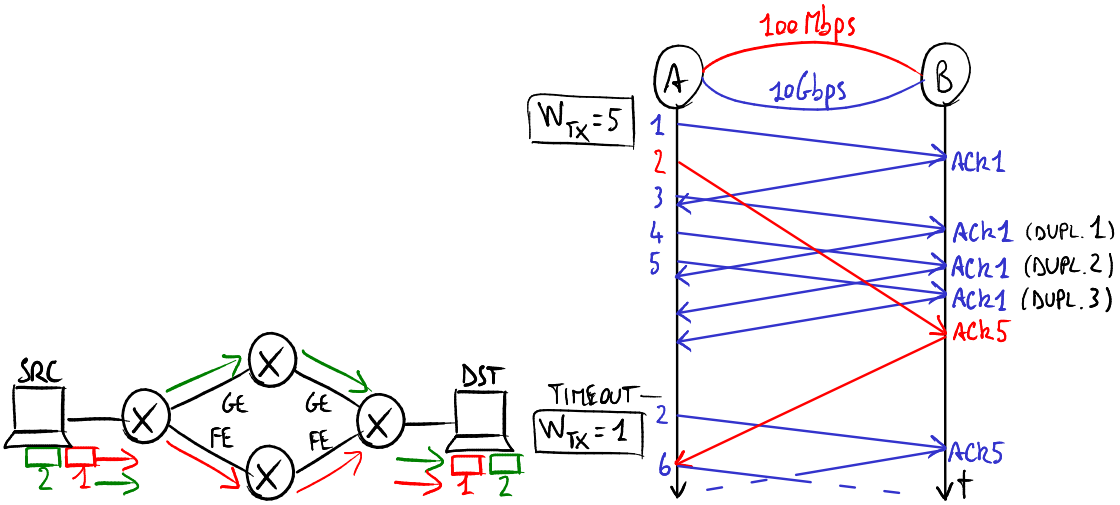
Un percorso alternativo viene usato solo se ha un costo c_{sec} esattamente uguale al costo c_{prim} del percorso primario ($K = 1$):

$$\text{dato } c_{\text{sec}} \geq c_{\text{prim}} :\ \text{sec usato} \Leftrightarrow c_{\text{sec}} = c_{\text{prim}}$$

Il traffico viene partizionato equamente su entrambi i percorsi (50%).

Problemi Se il primo pacchetto segue il percorso lento e il secondo pacchetto segue il percorso veloce, i meccanismi del TCP possono causare il peggioramento delle prestazioni complessive:

- **problema di TCP reordering**: i pacchetti possono arrivare fuori sequenza: il secondo pacchetto arriva alla destinazione prima del primo pacchetto ⇒ il processo di riordinamento dei numeri di sequenza impegna le risorse computazionali del ricevitore;



(a) Problema di TCP reordering. *(b) Riduzione della velocità di trasmissione.*

Figura 2.4: Problemi causati dal multipath routing a costi equivalenti.

- **riduzione della velocità di trasmissione**: se il pacchetto di acknowledgment (ACK) del primo pacchetto arriva troppo in ritardo, la sorgente crede che il primo pacchetto sia andato perso ⇒ quando arrivano 3 ACK (cumulativi) duplicati e scade il timeout, entrano in azione i meccanismi a finestra di scorrimento del TCP:[2]
  - fast retransmit: la sorgente ritrasmette inutilmente il pacchetto ⇒ il pacchetto diventa duplicato;
  - fast recovery: la sorgente crede che la rete sia congestionata ⇒ rallenta l'invio di pacchetti limitando la finestra di trasmissione: imposta il valore di soglia alla metà del valore corrente della finestra di congestione, e fa ripartire la finestra di congestione dal valore 1 (cioè viene mandato un solo pacchetto per volta e se ne attende l'ACK prima di mandare il pacchetto successivo).

Criteri

Le implementazioni reali del multipath routing suddividono il traffico in modo che il traffico verso una stessa destinazione segua lo stesso percorso:

- **in base al flusso**: ogni sessione di livello trasporto è identificata dalla quintupla:
  1. `<indirizzo IP sorgente>`
  2. `<indirizzo IP di destinazione>`
  3. `<tipo di protocollo di livello trasporto>` (ad es. TCP)
  4. `<porta sorgente>`
  5. `<porta di destinazione>`

  e il router memorizza una tabella che mappa gli identificativi di sessione con le porte di uscita:
  - l'estrazione dal pacchetto dei campi che formano l'ID di sessione è onerosa, a causa della varietà di formati di pacchetto supportati (VLAN, MPLS...);

[2] Si rimanda alla sezione *Uso dei protocolli a finestra* nel capitolo *Livello Trasporto: Protocolli TCP-UDP* negli appunti di "Reti di calcolatori".

  - le informazioni sulle porte di livello trasporto non sono disponibili in caso di pacchetti IP frammentati;
  - la ricerca della quintupla nella tabella degli ID di sessione è onerosa;
  - spesso la chiusura delle connessioni TCP non è "graceful leaving", cioè non vengono inviati i pacchetti FIN e ACK ⇒ le entry nella tabella degli ID di sessione non vengono cancellate ⇒ occorre un thread che ogni tanto effettui la pulizia delle entry vecchie guardandone i timestamp;

- **in base al pacchetto**: il router invia i pacchetti con indirizzo IP (di destinazione o sorgente o entrambi) pari su un percorso, e con indirizzo IP dispari sull'altro percorso ⇒ l'operazione di hashing è molto rapida.

Problema Il traffico verso una stessa destinazione non può usare entrambi i percorsi allo stesso tempo ⇒ ci sono dei problemi in caso di traffico molto grande verso una certa destinazione (ad es. un backup notturno tra due server).

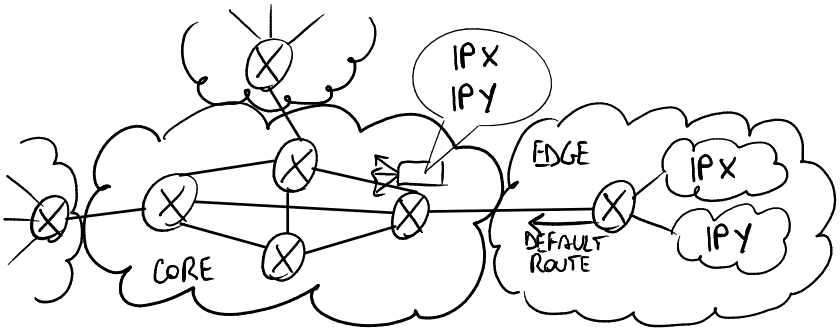
2.4 Algoritmi non adattativi

2.4.1 Instradamento statico

L'amministratore della rete configura manualmente su ogni router la tabella di instradamento.

Svantaggio Se avviene un cambiamento nella rete, bisogna aggiornare manualmente le tabelle di instradamento.

Applicazione L'instradamento statico è utilizzato principalmente sui router ai margini della rete:



- ai router edge non è consentito propagare le informazioni di instradamento verso il backbone: il router core blocca tutti gli annunci in arrivo dalla periferia, altrimenti un utente potrebbe annunciare una rete con lo stesso indirizzo di una rete esistente (ad es. google.com) e dirottare verso di lui una parte del traffico diretto a quella rete.
  Dato che gli utenti non possono annunciare le proprie reti, come fanno a essere raggiungibili dall'esterno? L'ISP, che sa quali indirizzi usano le reti che ha venduto agli utenti, deve configurare il router core in modo che annunci quelle reti agli altri router anche se esse non sono localmente connesse;

- ai router edge non è consentito ricevere le informazioni di instradamento dal backbone: un router edge è tipicamente connesso con un singolo link a un router core ⇒ una rotta predefinita globale è sufficiente per raggiungere l'intera Internet.

2.4.2 Random walk

Quando arriva un pacchetto, il router sceglie una porta a caso (tranne quella da cui è stato ricevuto) e lo manda in uscita su quella porta.

Applicazioni È utile quando la probabilità che il pacchetto raggiunga la destinazione è elevata:

- reti peer-to-peer (P2P): per la ricerca dei contenuti;
- reti di sensori: l'invio di messaggi deve essere un'operazione a basso consumo.

2.4.3 Flooding

Quando arriva un pacchetto, il router lo manda in uscita su tutte le porte (tranne quella da cui è stato ricevuto).

I pacchetti possono avere un campo "numero di hop" per limitare il flooding a una porzione della rete.

Applicazioni

- applicazioni militari: in caso di attacco la rete potrebbe essere danneggiata ⇒ è fondamentale che il pacchetto arrivi a destinazione, anche a costo di avere un'enorme quantità di traffico duplicato;
- algoritmo Link State: ogni router quando riceve la mappa della rete da un vicino deve propagarla agli altri vicini (si rimanda alla sezione 4.1.3).

2.4.4 Flooding selettivo

Quando arriva un pacchetto, il router prima controlla se lo ha già ricevuto e mandato in flooding in passato:

- pacchetto vecchio: lo scarta;
- pacchetto nuovo: lo memorizza e lo manda in uscita su tutte le porte (tranne quella da cui è stato ricevuto).

Ogni pacchetto è riconosciuto tramite l'identificativo del mittente (ad es. l'indirizzo IP sorgente) e il numero di sequenza:

- se il mittente si disconnette dalla rete o si spegne, quando si riconnette il numero di sequenza ricomincia da capo ⇒ il router vede tutti i pacchetti ricevuti come pacchetti vecchi;
- i numeri di sequenza sono codificati su un numero limitato di bit ⇒ lo spazio dei numeri di sequenza deve essere scelto in modo da minimizzare l'errato riconoscimento di pacchetti nuovi come pacchetti vecchi.

Spazi dei numeri di sequenza

Spazio lineare Può essere tollerabile se il selective flooding è usato per pochi messaggi di controllo:

- quando il numero di sequenza raggiunge il valore massimo possibile, si verifica un errore di overflow;
- pacchetto vecchio: il numero di sequenza è minore di quello corrente;
- pacchetto nuovo: il numero di sequenza è maggiore di quello corrente.

Spazio circolare Risolve il problema dell'esaurimento dello spazio dei numeri di sequenza, ma fallisce se arriva un pacchetto con numero di sequenza troppo lontano da quello corrente:

- quando il numero di sequenza raggiunge il valore massimo possibile, il numero di sequenza ricomincia dal valore minimo;
- pacchetto vecchio: il numero di sequenza è nella metà precedente a quello corrente;
- pacchetto nuovo: il numero di sequenza è nella metà successiva a quello corrente.

Spazio lecca-lecca La prima metà dello spazio è lineare, la seconda metà è circolare.

2.5 Algoritmi adattativi

2.5.1 Instradamento centralizzato

Tutti i router sono collegati a un nucleo di controllo centralizzato chiamato **Routing Control Center** (RCC): ogni router dice al RCC quali sono i propri vicini, e il RCC usa queste informazioni per creare la mappa della rete, calcolare le tabelle di instradamento e comunicarle a tutti i router.

Vantaggi

- prestazioni: i router non devono avere un'elevata capacità di calcolo, concentrata tutta su un singolo apparato;
- debug: l'amministratore di rete può ottenere la mappa dell'intera rete da un singolo apparato per verificarne la correttezza;
- manutenzione: l'intelligenza è concentrata nel centro di controllo ⇒ per aggiornare l'algoritmo basta aggiornare il centro di controllo.

Svantaggi

- tolleranza ai guasti: il centro di controllo è un singolo punto di guasto ⇒ un malfunzionamento del centro di controllo impatta su tutta la rete;
- scalabilità: più router compongono la rete, più il lavoro per il centro di controllo aumenta ⇒ non è adatto per reti ampie come Internet.

Applicazione Principi simili sono utilizzati nelle reti telefoniche.

2.5.2 Instradamento isolato

Non c'è alcun centro di controllo, ma tutti i nodi sono paritetici: ogni nodo decide i percorsi autonomamente senza scambiare informazioni con gli altri router.

Vantaggi e svantaggi Sono praticamente gli opposti di quelli dell'instradamento centralizzato.

Backward learning

Ogni nodo apprende informazioni sulla rete in base all'indirizzo sorgente dei pacchetti:[3]

- funziona bene solo con nodi "loquaci";

[3] Si rimanda alla sezione *Transparent bridge* nel capitolo *Repeater e bridge* negli appunti di "Progetto di reti locali".

- non è facile capire di dover passare a un percorso alternativo quando il percorso migliore diventa non più disponibile;
- non è facile rilevare le destinazioni diventate irraggiungibili ⇒ è richiesto un timer speciale per eliminare le entry vecchie.

2.5.3 Instradamento distribuito

L'instradamento distribuito utilizza un modello "paritetico": prende i vantaggi dell'instradamento centralizzato e quelli dell'instradamento isolato:

- instradamento centralizzato: i router partecipano nello scambio di informazioni riguardanti la connettività;
- instradamento isolato: i router sono equivalenti e non c'è un router "migliore".

Applicazioni I moderni protocolli di instradamento utilizzano due principali algoritmi di instradamento distribuito:

- Distance Vector: ogni nodo dice a tutti i vicini che cosa sa della rete (capitolo 3);
- Link State: ogni nodo dice a tutta la rete che cosa sa dei propri vicini (capitolo 4).

Confronto

Vicini

- LS: ha bisogno del protocollo di "hello";
- DV: conosce i suoi vicini attraverso il DV stesso.

Tabella di instradamento DV e LS creano la stessa tabella di instradamento, solo calcolata in modi differenti e con differente durata (e comportamento) del transitorio:

- LS: i router cooperano per mantenere la mappa della rete aggiornata, quindi ognuno di essi calcola il proprio albero ricoprente: ogni router conosce la topologia della rete, e conosce il percorso preciso per raggiungere una destinazione;
- DV: i router cooperano per calcolare la tabella di instradamento: ogni router conosce solo i suoi vicini, e si fida di loro per determinare il percorso verso la destinazione.

Semplicità

- DV: singolo algoritmo facile da implementare;
- LS: incorpora molti componenti diversi.

Debug Migliore in LS: ogni nodo ha la mappa della rete.

Consumo di memoria (in ogni nodo) Possono essere considerati equivalenti:

- LS: ognuno degli N LS ha A adiacenze (Dijkstra: $\sim N \cdot A$);
- DV: ognuno degli A DV ha N destinazioni (Bellman-Ford: $\sim A \cdot N$).

Traffico Migliore in LS: i pacchetti Neighbor Greeting sono molto più piccoli dei DV.

Convergenza Migliore in LS: il rilevamento dei guasti è più rapido perché è basato sui pacchetti Neighbor Greeting inviati ad alta frequenza.

Capitolo 3

L'algoritmo Distance Vector

L'algoritmo **Distance Vector** (DV) è basato sulla distribuzione nell'intorno del router di informazioni sull'intera rete.

Ogni router genera periodicamente un DV, che è un insieme di coppie destinazione-costo:

- destinazione: tutte le destinazioni conosciute dal router generante (nelle reti IP reali sono indirizzi di rete con netmask);
- costo: il costo del percorso dal router generante alla destinazione.

Il router ricevente apprende da ogni DV:

- destinazioni raggiungibili: si aggiungono a quelle già note localmente;
- direzione: quelle destinazioni sono raggiungibili tramite il router generante;
- costo: quello riportato dal router generante più il costo del link tra il router ricevente e il router generante.

Ogni nodo memorizza tutti i DV che arrivano dai vicini, e li integra selezionando i costi migliori per ogni destinazione al fine di costruire la sua tabella di instradamento e il suo DV:

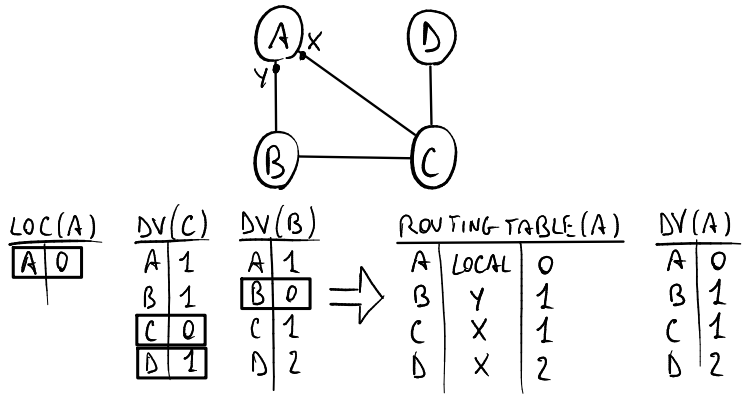


Figura 3.1: Processo di generazione della tabella di instradamento e del nuovo DV per il nodo A.

3.1 Algoritmo base

- processo principale:
  1. il DV viene annunciato ai router adiacenti;

2. si attende per il timeout;
3. si ritorna al punto 1;

- alla ricezione di un nuovo DV:
  1. il DV viene salvato in memoria;
  2. il DV viene unito con i DV memorizzati;
- al guasto di un link (rilevato a livello fisico):
  1. tutti i DV arrivati da quel link vengono eliminati;
  2. i DV rimasti vengono uniti;
- quando un DV non è ricevuto entro il timeout:
  1. il DV mancante viene eliminato;
  2. i DV rimasti vengono uniti.

Osservazioni

- affidabilità: i timeout evitano l'uso dei segnali di link-up che possono non essere sempre disponibili (ad es. se il guasto avviene al di là di un hub);
- efficienza: al guasto di un link, il router ottiene la nuova tabella di instradamento senza scambiare alcun DV con i nodi adiacenti;
- velocità di convergenza: quando un router cambia il suo DV, non lo annuncia ai vicini fino al prossimo timeout del processo principale (no triggered update).

3.2 Triggered update

Un router può inviare il suo DV aggiornato non appena aggiorna la sua tabella di instradamento, senza aspettare per il timeout predefinito, per migliorare il tempo di convergenza. Può annunciare l'intero DV o, com'è più frequente nelle implementazioni reali, solo le rotte cambiate.

Il triggered update non reimposta il timeout del processo principale, per evitare che i router comincino a generare i DV allo stesso tempo (**sincronizzazione**).

3.3 Count to infinity

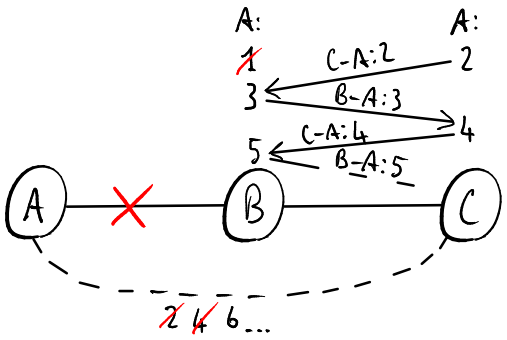


Figura 3.2: Esempio di count to infinity.

Si scatena un **count to infinity** quando il costo per raggiungere una destinazione, che non è più raggiungibile, viene progressivamente incrementato all'infinito.

Esempio In figura 3.2, un guasto sul link tra A e B scatena un count to infinity:

1. B rileva il guasto a livello fisico ed elimina il DV di A, ma C non è in grado di rilevare il guasto a livello fisico;
2. C annuncia a B di poter raggiungere A attraverso un percorso a costo 2, che in realtà era quello vecchio passante per B;
3. B aggiorna il DV di C, sembrando che A sia tornato raggiungibile tramite un percorso alternativo a costo 3 che passa per C;
4. B a sua volta invia il suo DV a C, il quale lo aggiorna e incrementa il costo a 4, e così via.

Effetto B crede di poter raggiungere A tramite C, mentre C crede di poter raggiungere A tramite B ⇒ un pacchetto che è diretto ad A comincia a rimbalzare tra B e C (bouncing effect) saturando il link tra B e C finché il TTL non va a 0.

Causa A differenza del buco nero e del routing loop, il count to infinity è un problema specifico dell'algoritmo DV, dovuto al fatto che le informazioni contenute nel DV non tengono conto della topologia della rete.

Soluzioni possibili

- soglia per l'infinito: limite superiore al count to infinity;
- algoritmi aggiuntivi: impediscono il count to infinity, ma rendono il protocollo più pesante e tendono a ridurne l'affidabilità perché non possono prevedere tutti i possibili guasti:
  - route poisoning: le cattive notizie sono meglio di nessuna notizia;
  - orizzonte limitato: se C raggiunge la destinazione A attraverso B, non ha senso per B cercare di raggiungere A attraverso C;
  - path hold down: si lascia che il rumore si calmi in attesa della verità.

3.3.1 Soglia per l'infinito

È possibile definire un valore soglia: quando il costo raggiunge il valore soglia, la destinazione è considerata non più raggiungibile.

Ad esempio il RIP ha un valore di soglia pari a 16: non possono essere collegati più di 15 router in cascata.

Protocolli con metriche complesse (ad es. IGRP) richiedono un valore soglia molto elevato per tenere conto dei costi differenziati: ad esempio una metrica basata sulla larghezza di banda può risultare in un ampio intervallo di valori di costo.

Se il bouncing effect ha luogo su un link a basso costo, è necessario troppo tempo per aumentare i costi fino al valore soglia ⇒ è possibile utilizzare due metriche allo stesso tempo:

- una metrica per i costi dei percorsi (ad es. basata sulla larghezza di banda del link);
- una metrica per il count to infinity (ad es. basata sul numero di hop).

Quando la metrica usata per il count to infinity restituisce "infinito", la destinazione è considerata non raggiungibile a prescindere dal costo del percorso.

3.3.2 Route poisoning

Il router che ha rilevato il guasto propaga le destinazioni non più raggiungibili con costo pari a infinito ⇒ gli altri router apprendono del guasto e propagano a loro volta l'informazione "avvelenata".

3.3.3 Orizzonte limitato

Ogni router differenzia i DV inviati ai suoi vicini: in ogni DV omette le destinazioni che sono raggiungibili tramite un percorso che passa per il vicino a cui lo sta inviando ⇒ non si innesca la comparsa di percorsi "fantasma" verso una destinazione non più raggiungibile in seguito all'invio di informazioni obsolete nel DV.

Caratteristiche

- evita il count to infinity tra due nodi (tranne in caso di cicli particolari);
- migliora il tempo di convergenza dell'algoritmo DV;
- i router devono calcolare un DV diverso per ogni link.

Orizzonte limitato con poisoned reverse

Nelle implementazioni reali, il DV può essere frammentato in più pacchetti ⇒ se vengono omesse delle entry nel DV, il nodo ricevente non sa se quelle entry sono state volontariamente omesse dal meccanismo dell'orizzonte limitato o se i pacchetti in cui erano contenute sono andati persi.

Nell'orizzonte limitato con poisoned reverse, le destinazioni invece di essere omesse vengono trasmesse comunque ma "avvelenate" con costo infinito, così il nodo ricevente è sicuro di aver ricevuto tutti i pacchetti che compongono il DV ⇒ aumenta la velocità di convergenza.

3.3.4 Path hold down

Se il percorso verso una destinazione aumenta di costo, probabilmente si sta per innescare un count to infinity ⇒ quella entry viene "congelata" per uno specifico periodo di tempo in attesa che il resto della rete trovi un eventuale percorso alternativo, dopodiché se nessuno annuncia più quella destinazione essa sarà considerata non raggiungibile e la sua entry sarà cancellata.

3.4 DUAL

Il **Diffusing Update Algorithm** (DUAL) è un algoritmo aggiuntivo che mira a migliorare la scalabilità dell'algoritmo DV garantendo l'assenza di routing loop anche durante il transitorio:

- cambiamento di stato positivo: se un qualsiasi nodo vicino annuncia un percorso alternativo a costo inferiore, esso viene subito accettato perché sicuramente non provocherà un routing loop;
- cambiamento di stato negativo: se
  - il next hop corrente annuncia l'incremento del costo della rotta corrente (gli annunci peggiorativi da parte di altri nodi vicini sono ignorati), oppure
  - il router rileva a livello fisico un guasto sul link appartenente alla rotta corrente

  allora è necessario attivare l'algoritmo DUAL:
  1. **selezione di un vicino accettabile**: viene selezionato un altro vicino solo se garantisce che il percorso alternativo attraverso di esso non provocherà routing loop;
  2. **processo di diffusione**: se non si riesce a trovare alcun vicino accettabile, il nodo entra in una sorta di "modalità di panico" e chiede aiuto ai vicini, aspettando che qualcuno segnali un percorso ammissibile verso quella destinazione.

3.4.1 Selezione di un vicino accettabile

Se la rotta corrente non è più disponibile a causa di un cambiamento di stato negativo, un percorso alternativo è selezionato solo se è possibile dimostrare che il nuovo percorso non crea dei cicli, cioè se è certo che il nuovo next hop non utilizza il nodo stesso per raggiungere la destinazione.

Un nodo vicino K è un **vicino accettabile** per il router R se e solo se la sua distanza verso la destinazione D è più piccola della distanza che il router R aveva prima del cambiamento di stato:

$$d(K, D) < d(R, D)$$

Ciò garantisce che il vicino K può raggiungere la destinazione D utilizzando un percorso che non passa per il router R: se il percorso $K \to D$ passasse da R, il suo costo non potrebbe essere inferiore a quello del sottopercorso $R \to D$.

In caso esista più di un vicino accettabile, viene selezionato il vicino X che offre il percorso a più basso costo verso la destinazione D:

$$\min \{L(R, X) + d(X, D)\}$$

dove:

- $L(R, X)$ è il costo del link tra il router R e il suo vicino X;
- $d(X, D)$ è la distanza tra il vicino X e la destinazione D.

Il vicino accettabile selezionato non è garantito essere il vicino attraverso il quale passa il percorso migliore possibile verso la destinazione. Se il meccanismo non seleziona il vicino migliore, quest'ultimo continuerà ad annunciare il percorso veramente migliore senza variare il suo costo $\Rightarrow$ il router riconoscerà l'esistenza di un nuovo percorso migliore che non era stato selezionato e adotterà il nuovo percorso (cambiamento di stato positivo).

3.4.2 Processo di diffusione

Se il router R non riesce a trovare alcun vicino accettabile per la destinazione:

1. congela temporaneamente la entry nella tabella di instradamento relativa alla destinazione $\Rightarrow$ i pacchetti continuano a prendere il percorso vecchio, che di sicuro è privo di cicli e al più non è più in grado di condurre a destinazione;
2. entra in uno **stato attivo**:
   (a) invia a ognuno dei suoi vicini, eccetto il next hop del percorso vecchio, un messaggio di **query** chiedendo a esso se riesce a trovare un percorso che sia migliore del suo percorso vecchio e che sia sicuramente privo di cicli;
   (b) attende la ricezione di un messaggio di **reply** da ognuno dei suoi vicini;
   (c) sceglie il percorso migliore uscendo dallo stato attivo.

Ogni router vicino X che riceve il messaggio di query dal router R invia in risposta un messaggio di reply contenente il suo DV relativo a un percorso attraverso di esso:

- se il router R non è il suo next hop verso la destinazione, e quindi il costo del suo percorso verso la destinazione non è cambiato, allora il router X segnala che il router R può usare quel percorso;
- se il router R è il suo next hop verso la destinazione, allora il router X deve mettersi a sua volta alla ricerca di un percorso nuovo, selezionando un vicino accettabile o entrando anch'esso nello stato attivo.

3.5 Vantaggi e svantaggi

Vantaggi

- molto facile da implementare, e i protocolli basati sull'algoritmo DV sono semplici da configurare;
- richiede risorse di elaborazione limitate ⇒ hardware nei router economico;
- adatto per reti piccole e stabili con cambiamenti di stato negativi non troppo frequenti;
- l'algoritmo DUAL garantisce reti libere da cicli: non possono avvenire routing loop, neanche nel transitorio (anche se i buchi neri sono ancora tollerati).

Svantaggi

- l'algoritmo ha un caso peggiore esponenziale e ha un funzionamento normale tra $O\left(n^2\right)$ e $O\left(n^3\right)$;
- la convergenza può essere piuttosto lenta, proporzionale al link più lento e al router più lento nella rete;
- difficile capire e predire il suo comportamento nelle reti grandi e complesse: nessun nodo ha una mappa della rete ⇒ è difficile rilevare eventuali routing loop;
- può scatenare routing loop dovuti a particolari cambiamenti nella topologia;
- le tecniche aggiuntive per migliorare il suo funzionamento rendono il protocollo più complesso, e comunque non risolvono completamente il problema della mancanza di conoscenza della rete;
- la soglia "infinito" limita l'utilizzo di questo algoritmo alle sole reti piccole (ad es. con pochi hop).

3.6 L'algoritmo Path Vector

L'algoritmo **Path Vector** (PV) aggiunge informazioni sulle rotte annunciate: viene annunciato anche il percorso, ovvero la lista dei nodi attraversati lungo di esso:

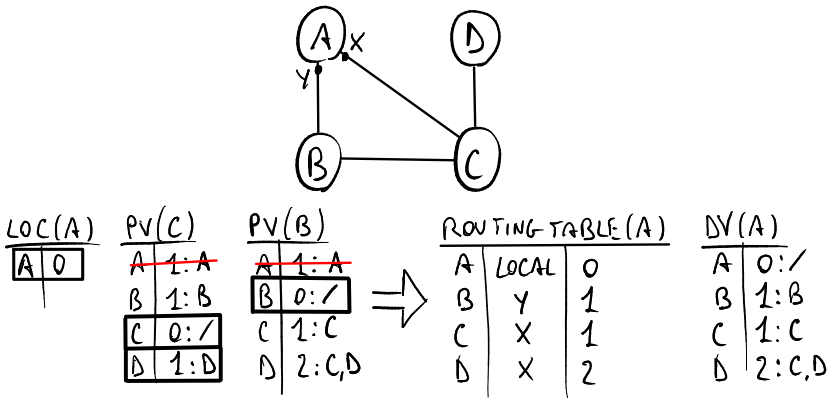


Figura 3.3: Processo di generazione della tabella di instradamento e del nuovo PV per il nodo A.

La lista dei nodi attraversati permette di evitare la comparsa di routing loop: il nodo ricevente è in grado di rilevare che la rotta annunciata passa attraverso di esso osservando la presenza del

suo identificativo nella lista, scartandola invece di propagarla ⇒ non possono formarsi percorsi che passano due volte per lo stesso nodo.

Il Path Vector è un algoritmo intermedio tra il Distance Vector e il Link State: aggiunge le informazioni strettamente necessarie sui percorsi annunciati senza avere la complessità associata al Link State dove è necessario conoscere l'intera topologia della rete.

Applicazione L'algoritmo PV viene utilizzato nell'instradamento inter-dominio dal protocollo BGP (si rimanda alla sezione 12.1.1).

Capitolo 4

L'algoritmo Link State

L'algoritmo **Link State** (LS) è basato sulla distribuzione nell'intera rete di informazioni sull'intorno del router. Ogni nodo può creare la mappa della rete (uguale per tutti i nodi), dalla quale è necessario ricavare la tabella di instradamento.

4.1 Componenti

- Neighbor Greeting (sezione 4.1.1)
- Link State (sezione 4.1.2)
- algoritmo di flooding (sezione 4.1.3)
- algoritmo di Dijkstra (sezione 4.1.4)
- riallineamento delle adiacenze (sezione 4.1.5)

4.1.1 Neighbor Greeting

I Neighbor Greeting sono messaggi scambiati periodicamente tra nodi adiacenti per raccogliere le informazioni sulle adiacenze. Ogni nodo:

- invia i Neighbor Greeting per segnalare la propria esistenza ai suoi vicini;
- riceve i Neighbor Greeting per apprendere quali sono i suoi vicini e i costi per raggiungerli.

I Neighbor Greeting implementano il **rilevamento dei guasti** basato su un numero massimo di Neighbor Greeting consecutivi non ricevuti:

- rapido: i Neighbor Greeting possono essere inviati ad alta frequenza (ad es. ogni 2 secondi) per riconoscere le variazioni sulle adiacenze in un tempo molto breve:
  - una volta ricevuti, non sono propagati ma si fermano al primo hop ⇒ non intasano la rete;
  - sono pacchetti di piccole dimensioni perché non contengono informazioni su altri nodi oltre al nodo generante;
  - richiedono un overhead basso per i router, che non sono costretti a ricalcolare la tabella di instradamento ogni volta che ne ricevono uno;
- affidabile: non si affida al segnale "link-up", non disponibile in presenza di hub.

4.1.2 Link State

Ogni router genera un LS, che è un insieme di coppie adiacenza-costo:

- adiacenza: tutti i vicini del router generante;
- costo: il costo del link tra il router generante e il vicino.

Ogni nodo memorizza tutti i LS che arrivano da tutti i nodi della rete nel **Link State Database**, quindi scandisce la lista di tutte le adiacenze e costruisce un grafo unendo i nodi (router) con degli archi (link) al fine di costruire la mappa della rete.

La generazione dei LS è principalmente **basata sugli eventi**: un LS viene generato in seguito a un cambiamento nella topologia locale (= nell'intorno del router):

- il router ha un nuovo vicino;
- è cambiato il costo per raggiungere un vicino;
- il router ha perduto la connettività ad un vicino precedentemente raggiungibile.

La generazione basata sugli eventi:

- permette un migliore utilizzo della rete: non consuma la larghezza di banda;
- richiede il componente di "hello", basato sui Neighbor Greeting, poiché il router non può più utilizzare la generazione periodica per rilevare i guasti verso i suoi vicini.

In aggiunta, i router implementano anche una generazione periodica, con frequenza molto bassa (dell'ordine di decine di minuti):

- aumenta l'affidabilità: se un LS per qualche motivo va perso, può essere nuovamente inviato senza dover aspettare l'evento successivo;
- permette di includere un campo "age": la entry relativa a una destinazione scomparsa rimane nella tabella di instradamento e i pacchetti continuano a essere mandati a quella destinazione fino a quando l'informazione, se non rinfrescata, non invecchia abbastanza da poter essere cancellata.

4.1.3 Algoritmo di flooding

Ogni LS deve essere inviato in "broadcast" a tutti i router della rete, che devono riceverlo invariato ⇒ i protocolli reali implementano una sorta di **flooding selettivo**, che rappresenta l'unico modo per raggiungere tutti i router con gli stessi dati e con overhead minimo. Il broadcast è limitato ai soli LS, per evitare l'intasamento della rete.

La propagazione dei LS avviene a velocità elevata: al contrario dei DV, ogni router può subito propagare il LS ricevuto e in un secondo momento elaborarlo localmente.

I protocolli reali implementano un meccanismo affidabile per la propagazione dei LS: ogni LS deve essere confermato "hop by hop" con un acknowledgment, perché il router deve essere sicuro che il LS inviato ai suoi vicini sia stato ricevuto, anche considerando che i LS sono generati con una frequenza bassa.

4.1.4 Algoritmo di Dijkstra

Dopo aver costruito la mappa della rete dalla lista delle adiacenze, ogni router è in grado di calcolare l'**albero ricoprente** del grafo, cioè l'albero con i percorsi a costo minimo avente il nodo come radice, grazie all'algoritmo di Dijkstra: a ogni iterazione vengono considerati tutti i link che collegano i nodi già selezionati con i nodi non ancora selezionati, e viene selezionato il nodo adiacente più vicino.[1]

[1] Si rimanda alla sezione *Algoritmo di Dijkstra* nel capitolo *I cammini minimi* negli appunti di "Algoritmi e programmazione".

Tutti i nodi hanno lo stesso Link State Database, ma ogni nodo ha un albero di instradamento diverso verso le destinazioni, perché al cambiare del nodo scelto come radice cambia l'albero ricoprente ricavato:

- migliore distribuzione del traffico: ragionevolmente non ci sono link inutilizzati (a differenza dello Spanning Tree Protocol);
- ovviamente l'albero di instradamento deve essere consistente tra i vari nodi.

4.1.5 Riallineamento delle adiacenze

Il riallineamento delle adiacenze è richiesto per sincronizzare i Link State Database dei router quando viene rilevata una nuova adiacenza:

- un nuovo nodo si collega alla rete: il nodo adiacente comunica ad esso tutti i LS relativi alla rete, per popolare il suo Link State Database da cui potrà calcolare la sua tabella di instradamento;
- due sottoreti partizionate (ad es. a causa di un guasto) vengono riconnesse insieme: ognuno dei due nodi alle estremità del link comunica all'altro nodo tutti i LS relativi alla sua sottorete.

Procedura

1. una nuova adiacenza viene rilevata dal protocollo di "hello", che tiene le adiacenze sotto controllo;
2. la sincronizzazione è un processo punto-punto, cioè interessa solo i due router alle estremità del nuovo link;
3. gli LS che erano precedentemente sconosciuti vengono inviati agli altri nodi della rete in flooding.

4.2 Comportamento su reti broadcast di livello data-link

L'algoritmo LS modella la rete come un insieme di link punto-punto ⇒ soffre in presenza di reti broadcast[2] di livello data-link (come Ethernet), dove ogni entità ha accesso diretto a ogni altra entità sullo stesso collegamento dati (bus condiviso), creando perciò un insieme di adiacenze a maglia completa (N nodi $\rightarrow \frac{N(N-1)}{2}$ link punto-punto).

Il numero elevato di adiacenze ha un impatto importante sull'algoritmo LS:

- problemi computazionali: la convergenza dell'algoritmo di Dijkstra dipende dal numero di link ($L \cdot \log N$), ma il numero di link esplode su reti broadcast;
- overhead non necessario nella propagazione dei LS: ogni volta che un router deve inviare il suo LS sulla rete broadcast, deve generare $N-1$ LS, uno per ogni vicino, anche se sarebbe sufficiente inviarlo una volta sola sul canale condiviso per raggiungere tutti i vicini, poi ogni vicino a sua volta propagherà più volte il LS ricevuto ($\sim N^2$);
- overhead non necessario nel riallineamento delle adiacenze: ogni volta che viene aggiunto un nuovo router alla rete broadcast, deve iniziare $N-1$ fasi di riallineamento, una per ogni vicino, anche se sarebbe sufficiente riallineare il database con uno solo di essi.

La soluzione è trasformare la topologia broadcast in una **topologia a stella**, aggiungendo uno pseudo-nodo (NET): la rete è considerata un componente attivo che inizierà ad annunciare le sue adiacenze, diventando il centro di una topologia a stella virtuale:

[2] Per essere precisi, su tutte le reti di livello data-link con accesso multiplo (ad es. anche su Non-Broadcast Multiple Access [NBMA]).

- viene “promosso” uno dei router che sarà responsabile di inviare anche quei LS aggiuntivi a nome della rete broadcast;
- tutti gli altri router annunciano un’adiacenza solamente a quel nodo.

La topologia a stella virtuale è valida solo per la propagazione dei LS e il riallineamento delle adiacenze, mentre il normale traffico dati utilizza ancora la topologia broadcast reale:

- propagazione dei LS: il nodo generante invia un LS allo pseudo-nodo, il quale lo invia agli altri nodi ($\sim N$);
- riallineamento delle adiacenze: il nuovo nodo attiva una fase di riallineamento solo con lo pseudo-nodo.

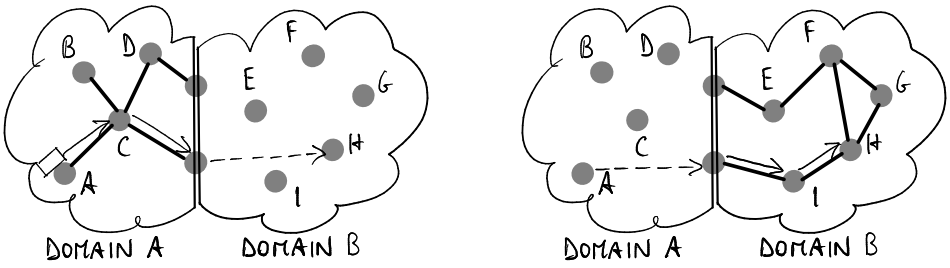
4.3 Vantaggi

- convergenza rapida:
  - i LS vengono velocemente propagati senza alcuna elaborazione intermedia;
  - ogni nodo ha informazioni certe perché provenienti direttamente dalla fonte;
- breve durata dei routing loop: possono accadere durante il transitorio per una quantità di tempo limitata;
- facilità di debug: ogni nodo ha una mappa della rete, e tutti i nodi hanno database identici $\Rightarrow$ basta interrogare un singolo router per avere la mappa completa della rete in caso di necessità;
- buona scalabilità, anche se è meglio non avere domini grandi (ad es. OSPF suggerisce di non avere più di 200 router in una singola area).

Capitolo 5

Instradamento gerarchico

L'**instradamento gerarchico** permette di partizionare la rete in domini di instradamento autonomi. Un **dominio di instradamento** è la porzione della rete che è controllata dalla stessa istanza di un protocollo di instradamento.



(a) Vista dal dominio A. *(b) Vista dal dominio B.*

Figura 5.1: Esempio: inoltro di un pacchetto dal nodo A al nodo H.

I router appartenenti a un dominio non conoscono la topologia esatta di un altro dominio, ma conoscono solo la lista delle destinazioni contenute in esso con i relativi costi (a volte fittizi) ⇒ una buona scelta per raggiungerle è prendere il percorso di uscita migliore verso il dominio di destinazione attraverso un router di frontiera.

Ogni **router di frontiera** ha visibilità su entrambi i domini che esso interconnette:

- è chiamato **egress router** quando il pacchetto esce dal dominio;
- è chiamato **ingress router** quando il pacchetto entra nel dominio.

L'instradamento gerarchico introduce una nuova regola per gestire le rotte relative alle destinazioni che si trovano al di fuori del dominio corrente:

- destinazioni interne: se la destinazione si trova all'interno dello stesso dominio di instradamento, devono essere utilizzate le informazioni di instradamento generate dal protocollo di instradamento "interno";
- destinazioni esterne: se la destinazione si trova all'interno di un altro dominio di instradamento, il traffico deve essere inoltrato verso l'egress router più vicino in uscita dal dominio corrente al dominio di destinazione, e poi quest'ultimo sarà responsabile di consegnare il pacchetto a destinazione utilizzando le informazioni di instradamento interne.

Il sotto-percorso dalla sorgente all'egress router più vicino e il sotto-percorso da qui alla destinazione finale presi singolarmente sono ottimali, ma il percorso complessivo che è la loro concatenazione è non ottimale: data una destinazione, la prima parte del percorso (sorgente-router di frontiera) è uguale per tutte le destinazioni nel dominio remoto.

Motivazioni

- interoperabilità: possono essere interconnessi domini controllati da protocolli di instradamento differenti;
- visibilità: un ISP non vuole far conoscere i dettagli sulla sua rete a un concorrente;
- scalabilità: una porzione di rete troppo ampia non può essere controllata da una singola istanza di un protocollo di instradamento, ma ha bisogno di essere partizionata:
  - memoria: esclude le informazioni sulla topologia precisa dei domini remoti, riducendo la quantità di informazioni che ogni router deve mantenere in memoria;
  - sommarizzazione: consente di annunciare una destinazione "virtuale" (con un costo convenzionale) che raggruppa insieme diverse destinazioni "reali" (ad es. nelle reti IP più indirizzi di rete si possono aggregare in un indirizzo di rete con una netmask più lunga);
  - isolamento: se all'interno di un certo dominio avviene un guasto o viene aggiunto un nuovo link, i cambi di rotta non perturbano gli altri domini, cioè le tabelle di instradamento nei router dei domini remoti rimangono invariate ⇒ meno transitori, rete più stabile, convergenza più rapida.

Implementazioni L'instradamento gerarchico può essere implementato in due modi (non mutualmente esclusivi):

- automatico: alcuni protocolli (come OSPF, IS-IS) partizionano automaticamente la rete in domini di instradamento (detti "aree" in OSPF, si rimanda al capitolo 10);
- manuale: è possibile attivare il processo di ridistribuzione su un router di frontiera per interconnettere domini controllati da protocolli di instradamento anche differenti (sezione 5.2).

5.1 Domini partizionati

Un dominio diventa **partizionato** se a partire da un router di frontiera non è più possibile raggiungere tutte le sue destinazioni interne attraverso percorsi che rimangano sempre all'interno del dominio stesso.

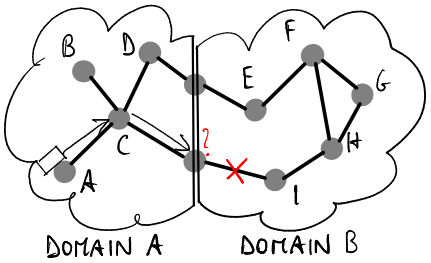


Figura 5.2: Esempio di domini partizionati.

Nell'esempio in figura 5.2, il pacchetto inviato dal nodo A esce appena possibile dal dominio di instradamento A, ma una volta entrato nel dominio B non può raggiungere la destinazione finale H a causa del guasto del link tra il router di frontiera e il nodo I. Esiste in realtà un percorso alternativo che conduce alla destinazione H attraverso l'altro router di frontiera, ma non può essere preso perché richiederebbe al pacchetto di uscire dal dominio B ed attraversare il dominio A.

Inoltre, i percorsi possono essere asimmetrici: il pacchetto di risposta potrebbe prendere un percorso diverso da quello preso dal pacchetto di query, passando per un router di frontiera

differente ⇒ i dati possono venire ricevuti, ma gli ACK di conferma della loro ricezione possono andare persi.

Soluzioni

- è possibile ridondare i link all'interno di ogni dominio per renderlo fortemente connesso, per evitare che il guasto di un link possa causare il partizionamento del dominio;
- il protocollo OSPF permette la configurazione manuale di una sorta di tunnel virtuale tra due router di frontiera chiamato **Virtual Link** (si rimanda alla sezione 10.1.2).

5.2 Ridistribuzione

La **ridistribuzione** è il processo software, in esecuzione su un router di frontiera, che permette di trasferire le informazioni di instradamento da un dominio di instradamento a un altro.

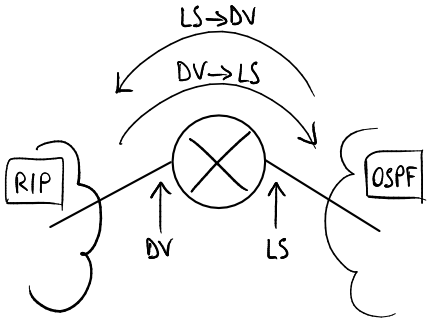


Figura 5.3: Esempio di ridistribuzione.

Nell'esempio in figura 5.3, le destinazioni apprese in un dominio RIP possono essere iniettate in un dominio OSPF e viceversa.

Osservazioni

- Il comando di ridistribuzione è unidirezionale ⇒ è possibile fare una ridistribuzione selettiva in una sola direzione (ad esempio l'ISP non accetta rotte non fidate annunciate dal cliente).
- La ridistribuzione può essere effettuata anche tra domini controllati da istanze dello stesso protocollo.
- Le rotte apprese con il processo di ridistribuzione possono essere contrassegnate come "rotte esterne" dal protocollo di instradamento.

5.2.1 Costi

I router in un dominio conosceranno un insieme più ampio di destinazioni, anche se alcune di esse possono avere una topologia "errata" (semplificata): infatti il processo di ridistribuzione può

- mantenere il costo della rotta originale, al più aggiustata con un coefficiente, oppure
- impostare il costo a un valore convenzionale quando:
  - i due protocolli utilizzano metriche differenti: ad esempio non è possibile convertire un costo appreso in "numero di hop" in un altro che utilizza i "ritardi";
  - più destinazioni con costi diversi vengono aggregate in una rotta sommarizzata.

Sorgente della rotta		**Distanza amministrativa**
interfaccia connessa		0
rotta statica		1
rotta dinamica	BGP esterno	20
	EIGRP interno	90
	IGRP	100
	OSPF	110
	RIP	120

Tabella 5.1: Principali distanze amministrative predefinite nei router Cisco.

Quando una destinazione è annunciata come raggiungibile da entrambi i domini, controllati da protocolli di instradamento con metriche diverse, il router di frontiera come può confrontare i costi per determinare la rotta migliore verso quella destinazione? Ogni protocollo di instradamento ha un **costo intrinseco** pre-assegnato dal costruttore dell'apparato ⇒ il router sceglie sempre il protocollo con costo intrinseco più basso (anche se la rotta selezionata potrebbe non essere la migliore).

Capitolo 6

Instradamento inter-dominio

L'**instradamento inter-dominio** si occupa di decidere e propagare le informazioni sulle **rotte esterne** fra più AS interconnessi nella rete.

6.1 Autonomous System

Un **Autonomous System** (AS) è un insieme di reti IP che sono sotto il controllo di un insieme di entità che concordano nel presentarsi come un'entità unica, adottando tutte lo stesso insieme di politiche di instradamento.

Dal punto di vista dell'instradamento inter-dominio, Internet è organizzata in AS: un AS rappresenta un'entità amministrativa omogenea, generalmente un ISP, al livello gerarchico più alto sulla rete. Ogni AS è univocamente identificato da un numero a 32 bit (era a 16 bit in passato) assegnato da IANA.

Ogni AS è completamente indipendente: può decidere l'instradamento interno secondo le proprie preferenze, e i pacchetti IP sono instradati al suo interno secondo le regole interne. Ogni AS può avere uno o più domini di instradamento interni serviti da protocolli IGP: ogni dominio può adottare il suo protocollo IGP preferito, e grazie alla ridistribuzione può scambiare le informazioni di instradamento con gli altri domini.

Una rete che è AS può tenere sotto controllo il traffico in entrata e in uscita grazie alle politiche di instradamento, ma è soggetta a una maggiore responsabilità: l'instradamento è più difficile da configurare, ed eventuali errori di configurazione possono influenzare il traffico di altri AS.

Per le porzioni di rete che intendono diventare degli AS, in passato erano applicate alcune regole aggiuntive che oggi sono state rilassate:

- tutta la rete deve essere nello stesso dominio amministrativo:

  oggi l'entità amministrativa di un AS non coincide necessariamente con l'organizzazione che effettivamente gestisce internamente la rete: ad esempio, la rete del Politecnico di Torino, pur essendo di proprietà dell'università ed essendo sotto il controllo di organi interni ad essa, è una delle sottoreti all'interno dell'AS amministrato dall'ente di ricerca GARR, a cui è affidato il compito di decidere le interconnessioni a lunga distanza verso gli altri AS;

- la rete deve essere almeno di una dimensione data:

  negli ultimi anni i content provider hanno avuto la necessità di avere degli AS sparsi per il mondo di dimensioni molto ridotte: ad esempio, Google possiede alcuni server Web in Italia che distribuiscono i contenuti personalizzati per il pubblico italiano (ad es. annunci pubblicitari) e che, essendo più vicini agli utenti, restituiscono più velocemente i risultati di ricerca agendo come una cache (si rimanda al capitolo 15) ⇒ se quei server

Web costituiscono da soli un AS, Google ha il controllo sulla distribuzione dei suoi contenuti agli ISP italiani, e può realizzare degli accordi commerciali con questi ultimi privilegiandone alcuni a scapito di altri;

- l'AS deve essere connesso con almeno due altri AS per garantire, almeno tecnicamente, il transito attraverso di esso del traffico da un AS all'altro:

  un ISP locale di piccole dimensioni (Tier 3) può acquistare da un ISP nazionale di grandi dimensioni (Tier 2) l'intera connettività verso Internet (si rimanda alla sezione 11.1).

6.2 Classe di protocolli EGP

Un singolo router di frontiera posto tra AS appartenenti ad ISP diversi dà origine ad alcune problematiche:

- a chi appartiene? chi lo configura?
- chi è responsabile in caso di guasto?
- come impedire a un ISP di raccogliere informazioni sulla rete di un concorrente?

La soluzione è utilizzare due router di frontiera, ciascuno amministrato da uno dei due ISP, separati da una sorta di "zona franca" intermedia controllata da una terza istanza di protocollo di instradamento di tipo **Exterior Gateway Protocol** (EGP).

Attraverso un protocollo EGP, ogni router di frontiera al bordo di un AS scambia informazioni di instradamento esterno con altri router di frontiera:

- propaga agli altri AS le informazioni sulle destinazioni che si trovano all'interno del suo AS;
- propaga agli altri AS le informazioni sulle destinazioni che si trovano all'interno di altri AS ma che possono essere raggiunte attraverso il suo AS.

I protocolli EGP si differenziano dai protocolli IGP soprattutto per il supporto alle **politiche di instradamento** che riflettono gli accordi commerciali tra gli AS (si rimanda alla sezione 11.2).

6.2.1 Protocolli EGP

- **instradamento statico**: configurazione manuale dei router:
  - \+ è l'"algoritmo" migliore per implementare delle politiche complesse e per avere il completo controllo sui percorsi di rete;
  - \+ non è necessario traffico di controllo: si evita lo scambio di informazioni sulle destinazioni;
  - − non reagisce ai cambiamenti topologici;
  - − è facile introdurre inconsistenze;
- **Exterior Gateway Protocol** (EGP)[1]: fu il primo protocollo completamente dedicato all'instradamento tra domini, ma nessuno attualmente lo usa perché fornisce solo informazioni sulla raggiungibilità e non sulla distanza:
  - − se la raggiungibilità di una destinazione è annunciata attraverso più percorsi, non è possibile scegliere il percorso migliore a costo minore;

[1] Il protocollo EGP è uno dei protocolli appartenenti alla classe di protocolli EGP.

  - se la raggiungibilità di una destinazione è annunciata attraverso più percorsi, non è possibile garantire che tutti i router scelgano un percorso coerente ⇒ può essere usato solo in reti senza percorsi chiusi dove non si possono creare dei cicli;

- **Border Gateway Protocol** (BGP): è l'unico protocollo EGP che è stato adottato nell'intera Internet a scapito di altri protocolli EGP: tutti i router di frontiera nell'intera rete di AS interconnessi devono adottare lo stesso protocollo EGP per lo scambio delle rotte esterne, perché se due AS scegliessero di utilizzare protocolli EGP differenti, i loro router di frontiera non potrebbero comunicare tra loro (si rimanda al capitolo 12);

- **Inter-Domain Routing Protocol** (IDRP): fu creato come un'evoluzione di BGP al fine di supportare l'indirizzamento OSI, ma nessuno attualmente lo usa perché:

  - è composto da parti piuttosto complesse;
  - da allora i miglioramenti introdotti dall'IDRP sono stati portati nelle versioni successive di BGP;
  - non è compatibile con il BGP ⇒ la sua adozione da parte di un AS romperebbe l'interoperabilità con il resto della rete che usa ancora il BGP.

6.3 Ridistribuzione

Su ogni router di frontiera è in esecuzione un processo di ridistribuzione dal protocollo IGP all'interno dell'AS al protocollo EGP all'esterno dell'AS e viceversa ⇒ le rotte vengono ridistribuite prima da un AS alla zona intermedia e poi da qui all'altro AS:

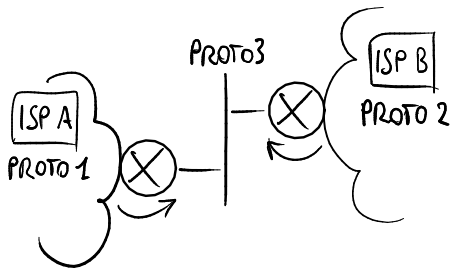


Figura 6.1: Esempio di ridistribuzione tra AS appartenenti a ISP diversi.

- il protocollo IGP apprende le **rotte esterne** verso le destinazioni presenti in altri AS, e le propaga nell'AS come rotte interne;

- il protocollo EGP apprende le **rotte interne** verso le destinazioni presenti nell'AS, e le propaga agli altri AS come rotte esterne.

La ridistribuzione definisce:

- quali reti interne devono essere note al mondo esterno: reti private ad esempio non devono essere propagate ad altri AS;

- quali reti esterne devono essere note all'interno dell'AS: è possibile ridurre la quantità di informazioni di instradamento annunciate evitando di includere i dettagli completi sulle reti esterne:

  - gli indirizzi annunciati possono essere "condensati" in rotte aggregate quando condividono una parte del loro prefisso di rete;
  - può venire annunciata una singola rotta di default quando l'AS ha un singolo punto di uscita.

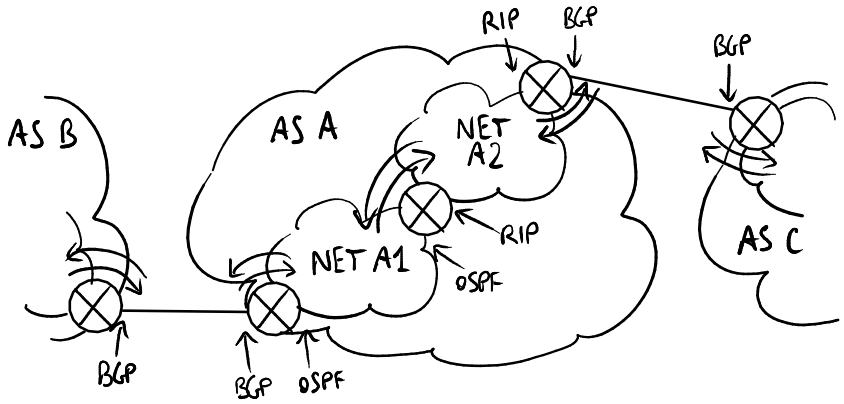


Figura 6.2: Le rotte vengono ridistribuite sia tra domini gerarchici all'interno dell'AS, sia tra l'AS stesso e il mondo esterno.

La ridistribuzione non deve introdurre delle incoerenze nell'instradamento:

- si può creare un routing loop se, ad esempio, una rotta appresa in IGP ed esportata in EGP viene poi re-importata in IGP apparendo come una rotta esterna;
- se un certo AS è raggiungibile attraverso più router di frontiera di uno stesso AS, questi router di frontiera devono accordarsi al fine di ridistribuire internamente un solo punto di uscita per quella rotta.

Spesso la ridistribuzione su un router di frontiera al bordo di un AS è attivata in una sola direzione dal protocollo IGP al protocollo EGP: le rotte interne vengono esportate al mondo esterno, mentre le rotte esterne sono rimpiazzate da una rotta di default.

Capitolo 7

Instradamento multicast

Il **multicast** è la possibilità di trasmettere la medesima informazione a più utenti finali senza essere costretti ad indirizzare questi ultimi singolarmente e senza avere, quindi, la necessità di duplicare per ciascuno di essi l'informazione da diffondere.

L'**instradamento multicast** si occupa di decidere e propagare le informazioni necessarie per inoltrare i pacchetti multicast al di fuori delle reti locali fra più **multicast router** (mrouter) interconnessi nella rete:

1. determinazione dell'esistenza di ricevitori su un particolare segmento di LAN: nel caso non esistano ricevitori, non è il caso di inoltrare quei pacchetti sulla LAN ⇒ le reti che non hanno ricevitori vengono tagliate dall'albero (**pruning**);
2. propagazione dell'esistenza e della localizzazione dei ricevitori nell'intera rete IP: l'instradamento multicast deve tenere traccia della localizzazione dei vari ricevitori, creando un "albero ricoprente", detto **albero di distribuzione**, in modo da minimizzare i costi e recapitare i pacchetti a tutti;
3. trasmissione e inoltro dei dati: i trasmettitori generano i pacchetti con un indirizzo di destinazione particolare multicast, e gli mrouter li inoltrano lungo l'albero di distribuzione fino ai ricevitori.

Gli algoritmi di instradamento multicast usano due tipi di alberi di distribuzione:

- **albero specifico della sorgente** (RPB, TRPB, RPM, Link State): c'è un albero per ogni mittente ⇒ i percorsi sono ottimali, ma l'aggiornamento è più complesso;
- **albero condiviso** (CBT): c'è un albero per ogni gruppo multicast, valido per tutti i mittenti ⇒ l'aggiornamento è più semplice, ma i percorsi non sono ottimali.

Algoritmi di instradamento multicast

- flooding selettivo (sez. 2.4.4)
- Distance Vector (sez. 7.1):
  - reverse path forwarding (RPF)
  - reverse path broadcasting (RPB)
  - truncated reverse path broadcasting (TRPB)
  - reverse path multicasting (RPM)
- Link State (sez. 7.2)
- core-based tree (CBT) (sez. 7.3)
- gerarchici (sez. 7.4)

7.1 Instradamento multicast Distance Vector

7.1.1 Reverse path forwarding

Quando un router riceve un pacchetto multicast, lo invia su tutte le altre interfacce, a patto che quella da cui è arrivato sia sul cammino più breve tra il router e la sorgente.

Problemi

- traffico: carica la rete in modo inaccettabile:
  - nessun albero di instradamento: su una LAN possono transitare più copie dello stesso pacchetto se due router collegati alla LAN hanno la stessa distanza minima dalla sorgente;
  - nessun pruning: il pacchetto viene sempre distribuito su tutti i link, senza tenere conto del fatto che ci siano ascoltatori o meno;
- rete simmetrica: considera il costo del percorso inverso dal router alla sorgente, che potrebbe essere diverso dal costo del percorso dalla sorgente al router per la presenza di link unidirezionali.

7.1.2 Reverse path broadcasting

Si costruisce un albero ricoprente di distribuzione basato sulla sorgente (nodo radice), e i pacchetti raggiungono tutte le destinazioni passando sui rami di quest'albero:

- interfaccia padre: l'interfaccia a distanza minore verso la sorgente, da cui vengono ricevuti i pacchetti dai livelli superiori;
- interfacce figlie: le altre interfacce del router, su cui vengono inviati i pacchetti verso i sottoalberi (eventuali pacchetti ricevuti vengono sempre scartati).

Su una LAN transita una sola copia dello stesso pacchetto: tra i router aventi interfacce figlie sulla LAN, il router che ha distanza inferiore verso la sorgente viene eletto come il **router designato** per quel link (in caso di costo uguale, viene presa l'interfaccia con indirizzo IP più basso).

Problemi

- traffico: carica la rete in modo inaccettabile:
  - nessun pruning: il pacchetto viene sempre distribuito su tutti i link, senza tenere conto del fatto che ci siano ascoltatori o meno;
- rete simmetrica: considera il costo del percorso inverso dal router alla sorgente, che potrebbe essere diverso dal costo del percorso dalla sorgente al router per la presenza di link unidirezionali.

7.1.3 Truncated reverse path broadcasting

Gli host interessati inviano dei membership report per iscriversi al gruppo multicast ⇒ i router invieranno i pacchetti multicast solo agli host interessati, ed elimineranno dall'albero i rami su cui non è stato ricevuto alcun membership report (**pruning**).

Sfortunatamente l'albero di distribuzione dipende, oltre che dalla sorgente, anche dal gruppo multicast, risultando in requisiti di banda per i report e di memoria dei router nell'ordine del numero totale di gruppi moltiplicato per il numero totale di possibili sorgenti ⇒ per ridurre i requisiti di banda e di memoria, vengono eliminate dall'albero solo le LAN foglie che non hanno

ascoltatori: una **LAN foglia** è una rete non usata da alcun altro router per raggiungere la sorgente multicast.

Come determinare se una certa LAN è una LAN foglia? Nell'orizzonte limitato con poisoned reverse[1], le destinazioni raggiunte attraverso il link sul quale l'annuncio è inviato vengono poste con distanza pari a infinito: se almeno un router a valle propaga la entry relativa alla sorgente in esame con distanza infinita, allora quel router usa quel link come strada più breve per raggiungere la sorgente ⇒ quel link non è una foglia, e quindi potrebbero esserci delle LAN foglie più a valle con degli ascoltatori.

Problema Non è possibile eseguire il pruning di interi sottoalberi, ma sono eliminate solo le LAN foglie ⇒ sui nodi interni dell'albero viaggia del traffico inutile.

7.1.4 Reverse path multicasting

È possibile eseguire il pruning di un intero sottoalbero:

1. il primo pacchetto inviato dalla sorgente viene propagato secondo l'algoritmo TRPB;
2. se il primo pacchetto raggiunge un router collegato solo a LAN foglie prive di ascoltatori per quel gruppo, il router invia un messaggio di non-membership report (NMR) al router padre;
3. se il router padre riceve dei messaggi NMR da tutti i suoi figli, genera a sua volta un messaggio NMR verso il padre.

I messaggi NMR hanno validità limitata: quando scade il timeout, viene adottato nuovamente l'algoritmo TRPB. Quando in un ramo potato si aggiunge un ascoltatore per quel gruppo, il router invia al nodo padre un messaggio di membership report per attivare rapidamente il ramo dell'albero senza aspettare il timeout.

Problemi

- broadcast storm periodici: sono dovuti all'algoritmo TRPB a ogni scadenza del timeout;
- scalabilità: è critica, perché ogni router deve tenere molte informazioni per ogni coppia (sorgente, gruppo).

7.2 Instradamento multicast Link State

Grazie alla mappa completa della rete costruita da un protocollo di instradamento (unicast) di tipo LS, ogni router è in grado di calcolare l'albero di distribuzione da ogni sorgente verso ogni potenziale ricevitore.

Non è più necessario il "flood and prune", ma ogni router è in grado di determinare autonomamente se si trova lungo l'albero di distribuzione:

1. in una LAN foglia priva di ascoltatori, un host comunica di essere interessato al gruppo;
2. il router collegato invia in flooding un pacchetto LS che annuncia l'esistenza di un LAN con ascoltatori e la sua posizione all'interno della rete;
3. gli altri nodi della rete memorizzano il pacchetto LS e lo propagano a loro volta in flooding a tutta la rete;
4. quando il primo pacchetto della trasmissione arriva ad un router, prima di poterlo inoltrare deve calcolare l'albero dei cammini minimi per sapere se si trova lungo l'albero di distribuzione e, in caso affermativo, su quali link deve inoltrare il pacchetto;

[1] Si veda la sezione 3.3.3.

5. per i pacchetti successivi questo calcolo non è più necessario in quanto l'informazione si troverà in cache.

Problemi

- l'instradamento del primo pacchetto di una trasmissione può richiedere parecchio tempo: ogni router deve calcolare l'albero dei cammini minimi per la coppia (sorgente, gruppo);
- risorse di memoria: ogni sorgente ha un albero distinto verso ogni destinazione ⇒ è presente nella tabella di instradamento una entry per ogni coppia (sorgente, gruppo) attiva;
- risorse della CPU: l'esecuzione dell'algoritmo di Dijkstra per il calcolo dell'albero di instradamento è pesante per i router.

7.3 Instradamento multicast con algoritmo core-based tree

L'albero di distribuzione multicast è unico per tutto il gruppo multicast e indipendente dalla sorgente (**albero condiviso**). Il **core router** è il router principale dell'albero di distribuzione.

Costruzione dell'albero

1. un host segnala al suo router periferico (leaf router) che vuole agganciarsi al gruppo multicast (sia come ricevitore sia come trasmettitore);
2. il router periferico invia un messaggio di **Join Request** al core router;
3. i router intermedi che ricevono il messaggio di Join Request marcano l'interfaccia dalla quale è arrivato il messaggio come una delle interfacce da usare per l'inoltro dei pacchetti multicast per quel gruppo;
4. quando il core router riceve il messaggio di Join Request, anch'esso marca quell'interfaccia per l'inoltro e la segnalazione si ferma.
   Nel caso in cui il messaggio raggiunga un router che fa già parte dell'albero, la segnalazione si ferma prima di raggiungere il core router, e all'albero precedente viene aggiunto un nuovo ramo.

Inoltro dei dati

1. un membro del gruppo invia semplicemente il pacchetto in multicast;
2. il pacchetto viene inoltrato prima lungo il ramo dalla sorgente al core router, poi sui rami dal core router agli altri membri del gruppo: ogni router che riceve il pacchetto, compreso il core router, lo invia su tutte le interfacce appartenenti a quel gruppo multicast definite nella fase di costruzione dell'albero (tranne quella da cui il pacchetto è arrivato).

Vantaggio scalabilità: poche informazioni di stato nei router.

Svantaggi

- utilizzo di "stati forti": il core router è fisso, e non vengono inviati periodici messaggi di refresh sullo stato dei gruppi multicast ⇒ poco adatto a situazioni altamente variabili;
- il core router è un singolo punto di guasto (anche se si può eleggere un altro router);
- la posizione del core router influenza pesantemente le prestazioni dell'algoritmo: il core router può diventare un collo di bottiglia perché tutto il traffico passa attraverso di esso;
- i percorsi non sono ottimizzati: l'albero di distribuzione non è costruito in base alla posizione della sorgente, ma tutti i membri del gruppo possono essere sorgenti.

7.4 Instradamento multicast gerarchico

Per l'instradamento inter-dominio sono necessari algoritmi di tipo gerarchico: la complessità degli algoritmi tradizionali (e le informazioni di stato da tenere) non permettono la scalabilità all'intera Internet.

In generale, entrano in gioco le politiche di instradamento, e gli "host" sono sostituiti dai "domini":

- instradamento non gerarchico: l'host X vuole ricevere i gruppi A, B e C;
- instradamento gerarchico: il dominio Y vuole ricevere i gruppi A, B e C.

Parte II

Protocolli di instradamento

Capitolo 8

Routing Information Protocol

Il **Routing Information Protocol** (RIP) è un protocollo di instradamento intra-dominio basato sull'algoritmo Distance Vector (DV). RIP versione 1 fu definito nel 1988 e fu il primo protocollo di instradamento utilizzato su Internet.

RIP, a seconda dell'implementazione, include i meccanismi dell'orizzonte limitato, del route poisoning e del path hold down per limitare la propagazione di informazioni di instradamento non corrette.

Il RIP è adatto per reti piccole, stabili e omogenee:

- piccole: la metrica è basata semplicemente sul **numero di hop** (ogni link ha costo 1), ma non si può superare il limite di 16 hop ⇒ non sono ammessi più di 15 router in cascata in uno stesso dominio RIP;
- stabili: cambiamenti di stato possono scatenare dei transitori di lunga durata;
- omogenee:
  - link omogenei: non è possibile differenziare i costi su link diversi in base alla larghezza di banda;
  - router omogenei: ogni router deve terminare l'elaborazione prima di produrre il suo nuovo DV ⇒ la durata del transitorio è legata alle prestazioni del router più lento.

8.1 Formato dei pacchetti

I pacchetti RIP hanno il formato seguente:

8	16	32	
Command	Version (1)	0	
Address Family Identifier		0	
IP Address			× N ≤ 25
0			
0			
Metric			

Tabella 8.1: Formato di un pacchetto RIP (da 24 a 504 byte).

dove i campi più significativi sono:

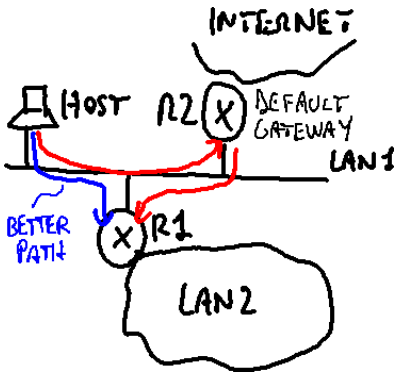
- campo Command (1 byte): specifica il tipo di messaggio:
  - valore "Response": il pacchetto trasporta un DV contenente uno o più indirizzi;

  - valore “Request”: un router appena connesso alla rete notifica i vicini della sua presenza ⇒ i vicini invieranno in risposta i loro DV senza dover attendere il timeout, aumentando la velocità di convergenza;

- campo Address Family Identifier (2 byte): specifica il protocollo di livello rete utilizzato (ad es. valore 2 = IP);
- campo IP Address (4 byte): specifica l'indirizzo IP annunciato (senza netmask). Possono essere annunciati fino a 25 indirizzi in un singolo pacchetto RIP;
- campo Metric (4 byte): specifica il costo relativo all'indirizzo annunciato.

Incapsulamento Il pacchetto RIP è incapsulato in un pacchetto UDP:

- la porta UDP di destinazione è la porta 520, che all'epoca fu scelta come meccanismo di sicurezza dato che le porte inferiori a 1024 possono essere utilizzate solo sotto privilegi amministrativi;
- l'indirizzo IP di destinazione è l'indirizzo broadcast (255.255.255.255) ⇒ tutti gli apparati possono riceverlo, compresi gli host, anche se conviene disabilitare i protocolli di instradamento al lato host per proteggerli da attacchi malevoli e apprendere rotte migliori ascoltando gli eventuali messaggi ICMP Redirect:



8.2 Timer

Il RIP è pesantemente basato su timer:

- è difficile rispettare precisamente dei timer fissati perché la CPU potrebbe essere impegnata ⇒ le incertezze introducono degli ulteriori ritardi;
- tutti i router nella rete devono usare gli stessi timer, altrimenti i router possono interagire in modo non coordinato.

8.2.1 Routing update timer (predefinito 30 s)

Definisce ogni quanto tempo vengono inviati i messaggi Response gratuiti contenenti le informazioni sui DV.

Sincronizzazione dei router Si cerca di evitarla non reimpostando il routing update timer all'invio di un triggered update e inviando i messaggi Response gratuiti con un ritardo variabile tra 25 e 35 secondi.

8.2.2 Route invalid timer (predefinito 180 s)

Definisce il tempo per cui una entry può rimanere valida nella tabella di instradamento senza che venga rinfrescata. Quando scade il route invalid timer, il numero di hop della entry viene impostato al costo infinito (16), contrassegnando la destinazione come irraggiungibile.

Rilevamento dei guasti Il router invalid timer è soprattutto utile per rilevare una connettività mancante verso un vicino quando il segnale "link down" non è disponibile.

8.2.3 Route flush timer (predefinito 240 s)

Definisce il tempo per cui una entry può rimanere nella tabella di instradamento senza che venga rinfrescata. Quando scade il route flush timer, la entry viene cancellata dalla tabella di instradamento.

Route poisoning Quando il route invalid timer scade e la entry viene contrassegnata come non valida, rimangono 60 s (con i valori predefiniti) in cui il router può annunciare la destinazione a costo infinito, per informare gli altri router prima che la entry venga cancellata.

8.2.4 Hold down timer (predefinito 180 s)

Definisce il tempo per cui una entry non può subire modifiche in seguito a un sospetto inizio di count to infinity. L'hold down timer è una funzionalità proprietaria di Cisco.

Path hold down L'hold down timer parte quando il numero di hop sta salendo a un valore più alto, per evitare di innescare un count to infinity e consentire alla rotta di stabilizzarsi.

Route poisoning Anche una destinazione bloccata dall'algoritmo path hold down può essere propagata a costo infinito.

8.3 Limitazioni

8.3.1 Netmask

La prima versione di RIP fu definita quando era ancora in uso l'**indirizzamento classful**, dove la subnet mask può essere ricavata automaticamente dall'indirizzo di rete stesso ⇒ gli indirizzi di rete annunciati nei DV sono privi di informazioni sulle netmask ⇒ la versione 1 di RIP può essere usata solo in reti dove ogni indirizzo appartiene a una **classe di indirizzi** secondo le vecchie regole dell'indirizzamento classful.

È possibile adottare uno stratagemma per far funzionare la versione 1 di RIP in reti con indirizzi a netmask di lunghezza variabile: dato un indirizzo di rete annunciato, il router esamina gli indirizzi di rete assegnati alle interfacce connesse:

- se ad almeno un'interfaccia è assegnato un indirizzo avente una subnet mask uguale alla subnet mask dell'indirizzo annunciato, il router assume come netmask dell'indirizzo annunciato la netmask dell'indirizzo dell'interfaccia;
- se a nessuna interfaccia è assegnato un indirizzo avente una subnet mask uguale alla subnet mask dell'indirizzo dato, il router assume come netmask dell'indirizzo annunciato la sua subnet mask.

Problemi Può essere assunta una netmask errata se:

- nessuna delle interfacce del router ha la subnet mask cercata;
- l'indirizzo annunciato ha in realtà una netmask diversa da quella dell'indirizzo dell'interfaccia selezionata.

8.3.2 Limite di numero di hop

Il RIP definisce il limite di numero di hop uguale a 16 ⇒ le destinazioni la cui distanza è maggiore di 15 sono considerate non raggiungibili.

Un valore massimo così basso è stato scelto per limitare il ben noto problema del count to infinity degli algoritmi basati su DV: quando il costo di una rotta raggiunge il valore 16, la rotta è considerata irraggiungibile e il costo non può salire ancora di più.

Questo non significa che la rete non può avere più di 15 router in cascata: l'unico effetto è che due router troppo distanti non possono comunicare tra loro direttamente. È possibile risolvere questo problema partizionando la rete in due domini di instradamento, controllati da due differenti istanze di protocollo RIP, e attivando il processo di ridistribuzione tra di essi in modo da "falsificare" i costi delle rotte esterne.

8.3.3 Mancanza del campo "age"

Il RIP non associa un campo "age" alle rotte nei DV ⇒ le informazioni annunciate possono essere vecchie, ma il router che le riceve le assume come nuove e reimposta i timer a zero ⇒ la durata del transitorio aumenta tanto più quanto ci si allontana dal cambiamento di stato.

Esempio In una rete con topologia A—B—C:[1]

- tempo 0 s: avviene un guasto sul link A—B ⇒ il nodo A non è più raggiungibile;
- tempo 179 s: il nodo B annuncia al nodo C il suo DV, l'ultimo a contenere la destinazione A;
- tempo 180 s: il nodo B contrassegna la destinazione A come non raggiungibile;
- tempo 359 s: il nodo C contrassegna la destinazione A come non raggiungibile.

8.4 RIP versione 2

RIP versione 2 estende la prima versione di RIP sfruttando alcuni campi che erano inutilizzati nei messaggi:

8	16	32	
Command	Version (2)	Routing Domain	
Address Family Identifier		Route Tag	
IP Address			1 ≤ N ≤ 25
Subnet Mask			
Next Hop			
Metric			

Tabella 8.2: Formato di un pacchetto RIP versione 2 (da 24 a 504 byte).

dove i nuovi campi sono:

- campo Routing Domain (2 byte): specifica il dominio di instradamento a cui è destinato questo messaggio RIP per la gestione di più domini di instradamento sullo stesso router di frontiera:

[1] Si assumono: valori dei timer predefiniti, no triggered update, no route poisoning.

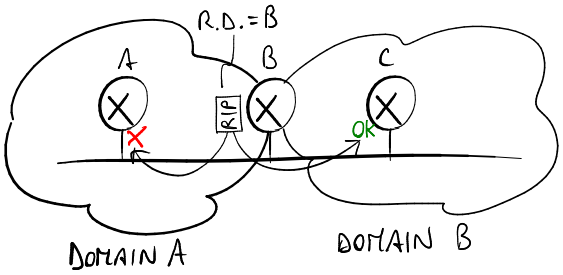


Figura 8.1: Il router A scarta i messaggi destinati al dominio B.

- campo Route Tag (2 byte): specifica se l'indirizzo annunciato è una **rotta esterna**, ovvero è stata appresa tramite un processo di ridistribuzione da un altro dominio di instradamento;
- campo Subnet Mask (4 byte): contiene la **netmask** associata all'indirizzo di rete annunciato per supportare l'indirizzamento classless;
- campo Next Hop (4 byte): ottimizza l'instradamento quando più router RIP appartengono alla stessa LAN ma a due domini RIP diversi, e quindi il traffico da un dominio all'altro passerebbe sempre dal router di frontiera ⇒ il router di frontiera può annunciare di mandare il traffico direttamente al router next hop nell'altro dominio:

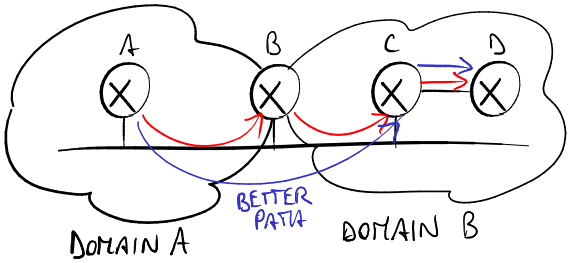


Figura 8.2: Il router di frontiera B insegna al router A di usare il router C come next hop per la destinazione D.

8.4.1 Autenticazione

RIP versione 2 introduce un meccanismo di **autenticazione basata su password**: un router deve essere autenticato per poter annunciare il suo DV ai suoi vicini.

Se la prima entry nel pacchetto RIP ha il campo "Address Family Identifier" uguale al valore 0xFFFF, allora il resto della entry contiene le informazioni di autenticazione:

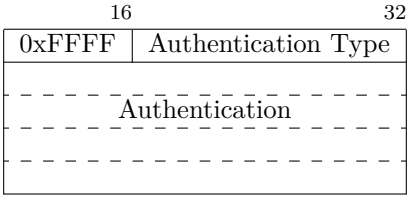


Tabella 8.3: Formato della entry di autenticazione in un pacchetto RIP.

dove i campi sono:

- campo Authentication Type (2 byte): solo il tipo "password semplice" è stato definito (valore 2);

- campo Authentication (16 byte): contiene la password in chiaro.

Questo meccanismo di autenticazione è piuttosto debole perché la password può essere facilmente sniffata ⇒ è raramente utilizzato. Meccanismi di autenticazione più complessi non sono possibili a causa della mancanza di spazio nel messaggio RIP.

8.4.2 Multicast

RIP versione 1 invia il DV in broadcast ⇒ tutte le entità, compresi gli host, devono elaborare i messaggi RIP.

RIP versione 2 definisce un indirizzo IP multicast (224.0.0.9) di destinazione, in modo che il pacchetto RIP venga ricevuto solo dalle entità che sono iscritte al **gruppo multicast** ⇒ gli host e i router che non utilizzano il protocollo RIP possono scartare il pacchetto a livello data-link.

8.5 Vantaggi

- è adatto per reti piccole, stabili e omogenee;
- richiede poche risorse di elaborazione;
- è semplice da implementare;
- è semplice da configurare (non ci sono i sottodomini come in OSPF);
- è disponibile su un'ampia gamma di apparati, anche su router economici.

Capitolo 9

IGRP e EIGRP

L'**Interior Gateway Routing Protocol** (IGRP) è un protocollo di instradamento intra-dominio, proprietario di Cisco, basato sull'algoritmo Distance Vector (DV).

Anche nell'IGRP è assente il supporto all'**indirizzamento classless** (netmask), ma rispetto al RIP ha delle funzionalità aggiuntive "orientate al marketing" che tuttavia nascondono degli errori tecnici non previsti:

- metriche più sofisticate: introducono maggiore complessità e minore stabilità delle rotte;
- multipath routing: il multipath routing a costi differenziati può dare origine a cicli;
- supporto per reti eterogenee: un intervallo ampio per i costi dei link può rallentare la convergenza ad infinito;
- meno traffico legato al protocollo di instradamento: l'aggiornamento dei DV avviene ogni 90 secondi;
- maggiore stabilità: i triggered update sono inviati solo se il costo è cambiato più del 10% per evitare la frequente riconfigurazione della rete;
- non più di una frammentazione IP: i messaggi IGRP trasportano anche informazioni sulle MTU supportate dai router lungo il percorso ⇒ il pacchetto può essere frammentato subito in base alla minima MTU, evitando che in un secondo tempo venga ri-frammentato con una MTU più piccola.

9.1 Metriche

Il costo C è ottenuto dalla combinazione di 4 metriche:

$$\begin{cases} C = \dfrac{10^7}{B}\left(k_1 + \dfrac{k_2}{256 - L}\right) + k_3 D & \text{se } k_5 = 0 \\ C = \left[\dfrac{10^7}{B}\left(k_1 + \dfrac{k_2}{256 - L}\right) + k_3 D\right] \dfrac{k_5}{R + k_4} & \text{se } k_5 \neq 0 \end{cases}$$

B - **larghezza di banda**: è direttamente proporzionale alla larghezza di banda del link (valori da 1 a 2^{24} con 1 = 1,2 kbit/s);

D - **ritardo**: è inversamente proporzionale alla larghezza di banda del link, e considera solo il ritardo di trasmissione ignorando altri componenti come il ritardo di propagazione e il ritardo di accodamento (valori da 1 a 2^{24} con 1 = 10 ms);

R - **affidabilità**: può essere piuttosto variabile nel tempo (valori da 1 a 255 con 255 = 100%);

L - **carico**: dipende dal traffico istantaneo (valori da 1 a 255 con 255 = 100%).

Con i valori predefiniti dei coefficienti $k_{1...5}$, il costo tiene conto solo del ritardo D e della larghezza di banda B:

$$k_1 = k_3 = 1,\ k_2 = k_4 = k_5 = 0 \Rightarrow C = \frac{10^7}{B} + D$$

I comandi IGRP richiedono la specificazione di una classe di servizio (TOS), ma in pratica l'instradamento basato sulle classi di servizio non è stato mai implementato in questo protocollo, perché richiederebbe una tabella di instradamento e una funzione di costo differenti per ogni classe di servizio.

Problemi Una metrica così sofisticata soffre in realtà di alcuni problemi dal punto di vista tecnico:[1]

- è difficile capire le scelte di instradamento: gli esseri umani guardano la topologia della rete e misurano la distanza in "numero di hop" ⇒ non è facile determinare qual è il percorso migliore quando viene adottata una metrica più sofisticata;
- è difficile capire come regolare i coefficienti $k_{1...5}$: che cosa succede alla rete quando i parametri vengono modificati? quali valori bisogna dare a essi al fine di ottenere il comportamento voluto?
- alcune metriche (ad es. carico), non essendo molto stabili, forzano la rete ad adattare continuamente i percorsi perché questi ultimi cambiano spesso di costo ⇒ la necessità di aggiornare frequentemente le rotte porta a più transitori con conseguenti buchi neri e bouncing effect, più traffico di instradamento e più risorse della CPU dedicate ai protocolli di instradamento;
- è difficile definire il giusto valore soglia per l'infinito: IGRP lo definisce a 2^{24}, ma è necessario troppo tempo per aumentare i costi fino al valore soglia quando sono coinvolti link a basso costo.[2]

9.2 Multipath routing

L'IGRP supporta il **multipath routing a costi differenziati**: sono ammesse più rotte per la stessa destinazione, anche se quelle rotte hanno costi differenti ($c_{\text{sec}} \leq V \cdot c_{\text{prim}}$), e il carico è distribuito proporzionalmente al costo della rotta.[3]

Problema Il traffico può entrare in un ciclo quando percorsi differenti vengono scelti da due router: uno può scegliere il percorso primario (rotta ottimale) e l'altro il percorso secondario (rotta sub-ottimale) ⇒ nell'ultima versione di IGRP è consentito solo il multipath routing a costi equivalenti (il coefficiente V è impostato a 1) al fine di impedire queste problematiche.

9.3 EIGRP

L'**Enhanced IGRP** introduce diversi miglioramenti all'IGRP, soprattutto dal punto di vista della scalabilità:

- supporta l'**indirizzamento classless**: le reti sono finalmente annunciate con le coppie indirizzo-netmask corrette;
- implementa il **Diffusing Update Algorithm** (DUAL): la rete è libera da cicli, anche durante i transitori, e la convergenza è più rapida (nessun fenomeno di count to infinity);[4]

[1] Si veda la sezione 2.1.
[2] Si veda la sezione 3.3.1.
[3] Si veda la sezione 2.3.1.
[4] Si veda la sezione 3.4.

- disaccoppia la funzione di **neighbor discovery** dal meccanismo di aggiornamento delle rotte: i router si scambiano periodicamente dei piccoli messaggi di Hello, mentre i DV vengono generati solo quando è cambiato qualcosa nella rete:
  - i messaggi di Hello possono essere mandati ad alta frequenza, rendendo più veloci il rilevamento dei guasti e quindi la convergenza, perché:
    - consumano meno banda ⇒ il traffico di instradamento è ridotto;
    - consumano meno risorse della CPU rispetto all'elaborazione e al calcolo dei DV;
  - i DV devono essere inviati tramite un protocollo affidabile: ogni DV deve essere confermato da un messaggio di acknowledgment, e deve essere ritrasmesso se è andato perso.

Capitolo 10

Open Shortest Path First

L'**Open Shortest Path First** (OSPF) è un protocollo di instradamento intra-dominio basato sull'algoritmo Link State (LS). Le prime due versioni sono utilizzate con IPv4, mentre la versione 3 è pensata per IPv6.

Il principale vantaggio di OSPF rispetto agli altri protocolli di instradamento intra-dominio è la scalabilità (fino a qualche centinaio di router):

- algoritmo LS: la conoscenza della topologia della rete permette una maggiore stabilità rispetto ai protocolli basati sull'algoritmo Distance Vector;
- instradamento gerarchico: OSPF suggerisce di non avere più di 200 router in una singola area:
  - le informazioni di instradamento sulle altre aree possono essere sommarizzate;
  - i cambiamenti delle rotte in un'area non perturbano le altre aree.

10.1 Aree

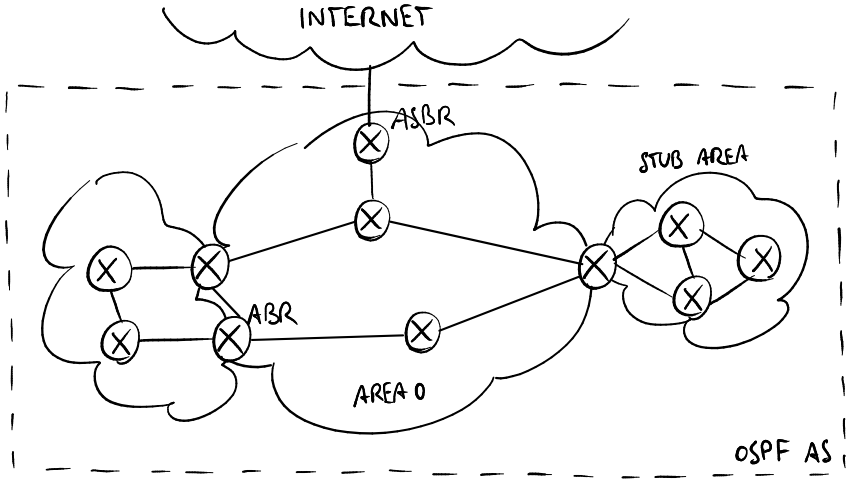


Figura 10.1: Esempio di rete OSPF.

OSPF definisce la propria terminologia, che non è sempre allineata a quella degli altri protocolli:

Autonomous System (AS) un dominio sotto il controllo di una singola entità dal punto di vista amministrativo (ad es. GARR)[1]

[1] Si veda la sezione 6.1.

Autonomous System OSPF un dominio sotto il controllo di una singola entità dal punto di vista tecnico (cioè di configurazione degli apparati), e gestito da una singola istanza di protocollo OSPF

Autonomous System boundary router (ASBR) un router di frontiera posto tra l'AS OSPF (tipicamente l'area 0) e un dominio di instradamento esterno (EGP, oppure IGP se l'AS OSPF convive all'interno dell'AS stesso insieme ad altri domini di instradamento con protocolli IGP anche differenti)

area edge uno dei sotto-domini gerarchici in cui è suddiviso un AS OSPF, composto da una rete fisicamente contigua: ogni router interno può comunicare con qualsiasi altro router nella stessa area senza dover uscire dall'area stessa

area 0 l'area di backbone, non necessariamente fisicamente contigua[2], attraverso la quale deve passare tutto il traffico tra un'area edge e l'altra o tra un'area edge e l'esterno dell'AS OSPF:

- no collo di bottiglia: i link non devono essere sottodimensionati
- robusta: non deve diventare un'area partizionata

area border router (ABR) un router di frontiera posto tra l'area 0 e un'area edge

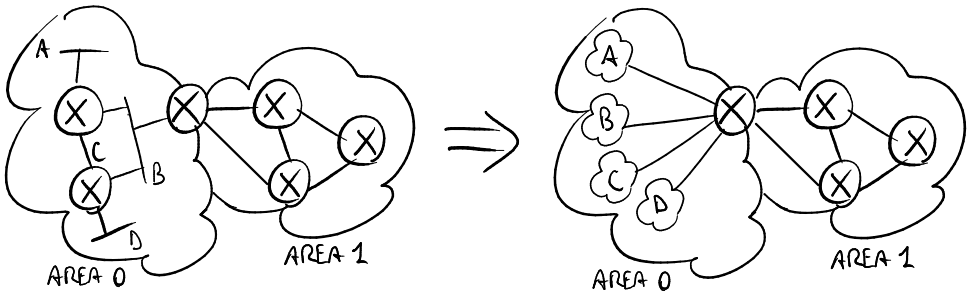


Figura 10.2: Vista della rete dall'area 1.

Ogni router conosce perfettamente la topologia dell'area a cui appartiene, ma la topologia precisa delle altre aree non è nota: il router può conoscere la lista delle destinazioni raggiungibili al di fuori della sua area, che possono venire sommarizzate o sostituite da una rotta di default.

Il database di un router interno contiene tre tipi di record:

- Link State: sono generati dagli altri router interni all'area e contengono le rotte interne all'area, incluse le informazioni sulla topologia, che non vengono mai sommarizzate.
  Un ABR conosce i Link State di entrambe le aree che connette: ha più database, uno per ogni area, che danno origine naturalmente a una singola tabella di instradamento;
- Summary/External Record: contengono le rotte esterne all'area, escluse le informazioni sulla topologia (solo indirizzo di rete + netmask), che possono essere sommarizzate:
  - Summary Record: sono generati dall'ABR e contengono le rotte esterne in altre aree dello stesso AS OSPF (compresa l'area 0);
  - External Record: sono generati dall'ASBR e contengono le rotte esterne al di fuori dell'AS OSPF.

I router nell'area 0 sono solitamente configurati al fine di aggregare gli indirizzi di rete, così da propagare i riassunti delle reti da un'area all'altra. Tuttavia l'aggregazione deve essere specificata manualmente dall'operatore, al fine di non avere problemi con la sommarizzazione di rete.

[2]Si rimanda alla sezione 10.1.2.

10.1.1 Aree stub

Un'area edge normale, oltre a conoscere i dettagli sulla topologia di tutte le rotte interne all'area stessa, importa tutte le rotte esterne dall'area 0 senza alcuna ulteriore aggregazione.

Un'area è **stub** quando alcune rotte esterne sono sostituite da una singola rotta di default per ridurre le informazioni di instradamento importate dall'esterno:

- area stub: mantiene i Summary Record, ma rimuove gli External Record;
- area totally stubby: rimuove sia i Summary Record sia gli External Record, lasciando solo i Link State e la singola rotta di default verso l'uscita;
- area not-so-stubby: è simile all'area stub, ma può iniettare in altre aree le rotte esterne all'AS OSPF (senza importarle nell'area).

Le aree stub sono attivate su esplicita configurazione del gestore della rete:

- area stub: `area xx stub`
- area totally stubby: `area xx stub no summary`
- area not-so-stubby: `area xx nssa`

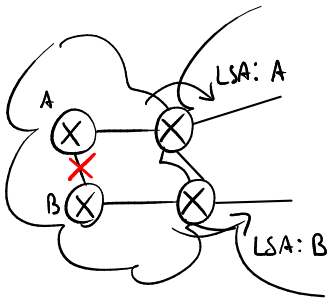
Sebbene OSPF non impedisca di avere un'area stub con più di un ABR, ha più senso configurare un'area stub quando è connessa all'area 0 tramite un solo ABR: non è necessario propagare le rotte esterne perché c'è un solo percorso che connette l'area al resto della rete, mentre le rotte esterne sono utili solo se esiste più di un router egress.

10.1.2 Virtual Link

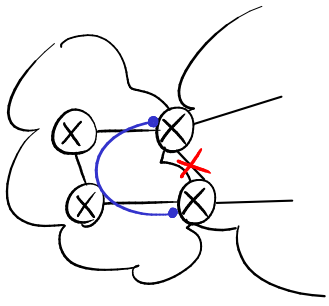
Il **Virtual Link** è una sorta di "tunnel" tra due router, di cui almeno uno appartenente all'area 0, che logicamente appartiene all'area 0, ma fisicamente è composto da una sequenza di link all'interno di un'area edge. Lo scopo è far credere a OSPF che quei due router siano collegati da un link fittizio in area 0.

L'attivazione del Virtual Link richiede solo l'area da attraversare (una sola) e i Router ID dei due router coinvolti, non gli indirizzi IP delle loro interfacce: OSPF deriverà automaticamente gli indirizzi IP corretti. I messaggi di instradamento OSPF sono incapsulati in pacchetti unicast IP che attraversano il link ⇒ per avere un tunnel bidirezionale, occorre configurare il Virtual Link su entrambi i router alle estremità.

Aree partizionate



(a) Partizionamento di un'area edge.



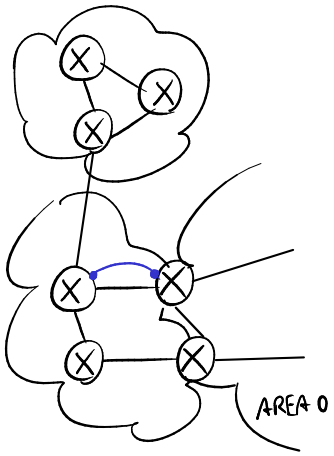
(b) Partizionamento dell'area 0.

Il problema delle aree partizionate[3] è gestito in OSPF differentemente a seconda del tipo di area:

[3] Si veda la sezione 5.1.

- area edge: l'ABR non sommarizza le informazioni su tutte le reti presenti nell'area edge, ma annuncia solo le reti che è in grado di raggiungere ⇒ i pacchetti verso la partizione verranno inviati solo agli ABR per cui esiste un percorso interno per giungere a destinazione;
- area 0: OSPF non è in grado di risolvere automaticamente i problemi di aree partizionate nel backbone (viene scelto sempre l'ABR più vicino come punto di uscita), ma in alcuni casi l'operatore può attivare manualmente tra due ABR un Virtual Link che fisicamente passa attraverso un'area edge: quando un pacchetto arriva a un ABR, rientra nell'area andando all'ABR all'altra estremità del tunnel per poi finalmente entrare nell'area 0 ⇒ l'area 0 deve sempre essere logicamente contigua, ma non necessariamente lo è fisicamente.

Estensione dell'area di backbone



Un link tra due router appartenenti ad aree edge differenti normalmente non può essere usato: il traffico da un'area all'altra infatti deve passare sempre attraverso l'area 0.

Grazie al Virtual Link è possibile portare nel backbone un router di un'area edge che è collegato direttamente a un solo router di backbone: quel router interno diventa un ABR di accesso all'area 0, e quindi tutti i link ad esso collegati possono essere usati per il traffico.

10.2 Metriche e costi

OSPF supporta più di una metrica simultaneamente su un singolo link: il percorso migliore può essere, a seconda dei pacchetti, ad esempio:

- il percorso più breve;
- il percorso con la migliore capacità di banda;
- il percorso con il minore ritardo.

OSPF consente di definire le metriche a seconda del campo "Type of Service" (ToS) del pacchetto IP ⇒ in teoria, sono possibili 64 tipi di servizio e quindi 64 alberi di instradamento diversi, ma in pratica questa funzione è quasi inutilizzata perché il carico di elaborazione richiesto dall'algoritmo LS su ogni router sarebbe duplicato per ogni ToS.

OSPF adotta il multipath routing a costi equivalenti. A differenza di IGRP, OSPF non definisce un modo non ambiguo per calcolare il costo di un link: il costo è assegnato dal produttore dell'apparato di rete ⇒ ogni produttore ha i propri valori predefiniti, creando possibili inconsistenze in reti multi-vendor ⇒ è meglio personalizzare i valori di costo sui link più importanti (su entrambe le estremità).

10.3 Router ID

Ogni router OSPF è identificato univocamente da un **Router ID**, che è usato come "nome" dei router nei pacchetti OSPF (ad es. come sorgente nell'intestazione OSPF).

OSPF non specifica come deve essere determinato il Router ID, ma si limita a specificare che deve essere un identificativo univoco lungo 32 bit. Sugli apparati Cisco, i Router ID possono essere ottenuti in due modi:

- manualmente: l'amministratore di rete configura esplicitamente il valore del Router ID (in IPv4 tipicamente non si fa, mentre in IPv6 è obbligatorio[4]);
- automaticamente: un algoritmo ricava il Router ID dagli indirizzi IPv4 del router:
  - se c'è almeno un'interfaccia di loopback, il Router ID è uguale all'indirizzo più grande tra quelli delle interfacce di loopback: le interfacce di loopback non dipendono dallo stato delle interfacce fisiche e sono così più stabili;
  - se non c'è alcuna interfaccia di loopback, il Router ID è uguale all'indirizzo più grande tra quelli delle interfacce di rete OSPF.

10.4 LSA

Il **Link State Advertisement** (LSA) è la struttura dati, contenuta nei pacchetti di Link State Update, che contiene le informazioni di instradamento OSPF:

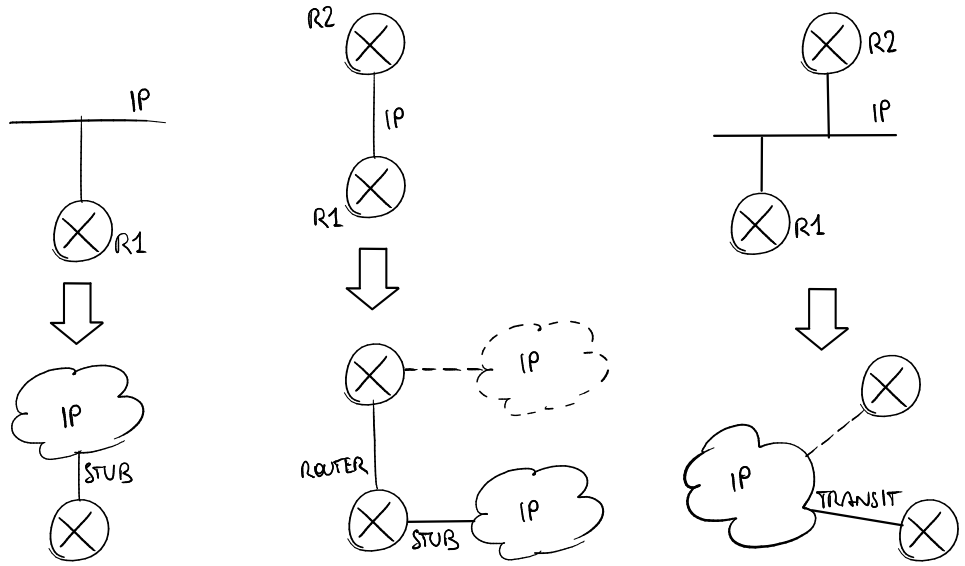
1. **Router LSA**: descrive un'adiacenza tramite un link punto-punto;
2. **Network LSA**: elenca i router collegati a una rete di transito, ed è generato dal router designato della rete di transito;
3. **Network Summary LSA**: contiene le rotte esterne in altre aree dello stesso AS OSPF (Summary Record), ed è generato da un ABR;
4. **ASBR Summary LSA**: comunica la posizione dell'ASBR se esso non si trova nell'area 0 ma in un'area edge.
5. **AS External LSA**: contiene le rotte esterne al di fuori dell'AS OSPF (External Record), ed è generato da un ASBR.

10.4.1 Router LSA

OSPF definisce due tipi di link:

- **router link** (predefinito) (figura 10.4b): in presenza di un link punto-punto tra due router (ad es. interfaccia seriale), ciascuno dei router lo vede suddiviso logicamente in due link punto-punto:
  - una connessione punto-punto con il router adiacente, identificato dal suo Router ID;
  - una connessione punto-punto con la rete IP adiacente, chiamata **rete stub**[5], cioè quella a cui appartiene l'interfaccia di rete del router.
    Se l'interfaccia di un router è attiva ma non è collegata ad alcun altro router, solo la connessione con la rete stub è presente (figura 10.4a);
- **network link** (figura 10.4c): in presenza di una rete broadcast (ad es. Ethernet), chiamata **rete di transito**, ciascuno dei due[6] o più router ad essa collegati la vede logicamente come una connessione punto-punto con la rete di transito adiacente.

[4] Si rimanda alla sezione 13.2.2.
[5] La "rete stub" non è da confondere con la "area stub".
[6] OSPF non impedisce di configurare un link punto-punto tra due router come una rete di transito.



(a) Router singolo. *(b) Router link.* *(c) Network link.*

Figura 10.4: Le adiacenze viste dal router R1.

I Router LSA possono descrivere diversi tipi di adiacenze tramite link punto-punto:

1. adiacenza con un router: è generato da ciascuno dei due router adiacenti tra loro;
2. adiacenza con una rete di transito: è generato da ciascuno dei router adiacenti alla rete di transito (compreso il router designato);
3. adiacenza con una rete stub: è generato dal router la cui interfaccia è adiacente alla rete stub;
4. adiacenza con un Virtual Link: è generato da ciascuno dei router alle estremità del Virtual Link.

10.5 Pacchetti OSPF

Tutti i pacchetti OSPF vengono incapsulati direttamente in IP (Protocol Type = 89), senza l'ausilio di un protocollo di trasporto intermedio.

Un'intestazione OSPF, uguale per tutti i pacchetti, specifica il tipo del pacchetto OSPF trasportato:

- tipo 1: **Hello**
- tipo 2: **Database Description**
- tipo 3: **Link State Request**
- tipo 4: **Link State Update**
- tipo 5: **Link State Acknowledgement**

10.5.1 Protocollo di hello

Il **protocollo di hello** effettua il rilevamento dei guasti senza affidarsi al livello fisico.

I pacchetti Hello vengono inviati ogni HelloInterval (predefinito = 10 s), e l'adiacenza è considerata scomparsa e non viene più annunciata:

- appena il guasto è rilevato a livello fisico;
- dopo un certo numero di pacchetti Hello persi (RouterDeadInterval predefinito = 40 s, equivalente a 4 pacchetti Hello), se il guasto non può essere rilevato a livello fisico.

L'LSA tuttavia rimane nel database OSPF, e se non rinnovato scade dopo un tempo pari al MaxAge (predefinito = 1 ora).

Il Router ID è calcolato all'avvio del processo OSPF, e non viene modificato anche se vengono modificati gli indirizzi IP sul router ⇒ il router potrebbe apparire con un Router ID diverso al riavvio del processo OSPF (ad es. in seguito a un guasto o a un'interruzione dell'alimentazione) ⇒ nella topologia calcolata dall'algoritmo LS rimane un nodo non più esistente, fino a quando non scade il relativo LSA.

10.5.2 Protocollo di exchange

Il **protocollo di exchange** è usato per effettuare il riallineamento delle adiacenze, cioè per sincronizzare il database di due router quando diventano adiacenti.

Il riallineamento delle adiacenze viene svolto solo quando necessario, cioè quando un router ha delle informazioni non aggiornate nel suo database. La verifica della necessità di un aggiornamento del database viene fatta:

- in seguito a un cambiamento nella rete (ad es. in fase di avvio del sistema o quando diventa attivo un nuovo link);
- a ogni scadenza del timer LSA Refresh (predefinito = 30 minuti), per rinfrescare l'age degli LSA ancora validi prima che scadano.

Durante il riallineamento delle adiacenze, vengono scambiati solo gli LSA vecchi o mancanti:

1. Database Description: il router master invia la lista dei numeri di sequenza di tutti gli LSA nel suo database;
2. Link State Request: il router slave invia la lista dei numeri di sequenza relativi agli LSA vecchi o mancanti nel suo database;
3. Link State Update: il router master invia gli LSA richiesti, e il pacchetto Link State Update viene propagato in selective flooding;
4. Link State Acknowledgement: il router slave conferma la ricezione del Link State Update.

Capitolo 11

Instradamento inter-dominio: peering e transito in Internet

Il traffico all'interno dell'AS è quasi "gratis", escludendo i costi dell'infrastruttura (manutenzione, amministrazione, elettricità, ecc.) ⇒ gli ISP provano a convincere gli utenti a trascorrere la maggior parte del loro tempo all'interno dell'AS.

Tuttavia, un AS deve connettersi ad altri AS per due motivi:

- un AS deve essere in grado di raggiungere tutte le destinazioni presenti su Internet per la legge di Metcalfe (= la rete dev'essere la più estesa possibile per essere utile);
- un AS vorrebbe raggiungere la resilienza nelle sue connessioni verso il mondo esterno.

Gli AS su Internet sono interconnessi con un'organizzazione gerarchica:

- **Tier 1** (ad es. Seabone, Sprint): operatore internazionale che interconnette le grandi città con link a larga banda e a lunga distanza e trasporta grandi flussi di traffico lungo le dorsali;
- **Tier 2** (ad es. Telecom Italia): operatore nazionale che raccoglie il traffico dai singoli utenti attraverso molti punti di accesso grazie alla sua presenza capillare sul territorio;
- **Tier 3**: operatore locale che serve un'area geografica molto ristretta.

11.1 Accordi commerciali tra AS

Le interconnessioni tra un operatore e un altro possono non essere gratuite: di solito, l'interconnessione tra due AS è stabilita solo al momento di un **accordo economico**. Due tipi di accordi sono possibili:

- transito: rappresenta la scelta più naturale dal punto di vista economico (sezione 11.1.1);
- peering: quando due AS scoprono che possono fare di meglio (sezione 11.1.2).

L'instradamento inter-dominio su Internet è principalmente guidato dagli accordi commerciali tra gli operatori ai vari livelli gerarchici:

- Tier 1: può annunciare, indipendentemente dalla copertura geografica della sua rete, la raggiungibilità della full route (0.0.0.0/0), cioè la raggiungibilità di (quasi) ogni AS di destinazione su Internet, senza dover acquistare il transito da altri provider né pagare qualche tassa di accesso;
- Tier 2: deve acquistare il transito da un operatore Tier 1 al fine di poter raggiungere l'intera Internet, e può stabilire molti accordi di peering con altri provider Tier 2;
- Tier 3: non ha alcun accordo di peering, e semplicemente acquista il transito da un provider Tier 2 (o Tier 1).

11.1.1 Transito

Un accordo è di **transito** quando un ISP deve pagare un altro ISP per connettersi al suo AS. L'ISP che riceve il denaro garantisce il "transito", cioè il diritto di utilizzare la sua rete, al traffico proveniente dall'altro AS.

L'accordo economico può stabilire:

- la modalità di pagamento:
  - tariffa a volume: una quantità massima di byte di dati al giorno o al mese, più un costo aggiuntivo per il traffico eccedente quella quantità;
  - tariffa flat: una tassa mensile per una banda massima (la larghezza di banda può essere limitata via software sull'interfaccia di accesso).
- quali destinazioni sono raggiungibili attraverso il transito:
  - full route: tutte le destinazioni nel mondo devono essere raggiungibili;
  - solo le destinazioni in una certa area geografica (ad es. USA): i pacchetti diretti verso altre destinazioni vengono scartati.

Il prezzo può essere influenzato dall'importanza dell'ISP che vende il transito:

- un ISP statunitense ha il controllo della parte più importante della rete perché all'interno del suo AS sono presenti i server Web più visitati al mondo;
- un ISP molto grande può offrire una buona raggiungibilità con il resto del mondo grazie all'elevato numero di interconnessioni.

11.1.2 Peering

Un accordo è di **peering** quando due ISP paritetici si mettono d'accordo per scambiare il traffico tra essi stessi senza dover pagare l'un l'altro.

Due ISP possono decidere di stipulare un accordo di peering se determinano che i costi per l'interconnessione diretta sono più bassi dei costi per l'acquisto del transito l'uno dall'altro: i costi di installazione e di manutenzione del link diretto tra gli AS sono equamente suddivisi tra i due ISP, che possono inviare dati alla massima velocità consentita dal link.

Gli operatori Tier 1 operano in un mercato molto competitivo:

- gli operatori Tier 2 possono stabilire dei nuovi accordi di peering tra loro non appena diventano più convenienti del transito;
- un operatore Tier 2 può in breve tempo spostarsi a un operatore Tier 1 più conveniente;
- un operatore dominante può essere costretto dal garante del mercato a offrire delle connessioni in peering con gli ISP minori.

11.2 Politiche di instradamento

Nell'instradamento inter-dominio, altri requisiti sono più importanti della semplice connettività di rete:

- economici (chi paga per la banda?): qualche volta si può preferire percorsi più lunghi ai percorsi migliori (sezione 11.2.1);
- amministrativi (è consentito passare?): qualche volta si può omettere alcuni percorsi all'altro party (sezione 11.2.2);
- di sicurezza (quel dominio amministrativo è fidato?): qualche volta si può preferire percorsi più sicuri (e più lunghi) ai percorsi migliori (sezione 11.2.3).

Il percorso scelto dal protocollo di instradamento non è necessariamente il percorso a costo inferiore dal punto di vista tecnico, ma è il percorso migliore tra quelli che soddisfano i vincoli stabiliti dalle **politiche di instradamento** configurate dall'amministratore di rete, che riflettono gli accordi commerciali tra gli AS.

Il **processo di decisione** sui router di frontiera è influenzato dalle politiche di instradamento:

- tabella di instradamento: può essere favorita la scelta di alcune rotte più economiche e scoraggiata la scelta di altre attraverso AS non fidati;
- annunci di rotte: le rotte annunciate verso altri AS possono non corrispondere alla topologia reale della rete.

11.2.1 Requisiti economici

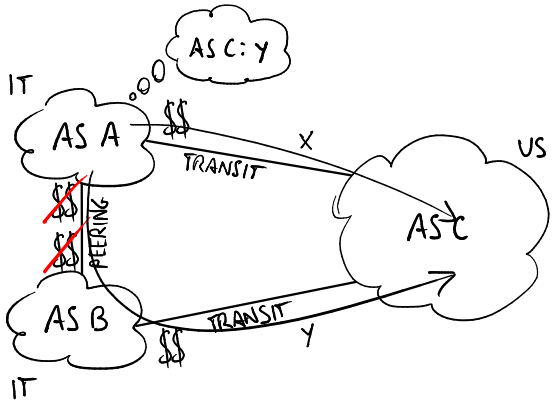


Figura 11.1: Esempio di freeriding.

L'invio del traffico su un link in transito costa ⇒ un AS può avvantaggiarsi di un link in peering, anche se non è un link diretto, per far pagare il costo di transito all'altro AS paritetico (**freeriding**).

Nell'esempio in figura 11.1, i due AS italiani A e B sono interconnessi in peering, e ciascuno di essi è connesso in transito con l'AS statunitense C. Il percorso migliore secondo le regole dell'instradamento tradizionale è il percorso x perché è costituito da un link diretto, ma A deve pagare per far passare il traffico su quel link. A può impostare una politica che preferisce il percorso y più economico: dirotta tutto il traffico diretto a C sul link verso B, che è un link a basso costo per A ⇒ B manderà il traffico di A sul suo link di transito verso C, pagando al posto di A.

11.2.2 Requisiti amministrativi

Un AS può impostare una politica amministrativa con lo scopo di non annunciare a un altro AS la connettività con alcun altro AS (**route hiding**).

Nell'esempio in figura 11.2, B ha un link in transito verso C e lo usa per il suo traffico, ma annuncia la connettività parziale omettendo l'informazione su questo link negli annunci che invia ad A, al fine di evitare che A si avvantaggi del link in peering per risparmiare sul costo di transito (e viceversa). A potrebbe non fidarsi di questo annuncio e a sua volta impostare una politica che forza staticamente tutto il traffico verso C a essere mandato comunque a B ⇒ B si può difendere impostando una **Access Control List** (ACL) sul suo router di frontiera per scartare tutti i pacchetti provenienti da A e diretti verso C.

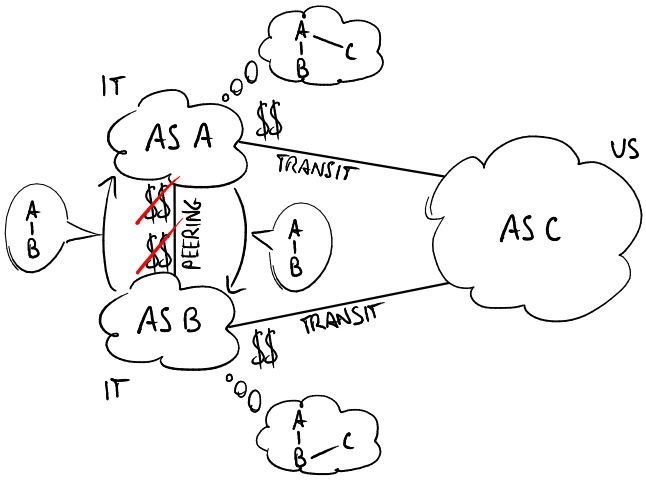


Figura 11.2: Esempio di route hiding.

11.2.3 Requisiti di sicurezza

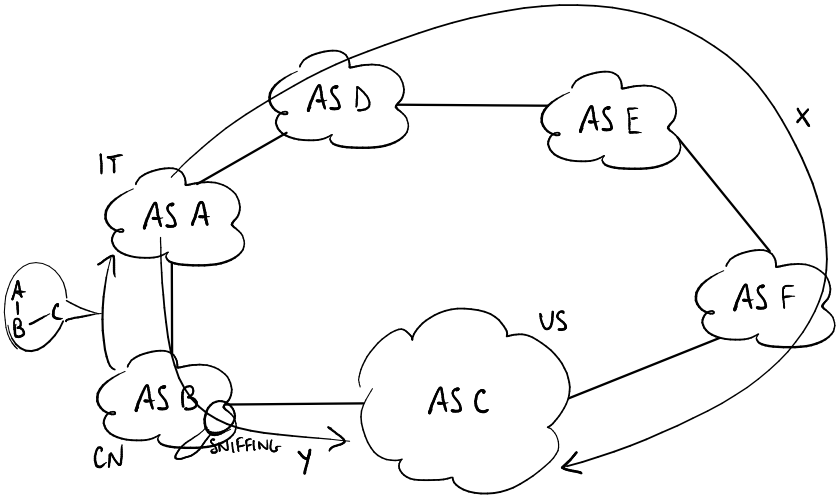


Figura 11.3: Esempio di operatore non fidato.

Un operatore di rete può rappresentare una minaccia di sicurezza perché ad esempio compie abitualmente delle azioni di sniffing sul traffico che passa attraverso il suo AS ⇒ un AS vuole evitare che il suo traffico diretto ad altri AS passi attraverso quell'operatore non fidato.

Nell'esempio in figura 11.3, A per raggiungere C preferisce il percorso x più lungo ma più sicuro perché non attraversa l'operatore B non fidato, anche se quest'ultimo annuncia il percorso y a basso costo verso C.

11.3 Internet Exchange Point

Non è conveniente interconnettere due AS per **connessione diretta**, cioè con un singolo link geografico tra di essi:

- costo del link: l'installazione può richiedere operazioni di scavo;
- costo delle interfacce sui router: devono inviare il segnale a lunghe distanze;

- flessibilità: è necessario intervenire sull'infrastruttura fisica per creare una nuova interconnessione.

Un **Internet Exchange Point** (IXP) consente a più router di frontiera di AS (ISP) differenti di scambiarsi informazioni di instradamento esterno in modo più dinamico e flessibile.

I router sono connessi tramite una rete locale intermedia **di livello data-link**: tecnicamente tutti i router sono direttamente raggiungibili, ma in pratica le politiche di instradamento definiscono le interconnessioni a seconda degli accordi commerciali tra gli AS ⇒ per creare una nuova interconnessione, è sufficiente configurare le politiche di instradamento sui singoli router senza dover modificare l'infrastruttura fisica. Una interconnessione può anche essere attiva ma utilizzata solo come backup (selezione fatta in BGP).

Di solito ogni AS paga una tassa mensile, che dipende ad esempio dalla velocità della connessione all'IXP. L'IXP è responsabile del funzionamento tecnico degli switch nella rete intermedia:

- singola ubicazione: spesso tutti i router sono concentrati all'interno di una stanza in un datacenter, dove vengono forniti:
  - rete di livello data-link ad alta velocità;
  - energia elettrica, sistema di condizionamento;
  - servizio di monitoraggio;
  - vicinanza alle dorsali in fibra ottica;
- infrastruttura distribuita: più punti di accesso sono disponibili nelle principali città sul territorio (ad esempio, il TOPIX si estende per l'intera regione piemontese).

L'IXP è anche noto come **Neutral Access Point** (NAP): l'IXP deve essere neutrale e non coinvolto negli affari dei suoi clienti. Un IXP può decidere di non consentire accordi di transito: ad esempio, il MIX di Milano è un'organizzazione non profit che ammette solamente accordi di peering per favorire la diffusione di Internet in Italia, ma ciò può limitare l'ammontare di traffico scambiato attraverso l'IXP perché gli ISP disponibili solo ad accordi di transito sceglieranno degli altri IXP.

11.4 Neutralità della rete

La **neutralità della rete** è il principio secondo il quale tutto il traffico deve essere trattato equamente, senza privilegiare o danneggiare una parte del traffico per interessi economici.

Gli operatori di rete possono essere tentati di dare un "trattamento preferenziale" a porzioni di traffico:

- privilegiare del traffico: offrire un servizio migliore a un certo tipo di traffico (ad es. maggiore velocità);
- danneggiare del traffico: offrire un servizio peggiore, oppure totalmente nessun servizio, a un certo tipo di traffico.

Una rete neutrale garantisce che tutte le entità (ad es. i content provider) abbiano lo stesso servizio, senza che qualche servizio venga ucciso a discrezione dell'operatore di rete, ma forzare la "pura" neutralità della rete implica che non è assolutamente possibile il controllo del traffico, che può essere utile in molti casi; dall'altro lato, se si ammette che la rete può non essere neutrale, all'operatore di rete è dato il potere di privilegiare del traffico o del contenuto. In un mercato aperto la palla è lasciata all'utente: se gli utenti non sono d'accordo che il loro traffico VoIP venga discriminato, possono passare a un altro operatore di rete (anche se in pratica ciò può non essere sempre possibile a causa dei cartelli tra gli operatori di rete).

Esempi di non neutralità

- content provider: gli ISP vorrebbero avere una parte degli utili dei content provider ⇒ un ISP può privilegiare il traffico diretto a un content provider con cui ha stipulato un contratto di ripartizione degli utili;
- peer-to-peer (P2P):
  - gli utenti finali non si preoccupano della destinazione del loro traffico, ma il traffico P2P può raggiungere ogni utente in ogni AS del mondo facendo pagare degli alti costi all'ISP ⇒ un ISP può privilegiare il traffico che è generato all'interno dell'AS (ad es. AdunanzA di Fastweb);
  - il traffico P2P è più simmetrico perché utilizza molto la banda in upload, mentre le reti sono state dimensionate per supportare un traffico asimmetrico ⇒ un ISP può privilegiare il traffico asimmetrico (ad es. normale traffico Web);
- qualità del servizio (QoS): un ISP può privilegiare il traffico con un livello di priorità maggiore (ad es. traffico VoIP);
- sicurezza: un ISP può bloccare il traffico da utenti malintenzionati (ad es. attacco DDoS).

Capitolo 12

Border Gateway Protocol

Il **Border Gateway Protocol** (BGP) è il protocollo di instradamento inter-dominio comunemente utilizzato in Internet.

Panoramica

- usa l'algoritmo Path Vector (PV) per registrare la sequenza degli AS lungo il percorso senza il rischio di routing loop (sezione 12.1.1);
- i router possono aggregare le informazioni di instradamento ricevute prima di propagarle (sezione 12.1.2);
- non scopre automaticamente l'esistenza di nuovi router vicini, ma le peering session vanno configurate a mano (sezione 12.2);
- scambia gli aggiornamenti di instradamento usando connessioni TCP affidabili (sezione 12.2.1);
- è un protocollo estensibile grazie al formato Type-Length-Value (TLV) degli attributi (sezione 12.3);
- supporta le politiche di instradamento (sezione 12.4).

12.1 Informazioni di instradamento

Il BGP scambia informazioni di instradamento inter-dominio sulle rotte esterne, che sono nella forma indirizzo di rete/lunghezza del prefisso (invece della netmask).

12.1.1 Algoritmo Path Vector

Siccome le politiche di instradamento sono definite in base ai percorsi, il BGP non può basarsi sull'algoritmo DV perché non è sufficiente sapere i loro costi. Il BGP ha scelto di adottare l'algoritmo **Path Vector** (PV)[1]: ogni AS costituisce un singolo nodo, identificato da un numero di 2 (o 4) byte, e l'informazione aggiuntiva è la lista degli AS attraversati.

L'algoritmo PV è più stabile poiché è facile rilevare dei cicli:

- se un router riceve un PV che contiene già il suo numero di AS, scarta il PV senza propagarlo, poiché si sta incorrendo in un routing loop;
- se no, il router inserisce il proprio numero di AS nel PV e quindi lo propaga ai suoi vicini.

[1] Si veda la sezione 3.6.

Il BGP non supporta una esplicita metrica di costo: il costo associato a ogni rotta è semplicemente pari al numero di AS attraversati inclusi nella lista ⇒ la rotta a minor costo può non essere quella effettivamente ottimale:

- gli AS possono avere requisiti diversi, quindi possono adottare metriche diverse l'uno dall'altro (ad es. larghezza di banda, ritardo di trasmissione) ⇒ è difficile calcolare costi coerenti per tutti gli AS;
- il costo annunciato può non corrispondere alla reale topologia della rete perché un ISP può voler nascondere a un concorrente le informazioni reali sulla propria rete per ragioni economiche.[2]

12.1.2 Aggregazione delle rotte

Quando un router di frontiera propaga le informazioni sulle rotte ricevute, può essere configurato manualmente per includere delle **rotte aggregate** nei messaggi di annuncio per ridurne le dimensioni: due rotte possono essere aggregate in una rotta con la parte comune del prefisso di rete.

Tuttavia non tutte le rotte che sono state condensate in una rotta aggregata possono avere la stessa sequenza di AS attraversati, ma ci può essere una rotta più specifica che segue un altro percorso:

- **rotta sovrapposta**: insieme alla rotta aggregata viene annunciata anche la rotta specifica, con la sua lista diversa di AS attraversati ⇒ le informazioni sono complete, e l'algoritmo "longest prefix matching" selezionerà la rotta più specifica nella tabella di instradamento;
- **rotta not precise**: viene annunciata solamente la rotta aggregata ⇒ le informazioni sono approssimative, perché la lista degli AS attraversati non corrisponde al percorso realmente seguito per tutti gli indirizzi di destinazione in quell'address range.

12.2 Peering session

Due router di frontiera che si scambiano dei messaggi BGP vengono chiamati **peer**, e la sessione basata sul TCP è chiamata **peering session**.

Una differenza chiave in confronto agli altri protocolli di instradamento è il fatto che i peer non si sono in grado di scoprirsi a vicenda automaticamente: è necessaria la configurazione manuale da parte dell'amministratore di rete, perché i peer potrebbero non essere connessi tramite un link diretto ma tra di essi potrebbero esistere altri router, per i quali gli aggiornamenti BGP sono dei normali pacchetti di dati da inoltrare a destinazione.

12.2.1 TCP

La trasmissione delle informazioni di instradamento è affidabile perché due peer stabiliscono una peering session instaurando una **connessione TCP** attraverso cui scambiare tutti i messaggi BGP:

- vengono riutilizzati dei componenti esistenti anziché ridefinire ancora un altro meccanismo specifico del protocollo;
- il BGP non deve avere a che fare direttamente con ritrasmissioni, messaggi persi, ecc.

L'utilizzo del TCP come protocollo di trasporto evita l'invio periodico degli aggiornamenti: un aggiornamento è inviato soltanto quando necessario, includendo solo le rotte che sono cambiate, e l'invio viene ripetuto solo se il messaggio è andato perso ⇒ viene ridotta la banda consumata per inviare le rotte.

[2] Si veda la sezione 11.2.2.

Siccome gli annunci non sono periodici, le rotte non scadono mai ⇒ è necessario informare esplicitamente che una rotta precedentemente annunciata è diventata non raggiungibile, per **ritirare** le rotte quando non sono più valide (analogamente al route poisoning dell'algoritmo DV[3]).

Tuttavia il TCP toglie all'applicazione il controllo sulle tempistiche, perché i pacchetti di controllo possono essere ritardati dai meccanismi stessi del TCP: in caso di congestione il TCP riduce il bit rate di trasmissione impedendo la loro trasmissione puntuale ⇒ è possibile configurare la qualità del servizio sui router interni all'AS in modo da dare priorità ai pacchetti BGP, considerando che sono pacchetti di servizio per permettere il funzionamento della rete.

Il TCP non fornisce l'informazione se il peer remoto è ancora raggiungibile ⇒ è richiesto un **meccanismo esplicito di keepalive** gestito dal BGP stesso. Anche i messaggi di keepalive si affidano ai meccanismi del TCP ⇒ la reattività alla scomparsa di un peer o al guasto di un link è limitata, ma è ancora accettabile considerando che questi eventi sono rari (ad es. i link tra router di frontiera sono fortemente ridondati).

12.2.2 I-BGP e E-BGP

Quando due router di frontiera instaurano tra di loro una peering session, ognuno comunica, tramite un messaggio di OPEN, il suo numero di AS all'altro party per determinare il tipo di sotto-protocollo:

- **Exterior BGP** (E-BGP): i peer sono router di frontiera appartenenti a due AS differenti, solitamente connessi con un link diretto;
- **Interior BGP** (I-BGP): i peer sono router di frontiera appartenenti allo stesso AS, solitamente connessi attraverso una serie di router interni.

L'elaborazione dei messaggi BGP e le rotte annunciate nelle peering session possono essere diverse a seconda di a quali AS appartengono i peer:

- E-BGP: quando un router di frontiera propaga un PV a un peer E-BGP, antepone il numero dell'AS corrente a ogni lista di AS attraversati:
  - rotte esterne: sono propagate agli altri peer E-BGP, tranne i peer il cui AS è sul percorso migliore verso quelle destinazioni;
  - rotte interne: sono propagate agli altri peer E-BGP;
- I-BGP: quando un router di frontiera propaga un PV a un peer I-BGP, trasmette la lista così com'è perché il numero dell'AS rimane invariato:
  - rotte esterne: sono propagate agli altri peer I-BGP secondo varie modalità;
  - rotte interne: non sono mai propagate agli altri peer I-BGP, ma ogni router di frontiera le apprende da un processo di ridistribuzione indipendente.

Le sessioni I-BGP sono utilizzate per scambiare le rotte esterne:

- indipendentemente dalle rotte scambiate dal protocollo interior: la connessione diretta tra i peer evita di disturbare il protocollo IGP quando la variazione di una rotta esterna non richiede il ricalcolo delle rotte interne ⇒ nessun transitorio, meno elaborazione;
- indipendentemente dal protocollo interior: se i router di frontiera quando apprendono le rotte esterne dall'E-BGP si limitassero a ridistribuirle al protocollo IGP, lasciando che quest'ultimo le ridistribuisca naturalmente agli altri router di frontiera, verrebbero perse alcune informazioni importanti di cui il BGP ha bisogno ⇒ servono dei messaggi BGP appositi, chiamati UPDATE, che includono queste informazioni nei loro attributi.

[3] Si veda la sezione 3.3.2.

Sincronizzazione IGP-BGP

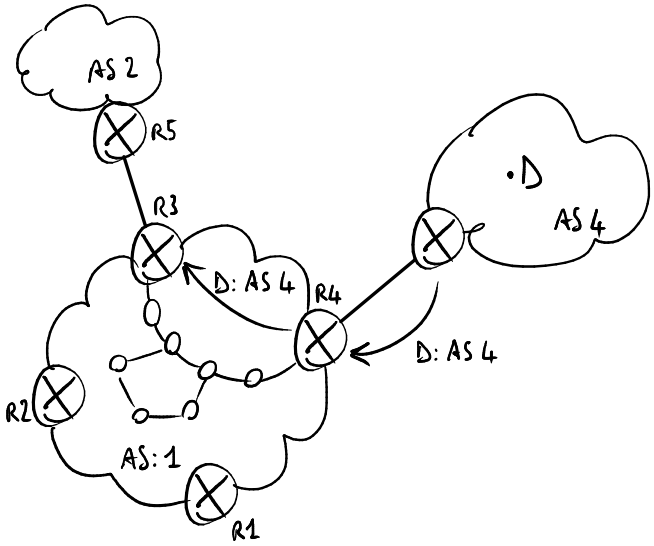


Figura 12.1: Esempio di sincronizzazione IGP-BGP.

I router BGP in un AS di transito apprendono le destinazioni esterne da altri router BGP attraverso l'I-BGP, ma l'instradamento dei pacchetti attraverso l'AS (verso il router BGP egress) si affida ai router interni, le cui tabelle di instradamento sono riempite dal protocollo IGP e non dal BGP ⇒ solo dopo che sono state annunciate anche dal protocollo IGP, le destinazioni esterne possono essere annunciate a router di frontiera di altri AS.

Nell'esempio in figura 12.1, il router R4 apprende la destinazione D tramite E-BGP e la annuncia al router R3 tramite I-BGP, ma R3 non può a sua volta annunciarla al router R5 tramite E-BGP finché la destinazione non è stata ridistribuita dal protocollo IGP a R3, altrimenti se R5 provasse a inviare un pacchetto verso D, R3 lo inoltrerebbe all'interno dell'AS dove i router interni lo scarterebbero.

Potrebbe andare bene disabilitare la sincronizzazione quando:

- l'AS non è di transito;
- tutti i router dell'AS utilizzano il BGP.

12.2.3 Routing loop

La mancanza delle informazioni sui router di frontiera attraversati quando essi appartengono allo stesso AS può essere causa di routing loop: un router di frontiera non può più fare affidamento sulla lista degli AS attraversati per rilevare i percorsi che passano due volte per lo stesso router di frontiera.

Nell'esempio in figura 12.2, si determina un ciclo negli annunci:

1. il router R4 apprende la rotta esterna verso la destinazione D;
2. R4 propaga D al peer R3;
3. R3 propaga D al peer R2;
4. R2 propaga D al peer R4, che è il router che ha inizialmente appreso e annunciato D.

Si crea così una situazione simile a quella che scatenava i count to infinity nell'algoritmo Distance Vector[4]: R4 non può stabilire se R2 sa raggiungere D passando attraverso R4 stesso

[4]Si veda la sezione 3.3.

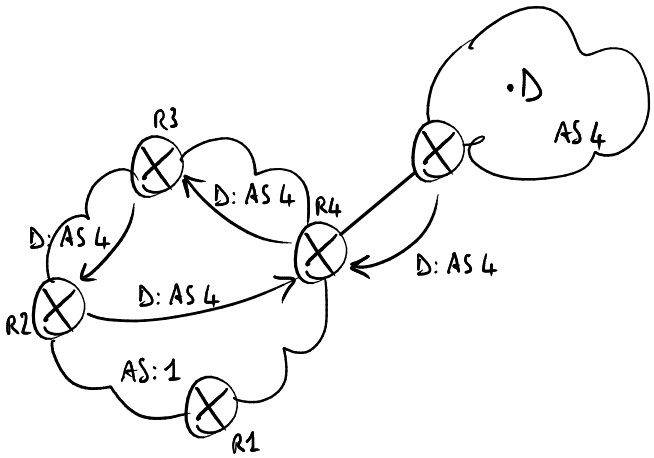


Figura 12.2: Esempio di routing loop.

oppure se esiste un percorso realmente alternativo ⇒ se si verifica un guasto sul link tra R4 e il router di frontiera dell'AS in cui si trova D, R4 crede che D sia ancora raggiungibile attraverso R2.

Le rotte esterne possono essere annunciate ai peer I-BGP secondo varie modalità: maglia completa, route reflector, AS confederation.

Maglia completa

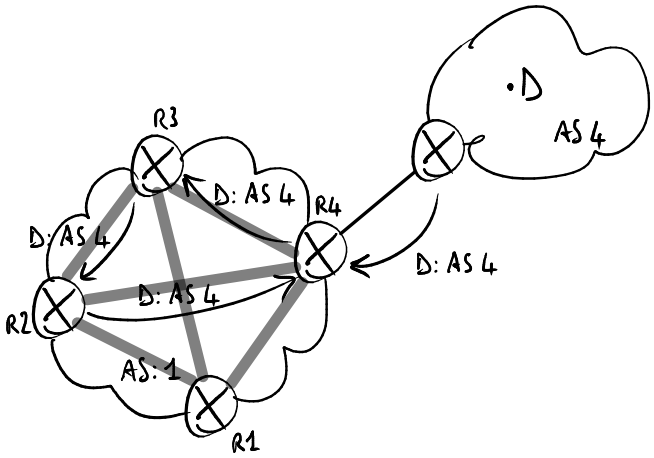


Figura 12.3: Esempio di maglia completa.

Ogni router di frontiera ha una peering session I-BGP con ogni altro router di frontiera del suo AS.

Quando un router di frontiera apprende una rotta esterna dall'E-BGP, la propaga a tutti gli altri, che a loro volta la propagano a tutti, e così via.

In presenza di più di 2 router di frontiera, si possono creare dei routing loop dovuti a cicli negli annunci, come nella figura 12.3.

Questa soluzione non è flessibile perché tutte le peering session vanno configurate a mano, anche se le peering session non cambiano molto nel tempo poiché i router di frontiera sono piuttosto fissi e resistenti ai guasti.

Route reflector

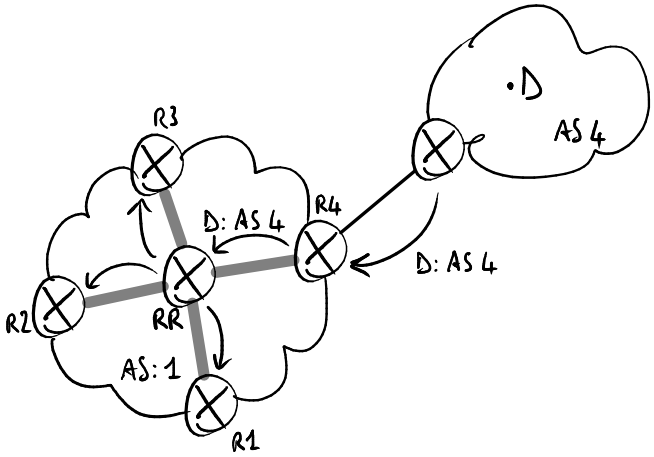


Figura 12.4: Esempio di route reflector.

Uno dei router di frontiera viene eletto **route reflector** (RR), e tutti gli altri router di frontiera instaurano delle peering session solamente con esso senza creare dei percorsi chiusi.

Quando un router di frontiera apprende una rotta esterna dall'E-BGP, la propaga solo al RR, il quale è responsabile di propagare a sua volta la rotta agli altri router di frontiera evitando i routing loop.

Il route reflector costituisce un singolo punto di guasto.

AS confederation

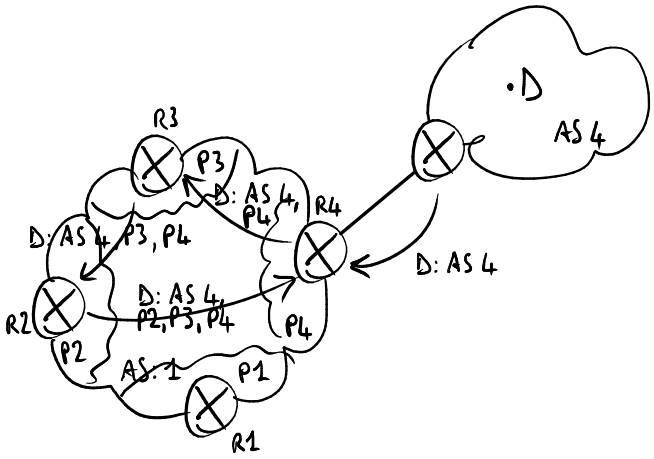


Figura 12.5: Esempio di AS confederation.

I router di frontiera hanno una maglia completa di peering session I-BGP (come nella prima modalità), ma l'AS è suddiviso in mini-AS, ognuno con un numero di AS privato, e quando un router di frontiera propaga il PV antepone nella lista il suo numero di AS privato.

Quando arriva un annuncio, il router di frontiera può guardare se nella lista è già presente il proprio numero di AS privato, in modo da scartare il pacchetto se viene rilevato un routing loop.

12.3 Attributi di percorso

Le informazioni BGP sulle rotte annunciate (ad es. la lista degli AS attraversati) sono contenute in **attributi di percorso** all'interno dei pacchetti di UPDATE.

Tutti gli attributi sono codificati nel **formato Type-Length-Value** (TLV) ⇒ il BGP è un **protocollo estensibile**: gli RFC di estensione possono definire dei nuovi attributi senza rompere la compatibilità con il mondo esistente e, se il router non supporta quell'attributo (codice di tipo non riconosciuto), può ignorarlo e saltare al successivo (grazie all'informazione sulla lunghezza).

Un attributo BGP può essere:

- **well-known**: deve essere compreso da tutte le implementazioni, e non può mai essere saltato (sezione 12.3.1):
  - mandatory: deve essere presente in tutti i messaggi;
  - discretionary: può non essere presente in tutti i messaggi;
- **optional**: può non essere compreso da tutte le implementazioni, e può essere saltato se non supportato (sezione 12.3.2):
  - transitive: se il router non supporta l'attributo, deve propagarlo comunque impostando il flag P;
  - non-transitive: se il router non supporta l'attributo, non deve propagarlo.

Ogni attributo ha il seguente formato TLV:

1	2	3	4	8	16	24/32	
O	T	P	E	0	Type	Length	Value

Tabella 12.1: Formato Type-Length-Value (TLV) di un attributo BGP.

dove i campi sono:

- flag Optional (O) (1 bit): specifica se l'attributo è optional o well-known;
- flag Transitive (T) (1 bit): specifica se l'attributo è transitive o non-transitive;
- flag Partial (P) (1 bit): specifica se almeno un router lungo il percorso ha incontrato un attributo optional transitive che non supportava;
- flag Extended Length (E) (1 bit): specifica se il campo "Length" è codificato con uno o due byte;
- campo Type (1 byte): contiene il codice di tipo che identifica l'attributo ⇒ un router può determinare se supporta quell'attributo senza dover analizzare il suo valore;
- campo Length (1 o 2 byte): contiene la lunghezza del valore dell'attributo ⇒ un router può saltare un attributo non supportato e passare al successivo avanzando del numero di byte indicato da questo campo;
- campo Value (lunghezza variabile): contiene il valore dell'attributo.

12.3.1 Attributi well-known

- attributo **ORIGIN** (tipo 1, mandatory): definisce l'origine dell'informazione di percorso:
  - IGP: la rotta è stata specificata manualmente come una rotta statica (comando `bgp network`);

  - EGP: la rotta è stata appresa dal protocollo EGP[5];
  - INCOMPLETE: la rotta è stata appresa da un protocollo IGP tramite un processo di ridistribuzione (comando `bgp redistribute`);

- attributo **AS_PATH** (tipo 2, mandatory): contiene la lista degli AS attraversati suddivisa in segmenti di percorso:
  - AS_ SEQUENCE: i numeri di AS nel segmento di percorso sono in ordine di attraversamento, e se il primo segmento nel pacchetto è in ordine un nuovo numero di AS va aggiunto all'inizio di quel segmento;
  - AS_ SET: i numeri di AS nel segmento di percorso non sono in ordine di attraversamento, e se il primo segmento nel pacchetto non è in ordine va aggiunto prima di quel segmento un nuovo segmento in ordine in cui va inserito il nuovo numero di AS;
- attributo **NEXT_HOP** (tipo 3, mandatory): ottimizza l'instradamento quando più router appartengono alla stessa LAN ma a due AS diversi, e quindi il traffico da un AS all'altro passerebbe sempre dal router di frontiera ⇒ il router di frontiera può annunciare di mandare il traffico direttamente al router next hop nell'altro AS:

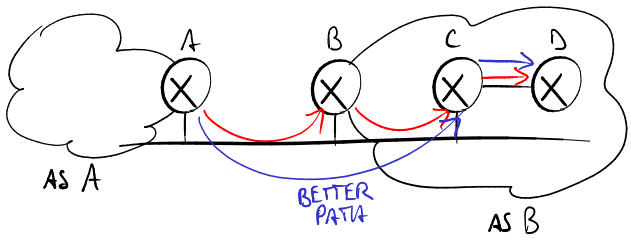


Figura 12.6: Il router di frontiera B insegna al router di frontiera A di usare il router C come next hop per la destinazione D.

- attributo **LOCAL_PREF** (tipo 5, discretionary): nell'I-BGP quando la destinazione esterna è raggiungibile tramite due router di frontiera egress, viene preferita la rotta con LOCAL_PREF più alto;
- attributo **ATOMIC_AGGREGATE** (tipo 6, discretionary): indica che la rotta annunciata è una rotta aggregata not precise.

12.3.2 Attributi optional

- attributo **MULTI_EXIT_DISC** (MED) (tipo 4, non-transitive): nell'E-BGP quando due AS sono connessi tramite più link, viene preferito il link con MED più basso e i link con MED più alti sono considerati link di backup;
- attributo **AGGREGATOR** (tipo 7, transitive): contiene il numero di AS e l'indirizzo IP del router che ha generato la rotta not precise;
- attributo **COMMUNITIES** (tipo 8, transitive): indica a quale gruppo di peer questa rotta deve essere annunciata (ad es. all'intera Internet, solo nell'AS corrente, a nessuno);
- attributo **MP_UNREACH_NLRI** (tipo 15, non-transitive): informa che una rotta precedentemente annunciata è diventata non raggiungibile (le rotte non scadono mai).

[5]Qui s'intende il protocollo, non la classe di protocolli (si veda la sezione 6.2.1).

12.4 Processo di decisione

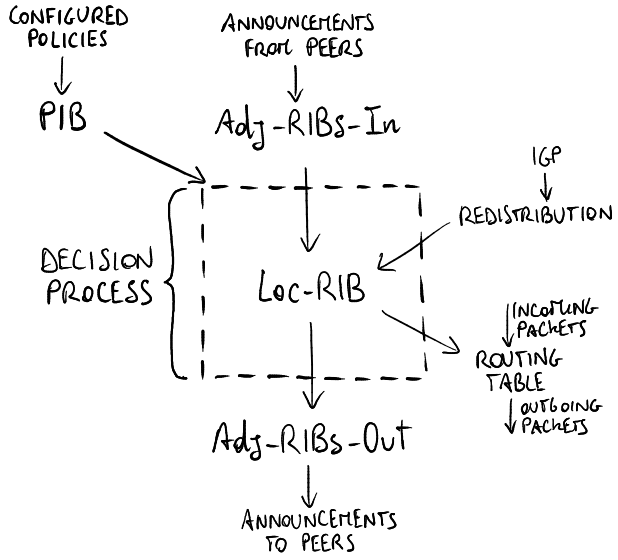


Figura 12.7: Il processo di decisione seleziona le rotte dalle Adj-RIB-In in input e le scrive nella Loc-RIB e nelle Adj-RIB-Out in output.

Il **processo di decisione** in esecuzione su ogni router di frontiera è responsabile di:

- selezionare quali rotte sono annunciate agli altri peer BGP;
- selezionare quali rotte sono usate localmente dal router di frontiera;
- aggregare le rotte per ridurre le informazioni.

Le basi di dati con cui il BGP ha a che fare sono:

- **Routing Information Base** (RIB): consiste di tre parti distinte:
  - **Adjacent RIB Incoming** (Adj-RIB-In): contiene tutte le rotte apprese dagli annunci ricevuti da un certo peer;
  - **Local RIB** (Loc-RIB): contiene le rotte selezionate dal processo di decisione con il loro grado di preferenza;
  - **Adjacent RIB Outgoing** (Adj-RIB-Out): contiene le rotte che verranno propagate negli annunci a un certo peer;
- **Policy Information Base** (PIB): contiene le politiche di instradamento definite per configurazione manuale;
- **tabella di instradamento**: contiene le rotte utilizzate dal processo di inoltro dei pacchetti.

Possono essere imposte delle politiche di instradamento molto complesse per influenzare il processo di decisione:

1. viene applicata a ogni rotta nelle Adj-RIB-In una certa funzione che, applicando le politiche definite sugli attributi, restituisce il **grado di preferenza** per quella rotta.
   Le politiche sono definite solo in base agli attributi della rotta corrente: il calcolo del grado di preferenza non è mai influenzato dall'esistenza, dalla non esistenza o dagli attributi di altre rotte;

2. per ogni destinazione, la rotta con il grado di preferenza maggiore viene selezionata e viene inserita nella Loc-RIB;

3. altre politiche determinano quali rotte vengono selezionate dalla Loc-RIB per essere inserite nelle Adj-RIB-Out.

Capitolo 13

Instradamento IPv6

Oggi i router sono per lo più pronti per IPv6, anche se le prestazioni in IPv6 sono ancora peggiori rispetto a quelle in IPv4 a causa della mancanza di esperienza e della più bassa domanda di traffico. Spesso l'instradamento IPv6 è disabilitato per impostazione predefinita anche se l'apparato supporta IPv6 (sui router Cisco si attiva con il comando `ipv6 unicast-routing`).

Due aspetti sono da considerare:

- tabelle di instradamento: come gestire l'inoltro dei pacchetti di dati? (sez. 13.1)
- protocolli di instradamento: come distribuire le rotte nella rete? (sez. 13.2)

13.1 Tabelle di instradamento

L'instradamento in IPv6 è effettuato nello stesso modo di IPv4 ma richiede due tabelle di instradamento distinte, una per le rotte IPv4 e l'altra per le rotte IPv6. Le tabelle di instradamento IPv6 possono memorizzare diversi tipi di entry, tra cui:

- entry indirette (codici O/S): specificano gli indirizzi, tipicamente link local, delle interfacce dei router next hop ai quali inviare i pacchetti indirizzati verso link remoti;
- entry dirette: specificano le interfacce del router stesso attraverso le quali inviare i pacchetti indirizzati verso i link locali:
  - reti connesse (codice C): specificano i prefissi dei link locali;
  - indirizzi di interfaccia (codice L): specificano gli interface identifier nei link locali.

13.1.1 Next hop

Come next hop nelle **rotte dinamiche** calcolate, i protocolli di instradamento utilizzano sempre gli indirizzi link local, anche se è configurato un indirizzo global sull'interfaccia vicina, per motivi di semplicità: gli indirizzi link local esistono sempre, mentre gli indirizzi global potrebbero non essere utilizzati in alcune porzioni della rete.

Tuttavia l'uso degli indirizzi link local rende difficile il compito di determinare la posizione di quell'indirizzo: l'indirizzo di rete di un indirizzo global permette almeno di identificare la rete in cui l'host dovrebbe essere presente e determinare così l'interfaccia di uscita, ma un indirizzo link local che inizia con FE80:: può essere ovunque ⇒ accanto all'indirizzo del next hop, i router stampano anche l'**interfaccia locale** di uscita per risolvere le ambiguità, come:

```
2001:1::/64 via FE80::1, FastEthernet0/0
```

Per le **rotte statiche**, la scelta è lasciata al gestore della rete, che può utilizzare l'indirizzo che preferisce come next hop:

```
ipv6 route indirizzo/netmask [interfaccia locale] [next hop] [distanza]
```

- interfaccia broadcast (ad es. Ethernet): è necessario specificare l'indirizzo del next hop:
  - indirizzo global: non è necessario specificare l'interfaccia locale perché può essere determinata dal prefisso di rete:
    ```
    ipv6 route 2001:1::/64 2001::1
    ```
  - indirizzo link local: è necessario specificare anche l'interfaccia locale per identificare l'ambito dell'indirizzo link local:
    ```
    ipv6 route 2001:1::/64 FastEthernet0/0 fe80::1
    ```
- interfaccia punto-punto (ad es. seriale): non è necessario specificare l'indirizzo del next hop perché è identificato univocamente dall'interfaccia locale:
  ```
  ipv6 route 2001:1::/64 serial 0
  ```

Siccome le rotte statiche non possono adattarsi ai cambiamenti della rete, è fortemente suggerito di utilizzare gli indirizzi global come next hop per le rotte statiche. Ciò evita che una rotta diventi non valida se il next hop cambia: ad esempio, se la scheda di rete sul router next hop viene sostituita a causa di un guasto hardware:

- indirizzo link local: dipende dall'indirizzo MAC della scheda ⇒ occorre modificare la rotta;
- indirizzo global: è sufficiente assegnare alla nuova interfaccia lo stesso indirizzo global.

13.2 Protocolli di instradamento

I protocolli di instradamento che supportano IPv6 possono adottare due approcci:

- **instradamento integrato** (IS-IS, MP-BGP4): il protocollo consente di scambiare informazioni di instradamento sia IPv4 sia IPv6 allo stesso tempo:
  - \+ efficienza: gli indirizzi IPv4 e IPv6 appartenenti alla stessa destinazione possono essere trasportati tramite un singolo messaggio;
  - − flessibilità: un singolo protocollo trasporta più famiglie di indirizzi;
  - \+ reattività: se avviene un guasto o un cambiamento nella rete, il protocollo lo scopre per entrambe le famiglie di indirizzi;
  - − bug: un problema nel protocollo influenza reti IPv4 e IPv6 allo stesso modo;
  - − migrazione: se il protocollo usa IPv4 per trasportare i pacchetti di Hello, non è possibile abolire IPv4 nella rete;
- **navi nella notte** (RIPng, EIGRP, OSPFv3): il protocollo consente di scambiare informazioni di instradamento solo IPv6:
  - − efficienza: data una destinazione, deve essere scambiato un messaggio per il suo indirizzo IPv4 e un altro messaggio per il suo indirizzo IPv6, e i messaggi sono completamente indipendenti tra loro;
  - \+ flessibilità: si possono usare due protocolli diversi, uno per le informazioni di instradamento IPv4 e un altro per le informazioni di instradamento IPv6;
  - − reattività: se avviene un guasto o un cambiamento nella rete, entrambi i protocolli devono scoprirlo, ciascuno con le proprie tempistiche e con messaggi duplicati;
  - \+ bug: un problema nel protocollo non influenza l'instradamento nell'altro;
  - \+ migrazione: ogni protocollo di instradamento genera messaggi della famiglia di indirizzi a cui appartiene.

13.2.1 RIPng

RIPng adotta l'approccio "navi nella notte", e apporta a RIP dei miglioramenti principalmente nell'interfaccia a riga di comando di Cisco:

- supporto alle istanze multiple: il campo `tag` permette di specificare l'istanza di protocollo;[1]
- configurazione per interfaccia: sono stati introdotti nuovi comandi:
  - `ipv6 rip <tag> enable`: sostituisce il comando `network` e configura automaticamente RIP su quell'interfaccia senza dover specificare un indirizzo;
  - `ipv6 rip <tag> default-information originate`: origina la rotta di default (::/0), cioè il resto del mondo è raggiungibile tramite questa interfaccia.

13.2.2 OSPFv3

OSPFv3 adotta l'approccio "navi nella notte", e differisce da OSPF principalmente per tre aspetti:

- configurazione per interfaccia: è stato introdotto il comando `ipv6 ospf <ID processo> area <ID area>` che specifica che tutte le reti e gli indirizzi configurati su questa interfaccia verranno annunciati come appartenenti all'area specificata;
- Router ID: sfortunatamente OSPFv3 usa ancora un Router ID da 32 bit, che non è neanche capace di impostare automaticamente quando nessun indirizzo IPv4 è disponibile ⇒ il comando `ipv6 router-id <numero>` diventa obbligatorio quando il router è solo IPv6 o è in una rete solo IPv6;
- tunnel: è possibile configurare un tunnel IPv6 su IPv4 per collegare tra loro isole IPv6 attraverso una rete IPv4.

13.2.3 IS-IS

IS-IS per IPv6 adotta l'approccio "instradamento integrato": usa infatti un proprio protocollo di livello 3 per trasportare i pacchetti specifici del protocollo, indipendentemente dalla versione del protocollo IP sottostante.

13.2.4 MP-BGP4

MP-BGP4 può adottare entrambi gli approcci a seconda della configurazione: i pacchetti TCP possono essere imbustati in IPv4 o in IPv6 a seconda delle necessità.

La distribuzione più comune segue l'approccio "instradamento integrato", per la necessità di utilizzare il numero dell'AS (che è uguale sia per IPv4 sia per IPv6) per il processo BGP. L'integrazione si riflette anche nelle politiche: è possibile mischiare indirizzi IPv4 e indirizzi IPv6 a piacimento.

[1] Il supporto per le istanze multiple era già presente in RIPv2, ma non era configurabile sui router Cisco.

Capitolo 14

Instradamento multicast

Protocolli di instradamento multicast

- DVMRP: DV (TRPB, RPM), indipendente dal protocollo unicast, albero specifico della sorgente (sez. 14.1);
- MOSPF: LS, opera sul protocollo OSPF, albero specifico della sorgente (sez. 14.2);
- PIM-DM: DV (RPM), indipendente dal protocollo unicast, albero specifico della sorgente;
- PIM-SM: CBT, indipendente dal protocollo unicast, ibrido tra albero condiviso e specifico della sorgente (sez. 14.3.1);
- BGMP (Border Gateway Multicast Protocol): protocollo di instradamento multicast inter-dominio basato sul BGP.

14.1 DVMRP

Il **Distance Vector Multicast Routing Protocol** (DVMRP) è stato il primo protocollo di instradamento multicast, ed era usato alle origini in **Multicast backbone** (Mbone), una rete virtuale nata nel 1992 in ambito IETF che si appoggia per la trasmissione sulla stessa struttura fisica di Internet per fornire agli utenti la possibilità di sfruttare il multicast per le comunicazioni multimediali.

DVMRP è basato sull'algoritmo DV:

- versione 1: adotta il TRPB[1];
- versione 3: adotta il RPM[2].

L'istanza di protocollo DVMRP è eseguita in parallelo all'istanza di protocollo unicast: DVMRP ignora le informazioni di instradamento degli altri protocolli, e calcola rotte che possono differire da quelle utilizzate per il traffico unicast.

Usa una metrica basata sul numero di hop, vale a dire il numero di mrouter che parlano DVMRP. In DVMRP possono essere configurati manualmente dei **tunnel**: il cammino che collega due vicini DVMRP può includere dei router che non supportano DVMRP (o su cui DVMRP è disabilitato):

- local end-point: specifica l'mrouter all'inizio del tunnel;
- remote end-point: specifica l'mrouter all'altra estremità del tunnel;
- metric: specifica la misura del costo del tunnel;

[1] Si veda la sezione 7.1.3.
[2] Si veda la sezione 7.1.4.

- threshold: specifica il valore minimo del TTL che un pacchetto deve avere per poter essere instradato attraverso il tunnel ⇒ permette di definire la visibilità dei pacchetti: i pacchetti che non devono uscire dalla rete aziendale vengono generati con un TTL pari alla threshold.

14.2 MOSPF

Il **Multicast OSPF** (MOSPF) è un protocollo di instradamento multicast basato sull'algoritmo LS[3] che estende OSPF, permettendo ai singoli router di avere una conoscenza completa della topologia della rete e dei costi relativi ai singoli link.

MOSPF è retrocompatibile con OSPF: i router MOSPF possono essere mischiati con router solo OSPF, anche se i pacchetti multicast sono inoltrati solo tra router MOSPF e i cammini scelti per i pacchetti multicast non passano per i router solo OSPF.

MOSPF aggiunge l'**LSA di tipo 6**:

- i router interni MOSPF producono LSA di tipo 6 per informare gli altri router nella loro area dei gruppi multicast attivi sulle loro reti;
- gli ABR MOSPF producono LSA di tipo 6 per informare gli altri router nell'area 0 dei gruppi multicast attivi sulle loro aree edge (anche con l'aggregazione).

14.3 PIM

Il **Protocol Independent Multicast** (PIM) si serve direttamente delle tabelle contenenti le informazioni di instradamento indipendentemente dal protocollo unicast sottostante (DV o LS) che le ha costruite.

Esiste in due versioni, tra loro incompatibili, che affrontano diverse problematiche relative alla spazialità dei ricevitori:

- **Dense Mode** (DM):
  - è adatto per reti piccole (LAN/MAN) con molti ricevitori concentrati;
  - non è parco in termini di banda: i pacchetti possono venire inoltrati in zone non interessate al particolare gruppo multicast;
  - adotta l'algoritmo RPM basato su DV (come DVMRPv3);
  - Implicit Join Protocol: in mancanza di espliciti messaggi di Prune, i pacchetti vengono inoltrati in una certa rete;
  - è necessario generare messaggi di Prune per interrompere il traffico multicast (come DVMRP);
- **Sparse Mode** (SM):
  - è adatto per reti estese (WAN) con pochi ricevitori sparsi;
  - limita il più possibile l'overhead di banda: non usa mai il flooding;
  - è un'evoluzione dell'algoritmo CBT[4]: inizia sempre da un albero condiviso, che diventa un albero specifico della sorgente quando vantaggioso;
  - Explicit Join Protocol: in mancanza di espliciti messaggi di Join, i pacchetti non vengono inoltrati in una certa rete (le informazioni di instradamento però vanno a tutti i router, non sono quelli che riceveranno traffico);
  - è necessario generare messaggi di Join per attivare il traffico multicast (come MOSPF).

[3] Si veda la sezione 7.2.
[4] Si veda la sezione 7.3.

14.3.1 PIM-SM

PIM-SM gestisce due tipi di alberi di distribuzione:

- **RP-Tree** (RPT): è l'albero condiviso usato inizialmente per tutti i pacchetti del gruppo multicast, ma non è ottimizzato in base alla sorgente ⇒ i primi pacchetti della trasmissione possono essere consegnati in breve tempo ai ricevitori senza dover aspettare il calcolo dell'albero:
  - non vengono usati i percorsi minimi;
  - il traffico è centralizzato;
  - ad ogni gruppo multicast corrisponde un albero;
- **Shortest-Path Tree** (SPT): è l'albero specifico della sorgente costruito in un secondo momento se i router lo ritengono conveniente (non è obbligatorio):
  - vengono usati i percorsi minimi;
  - il traffico è distribuito;
  - ad ogni coppia (gruppo, sorgente) corrisponde un albero.

PIM-SM definisce tre nodi speciali:

- **rendez-vous point** (RP): è il core router, eletto in base a una formula algoritmica a partire da un elenco chiamato RP-set, che si occupa di:
  - ricevere i messaggi di Join da parte dei DR;
  - ricevere le richieste di registrazione da parte delle sorgenti;
  - ricevere e inviare ai DR i primi pacchetti di dati multicast trasmessi da una sorgente;
- **designated router** (DR): è il router, eletto in base al percorso più breve verso il RP (o all'indirizzo IP più alto in caso di parità), che si occupa di:
  - ricevere le richieste di iscrizione da parte degli host in una certa LAN;
  - inviare il messaggio di Join al RP per unirsi al RPT;
  - inviare il messaggio di Join alla sorgente per unirsi al SPT;
  - inviare il messaggio di Join periodicamente per adattarsi ai cambiamenti dei gruppi;
- **bootstrap router** (BSR): è il router, eletto in base al costo amministrativo migliore (o all'indirizzo IP più alto in caso di parità), che si occupa di distribuire l'RP-set a tutto il dominio PIM-SM.

Algoritmo

RPT

1. iscrizione dei DR come ricevitori: ogni DR invia un messaggio di Join al RP;
2. registrazione della sorgente come trasmettitore: la sorgente invia una richiesta di registrazione al RP;
3. registrazione del RP come ricevitore: il RP invia un messaggio di Join alla sorgente;
4. trasmissione lungo il RPT: la sorgente invia in multicast i primi pacchetti di dati, inoltrati dai router intermedi fino al RP;
5. propagazione lungo il RPT: il RP propaga in multicast i pacchetti di dati ricevuti, inoltrati dai router intermedi fino ai DR.

SPT

1. iscrizione dei DR come ricevitori: ogni DR invia un messaggio di Join alla sorgente;
2. trasmissione lungo il SPT: la sorgente invia in multicast i pacchetti di dati, inoltrati dai router intermedi a ogni DR (oltre che al RP);
3. distacco dal RPT: ogni DR invia un messaggio di Prune al RP;
4. unione al SPT: un altro DR invia un messaggio di Join al RP, poi invia un messaggio di Join alla sorgente.

Capitolo 15

Content Delivery Network

Una **cache Web** è un dispositivo che archivia una copia locale dei contenuti (ad es. risorse HTTP) richiesti più di recente e reagisce come un server proxy alle richieste dei client:

- la cache Web è più vicina all'utente rispetto al server Web:
  - \+ prestazioni: la risposta è più veloce quando la risorsa richiesta si trova già in cache;
  - \+ banda: non vengono caricati costosi link a lunga distanza (ad es. link transoceanici);
- − soluzione reattiva: se la risorsa richiesta non si trova in cache, l'utente deve attendere che la cache Web la acquisisca (**pull**) dal server Web;
- − no trasparenza: il browser Web dell'utente deve essere configurato manualmente per contattare quella cache Web.

Una **Content Delivery Network** (CDN) è una overlay network[1] di cache Web sparse in giro per il mondo ma cooperanti con l'obiettivo di offrire all'utente una migliore quality of experience[2]:

- soluzione proattiva: il server Web copia (**push**) i contenuti (generalmente i più popolari) sulle cache Web prima che gli utenti li richiedano;
- trasparenza: l'utente si collega alla cache Web automaticamente, senza necessità di configurazione manuale sul suo client;
- prestazioni: l'utente, anche se si sposta, si collega sempre alla cache Web più vicina;
- bilanciamento del carico: l'utente si collega sempre alla cache Web meno carica;
- scalabilità: la distribuzione dei contenuti in più repliche permette un grande numero di richieste che un singolo server Web da solo non riuscirebbe a servire;
- accesso condizionato: è possibile personalizzare i contenuti restituiti in base all'utente (ad es. annunci pubblicitari mirati).

Le CDN sono l'ideale per contenuti che generano grandi quantità di traffico (ad es. risorse multimediali), ma non tutti i contenuti possono essere memorizzati in cache:

- pagine Web dinamiche (ad es. quotazioni di Borsa);
- pagine Web dal contenuto personalizzato (ad es. account utente).

Le CDN possono essere distribuite in vari modi:

[1] Una **overlay network** è una rete di calcolatori che è costruita al di sopra di un'altra rete.

[2] La **quality of experience** è una misura delle esperienze di un utente con un servizio (ad es. navigazione Web).

- **CDN basate sui DNS**: il traffico è reindirizzato alla replica migliore in base agli host name:
  - instradamento basato sui DNS (sez. 15.1.1): l'hosting provider deve stipulare accordi con i gestori dei server DNS;
  - approccio di Akamai (sez. 15.1.2): non è necessario intervenire sui server DNS;
- **CDN basate sugli URL**: il traffico è reindirizzato alla replica migliore in base agli URL completi:
  - server load balancing (sez. 15.2.1): il punto di terminazione della connessione TCP è vicino al server;
  - content routing (sez. 15.2.2): il punto di terminazione della connessione TCP è vicino al client.

15.1 CDN basate sui DNS

15.1.1 Instradamento basato sui DNS

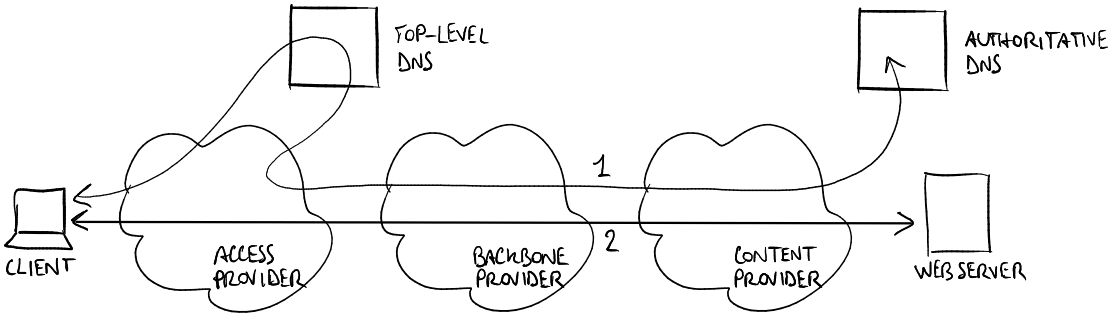


Figura 15.1: Navigazione tradizionale.

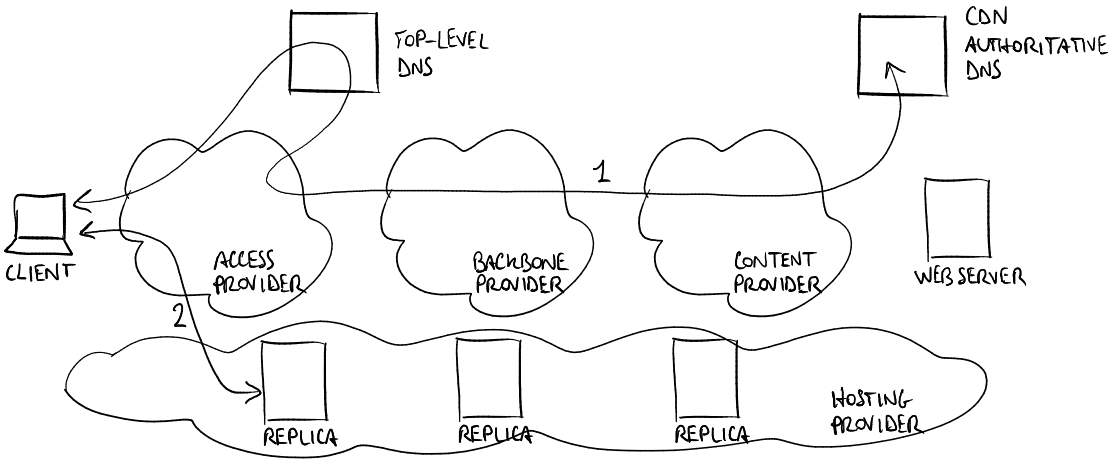


Figura 15.2: Navigazione CDN basata sui DNS.

La selezione della replica migliore ha luogo quando l'host name viene tradotto in un indirizzo IP. La risposta DNS a una query non dipende solo dall'host name, ma anche dalla sorgente: uno speciale server DNS calcola, in base a più metriche possibili (RTT, carico dei server, tempo di

risposta, ecc.), una tabella di instradamento delle repliche contenente entry del tipo:

$$\{\text{host name, indirizzo IP client}\} \rightarrow \text{indirizzo IP replica}$$

Il motore di instradamento nel server DNS "modificato" ha un'interfaccia standard per garantire la trasparenza: l'utente crede che l'indirizzo IP corrispondente all'host name sia l'indirizzo IP del server Web reale, mentre è l'indirizzo IP di una sua replica.

L'aggiunta di un nuovo attore, l'**hosting provider**, costituisce una nuova opportunità di business nel mondo delle reti:

- access provider: fornisce l'accesso alla rete agli utenti;
- backbone provider: fornisce la connettività a lungo raggio;
- hosting provider: fornisce il servizio di CDN ai content provider;
- content provider: fornisce i contenuti.

Problemi

- metriche: la misurazione delle metriche, soprattutto di quelle dinamiche, non è facile, e le metriche di livello 3 da sole non sono particolarmente significative;
- caching DNS: solo il server autoritativo conosce tutte le repliche e può selezionare la replica migliore in base alla posizione del client ⇒ i server DNS intermedi nella gerarchia non possono memorizzare in cache le risposte DNS;
- granularità: la granularità della redirezione è a livello di host name, non di singoli URL ⇒ il contenuto di grandi siti Web non può essere suddiviso in più cache, quindi due diverse pagine dello stesso sito Web saranno chieste alla stessa replica.

15.1.2 Approccio di Akamai

La CDN di Akamai sfrutta un algoritmo automatico proprietario per reindirizzare il traffico alle proprie repliche senza intervenire sui server DNS:

1. l'utente digita l'indirizzo di una pagina Web con il suo normale nome di dominio (ad es. `http://cnn.com/index.html`);
2. il server del content provider (ad es. CNN) restituisce una pagina Web in cui l'indirizzo di ogni risorsa multimediale (ad es. immagine) ha uno speciale nome di dominio corrispondente a una specifica replica su una cache di Akamai (ad es. `http://a128.g.akamai.net/7/23/cnn.com/a.gif` invece di `http://cnn.com/a.gif`);
3. il browser Web dell'utente durante il parsing della pagina effettua delle query DNS ai nuovi nomi di dominio e recupera le risorse multimediali dalle repliche più vicine.

15.2 CDN basate sugli URL

15.2.1 Server load balancing

I server reali contenenti le repliche sono visti dai client come un unico server virtuale con lo stesso indirizzo IP.

Il carico di traffico destinato al server virtuale è bilanciato tra i diversi server reali da un **Server Load Balancer** (SLB):

- commutazione a livello 4: le connessioni TCP non sono terminate dal SLB (**content-unaware**):
  - uno dei server reali risponde al three-way handshake con il client;

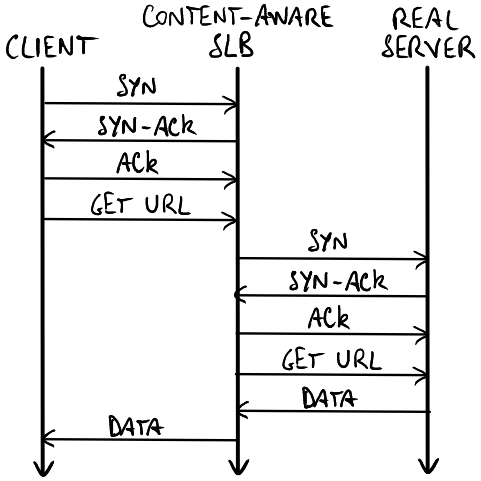


Figura 15.3: SLB content-aware.

  - tutte le query HTTP appartenenti alla stessa sessione TCP devono essere servite sempre dallo stesso server reale;
  - il bilanciamento del carico può essere basato sull'indirizzo IP sorgente, sulla porta TCP sorgente, ecc.;

- commutazione a livello 7: le connessioni TCP sono terminate dal SLB (**content-aware**), che agisce come un proxy:
  - il SLB risponde al three-way handshake con il client, per poter intercettare gli URL richiesti successivamente;
  - ogni query HTTP può essere servita dal server reale correntemente meno carico, in base alle decisioni del SLB;
  - il bilanciamento del carico è basato sull'URL completo.

Problemi

- connessioni cifrate (HTTPS): il SLB deve avere la chiave crittografica SSL privata del server, e deve sopportare il carico di elaborazione per criptare/decriptare i pacchetti in transito;
- connessioni sticky: alcune applicazioni richiedono che le connessioni TCP dallo stesso client siano reindirizzate allo stesso server (ad es. carrello della spesa) ⇒ occorre considerare anche i cookie;
- distribuzione geografica: tutte le repliche sono vicine tra loro e al SLB, che è lontano dal client.

15.2.2 Content routing

I **content router** sono dei router che instradano il traffico in base all'URL verso la replica migliore:

- TCP: i content router in sequenza terminino tutti delle connessioni TCP tra uno e l'altro ⇒ vengono introdotti troppo ritardi;
- content delivery control protocol: l'URL è estratto dal primo content router, ed è propagato da un protocollo specifico.

Problemi

- stateful: il primo content router deve terminare la connessione TCP per poter intercettare l'URL che l'utente richiederà;
- complessità degli apparati: il parsing dei pacchetti per recuperare l'URL è complicato ⇒ gli switch di livello 7 sono apparati complicati e costosi;
- complessità dei protocolli: i content delivery control protocol proposti sono veramente complicati;
- riservatezza: i content router leggono tutti gli URL richiesti dagli utenti.

Parte III

Elaborazione di rete

Capitolo 16

Cenni sull'architettura degli apparati di rete

L'architettura dei router si è evoluta nel corso del tempo per aumentarne sempre di più la capacità di elaborazione dei pacchetti:

- prima generazione (fino ad inizio 1990): inferiore a 500 Mbps (sez. 16.1);
- seconda generazione (inizio 1990): circa 5 Gbps (sez. 16.2);
- terza generazione (fine 1990): a partire da 50 Gbps (sez. 16.3);
- multi-chassis (tendenza recente): dell'ordine dei Tbps (sez. 16.4).

16.1 Prima generazione

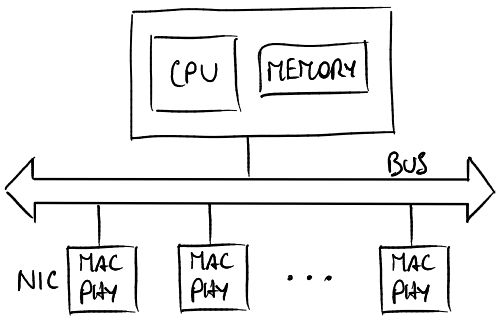


Figura 16.1: Architettura dei router di prima generazione.

I router di prima generazione erano sostanzialmente dei PC modificati:

- interfacce di rete: sono delle normali NIC, in numero maggiore e di molteplici tipologie rispetto a quelle di un normale PC;
- memoria: si preferisce la memoria allo stato solido (ad es. scheda SD) perché meno soggetta a guasti rispetto a un disco fisso meccanico;
- sistema operativo: è ottimizzato specificamente per l'elaborazione del traffico di rete.

Vantaggio economia di scala: usa componenti fabbricati a larga scala anziché componenti dedicati al mondo del networking.

Svantaggi

- colli di bottiglia:
  - bus condiviso:
    - slow path: ogni pacchetto transita due volte sul bus;
    - arbitraggio: una NIC per volta può usare il bus ⇒ non possono lavorare più interfacce in parallelo;
  - accesso a memoria: la tabella di instradamento è memorizzata in una struttura dati nella memoria generica ⇒ gli accessi non sono così localizzati come nei PC tradizionali, e la cache è poco usata;
  - capacità di elaborazione: la CPU lancia un interrupt al sistema operativo ogni volta che arriva un pacchetto;
- interfacce di rete: le schede di rete tradizionali tipicamente non hanno molte porte e supportano solo i tipi di interfacce più comuni;
- costi di manutenzione: l'uso di componenti open-source (ad es. client PPP) porta problemi di integrazione e variegate modalità di configurazione.

Attualmente questa architettura è usata per i router di fascia medio-bassa, le cui prestazioni sono adeguate per i piccoli uffici.

16.2 Seconda generazione

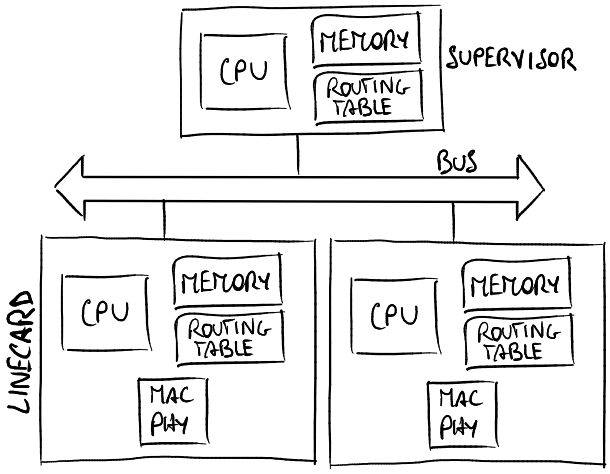


Figura 16.2: Architettura dei router di seconda generazione.

I router di seconda generazione cercano di risolvere il problema delle prestazioni spostando il carico di elaborazione dalla CPU centrale alle unità periferiche: le NIC sono rimpiazzate dalle **line card**, schede di rete "intelligenti" che aggiungono ai componenti fisici/MAC dei moduli per l'inoltro:

- CPU: il **packet processor** è un processore dedicato all'elaborazione dei pacchetti "semplici";
- memoria: ogni pacchetto in arrivo viene memorizzato in una memoria locale separata da quella (generalmente TCAM) contenente la tabella di instradamento;
- tabella di instradamento: ogni tanto la tabella di instradamento aggiornata viene copiata dall'unità centrale alle line card.

I pacchetti possono seguire due strade:

- **fast path**: i pacchetti che rientrano nel caso tipico (ad es. IPv4) transitano una volta sola sul bus: vengono elaborati direttamente dal packet processor e vengono inviati subito all'interfaccia di uscita;
- **slow path**: i pacchetti meno frequenti (ad es. IPv6, OSPF) vengono affidati alla CPU centrale per un'elaborazione più sofisticata, al costo di un throughput decisamente inferiore.

Il lavoro delle line card deve essere coordinato dalla CPU centrale, su cui inoltre girano i protocolli di instradamento (ad es. OSPF), che è montata su una scheda detta **supervisor** nei sistemi di fascia alta.

Vantaggi

- CPU centrale: lavora solo per i pacchetti lungo il fast path ⇒ può avere una potenza inferiore;
- ottimizzazione del fast path: i primi pacchetti della connessione seguono lo slow path, poi il sistema centrale segna la connessione come idonea al fast path e tutti i pacchetti successivi seguiranno il fast path;
- flessibilità: l'introduzione di un nuovo protocollo (ad es. IPv6) richiede solo l'aggiornamento del software nel sistema centrale.

Svantaggio arbitraggio del bus condiviso: una line card per volta può usare il bus ⇒ non possono lavorare più interfacce in parallelo.

16.3 Terza generazione

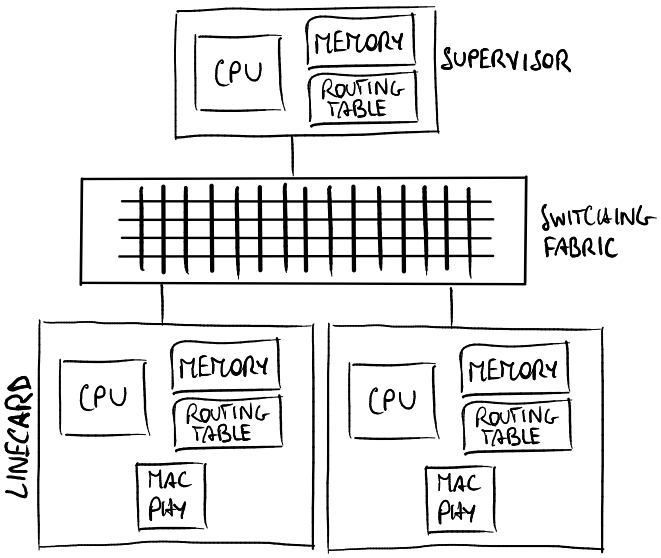


Figura 16.3: Architettura dei router di terza generazione.

I router di terza generazione si concentrano sul problema della parallelizzazione del lavoro rimpiazzando il bus condiviso con una **switching fabric** (o crossbar) in grado di gestire trasferimenti multipli: una line card è collegata a un'altra line card mettendo in cortocircuito l'interruttore all'intersezione dei relativi "fili" della matrice di commutazione.

Due line card non possono essere collegate ad un'altra stessa line card allo stesso tempo ⇒ è necessario un **arbitro** che pilota i punti di commutazione e risolve le situazioni di contesa senza collisioni.

Quando le interfacce di uscita sono lente nell'invio dei pacchetti rispetto alla velocità di commutazione della switching fabric, i pacchetti possono essere accodati in diversi modi:

- output queuing: i buffer sono posizionati sulle interfacce di uscita ⇒ caso peggiore: switching fabric N volte più veloce della velocità di ricezione, con N = numero di interfacce di ingresso;
- input queuing: i buffer sono posizionati sulle interfacce di ingresso ⇒ la velocità di commutazione della switching fabric non dipende dal numero di interfacce di ingresso, ma soffre il problema dell'**head-of-line blocking**:
  1. due pacchetti in due code di ingresso diverse sono destinati alla stessa coda di uscita;
  2. l'arbitro fa passare uno dei pacchetti bloccando l'altro;
  3. tutti i pacchetti successivi nella coda di ingresso, anche quelli destinati a code di uscita libere, devono aspettare il "primo della fila";
- virtual output queuing: risolve il problema dell'head-of-line blocking mantenendo ad ogni ingresso una coda per uscita;
- buffered fabric: i buffer sono posizionati all'interno della crossbar in corrispondenza dei nodi di commutazione.

Svantaggi

- arbitro:
  - costituisce un collo di bottiglia, soprattutto in caso di decisioni frequenti dovute a pacchetti piccoli (ad es. ATM);
  - le politiche di schedulazione per la qualità del servizio possono complicarne l'hardware;
- buffer per le code: sono memorie costose perché devono essere veloci.

16.4 Router multi-chassis

I **router multi-chassis** puntano sulla scalabilità: più rack affiancati sono collegati in modo da apparire come un router unico:

- la potenza di elaborazione è moltiplicata per il numero di chassis;
- c'è più spazio fisico per le interfacce di rete ⇒ a uno stesso apparato può essere collegato un gran numero di link;
- l'instradamento è concentrato in una singola centrale (come nel mondo telefonico) anziché in tanti piccoli router sparsi per la rete ⇒ la centrale dati può servire tutti gli utenti in un'area geografica estesa (ad es. Piemonte).

16.5 Service card

Un'altra tendenza recente punta sulla flessibilità: i router non sono più degli apparati puramente di livello 3, poiché stanno aggiungendosi molte altre funzioni che non appartengono al livello 3, come cache, firewall, monitor di rete, ACL, ecc. L'obiettivo è personalizzare il servizio offerto dagli ISP, facendo ritornare una parte del business attualmente in mano ai content provider.

Una soluzione oggi comune sono le **service card**, line card arricchite con servizi a valore aggiunto preconfigurati dal costruttore. Le service card non sono però così flessibili: non essendo programmabili, per aggiungere un servizio occorre acquistare una nuova service card.

16.6 Maggiori problematiche attuali

Il throughput ormai non è più una problematica attuale, ma le maggiori problematiche attuali sono:

- costi di acquisto: per ridurre il costo degli apparati di rete si sta passando da componenti dedicati a componenti general-purpose prodotti su larga scala:
  - le CPU di massa hanno cicli di vita del prodotto troppo brevi e tempi di ricambio troppo lunghi ⇒ Intel ha proposto di recente delle CPU embedded con cicli di vita del prodotto più lunghi e tempi di ricambio più brevi rispetto alle CPU di massa;
  - nei sistemi dedicati il programmatore poteva sfruttare la gerarchia di memorie, memorizzando la RIB in una memoria lenta e la FIB in una memoria veloce (ad es. TCAM), mentre i sistemi general-purpose decidono autonomamente che cosa mettere nella cache ⇒ occorre rivolgersi alle GPU, dotate di memorie partizionate ma di versatilità di elaborazione limitata;
- costi operativi: il consumo di elettricità e la dissipazione termica possono essere significativi;
- flessibilità: bisogna muoversi verso la separazione delle funzioni di controllo dall'hardware fisico (si rimanda al capitolo 18).

Capitolo 17

Filtraggio dei pacchetti basato su software

Il software in grado di analizzare i campi nei pacchetti trova spazio, eseguito su circuiti integrati o microprocessori, in varie applicazioni:

- switch: gli algoritmi di apprendimento si basano sugli indirizzi MAC sorgente e di destinazione delle trame[1], e l'inoltro delle trame si basa sugli indirizzi MAC di destinazione;
- router: l'inoltro dei pacchetti si basa sugli indirizzi IP sorgente e di destinazione;
- firewall: se una regola basata sui campi del pacchetto è soddisfatta, lancia l'azione associata di filtraggio (ad es. scarta);
- NAT: converte gli indirizzi IP tra privati e pubblici e le porte TCP/UDP di ogni pacchetto in transito;[2]
- URL filter: blocca il traffico HTTP da/verso URL di siti Web in una black list;
- pila protocollare: il sistema operativo consegna il pacchetto alla pila di livello rete appropriata (ad es. IPv4 o IPv6), poi il pacchetto passa alla pila di livello trasporto appropriata (ad es. TCP o UDP), infine in base alla quintupla che identifica la sessione il pacchetto viene reso disponibile all'applicazione attraverso il socket giusto;
- cattura dei pacchetti: le applicazioni per la cattura del traffico (ad es. Wireshark, tcpdump) possono impostare un filtro per ridurre la quantità di pacchetti catturati.

17.1 Architettura tipica di un sistema di filtraggio dei pacchetti

Componenti di livello kernel

- **network tap**: intercetta i pacchetti dalla scheda di rete e li consegna a una o più[3] pile di filtraggio;
- **packet filter**: lascia passare solo i pacchetti che soddisfano il filtro specificato dall'applicazione di cattura, aumentando l'efficienza della cattura: i pacchetti non voluti vengono scartati subito, e nel kernel buffer viene copiato un minor numero di pacchetti;

[1] Si rimanda alla sezione *Transparent bridge* nel capitolo *Repeater e bridge* negli appunti di "Progetto di reti locali".
[2] Si rimanda al capitolo *NAT* negli appunti di "Reti di calcolatori".
[3] Ogni applicazione di cattura ha la sua pila di filtraggio, ma tutte condividono la stessa network tap.

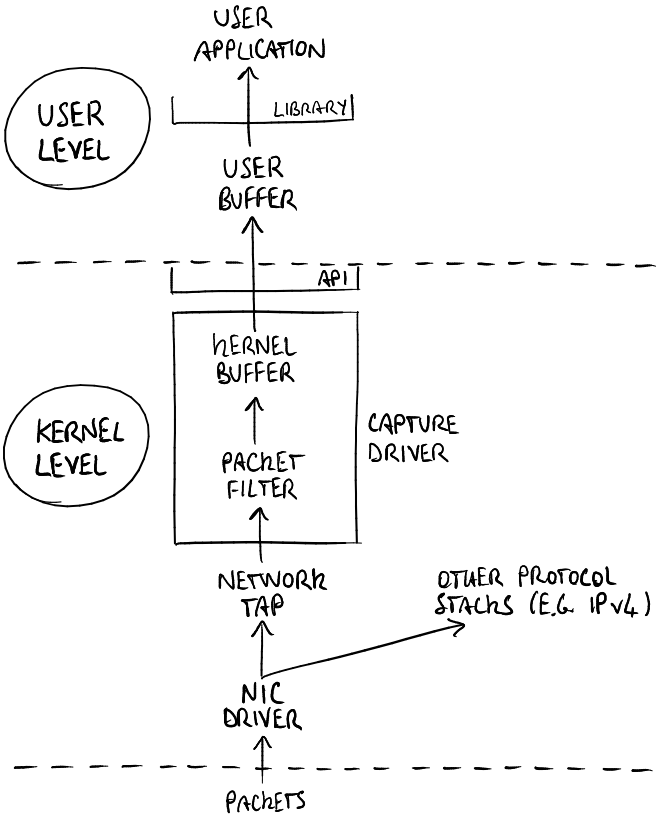


Figura 17.1: Architettura tipica di un sistema di filtraggio dei pacchetti.

- **kernel buffer**: memorizza i pacchetti prima che vengano consegnati al livello user;
- **API di livello kernel**: fornisce al livello user le primitive, tipicamente chiamate di sistema `ioctl`, necessarie per accedere ai livelli sottostanti.

Componenti di livello user

- **user buffer**: memorizza i pacchetti nello spazio di indirizzamento dell'applicazione utente;
- **libreria di livello user** (ad es. libpcap, WinPcap): esporta delle funzioni che sono mappate con le primitive fornite dalla API di livello di kernel, e fornisce un compilatore di alto livello per creare al volo il codice pseudo-assembler da iniettare nel packet filter.

17.2 Principali sistemi di filtraggio dei pacchetti

17.2.1 CSPF

Il **CMU/Stanford Packet Filter** (CSPF, 1987) fu il primo packet filter, ed era implementato in parallelo alle altre pile protocollari.

Introdusse alcune migliorie chiave:

- implementazione a livello kernel: l'elaborazione è più veloce perché si evita il costo della commutazione di contesto tra il kernel space e lo user space, anche se è più facile danneggiare l'intero sistema;
- **packet batching**: il kernel buffer non consegna subito un pacchetto arrivato all'applicazione, ma aspetta che ne siano memorizzati un certo numero e poi li copia tutti insieme nello user buffer per ridurre il numero di commutazioni di contesto;

- **macchina virtuale**: i filtri non sono più hard-coded, ma il codice di livello user può istanziare in fase di esecuzione un pezzo di codice in linguaggio pseudo-assembler che specifica le operazioni di filtraggio per determinare se il pacchetto può passare o deve essere scartato, e una macchina virtuale nel packet filter, composta in pratica da uno `switch case` su tutte le istruzioni possibili, emula un processore che interpreta quel codice per ogni pacchetto in transito.

17.2.2 BPF/libpcap

Il **Berkeley Packet Filter** (BPF, 1992) fu la prima implementazione seria di un packet filter, adottato storicamente dai sistemi BSD e usato ancora oggi in coppia con la libreria **libpcap** in user space.

Architettura

- network tap: è integrata nel driver della NIC, e può essere chiamata con esplicite chiamate ai componenti di cattura;
- kernel buffer: è suddiviso in due aree di memoria separate, in modo che i processi di livello kernel e di livello user possano operare in modo indipendente (il primo scrive mentre il secondo legge) senza necessità di sincronizzazione sfruttando due core della CPU in parallelo:
  - lo **store buffer** è l'area in cui scrive il processo di livello kernel;
  - l'**hold buffer** è l'area da cui legge il processo di livello user.

17.2.3 NPF/WinPcap

La libreria **WinPcap** (1998), sviluppata inizialmente al Politecnico di Torino, può essere considerata un porting per Windows dell'intera architettura BPF/libpcap.

Architettura

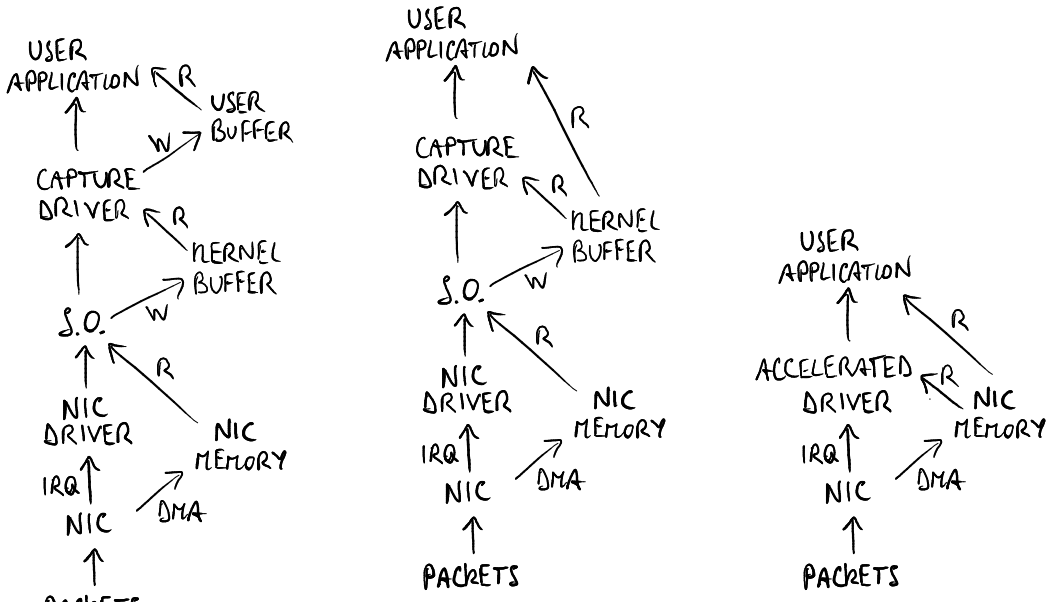
- **Netgroup Packet Filter** (NPF): è il componente di livello kernel e include:
  - network tap: giace sopra i driver della NIC, registrandosi come nuovo protocollo di livello rete accanto ai protocolli standard (come IPv4, IPv6);
  - packet filter: la macchina virtuale è un **compilatore just in time** (JIT): invece di interpretare il codice, lo traduce in istruzioni native del processore x86;
  - kernel buffer: è implementato come un **buffer circolare**: i processi di livello kernel e di livello user scrivono nella stessa area di memoria, e il processo di livello kernel sovrascrive i dati già letti dal processo di livello user ⇒ ottimizza lo spazio in cui memorizzare i pacchetti, ma:
    * se il processo di livello user è troppo lento a leggere i dati, il processo di livello kernel potrebbe sovrascrivere dei dati non ancora letti (**cache pollution**) ⇒ è necessaria la sincronizzazione tra i due processi: il processo scrivente deve ispezionare periodicamente una variabile condivisa contenente la posizione di lettura corrente;
    * l'area di memoria è condivisa tra i core della CPU ⇒ il buffer circolare è meno efficiente dal punto di vista della CPU;
- **Packet.dll**: esporta a livello user delle funzioni, indipendenti dal sistema operativo, che sono mappate con le primitive fornite dalla API di livello di kernel;
- **Wpcap.dll**: è la libreria a collegamento dinamico con cui l'applicazione interagisce direttamente:

– offre al programmatore le funzioni di libreria di alto livello necessarie per accedere ai livelli sottostanti (ad es. `pcap_open_live()`, `pcap_setfilter()`, `pcap_next_ex()`/`pcap_loop()`);
– include il compilatore che, dato un filtro definito dall'utente (ad es. stringa `ip`), crea il codice pseudo-assembler (ad es. "se il campo 'EtherType' è uguale a 0x800 restituisci vero") da iniettare nel packet filter per il compilatore JIT;
– implementa lo user buffer.

Nuove funzionalità

- **statistics mode**: registra dati statistici nel kernel senza commutazioni di contesto;
- **packet injection**: invia pacchetti attraverso l'interfaccia di rete;
- **remote capture**: attiva un server remoto che cattura i pacchetti e li consegna localmente.

17.3 Ottimizzazioni prestazionali



(a) Architettura tradizionale. *(b) Architettura con shared buffer.* *(c) Architettura con driver accelerato.*

Figura 17.2: Evoluzione delle tecniche di ottimizzazione delle prestazioni.

Negli ultimi anni il traffico sulle reti è cresciuto più velocemente rispetto alle prestazioni dei computer (memoria, CPU). Le prestazioni dell'elaborazione dei pacchetti possono essere migliorate in vari modi:

- incrementare le prestazioni della cattura: migliorare la capacità di recapito dei dati al software;
- creare componenti di analisi più intelligenti: solamente i dati più interessanti vengono recapitati al software (ad es. l'URL per un URL filter);
- ottimizzare l'architettura: cercare di sfruttare le caratteristiche dell'applicazione per migliorare le prestazioni.

Dati di profiling (WinPcap 3.0, pacchetti da 64 byte)

− [49,02%] driver NIC e sistema operativo: quando entra nella NIC, il pacchetto impiega un sacco di tempo solo per arrivare alla pila di cattura:

1. la NIC trasferisce il pacchetto nella sua zona di memoria kernel tramite DMA (non usa la CPU);
2. la NIC lancia un **interrupt** (IRQ) al driver della NIC, interrompendo il programma correntemente in esecuzione;
3. il driver della NIC copia il pacchetto dalla memoria della NIC in una zona di memoria kernel del sistema operativo (usa la CPU);
4. il driver della NIC invoca il sistema operativo cedendo il controllo ad esso;
5. il sistema operativo chiama le varie pile protocollari registrate, incluso il driver di cattura;

− [17,70%] tap processing: operazioni svolte dal driver di cattura al principio della pila di cattura (ad es. ricevere i pacchetti, impostare gli interrupt);

− [8,53%] timestamping: al pacchetto viene associato il timestamp;

\+ [3,45%] packet filter: i costi di filtraggio sono proporzionalmente bassi grazie al compilatore JIT;

− doppia copia nei buffer: il costo delle copie aumenta tanto più i pacchetti sono grandi:

1. [9,48%] copia nel kernel buffer: il pacchetto è copiato dalla memoria del sistema operativo al kernel buffer;
2. [11,50%] copia nello user buffer: il pacchetto è copiato dal kernel buffer allo user buffer;

\+ [0,32%] commutazione di contesto: ha un costo insignificante grazie al packet batching.

17.3.1 Interrupt

In tutti i sistemi operativi, ad un certo input rate la percentuale di pacchetti che arrivano all'applicazione di cattura non solo non aumenta più, ma diminuisce drasticamente a causa del **livelock**: gli interrupt sono così frequenti che il sistema operativo non ha il tempo di leggere i pacchetti dalla memoria della NIC e copiarli nel kernel buffer per consegnarli all'applicazione ⇒ il sistema è vivo e sta facendo del lavoro, ma non sta facendo del lavoro utile.

Esistono varie soluzioni per abbattere il costo degli interrupt:

- **interrupt mitigation** (basato su hardware): viene scatenato un interrupt solo quando è stato ricevuto un certo numero di pacchetti (un timeout evita la starvation se la soglia minima non è raggiunta entro un certo tempo);
- **interrupt batching** (basato su software): quando arriva un interrupt, il sistema operativo serve il pacchetto arrivato e poi lavora in modalità polling: serve immediatamente i pacchetti successivi arrivati nel frattempo, fino a quando non ci sono più pacchetti e l'interrupt può essere riabilitato sulla scheda;
- **device polling** (ad es. BSD [Luigi Rizzo]): il sistema operativo non attende più un interrupt, ma controlla autonomamente con un ciclo infinito la memoria della NIC ⇒ siccome un core della CPU è perennemente impegnato nel ciclo infinito, questa soluzione è adatta quando sono necessarie prestazioni veramente elevate.

17.3.2 Timestamping

Esistono due soluzioni per ottimizzare il timestamping:

- timestamp approssimato: il tempo reale è letto solo ogni tanto, e il timestamp è basato sul numero di cicli di clock trascorsi dall'ultima lettura ⇒ il timestamp dipende dalla frequenza di clock del processore, e i processori hanno frequenze di clock sempre maggiori;
- timestamp hardware: il timestamp è implementato direttamente nella scheda di rete ⇒ i pacchetti arrivano al software già dotati di timestamp.

17.3.3 Copia nello user buffer

La zona di memoria kernel in cui è presente il kernel buffer viene mappata in user space (ad es. tramite `nmap()`) ⇒ la copia dal kernel buffer allo user buffer non è più necessaria: l'applicazione può leggere il pacchetto direttamente dallo **shared buffer**.

Implementazione Questa soluzione è adottata in nCap di Luca Deri.

Problemi

- sicurezza: l'applicazione accede a delle zone di memoria kernel ⇒ può danneggiare il sistema;
- indirizzamento: il kernel buffer è visto attraverso due spazi di indirizzamento differenti: gli indirizzi usati in kernel space sono diversi dagli indirizzi usati in user space;
- sincronizzazione: l'applicazione e il sistema operativo devono lavorare su variabili condivise (ad es. le posizioni di lettura e scrittura dei dati).

17.3.4 Copia nel kernel buffer

Il sistema operativo non è fatto per supportare un grande traffico di rete, ma è stato ingegnerizzato per eseguire applicazioni utente con un limitato consumo di memoria. Prima di arrivare alla pila di cattura, ogni pacchetto è memorizzato in un'area di memoria del sistema operativo che è allocata dinamicamente come una lista linkata di piccoli buffer (`mbuf` in BSD e `skbuf` in Linux) ⇒ il costo di allocazione e di liberazione dei mini-buffer è troppo oneroso rispetto a un grande buffer allocato staticamente in cui memorizzare pacchetti di qualsiasi dimensione.

Le schede dedicate alla cattura non sono più viste dal sistema operativo, ma usano **driver accelerati** inglobati nella pila di cattura: la NIC copia il pacchetto nella sua zona di memoria kernel, e l'applicazione legge direttamente da quella zona di memoria senza l'intermediazione del sistema operativo.

Implementazioni Questa soluzione è adottata in netmap di Luigi Rizzo e in DNA di Luca Deri.

Problemi

- applicazioni: le altre pile protocollari (ad es. pila TCP/IP) sono scomparse ⇒ la macchina è completamente dedicata alla cattura;
- sistema operativo: è richiesta una modifica intrusiva al sistema operativo;
- NIC: il driver accelerato è fortemente legato alla NIC ⇒ non è possibile usare un altro modello di NIC;
- prestazioni: il collo di bottiglia rimane la banda del bus PCI;
- timestamp: non è preciso a causa dei ritardi dovuti al software.

17.3.5 Commutazione di contesto

L'elaborazione è spostata in kernel space, evitando la commutazione di contesto allo user space.

Implementazione Questa soluzione è adottata nel Data Plane Development Kit (DPDK) di Intel, con lo scopo di realizzare apparati di rete programmabili via software su hardware Intel.

Problemi

- packet batching: il costo della commutazione di contesto è ridicolo grazie al packet batching;
- debug: è più facile in user space;
- sicurezza: l'intera applicazione lavora con memoria kernel;
- programmazione: è più difficile scrivere il codice nel kernel space.

17.3.6 NIC intelligenti

L'elaborazione è svolta direttamente dalla NIC (ad es. Endace):

- elaborazione hardware: evita il collo di bottiglia del bus PCI, limitando lo spostamento dei dati (anche se il miglioramento prestazionale è limitato);
- precisione del timestamp: non c'è il ritardo dovuto al software ed è basato sul GPS ⇒ queste NIC sono adatte per catture su reti geograficamente estese.

17.3.7 Parallelizzazione in user space

FFPF ha proposto un'architettura che cerca di sfruttare le caratteristiche dell'applicativo per andare più veloce, aumentando il parallelismo in user space: l'applicazione di cattura è multi-thread ed è eseguita su CPU multi-core.

L'hardware può aiutare la parallelizzazione: la NIC può registrarsi al sistema operativo con più adapter, e ciascuno di essi è una coda logica distinta da cui escono i pacchetti a seconda della classificazione, effettuata da **filtri hardware**, in base ai loro campi (ad es. indirizzi MAC, indirizzi IP, porte TCP/UDP) ⇒ più software possono leggere da code logiche distinte in parallelo.

Applicazioni

- receive side scaling (RSS): la classificazione è basata sull'identificativo di sessione (quintupla) ⇒ tutti i pacchetti appartenenti alla stessa sessione vanno alla stessa coda ⇒ è possibile bilanciare il carico sui server Web;[4]
- virtualizzazione: ogni macchina virtuale (VM) su un server ha un indirizzo MAC diverso ⇒ i pacchetti entrano direttamente nella VM giusta senza essere toccati dal sistema operativo (hypervisor).[5]

[4] Si veda la sezione 15.2.1.
[5] Si rimanda alla sezione 18.2.3.

Capitolo 18

Introduzione alle Software-Defined Network

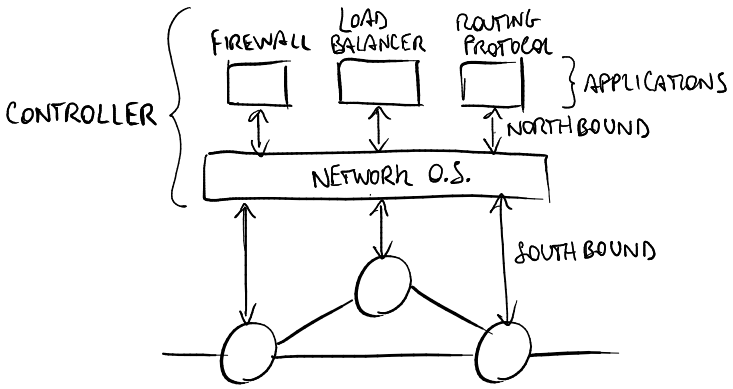


Figura 18.1: Componenti di una SDN.

Internet è ancora quella che è stata definita 30 anni fa: un tubo molto efficiente che trasporta bit ad alta velocità, con quasi gli stessi protocolli e la stessa filosofia.

Gli apparati di rete sono monolitici: ogni router contiene, oltre all'hardware specializzato per l'inoltro dei pacchetti, il suo sistema operativo e le sue applicazioni. Questa infrastruttura è chiusa alle innovazioni: le componenti software non sono installabili dal cliente ma sono imposte dal costruttore dell'hardware, il quale non è interessato a innovare se è il leader del mercato (cioè Cisco).

Le **Software-Defined Network** (SDN) introducono la possibilità di programmare la rete, e sono basate su tre pilastri:

- separazione delle funzioni di controllo e di inoltro: il software, la componente intelligente, è separato dall'hardware;
- centralizzazione del controllo: l'intera rete è coordinata da un **controller**, composto da un sistema operativo di rete e da applicazioni di rete definite dall'utente (ad es. protocollo di instradamento, load balancer, firewall);
- interfacce ben definite:
  - **northbound**: il sistema operativo di rete espone delle API alle applicazioni di rete;
  - **southbound**: il sistema operativo di rete pilota i nodi della rete, composti da hardware semplice per l'inoltro dei pacchetti.

Il **sistema operativo di rete** è un livello software che offre una visione globale e astratta della rete alle applicazioni superiori. La visione "dall'alto" della rete consente ad esempio il **traffic engineering**: le decisioni sono prese dalla logica centralizzata del load balancer e sono perciò coerenti per tutti i nodi di rete:

- modalità proattiva: prima che l'apparato cominci a inoltrare i pacchetti, il controller riempe a priori la tabella di inoltro con tutte le regole necessarie per tutte le sessioni;
- modalità reattiva: quando arriva il primo pacchetto di una sessione, l'apparato lo manda al controller, il quale prende una decisione e comunica all'apparato la regola necessaria per l'inoltro dei pacchetti di quella sessione.

Uno strato di **network slicing** può far vedere al software anche una topologia di rete diversa dalla reale infrastruttura fisica: può essere configurato in modo da mostrare a ogni istanza di sistema operativo topologie virtuali differenti (ad es. un sottoinsieme dei link reali) ⇒ le politiche per il traffico di una certa azienda hanno effetto solo sulla porzione di rete appartenente all'azienda.

Problemi

- controller: può costituire un singolo punto di guasto e un collo di bottiglia;
- versatilità: un firewall ha bisogno di ispezionare tutti i pacchetti, non solo il primo pacchetto della sessione ⇒ un sacco di traffico sarebbe generato tra l'apparato e il controller;
- scalabilità: l'hardware per l'inoltro non può essere troppo semplice per ottenere alte prestazioni;
- economia: la semplificazione dell'hardware va contro gli interessi economici dei principali fornitori di rete.

18.1 OpenFlow

OpenFlow, introdotto attorno al 2008, è una implementazione dell'interfaccia southbound.

Può essere distribuito in vari modi:

- regole: tipicamente sono basate sul flusso, cioè definite in base alla tupla (indirizzi MAC, indirizzi IP, porte TCP);
- controller: tipicamente è fisicamente centralizzato, ma può anche essere fisicamente distribuito (seppur sempre logicamente centralizzato);
- modalità: tipicamente è reattiva, ma nulla vieta di usare la modalità proattiva.

A ogni regola sono associate una o più **azioni**, ad esempio:

- inoltra il pacchetto sulla/e porta/e;
- incapsula e inoltra al controller;
- scarta il pacchetto;
- invia alla pipeline di elaborazione classica (cioè la tabella di instradamento classica);
- modifica i campi (ad es. NAT: cambia gli indirizzi e le porte).

OpenFlow 1.3 Ha introdotto alcune funzioni interessanti:

- la tabella di inoltro è suddivisa in varie sottotabelle (ad es. firewall, instradamento, ecc.), e ogni applicazione accede alla sua sottotabella ⇒ il matching di ogni pacchetto è effettuato più volte attraverso le sottotabelle in sequenza;
- **virtual switch** (vSwitch, ad es. Open vSwitch): invece di essere implementato in hardware, OpenFlow è eseguito su uno switch emulato da un processo software ⇒ è possibile creare un tunnel logico GRE tra due vSwitch su due server diversi attraverso una rete di switch tradizionali.[1]

Problemi

- piano dati: si occupa solo dell'inoltro dei pacchetti ⇒ è adatto per ambienti (ad es. datacenter) dove l'inoltro dei pacchetti è un aspetto preponderante rispetto al piano dati, ma non sembra appropriato per la rete di un ISP;
- utilità: l'interfaccia di southbound è meno interessante di quella di northbound: serve per gli sviluppatori di sistemi operativi di rete, non per gli sviluppatori delle applicazioni;
- costo hardware: le regole possono essere basate su un gran numero di campi che rendono le entry molto ampie ⇒ le TCAM necessarie sono costose e scaldano parecchio;
- flessibilità: in contrapposizione alla Open Networking Foundation (ONF) (VMware), il progetto OpenDaylight (Cisco) preferisce il **Network Configuration Protocol** (NETCONF) che, invece di esplicitare le regole, non conosce la semantica dei valori letti o impostati ⇒ può essere usato dal controller SDN per configurare delle funzionalità avanzate sugli apparati, come le "rotte di backup" in caso di guasti rilevati essere critici nella rete di un ISP.

18.2 Piano dati

Non è solo importante inoltrare i pacchetti nella giusta direzione, ma anche offrire dei **servizi** orientati al piano dati che elaborano i pacchetti (ad es. firewall, NAT, monitor di rete).

18.2.1 Service Function Chaining senza SDN

Figura 18.2: Service Function Chaining (SFC) senza SDN.

Oggi i servizi si possono aggiungere ai router di accesso (BNG), oltre che con le service card[2], collegando dei box chiamati appliance: un'**appliance** è un dispositivo hardware separato e discreto con software integrato (firmware) dedicato a fornire uno specifico servizio. Le appliance sono collegate in cascata con fili fisici a formare una **catena di servizi** statica, e ogni pacchetto deve essere elaborato da un servizio dopo l'altro prima di poter uscire dall'apparato.

[1] Si rimanda alla sezione 18.2.3.
[2] Si veda la sezione 16.5.

Svantaggi

- agilità nella fornitura di nuovi servizi: l'appliance deve essere collegata fisicamente all'apparato;
- flessibilità: per collegare una nuova appliance è necessario spezzare temporaneamente la catena interrompendo il servizio di rete;
- affidabilità: un'appliance guasta spezza la catena interrompendo il servizio di rete;
- ottimizzazione: ogni appliance ha a disposizione una quantità fissa di risorse, e nei picchi di lavoro non può sfruttare le risorse eventualmente lasciate libere in quel momento da un'altra appliance.

18.2.2 Service Function Chaining con SDN

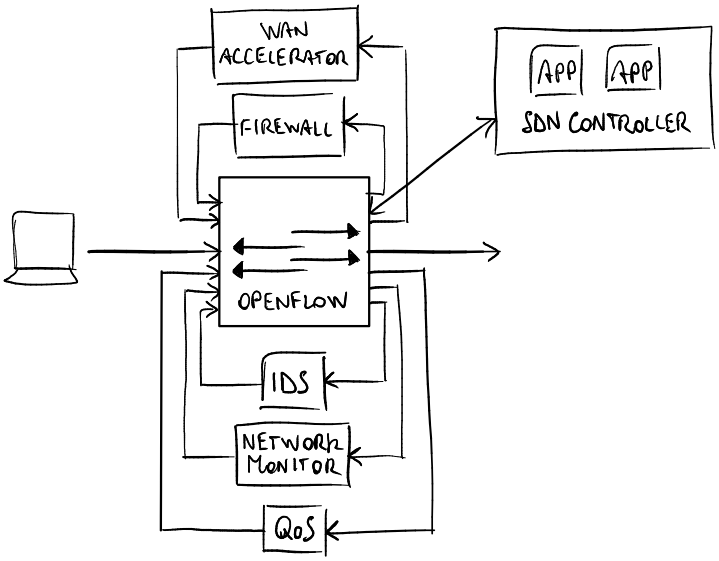


Figura 18.3: Service Function Chaining (SFC) con SDN.

Ogni appliance è collegata a una porta di uscita e una porta di ingresso di uno switch OpenFlow, e i flussi di traffico attraversano una catena di servizi decisa dinamicamente tramite regole OpenFlow che definiscono percorsi da una porta all'altra dello switch.

Vantaggi

- flessibilità: l'aggiunta di una nuova appliance richiede una modifica al volo della regola OpenFlow da parte del controller SDN senza interrompere il servizio di rete;
- affidabilità: è sufficiente una modifica al volo della regola OpenFlow da parte del controller SDN per ripristinare il servizio di rete;
- business: è possibile differenziare i percorsi in base al cliente (azienda) ⇒ il traffico passa solo per i servizi che il cliente ha acquistato.

Svantaggi

- agilità nella fornitura di nuovi servizi: l'appliance deve essere collegata fisicamente all'apparato;

- ottimizzazione: ogni appliance ha a disposizione una quantità fissa di risorse, e nei picchi di lavoro non può sfruttare le risorse eventualmente lasciate libere in quel momento da un'altra appliance;
- retrocompatibilità: occorre sostituire gli apparati con switch che supportano OpenFlow.

18.2.3 Network Function Virtualization

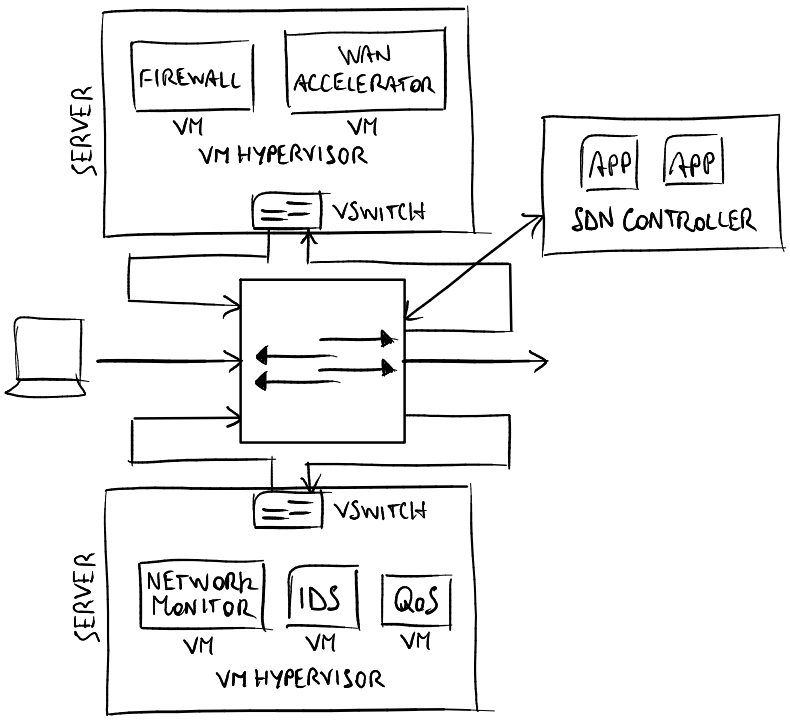


Figura 18.4: Network Function Virtualization (NFV).

I servizi sono implementati in un processo puramente software: lo switch è collegato a vSwitch OpenFlow emulati su più server remoti, e ogni server è dotato di un hypervisor in grado di eseguire delle **macchine virtuali** (VM) al cui interno sono eseguiti i servizi.

Scalamento Le prestazioni di un servizio possono essere migliorate in tre modi:

- **scale up**: vengono assegnate più risorse hardware alla VM ⇒ ciò può non essere sufficiente se il servizio non è in grado di sfruttare l'hardware disponibile adeguatamente (ad es. un programma single-thread non trae molto beneficio da un ambiente multi-thread);
- **scale out**: più VM sono eseguite in parallelo su uno stesso server fisico ⇒ è necessario un load balancer per mandare il traffico alla VM meno carica, e le VM necessitano sincronizzazione;
- più server: più VM sono eseguite in parallelo su più server fisici ⇒ è necessario un ulteriore load balancer per mandare il traffico al server meno carico.

Vantaggi

- agilità nella fornitura di nuovi servizi: un nuovo servizio può essere attivato dinamicamente scaricando ed avviando la sua immagine software;

- ottimizzazione: le risorse hardware del server sono condivise tra le VM;
- retrocompatibilità: se lo switch non supporta OpenFlow, è possibile sfruttare il tunnel GRE tra i vSwitch senza dover sostituire l'apparato;
- consolidamento: di notte è possibile ridurre il numero di VM eseguite in parallelo (**scale in**) e diminuire le risorse hardware assegnate (**scale down**).

Svantaggi

- traffico: il classico modello NFV può richiedere ai pacchetti di viaggiare da un server all'altro attraverso lo switch, intasando la rete su cui i server sono distribuiti;
- efficienza: i server hanno CPU general-purpose, non hardware dedicato (ad es. line card), e non sono attualmente disponibili tecnologie efficaci di accelerazione hardware;
- migrazione: quando l'utente si sposta, l'istanza della VM deve essere spostata sul server più vicino e deve essere avviata nel più breve tempo possibile;
- scalabilità: l'architettura è potenzialmente molto scalabile, ma soffre di problemi di sincronizzazione e di bilanciamento del carico quando più istanze del servizio sono eseguite in parallelo.

18.3 OpenStack

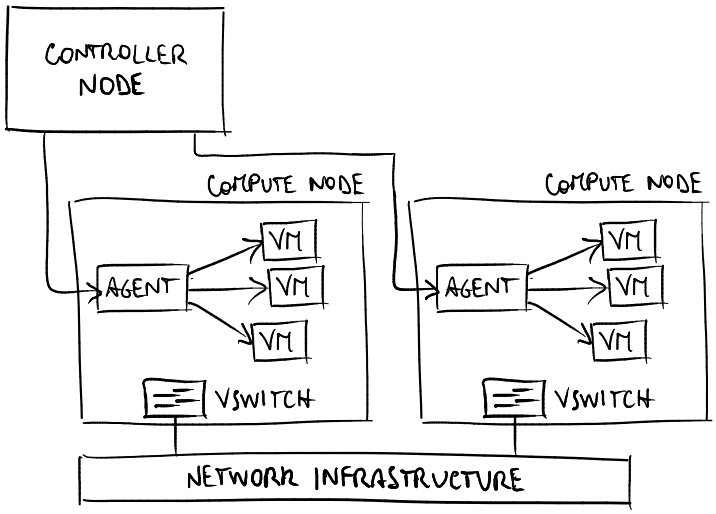


Figura 18.5: Componenti di un sistema OpenStack.

OpenStack, introdotto nel 2010, è un sistema operativo distribuito open source:

- Linux:
  - controlla il singolo host locale su cui è eseguito;
  - l'unità di esecuzione è il processo;
- OpenStack:
  - è eseguito su un server remoto, chiamato **controller node**;
  - controlla più server fisici distribuiti nella cloud, chiamati **compute node**;

– l'unità di esecuzione è la macchina virtuale.

Ogni **compute node** include i seguenti componenti:

- sistema operativo tradizionale: controlla l'hardware locale del server fisico;
- agente: riceve i comandi dal controller node, per esempio per lanciare le VM;
- vSwitch (ad es. Open vSwitch): connette il server all'infrastruttura di rete.

Uno dei compiti del **controller node** è lanciare le VM sul compute node correntemente meno carico.