UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica

TECNOLOGIE MPLS PER RETI DI TRASPORTO E INTERNET SERVICE PROVIDER

IMPLEMENTAZIONE DI UN TESTBED SU PIATTAFORMA LINUX

ANNO ACCADEMICO 2010-2011

Relatore: Prof. Massimo Maresca

Laureando: Alessandro Piva

Correlatore: Ing. Martino Fornasa

Ringraziamenti

Mi ritrovo in questa calda serata d'inizio luglio a scrivere finalmente la fatidica paginetta dei ringraziamenti. Dico "finalmente" perché questo significa che la mia data di laurea si avvicina e con essa la fine di un'era personale.

Sicuramente un percorso accademico costellato di molti sacrifici, un po' di studio e, non mi sbilancio, da qualche successo. Di certo, se sono arrivato fin qui lo devo anche e soprattutto a quelle persone che hanno sempre creduto in me e che ininterrottamente hanno appoggiato o criticato costruttivamente le mie scelte.

Il maggior ringraziamento va alla mia famiglia, in primis a papà e mamma, sempre pronti e disponibili ad incoraggiare i miei studi, anche nei momenti per me più bui.

Ringrazio di cuore mio fratello Marco per essere stato un punto di riferimento per la mia carriera universitaria e non, nonché per rappresentare sempre e comunque una fonte d'ispirazione ed innovamento personale. A lui va la mia promessa di tornare presto a nuotare con la Squadra Master.

Ai miei nonni va un grazie affettuoso, per avermi cresciuto così e per essermi sempre stati vicini fin da quando muovevo i primi passi.

Un grazie speciale va anche a Stefano e Chiara per avermi sopportato anche quando io stesso avrei fatto fatica a sopportarmi. Grazie Ragazzi!!

Ringrazio il mio Maestro Elio per avermi fatto scoprire, come solo lui sa fare, il piacere della buona musica. A lui si aggiungono anche i colleghi musicisti: Alberto, Carlo e Daniele per aver tenuto costantemente viva la mia vena artistica.

Un ringraziamento sentito lo devo al Prof. Massimo Maresca e ai ragazzi del Laboratorio Sintesi, Ing. Martino Fornasa e Ing. Michele Stecca per la loro disponibilità e professionalità. In particolare, sono riconoscente a Martino per avermi sempre seguito in questi mesi senza mai perdere la pazienza!

Mi sento in dovere di dire grazie anche all'Ing. Carlo Zaina per le sue mai noiose lezioni alla Cisco Networking Academy e ai miei compagni di corso, in particolare: Marco, Giordano e Giuliano.

Senza tutti voi non ce l'avrei mai fatta!!!!!

INDICE

INDICE delle FIGURE

INDICE delle TABELLE

1. Introduzione

Multi Protocol Label Switching (MPLS) [1] è una tecnologia di rete che utilizza un meccanismo di inoltro basato sulle etichette, o *label*. L'idea di fondo è di etichettare i pacchetti in ingresso in base al loro indirizzo di destinazione, o ad altri criteri decisi a priori, e a instradare il traffico su un'infrastruttura di rete comune. MPLS aggiunge al pacchetto in ingresso un'etichetta, o più specificatamente uno "*shim header*", incapsula il nuovo pacchetto etichettato in un frame di livello inferiore e ne esegue l'inoltro. Esso viene, pertanto, considerato un protocollo di "livello 2.5" del modello di riferimento ISO/OSI. Ormai da una decina d'anni gli aspetti implementativi di base sono stati standardizzati dall'IETF attraverso numerose *Request for Comments* (RFC), in particolare a partire dalla RFC 3031.

Le principali ragioni che hanno portato all'adozione di MPLS sono da ricondurre all'esigenza di avere un'*infrastruttura di rete unificata* basata su IP. Le grandi aziende stanno evolvendo verso un'architettura di questo tipo, sulla quale è possibile trasportare anche altri protocolli di livello 2 e fornire così collegamenti a livello Data-Link tra siti distanti geograficamente.

L'emulazione di un servizio end-to-end di layer 2 su una rete a commutazione di pacchetto, come IP/MPLS, prende il nome di *PseudoWire* (PW) [2]. Tra le numerose applicazioni di questa tecnologia, sono particolarmente rilevanti le reti *Ethernet over MPLS* (EoMPLS) [3] e *Virtual Private Lan Services* (VPLS) [4]. L'emulazione di link Ethernet in versione point-to-point o point-to-multipoint su lunghe distanze ha permesso un'ottimizzazione non solo del traffico dati, ma anche e soprattutto, di quello relativo al *Voice Over IP* (VOIP). La gestione di tale traffico di natura eterogenea ha richiesto l'integrazione di MPLS con applicazioni di *Traffic Engineering* (MPLS-TE) [5] per il controllo del *Quality of Service* (QoS) [6].

Negli ultimi anni, MPLS si è affermata sempre più come tecnologia leader del mercato, surclassando quasi definitivamente ATM [7] e Frame Relay [8] nelle reti di medie e grandi dimensioni. Essa, però, risultava ancora inadatta a reti geografiche di vastissima

estensione come quelle degli operatori di telecomunicazioni [9], le cui caratteristiche di altissima efficienza e velocità sono classificabili nell'ambito delle *reti di trasporto*, o "transport networks" [10]. Tali esigenze hanno richiesto una generalizzazione del concetto di label, riconducendolo ad un contesto più ampio di *risorsa*. In *Generalized MPLS* (GMPLS) [11], un'etichetta, non solo può essere un identificativo numerico come in MPLS, ma può corrispondere anche ad un dato slot TDM [12], ad una particolare lunghezza d'onda WDM [13] o ad una porta dello switch in fibra. Questo è reso possibile da un Control Plane comune per la gestione integrata delle metodologie trasmissive appena citate, generalizzando quanto proposto da MPLS-TE.

Dall'evoluzione congiunta delle precedenti tecnologie sono iniziati lo sviluppo e la standardizzazione di un nuovo profilo basato su MPLS, *Multi Protocol Label Switching - Transport Profile* (MPLS-TP) [14]. MPLS-TP ha l'obiettivo di introdurre un innovativo paradigma di trasporto per la realizzazione di reti a commutazione di pacchetto con le stesse potenzialità di quelle a commutazione di circuito, soprattutto in termini di *Operations, Administrations and Maintenance* (OAM) [15] e flessibilità per l'utilizzo ottimale della banda in presenza di dati a flusso non costante, come il traffico voce.

Il presente lavoro ha avuto come obiettivo quello di creare in laboratorio una rete pilota che permettesse di verificare le modalità di funzionamento delle tecnologie MPLS esistenti e che fornisse un ambiente per la sperimentazione di nuove tecnologie di trasporto. Tale piattaforma è stata sviluppata tramite lo studio preliminare delle tecnologie considerate e mediante la progettazione, la configurazione ed il test funzionale di una rete di PC operanti come apparati di rete, allo scopo di mantenere la necessaria flessibilità e configurabilità richieste da un ambiente sperimentale. A tale scopo sono stati selezionati ed impiegati dei software open source che implementassero gli stack protocollari richiesti, e tali software sono stati poi opportunamente integrati nell'ambiente di test.

La parte di studio preliminare ha permesso di fornire, per ogni tecnologia affrontata, non delle semplici riproposizioni di quanto disponibile in letteratura, ma anche un approccio algoritmico al funzionamento, descritto passo-passo al termine di ogni capitolo ed effettivamente riscontrabile attraverso l'implementazione del testbed.

Nella prima parte di questa tesi, sono stati forniti gli elementi per capire a fondo i princìpi ispiratori e le metodologie di funzionamento sia di MPLS che delle tecnologie da esso derivate. In questa parte teorica non ci si è limitati a riproporre semplicemente quanto già proposto dalla letteratura sull'argomento ma si è corredato il tutto con dettagli implementativi sulle procedure funzionali. Nello specifico, nel capitolo 2 vengono introdotti i concetti di ISP, di reti di trasporto e descritti nel dettaglio i protocolli di routing intra e inter-AS [16]. In particolare, viene data un'esauriente spiegazione dell'algoritmo di funzionamento di OSPF e di quali siano i passi

fondamentali per il processo decisionale di BGP. Nel capitolo 3 si introduce l'architettura di MPLS e si spiega come questo si collochi nell'Autonomous System [17]. L'interazione con LDP [1] e con gli altri protocolli di routing viene affrontata fino a descrivere dettagliatamente i passi necessari per la costituzione di un Label Switched Path (LSP) [1]. Nel capitolo 4 viene introdotto MPLS-TP, descrivendo le principali tecnologie che lo compongono: MPLS-TE, GMPLS e PW, dando anche un'overview dello stato dell'arte in termini di standardizzazione.

Infine, nella seconda parte si è implementato un testbed in laboratorio, attraverso l'uso degli applicativi open source *MPLS-Linux* [18] e *Quagga* [19]. Si è voluto prestare particolare attenzione al capitolo 5, tentando di fornire una guida esaustiva per il setup dell'ambiente di lavoro, e cercando anche di dare ordine alla vasta e non sempre corretta documentazione fruibile online. Si sono eseguite e commentate le catture del traffico relativo alla piattaforma, dando riscontro a quanto affrontato nella parte teorica. Inoltre, è stato possibile integrare la suite Quagga con un elemento indispensabile per l'ambiente di testing, LDP. La distribuzione delle etichette è stata monitorata e comparata rispetto a quanto visto nel caso di una assegnazione da management, verificando l'efficienza e rilevando quali siano le problematiche relative all'integrazione di protocolli di routing aggiuntivi alla stabilità futura del sistema.

2. Protocolli di Routing in Internet

2.1. Domini e Autonomous System

E' utile introdurre i concetti di "*sistema autonomo*" e "*dominio*", in quanto ad essi si farà spesso riferimento nei prossimi capitoli e, salvo alcuni casi generici, sono identificati da definizioni diverse [16][1]:

- Autonomous System: un sistema autonomo è un insieme di router in genere sotto una singola amministrazione tecnica, caratterizzato per l'impiego di uno o più protocolli di routing intra/inter Autonomous System. Ciascun AS adotta una singola e ben definita *policy* interna che ne regola il traffico, ed è identificato da un numero univoco rilasciato dal Regional Internet Registry (RIR) di riferimento: APNIC (Asia e Oceania), RIPE NCC (Europa), LACNIC (Sud America), ARIN (Nord America) e AfriNIC (Africa);

- *Dominio*: è una collezione di router gestita e coordinata, che adotta un unico protocollo di routing. Un AS comprende uno o più domini al suo interno. Un sistema autonomo potrebbe ammettere protocolli di routing diversi, ad esempio, OSPF nella sede centrale e RIP nella sua sede staccata. In tal caso questi sono considerati due domini distinti all'interno dello stesso AS.

E' possibile effettuare un prima suddivisione degli Autonomous System in:

- *Stub*: il cui traffico circolante all'interno ha sorgente o destinazione nell'AS stesso;

- *Transit*: il cui traffico entrante si limita ad attraversare il sistema e confluire verso uno o più punti d'uscita.

E' necessario definire anche i concetti [20] di :

- *Transito*: è una tipologia di contratto in cui, tipicamente, un cliente paga un ISP per ottenere accesso ad internet. Il traffico di transito è definito come l'insieme dei dati che attraversa l'ISP e che ha come sorgente, destinazione, o entrambe, un AS esterno all'ISP stesso;

- *Peering*: è una tipologia di contratto in cui due ISP scambiano mutuamente il traffico dei relativi clienti a titolo gratuito.

La seconda distinzione tra diverse tipologie [21] di sistemi autonomi è la seguente:

- *Single-homed* [Figura 1.]: è definito così un AS che possiede un solo punto d'uscita verso l'esterno. Solitamente esso coincide con uno Stub-AS, e può fare affidamento su un'unica *default route*, per la gestione del traffico in uscita;

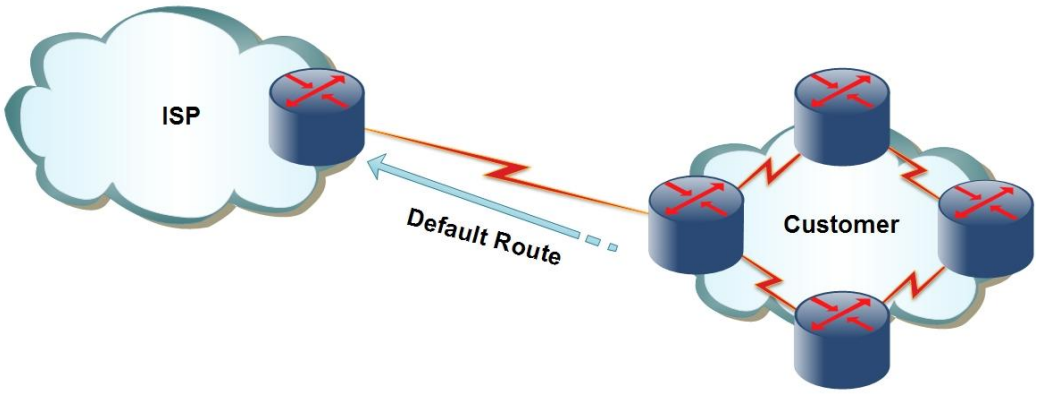

Figura 1. Single-homed Autonomous System

- *Multihomed Non-Transit* [Figura 2.]*:* questa tipologia di AS ha più punti d'uscita verso l'esterno. Un AS connesso a Internet può essere multihomed verso un singolo o più ISP. Questa tipologia di sistema autonomo non permette che il traffico in transito (che ha sorgente o destinazione al di fuori dell'AS di riferimento) passi attraverso di esso. E' chiaro che se una destinazione risulta raggiungibile attraverso due uscite diverse, l'AS in questione potrebbe preferire selezionarne in via prioritaria una piuttosto che l'altra. Inoltre, per non permettere il transito, esso dovrà configurare adeguatamente i propri filtri in entrata/uscita;

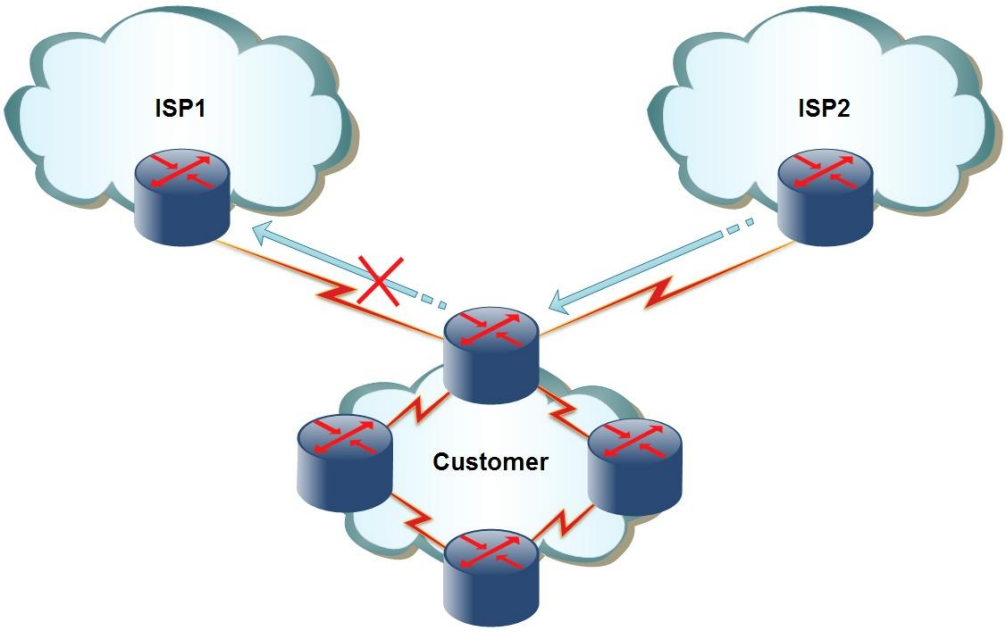

Figura 2. Multihomed Non-Transit Autonomous System

- *Multihomed* [Figura 3.]: tale tipologia di AS ha più punti di uscita verso l'esterno ed è intenzionalmente utilizzato da altri AS per il transito di traffico.

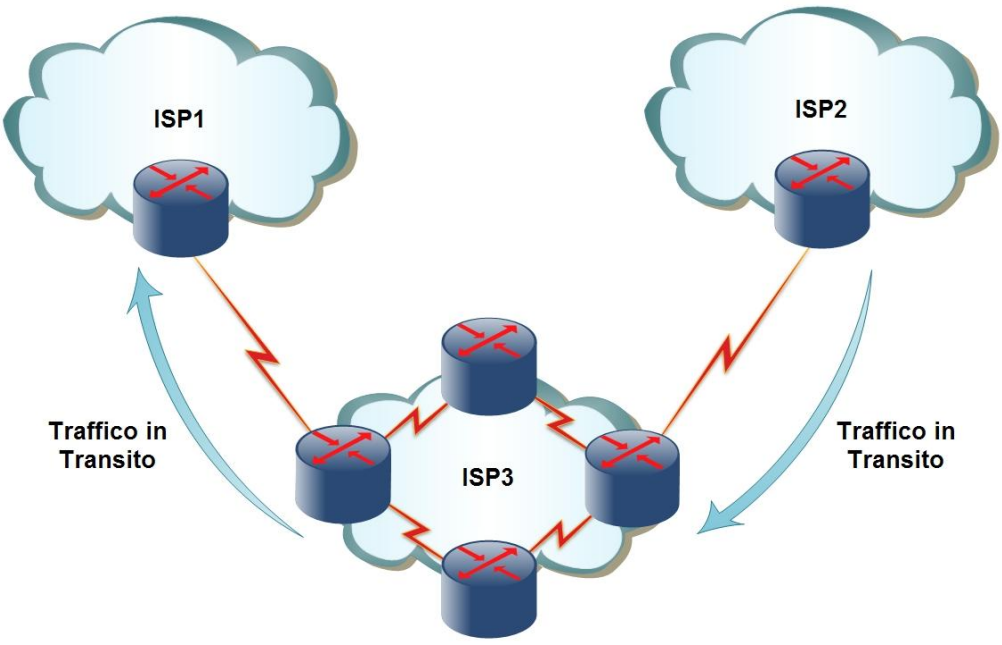

Figura 3. Multihomed Autonomous System (ISP3)

2.2. ISP e Reti di Trasporto

Per definizione, un ISP [9] *è un'organizzazione commerciale che si occupa di fornire ai propri utenti connettività o altri servizi Internet, quali: email, web-hosting, telefono, ecc.* Tale fornitura può avvenire attraverso la rete cablata (rame o fibra), o via wireless.

Per "rete di trasporto" [10] *s'intende l'insieme delle risorse, gestite da un ISP, che ha come obiettivo quello di convogliare il traffico generato dagli utenti da un luogo all'altro.*

Gli ISP possono essere suddivisi in base alla loro dimensione e diffusione sul territorio [22] [Figura 4.]:

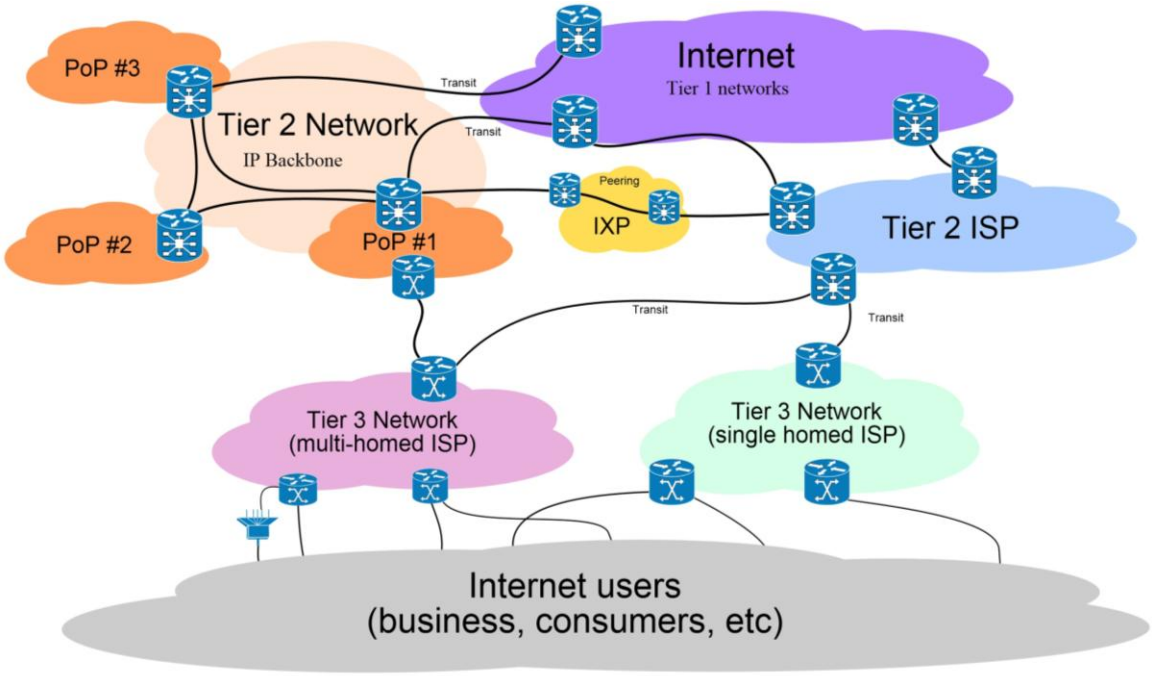

Figura 4. Suddivisione gerarchica degli ISP

1. *Tier 1*: appartengono a questa categoria gli ISP che si rapportano tra loro soltanto attraverso peering;
2. *Tier 2*: questo livello è formato da ISP che sono clienti di altri di Tier 1 (ISP del medesimo livello adottano tra loro il peering) mentre sono fornitori per quelli

di livello 3. A questa categoria solitamente appartengono provider nazionali o regionali (ad esempio *Telecom Italia*);

3. *Tier 3*: sono ISP di piccole dimensioni, grandi reti aziendali o fornitori di contenuti, che sono unicamente clienti di altri ISP di livello 2 o 3.

Infine, esistono particolari punti centrali di interconnessione chiamati *Internet eXchange Point* (IXP). Un IXP [23] è una infrastruttura di rete gestita da un'unica entità allo scopo di facilitare lo scambio di traffico Internet tra diversi ISP. Il numero di sistemi autonomi connessi deve essere almeno pari a tre e devono esistere politiche chiare ed aperte che consentano la partecipazione di nuovi AS al punto di inter-scambio.

2.3. Protocolli di Routing Intra-AS

Il *routing* [24] è l'azione di trasferire informazioni tra due o più reti, da una sorgente ad una destinazione. Tale trasferimento può essere circoscritto all'interno di un singolo AS (routing *intra-AS*) o tra AS distinti (routing *inter-AS*). Un sistema autonomo è governato da due famiglie distinte di protocolli [25]: la prima, *Interior Gateway Protocol* (IGP), responsabile del *routing intra-AS*; la seconda, *Exterior Gateway Protocol* (EGP), utilizzata nel contesto del *routing inter-AS*. Se ci si riferisce ad un Autonomous System come ad un unico dominio amministrativo, allora il routing intra/inter-AS diventa intra/inter-dominio [16].

OSPF è un protocollo per il routing intra-AS. Esso prevede che ogni router mantenga al suo interno il grafo topologico dell'intera rete e calcoli i percorsi di costo minimo per tutte le altre destinazioni all'interno dell'AS.

Architettura di OSPF

Open Shortest Path First (OSPF) [26], è un protocollo di routing di tipo *Link State* (LS), così classificato in quanto trasferisce le informazioni di instradamento a tutti i peer della rete, non soltanto ai propri vicini come accade per gli algoritmi di tipo *Distance Vector* (DV) [27]. Ciò è assicurato dal cosiddetto "*flooding*", o propagazione, di messaggi contenenti costi e altre informazioni relativamente ai propri link. Così facendo, ciascun router all'interno del dominio popola e dispone di uno stesso identico *Link State-DataBase* (LS-DB). OSPF prevede il calcolo dell'albero di costo minimo attraverso l'algoritmo di *Dijkstra* per lo *Shortest Path First* (SPF), prendendo come input il LS-DB e considerando sé stesso come radice dell'albero. Tutti i router OSPF eseguono tale algoritmo in parallelo. In aggiunta, se vengono identificati più cammini

di egual costo per una stessa destinazione, si possono adoperare semplici strategie di *load balancing* sui link, partizionando il traffico in maniera paritaria sui percorsi di costo minimo.

Il LS-DB descrive la topologia di rete e rappresenta un grafo orientato e pesato, i cui vertici possono essere un router o una rete (solitamente broadcast). Gli archi identificano il collegamento punto-punto tra due router o il link del router sulla rete ad esso collegata. Il grafo può essere rappresentato come una matrice di adiacenza, il cui numero di righe (e di colonne) è uguale alla somma dei router e delle reti presenti. In un dominio OSPF un costo, o *metrica*, viene associato all'interfaccia uscente e rappresenta il peso del grafo orientato. Infine, nel caso di collegamenti tra una rete e un router, la metrica associata è sempre posta uguale a zero.

OSPF calcola il costo di ogni collegamento, basandosi sulla capacità del link. L'equazione utilizzata è:

$$Metrica = 10^8 / capacità\ del\ link\ [bps]$$

Adottando questa metrica, si presenta un problema con le velocità di connessione superiori a 100 Mbps, come per le interfacce Fast Ethernet e Gigabit Ethernet, in quanto entrambi i link hanno valore 1. Per compensare tale imprecisione, dovuta all'età del protocollo, sui router si può configurare manualmente il valore di costo dell'interfaccia, o più generalmente, il valore a numeratore della precedente formula.

Le reti presenti nel grafo possono essere un end-point (*stub-network*) o una rete di transito (*transit-network*). Le route per reti esterne, acquisite da altre istanze di protocolli di routing, sono considerate come stub-network. Il diagramma di rete [Figura 5.] può essere considerato un esempio grafico di LS-DB:

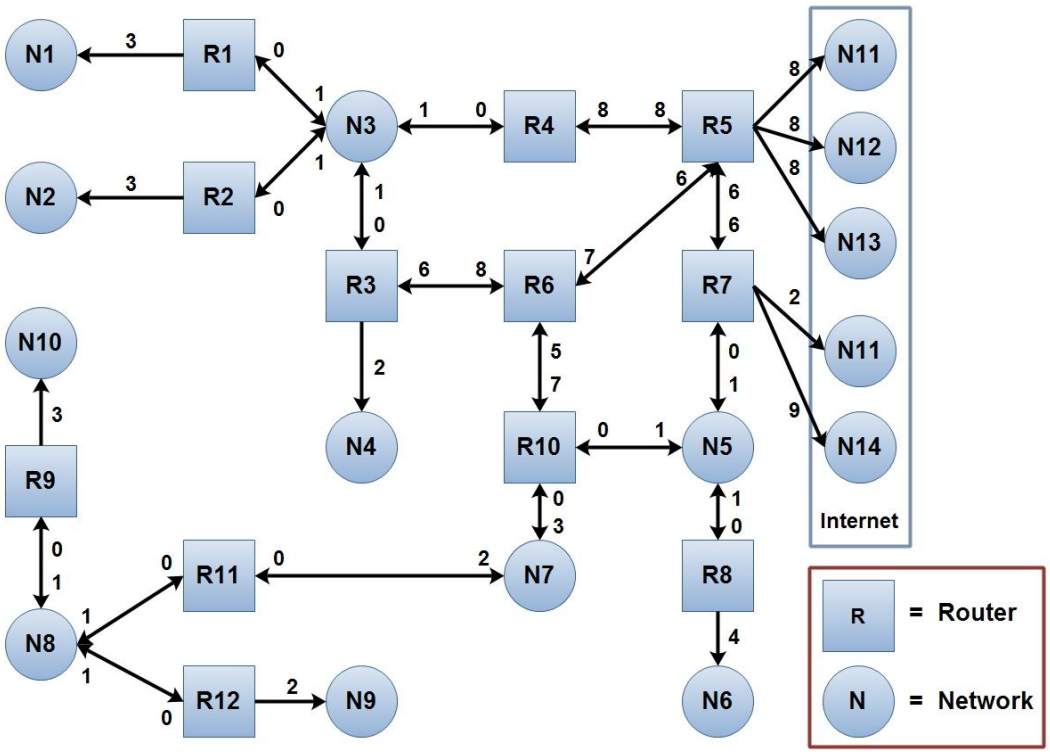

Figura 5. Rappresentazione grafica del LinkState-DataBase

Ciascun router OSPF computa l'albero di costo minimo attraverso l'algoritmo di Dijkstra, prendendo in input il proprio LS-DB, considerando se stesso come radice dell'albero, e producendo in output la tripla *<Destinazione, Next-Hop, Costo>,* dove la destinazione è un nodo del grafo, il next-hop è il router successivo del percorso, ed il costo è la somma dei costi sui link per arrivare a destinazione. Le destinazioni esterne al proprio dominio vengono in questo momento escluse dalla trattazione, per essere riprese in seguito. Vediamo un esempio di albero di costo minimo [Figura 6.], prendendo in input il grafo orientato in [Figura 5.], e considerando come radice R6:

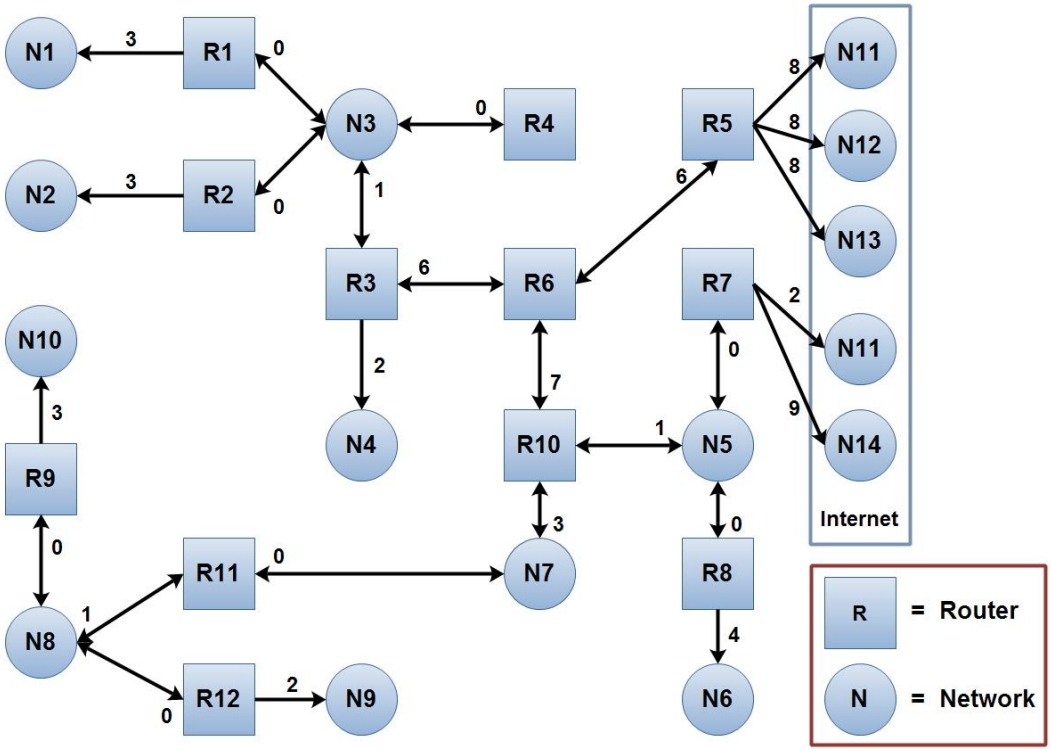

Figura 6. Rappresentazione grafica dell'albero di costo minimo (per R6)

L'output prodotto contiene tutte le informazioni necessarie per l'instradamento ed è pertanto detto tabella di routing, o *routing table* [Tabella 1.].

Destinazione	Next-Hop	Costo
N1	R3	10
N2	R3	10
N3	R3	7
N4	R3	8
N5	R10	8
N6	R10	12
N7	R10	10
N8	R10	11
N9	R10	13
N10	R10	14

Tabella 1. Routing table di un router OSPF (R6)

OSPF è progettato per operare correttamente con reti:

- Point-to-Point (PPP [52], HDLC [53]);
- Broadcast Multi-Access (LAN);
- Non-Broadcast Multi-Access (ATM [7], Frame Relay [8]).

Gerarchia su due livelli: le aree

Poiché ogni router conosce la topologia dell'intera rete, col crescere della stessa il ricalcolo con l'algoritmo di Dijkstra può impiegare eccessive risorse di CPU e memoria. Per questa ragione sono state introdotte le *aree*. Ogni router possiede, in realtà, un LS-DB relativo all'area a cui appartiene, non all'intera rete. In caso di espansione, possono essere create altre aree adiacenti all'area principale di riferimento, identificate da valori compresi nell'intervallo $1 \div 2^{32} - 1$. Il massimo numero di router presenti in un'area è 50.

OSPF prevede una gerarchia su due livelli:

- Area 0 (o area 0.0.0.0, in quanto le area-ID vengono spesso specificate nello stesso formato degli indirizzi IP), chiamata anche *area di backbone* ed è il livello più alto della gerarchia;
- Le altre aree sono al livello inferiore, dipendenti dall'area di backbone, a cui sono direttamente connesse.

Un gruppo di aree è detto anche "sistema autonomo OSPF" (OSPF-AS) [Figura 7.] o dominio OSPF. L'instradamento dei pacchetti effettuato in esso non è circoscritto ai router appartenenti ad una certa area, ma è da intendersi esteso a tutto l'OSPF-AS.

Un router OSPF può essere classificato logicamente in una o più delle seguenti tipologie:

- *Area Border Router* (ABR): connette due o più aree ed è membro di tutte quelle a cui è connesso. Per ognuna di queste tiene in memoria una copia del relativo LS-DB;
- *Internal router* (IR): appartiene ad una sola area e conosce solamente i router nella stessa;
- *Backbone router* (BR): appartiene all'area di backbone. Un ABR è anche un BR, anche se non necessariamente vale l'inverso;

- *Autonomous System Boundary Router* (ASBR): connette l'OSPF-AS con altri domini di routing. Gli ASBR tipicamente utilizzano anche un protocollo di routing EGP, come BGP [29]. Un ASBR può essere utilizzato per ridistribuire le route ricevute dagli altri AS attraverso il proprio ed il percorso per ciascun ASBR è noto a tutti i router dell'AS di riferimento. Questa classificazione è del tutto indipendente dalle precedenti in quanto un ASBR può essere IR o ABR, BR o non-BR;

- *Designated router* (DR): è eletto all'interno di un segmento di rete, ad esempio in una LAN. La principale funzione del DR è proprio quella di limitare il traffico scambiato all'interno del proprio segmento dovuto allo scambio degli LSA;

- *Backup Designated Router* (BDR): viene eletto DR se l'attuale diventa irraggiungibile, ad esempio in seguito ad un guasto.

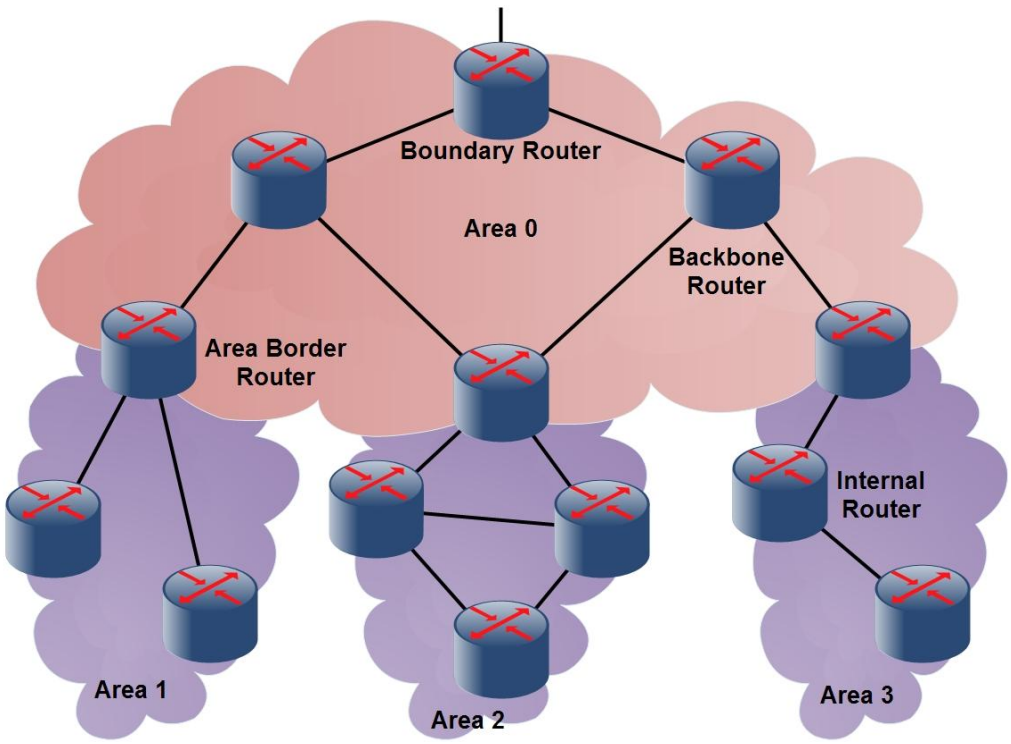

Figura 7. Aree OSPF

L'inoltro del traffico può avvenire in modalità *intra-area*, in tal caso sorgente e destinazione appartengano alla stessa area, o *inter-area* se sorgente, destinazione o entrambe appartengono ad aree diverse.

Protocolli Hello, Exchange e Update

Ogni router ha il compito di rendere noto lo stato dei propri link attraverso comunicazioni LS, dette *Link-State Advertisement* (LSA). Due nodi OSPF vengono detti "vicini", o *neighbors*, se sono connessi alla medesima rete (punto-punto o punto-multipunto) e se possono comunicare direttamente, mentre sono definiti "adiacenti" se scambiano tra loro informazioni di routing.

Per "adiacenza" [26] *si intende la relazione instaurata tra due router vicini, con lo scopo di scambiarsi informazioni topologiche. OSPF richiede che solo router adiacenti debbano rimanere sincronizzati sul database topologico che descrive l'area cui appartengono.*

In una rete ad accesso multiplo, ad esempio in una LAN, ognuno degli N router connessi allo switch è di fatto un vicino di tutti gli altri. Vengono pertanto complessivamente instaurate N (N – 1) / 2 adiacenze e il numero totale di LSA trasmessi è N^2 (N – 1). Conviene adottare una topologia a stella equivalente, inserendo un nodo virtuale che rappresenta la rete e ottenendo in totale solo N archi bidirezionali. In questo tipo di reti risulta molto più efficiente eleggere un router designato, o *Designated Router* (DR), fra gli N vicini. In tal modo, ognuno di questi è adiacente solo al DR, lo scambio di informazioni di routing avviene solo tra router adiacenti (in altre parole il DR fa da tramite) e il DR è l'unico a comunicare la raggiungibilità di router e host della LAN all'interno dell'area. Per ragioni di affidabilità occorre avere anche un router designato di backup, o *Backup Designated Router* (BDR), anch'esso adiacente a tutti i router locali e che subentra al DR in caso di guasto.

Il processo di sincronizzazione del LS-DB parte non appena si cerca di stabilire un'adiacenza. Un router ha conoscenza dei vicini grazie a pacchetti di *Hello*, periodicamente inviate dai router ad esso collegati per segnalare la propria presenza. Successivamente a questa prima parte di discovery, viene determinata una relazione master/slave. Come già accennato, le comunicazioni sullo stato dei link avvengono tramite degli annunci, o LSA.

I Link-State Advertisement (LSA) [26] *sono formalmente definiti come un'unità di dati che descrive lo stato locale di un router o di una rete.*

Il master invia dei pacchetti di descrizione del suo LS-DB, o *Database Description* (DD), contenenti una lista di header LSA, allo slave che risponde con pacchetti DD riguardanti il proprio. A questo punto è possibile che un router richieda le informazioni mancanti tramite *Link State Request* cui la controparte risponde con dei *Link State Update*. Concluso l'intero processo di sincronizzazione iniziale dei database topologici, l'adiacenza viene considerata pienamente funzionante e tenuta attiva tramite l'invio di

pacchetti Hello ad intervalli regolari. OSPF invia messaggi utilizzando IP e contrassegnandone il campo *Protocol* con il valore 89.

E' possibile identificare tre fasi di funzionamento per OSPF [Figura 8.]:

1. *Hello Protocol*: impiega un unico tipo di pacchetto, Hello (Type 1), ed è utilizzato per controllare l'operatività dei link, scoprire e mantenere le adiacenze fra vicini, eleggere DR e BDR. I pacchetti di Hello sono inviati sulle interfacce periodicamente secondo quanto specificato dal parametro *HelloInterval*. Si riesce così a conoscere se per ciascun vicino è presente un collegamento bidirezionale e se esso è ancora attivo;

2. *Exchange Protocol*: mira a sincronizzare gli LS-DB dei nodi adiacenti. Determinati il master e lo slave, il primo invia al secondo una serie di pacchetti Database Description (Type 2) contenenti la lista degli LSA del proprio database. Nell'elenco sono indicati il tipo di LSA, l'aging time, il router che l'ha generato e il numero di sequenza. Lo slave risponde con un'altra serie di Database Description e durante lo scambio ciascuno dei due router confronta i dati ottenuti con quelli in proprio possesso. Se nel proprio LS-DB sono presenti informazioni meno recenti rispetto a quanto ricevuto, queste vengono richieste con un successivo pacchetto Link State Request (Type 3);

3. *Flooding Protocol*: la propagazione degli LSA a tutti i router della rete avviene tramite l'invio di pacchetti Link State Update (Type 4). Ciò può succedere a fronte di un cambiamento nello stato di un collegamento, al ricevimento di una Link State Request o periodicamente (ogni 30 minuti). Si esegue il protocollo di flooding per propagare le informazioni LS a tutti i nodi del dominio e si continua ad inviare lo stesso update finché non viene confermata la sua ricezione dai vicini tramite il pacchetto Link State Acknowledgment (Type 5).

Prima di raggiungere la convergenza, cioè completare la procedura di adiacenza con i propri vicini, il router passa attraverso vari cambiamenti di stato:

- *Down*: è lo stato iniziale della conversazione con un vicino;
- *2-Way*: indica che è stata stabilita una prima comunicazione bidirezionale. Può essere stabilita un'adiacenza;
- *ExStart*: è il primo stato per la costruzione dell'adiacenza ed è in questo momento che si stabilisce la relazione master/slave;
- *Exchange*: indica che è possibile scambiarsi i pacchetti Database Description per iniziare la sincronizzazione dei database;

- *Loading*: denota la possibilità di inviare pacchetti di LSA-Request/LSA-Update;
- *Full*: gli LS-DB sono sincronizzati e l'adiacenza è pienamente attiva. In questo caso è possibile lanciare l'algoritmo di Dijkstra, tenendo conto che esso dovrà essere computato nuovamente ogni qual volta arriveranno nuove informazioni LS.

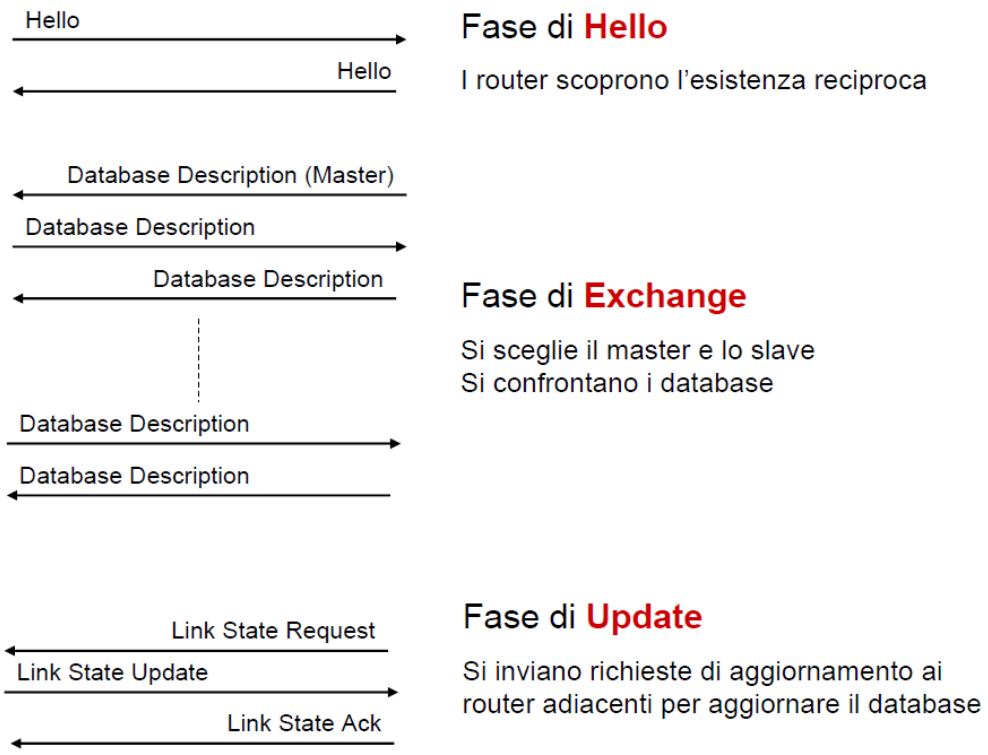

Figura 8. Fasi per l'instaurazione dell'adiacenza OSPF

Nei collegamenti Point-to-Point, non vi sono problemi nello stabilire le adiacenze, poiché, per definizione, sul link possono essere presenti solo due nodi. Al contrario, in una rete Broadcast Multi-Access, ovvero in un segmento in cui sono collegati più router, ciascuno di questi può raggiungere lo stato di Full solo dopo aver eletto DR e BDR. Tutti gli altri router del segmento assumono il ruolo di *DROther* e devono essere adiacenti a DR [Figura 9.a] e BDR [Figura 9.b]. Gli LSA da e verso questi ultimi sono inviati in multicast all'indirizzo 224.0.0.6; a sua volta il DR è incaricato di distribuire il cambiamento a tutti gli altri router OSPF del dominio, sempre in multicast, ma all'indirizzo 224.0.0.5. Oltre a ridurre il traffico nel segmento, questo meccanismo assicura che tutti i nodi ricevano le stesse informazioni, nello stesso momento e da una singola fonte. Il DR mantiene il suo ruolo sino ad un eventuale fallimento; se il DR

risulta irraggiungibile, il BDR prende il suo posto diventando DR e viene successivamente eletto un nuovo BDR.

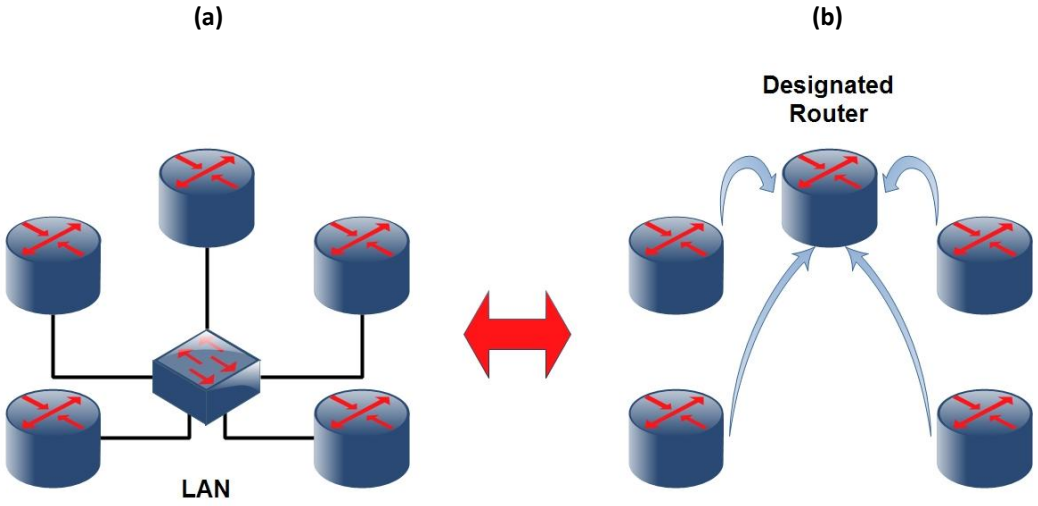

Figura 9. (a) Segmento di una rete OSPF (b) Designated Router per il segmento

Nel caso in cui l'amministratore di rete lo desideri, può forzare l'elezione del DR e del BDR configurando la priorità su ciascun router (un'azione di management è richiesta necessariamente per le reti Non-Broadcast Multi-Access). In maniera predefinita, i router OSPF hanno una priorità di valore 1; se questa viene cambiata, il router con la priorità più alta vince l'elezione indipendentemente dal router-ID. Il massimo valore assumibile è 255, mentre 0 significa che il nodo non partecipa all'elezione. All'interno della rete, viene eletto DR chi ha il router-ID più alto mentre il secondo più alto viene eletto BDR. Il router-ID è un indirizzo IP, scelto in questo modo:

- impostandolo tramite l'apposito comando;
- se nessun valore è impostato manualmente dall'amministratore, viene utilizzato il più alto indirizzo IP configurato sulle interfacce di loopback;
- se nessuna interfaccia di loopback è configurata, viene impiegato il più alto indirizzo IP sulle interfacce fisiche attive.

Per interfaccia di *loopback* s'intende un'interfaccia virtuale, manualmente configurata, che si utilizza di norma per scopi di management. Al contrario delle consuete "loopback interface" configurate negli host (il range di indirizzi IP 127.0.0.0/8), questo tipo di astrazione non è utilizzata dal router per dialogare con sé stesso. Essa viene mantenuta esclusivamente per scopi di management e non corrisponde a nessuna

interfaccia fisica. Se la loopback non fosse raggiungibile significherebbe che il router stesso è spento o presenta dei problemi di connettività. Oltre a quanto detto finora, l'interfaccia di loopback assicura una serie di vantaggi in ambito di stabilità e prevedibilità nelle reti di ISP.

Link-State Advertisement

Tutte le informazioni di instradamento vengono scambiate o meglio, annunciate, tramite messaggi chiamati Link-State Advertisement (LSA). Essi dispongono tutti di un intestazione comune (i primi 128 bit) [Tabella 2.] e successive istanze di un singolo LSA sono numerate in ordine crescente:

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	LS Age																LS Type															
32	Link State ID																															
64	Advertising Router																															
96	LS Sequence Number																															
128	LS Checksum																Length															
160	Dipende dal campo "LS Type".																															
192																																
~																																
~																																
~																																
~	…																															

Tabella 2. Header comune di un LSA

Gli LSA, di cui è riportato il payload, possono essere classificati come:

- *Router-LSA* (Type 1) [Tabella 3.]: servono a descrivere lo stato e l'identità dei link di un router all'interno di un'area. I tipi di connessione di cui esso può disporre sono sostanzialmente quattro: point-to-point, connessione ad una transit-network, connessione ad una stub-network e virtual link. Tali LSA sono alla base del routing intra-area;

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
160	0			Nt	x	V	E	B	Options																							
192	Type								0								Metric															
224	Interface ID																															
256	Neighbor Interface ID																															
288	Neighbor Router ID																															
320	…																															
~	Type								0								Metric															
~	Interface ID																															
~	Neighbor Interface ID																															
~	Neighbor Router ID																															
~	…																															

Tabella 3. Payload di un Router-LSA

- *Network-LSA* (Type 2) [Tabella 4.]: sono generati per ciascun segmento dal DR della rete stessa e sono propagati verso gli altri router adiacenti della propria area. Essi hanno lo scopo di descrivere lo stato delle connessioni elencando nell'LSA tutti i router adiacenti al router designato. Se il DR fallisce in favore del BDR, il primo dovrà rimuovere dal suo LS-DB tutti i Network-LSA da esso generati e al contempo il nuovo DR genererà i Network-LSA del caso;

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
160	0								Options																							
192	Attached Router																															
~	…																															

Tabella 4. Payload di un Network-LSA

- *Summary-LSA* (Type 3) [Tabella 5.]: vengono generati dagli ABR e propagati all'interno di una sola area per volta. Questi LSA indicano la disponibilità, da parte dell'ABR originante, ad inoltrare pacchetti aventi per destinazione gli indirizzi riportati nell'annuncio ad un costo anch'esso riportato nell'LSA. Da ciò si intuisce come tali LSA siano alla base del routing inter-area;

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
160	0								Metric																							
192	PrefixLength								PrefixOptions								0															
224	Address Prefix																															
256																																
288	…																															

Tabella 5. Payload di un Summary-LSA

- *AS-Boundary-Router-Summary-LSA* (Type 4) [Tabella 6.]: simili ai precedenti, contengono informazioni per raggiungere ogni ASBR dell'OSPF-AS;

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
160	0								Options																							
192	0								Metric																							
224	Destination Router ID																															

Tabella 6. Payload di un AS-Boundary-Router-Summary-LSA

- *AS-External-LSA* (Type 5) [Tabella 7.]: sono generati dagli ASBR ed indicano, all'interno dell'OSPF-AS, il range di indirizzi che l'ASBR originante è in grado di raggiungere. La destinazione riportata è accompagnata da un'indicazione del costo di routing inter-AS, espresso attraverso una metrica di tipo 1 o 2. Questi LSA sono propagati attraverso tutte le aree.

Bit/ Byte	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
160						E	F	T	Metric																							
192	PrefixLength								PrefixOptions								Referenced LS Type															
224	Address Prefix																															
256	…																															
288	Forwarding Address (Optional)																															
~																																
~																																
~																																
~	External Route Tag (Optional)																															
~	Referenced Link State ID (Optional)																															

Tabella 7. Payload di un AS-External-LSA

Riassumendo:

- I Router-LSA sono creati da ogni router e diffusi in una singola area; contengono le informazioni relative allo stato delle interfacce del router all'interno di quell'area;
- I Network-LSA sono creati dal DR di ogni rete ad accesso multiplo e diffusi in una singola area, all'esterno del proprio segmento. Contengono l'elenco dei router connessi alla LAN di cui quel router è membro designato;
- I Summary-LSA sono creati dagli ABR e diffusi in una singola area. Contengono informazioni di routing per destinazioni appartenenti ad altre aree dello stesso AS;
- Gli AS-Boundary-Router-Summary-LSA sono creati dagli ABR e propagati a tutte le aree. Contengono informazioni di routing per raggiungere gli AS-Boundary Router;
- Gli AS-External-LSA sono creati dagli ASBR e diffusi in tutto l'OSPF-AS; contengono informazioni di instradamento per destinazioni appartenenti ad altri AS, e comprendono anche l'eventuale default route.

Di norma, gli LSA sono propagati attraverso tutte le interfacce su una stessa area con esclusione dell'interfacce da cui sono arrivati ed hanno un aging time di 1 ora, oltre la quale sono cancellati. In ogni caso, i router inviano ogni 30 minuti dei refresh indipendentemente da eventuali cambiamenti di topologia.

Funzionamento di OSPF

In base a quanto visto finora a proposito di OSPF, è stato possibile delineare un possibile *algoritmo di funzionamento*, che ne indica sinteticamente e passo-passo le fasi, dalla scoperta dei vicini fino all'installazione delle entry nella routing table di ciascun nodo (si considera il dominio formato da un'unica area OSPF):

1. L'Hello Protocol viene utilizzato per individuare i vicini. Il router invia, ad intervalli regolari, pacchetti Hello su tutte le sue interfacce e a sua volta ne riceve;
2. Su tutte le reti Broadcast Multi-Access durante il protocollo Hello si devono eleggere DR e BDR, che hanno il compito di raccogliere le informazioni LS sulla propria LAN e propagarle agli altri nodi del dominio OSPF;

3. Ciascun router tenta di costituire le adiacenze con i propri vicini attraverso l'Exchange Protocol. I messaggi di request/update (LSA) sono inviati e ricevuti al fine di sincronizzare i propri database topologici con le informazioni LS (principalmente, il costo del link). I nodi di una LAN, invece, instaurano le proprie adiacenze solamente con DR e BDR;

4. Gli LSA vengono propagati attraverso il Flooding Protocol all'indirizzo Multicast 224.0.0.5, per poter tenere sempre aggiornato lo stato di tutti i nodi dell'OSPF-AS e affinché essi abbiano gli stessi LS-DB;

5. Ciascun router OSPF al termine della procedura ha la conoscenza di tutte le reti annunciate nel dominio e nel contempo mantiene attive le connessioni coi vicini attraverso l'invio periodico di messaggi Hello. Ad ogni cambiamento dello stato dei propri link, le nuove informazioni LS vengono propagate agli altri nodi del dominio;

6. Ogni router utilizza l'algoritmo di Dijkstra per calcolare l'albero di costo minimo, configurando sé stesso come radice e tutti gli altri router/reti del dominio come foglie;

7. E' possibile installare i percorsi di costo minimo all'interno della routing table, che sarà consultata ad ogni inoltro verso una data destinazione.

Bisogna precisare che in aree distinte vengono eseguiti istanze differenti dello stesso algoritmo di Dijkstra, in quanto esse presentano LS-DB distinti; per quei router che sono collegati su più aree si avrà un'istanza dell'algoritmo per ogni area. OSPF, di norma, aspetta 5 secondi prima di propagare le informazioni relative ad un link disconnesso. Esiste, inoltre, un timer che determina qual è il tempo minimo tra un ricalcolo e l'altro dell'albero di costo minimo che, di default, è settato a 10 secondi. Il meccanismo basato sui timer serve ad evitare il blocco dei router dovuto al continuo ricalcolo dei percorsi.

La routing table ottenuta dalla computazione di OSPF è composta dalle seguenti voci:

- Type: router o network, area border router, AS border router;
- Destination IP;
- Area Number;
- Path Type: intra-area, inter-area, external type 1, external type 2;
- Metric, derivante dal calcolo del percorso di costo minimo;
- IP Next Hop;

- Advertising Router.

Le route esterne

Per route esterna [26] *si intende qualsiasi annuncio generato originariamente da route statiche o da altri protocolli di routing quali RIP* [42], *IS-IS* [39], *BGP* [29], *ecc. e ridistribuito da OSPF all'interno del proprio dominio.*

Le route esterne vengono propagate a tutto l'OSPF-AS tramite uno o più router ASBR. Esse sono definibili come *costi esterni* e sono di due tipi:

- *External-Type 1*: metriche espresse nella stessa unità di misura di quelle interne;
- *External-Type 2*: metriche espresse in un'unità di misura di magnitudo maggiore rispetto a quelle esterne di tipo 1. Si assume, infatti, che queste prevalgano su qualsiasi altro costo interno, che in tale situazione verrà ignorato.

Nel caso vengano annunciate route esterne di tipo 1 distinte per la stessa destinazione, in ciascun nodo il calcolo del percorso minimo sarà computato sommando i costi esterni a quelli interni per raggiungere l'ASBR. E' possibile, quindi, che router diversi utilizzino punti d'uscita dall'AS diversi per la stessa destinazione. In caso di presenza di metriche esterne tutte di tipo 2, il costo interno invece viene ignorato, ed è scelto il punto di uscita con costo esterno minore, ignorando la metrica complessiva per raggiungere il router di bordo. Nel caso in cui i costi di tipo 2 assumano lo stesso valore per tutte le uscite dall'AS, si sceglie il percorso complessivo con minor costo interno.

Route multiple verso la stessa destinazione, disponibili attraverso molteplici modalità di routing, sono preferite nel seguente ordine: intra-area, inter-area, external-type 1, external-type 2.

Per impedire che gli ASBR annuncino route esterne non desiderate è possibile filtrarle. Tale meccanismo può essere effettuato in base al singolo protocollo originante le route o alla singola route statica, manualmente configurata sull'ASBR.

2.4. Protocolli di Routing Inter-AS

Nel contesto del routing inter-AS, i sistemi autonomi sono considerati delle black-box e il calcolo del percorso migliore non necessariamente implica il percorso più breve, anzi, molto spesso il percorso ottimo dipende dalle politiche interne all'AS. Exterior

Gateway Protocol (EGP), che era anche il nome del primo protocollo di questa famiglia, divenne presto obsoleto per lasciare il posto al nuovo, e di fatto unico standard [28], il *Border Gateway Protocol* (BGP) [29][29]. Gli ISP offrono un servizio di connettività previo pagamento e uno degli scopi di BGP è permettere la definizione di policy da parte degli AS, precedentemente non previste da EGP. BGP calcola, attraverso un processo decisionale, il miglior percorso inter-AS per una data destinazione, tenendo conto degli annunci ricevuti, delle politiche amministrative interne e ignorando il trattamento del traffico nell'AS stesso. BGP viene considerato un protocollo di tipo *path vector* [29].

Architettura di BGP

Formalmente una "route" [29], *è definita come un'unità di informazione che associa un insieme di destinazioni agli attributi di un percorso per quelle destinazioni. Queste ultime sono le reti i cui indirizzi IP sono compresi in un prefisso del campo NLRI di ogni annuncio BGP, e anche il percorso è un campo specificato nel medesimo messaggio.*

Una Network Layer Reachability Information (NLRI) [29] *è la lista di coppie <length, IP prefix> delle destinazioni annunciate per quella route.*

In BGP, l'annuncio delle route avviene attraverso la comunicazione tra peer. Esso utilizza TCP sulla porta 179 per stabilire connessioni affidabili e gli indirizzi IP dei vicini vengono impostati manualmente in fase di configurazione su ogni router. Ciascuno di questi possiede una tabella che include tutte le informazioni ricevute e si parla di full-mesh di connessioni BGP tra router di bordo di uno stesso AS, che possono essere ottenute adottando tecnologie di tunnelling di TCP su MPLS.

Adottando il Border Gateway Protocol, i router non necessariamente hanno next-hop direttamente connessi. IGP è utilizzato proprio per "risolvere" questi next-hop. Infatti, la differenza fondamentale rispetto agli altri protocolli di routing, è proprio che i peer BGP non si "riconoscono" automaticamente e che quindi l'interazione deve essere ben definita da management.

Gli annunci scambiati tra router di bordo di uno stesso sistema autonomo rientrano nell'internal BGP, o *iBGP*. Quelli scambiati tra peer di AS distinti sono classificati come external BGP, o *eBGP*. Inoltre, si ricorda che sia N il numero di nodi iBGP, ognuno di essi esegue N - 1 connessioni con i vicini per instaurare sessioni bidirezionali. Per cui, in totale si avranno N (N – 1) / 2 sessioni iBGP distinte [Figura 10.].

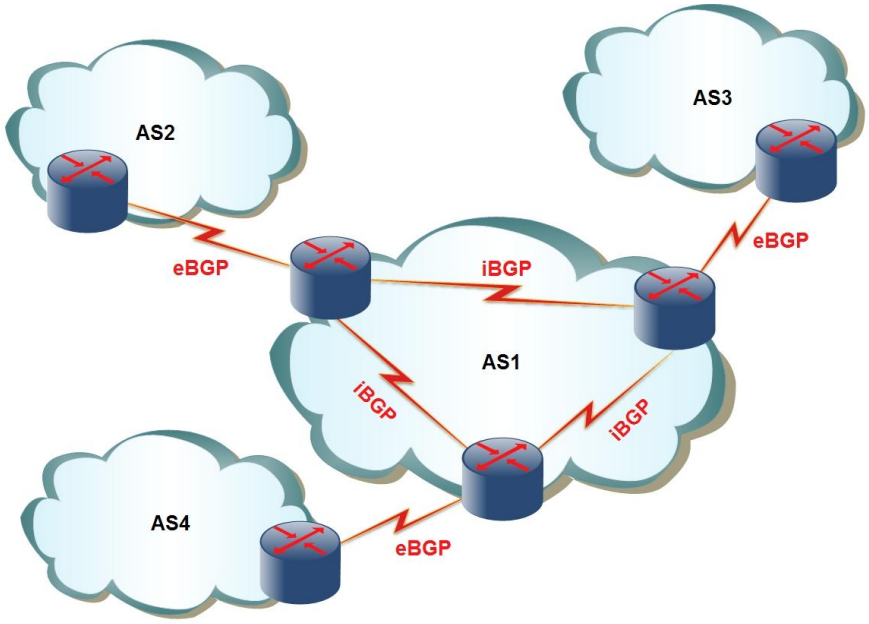

Figura 10. Connessioni eBGP e iBGP

Tipologie di messaggi BGP

Quando per la prima volta un processo BGP viene avviato, esso stabilisce una connessione con gli altri peer precedentemente configurati, con cui instaura un rapporto di vicinato. Per prima cosa riceve le routing table dagli stessi vicini con cui, successivamente, scambierà solo aggiornamenti minori.

(a) Open

Byte:	16	2	1	1	2	2	4	1	7
	Marker	Length	Type	Version	AS	Hold Time	BGP ID	Optional Length	Optional

(b) Update

Byte:	16	2	1	2	variabile	2	variabile	variabile
	Marker	Length	Type	Unfeasible Routes Length	Withdrawn Routes	Attribute Length	Attributes	NLRI

(c) Notification

Byte:	16	2	1	1	1	variable
	Marker	Length	Type	Error Code	Error SubCode	Diagnostic Data

(d) KeepAlive

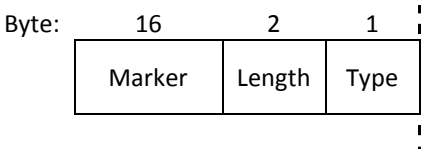

Tabella 8. Messaggi BGP (a) Open (b) Update (c) Notification (d) KeepAlive

Esistono diverse tipologie di messaggi che possono essere scambiati tra router BGP:

1. *Open* (type 1) [Tabella 8.a]: responsabile dell'instaurazione della sessione, è il primo messaggio inviato dopo che la connessione TCP è stata stabilita. Se il messaggio Open (spedito da entrambi i peer) risulta accettabile, viene scambiato successivamente un KeepAlive. Tra i campi più significativi troviamo:

   - AS: indica il numero dell'AS mittente;
   - BGP ID: è l'identificativo del router e corrisponde a un indirizzo IP impostato per quel peer. Esso è solitamente l'indirizzo IP di loopback più alto;

2. *Update* (type 2) [Tabella 8.b]: lo scopo di questi messaggi è annunciare le informazioni di routing tra peer BGP, utili per tracciare il grafo che descrive le relazioni tra i vari AS. Questa tipologia di messaggio si occupa della cancellazione di una route (router spento) o della comunicazione di una nuova route. I messaggi di update contengono principalmente i campi:

   - *Network Layer Reachability Information* (NLRI): è la lista di coppie *<length, IP prefix>* delle destinazioni annunciate per quella route. Un messaggio di update distribuisce una sola route ma può includere molti prefissi di destinazione.

     Ad esempio: </25, 204.149.16.128>, </23, 206.134.32>, </8, 10.0.0.0>;

   - *Withdrawn Routes*: è la lista di coppie *<length, IP prefix>* precedentemente annunciate e non più raggiungibili. In alternativa altre route possono essere cancellate se il peer BGP che aveva creato l'annuncio risulta ora irraggiungibile;

   - *Attributes:* è un insieme di attributi, utili a descrivere la route. Si divide in:

     - *Origin*: descrive l'origine della NLRI, può essere di tipo IGP (type 0), EGP (type 1) o altro (type 2) ed è compilato dal peer BGP che specifica la NLRI. Questo attributo era stato introdotto inizialmente come

informazione aggiuntiva di natura locale non modificabile, utile a specificare l'origine dell'NLRI annunciata, e risulta ormai deprecato;

- *AS-Path* [Figura 11.]: elenca gli AS da attraversare per raggiungere la destinazione. Esso non è modificato se l'annuncio viene propagato via iBGP mentre deve essere integrato con il numero del proprio AS, nel caso di propagazione eBGP;

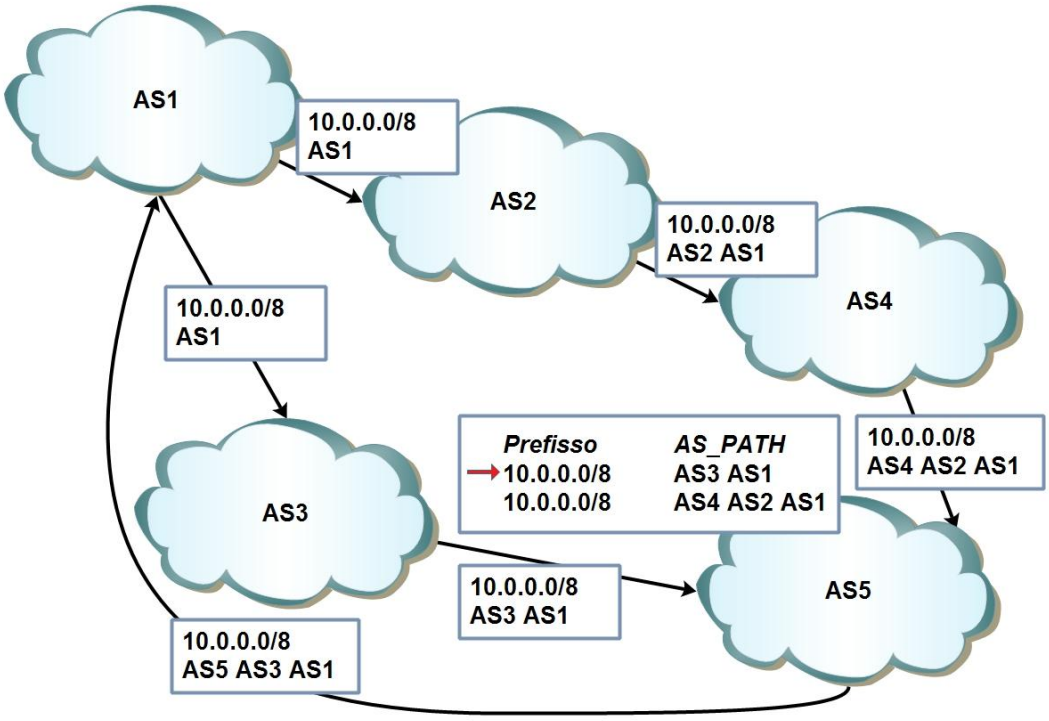

Figura 11. Attributo AS_PATH di BGP

- *Multi-Exit Discriminator* [Figura 12.]: è un parametro opzionale, non transitivo, utilizzato nel caso si abbiano punti d'ingresso multipli verso un AS vicino. Esso vuole indicare qual è il router di bordo dell'AS mittente più vicino alla destinazione, in quanto si basa su metriche scoperte via IGP e non viene propagato ad altri AS (ma reso noto solo all'interno dell'AS ricevente). Eventualmente, valori di MED diversi vengono pubblicizzati su link diversi in base alla policy dell'AS mittente, per forzare la scelta di un ingresso all'AS rispetto ad un altro. Vanno sempre preferite le route con MED minore;

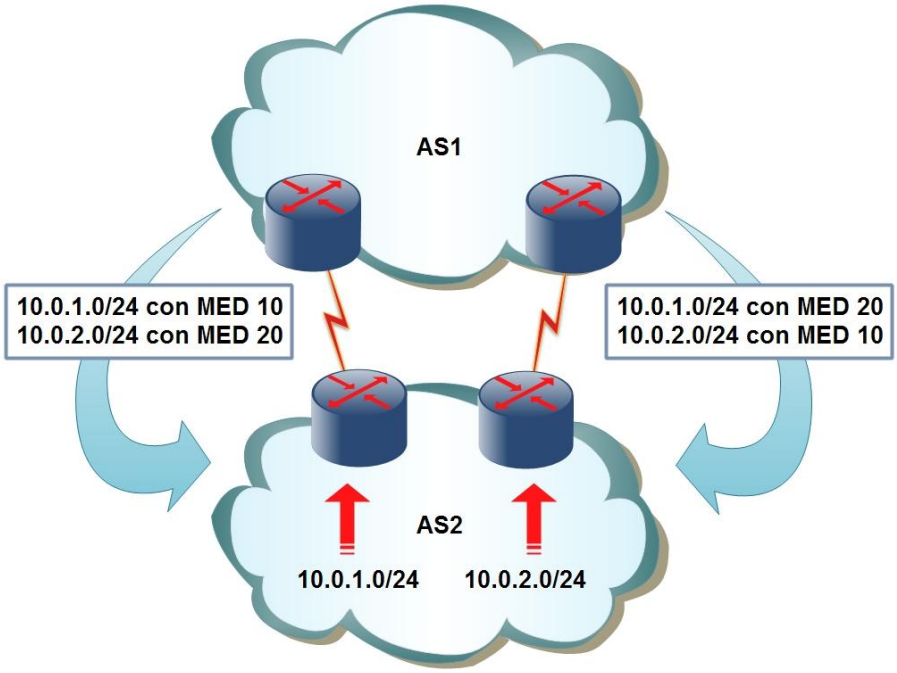

Figura 12. Attributo MED di BGP

- *Local-Pref* [Figura 13.]: è obbligatorio e presente solo negli annunci interni, mai propagato all'esterno. Esprime il grado di preferenza del proprio AS nell'utilizzo della route. Se la stessa route viene annunciata da peer iBGP con Local-Pref differente, viene preferita quella con valore più alto;

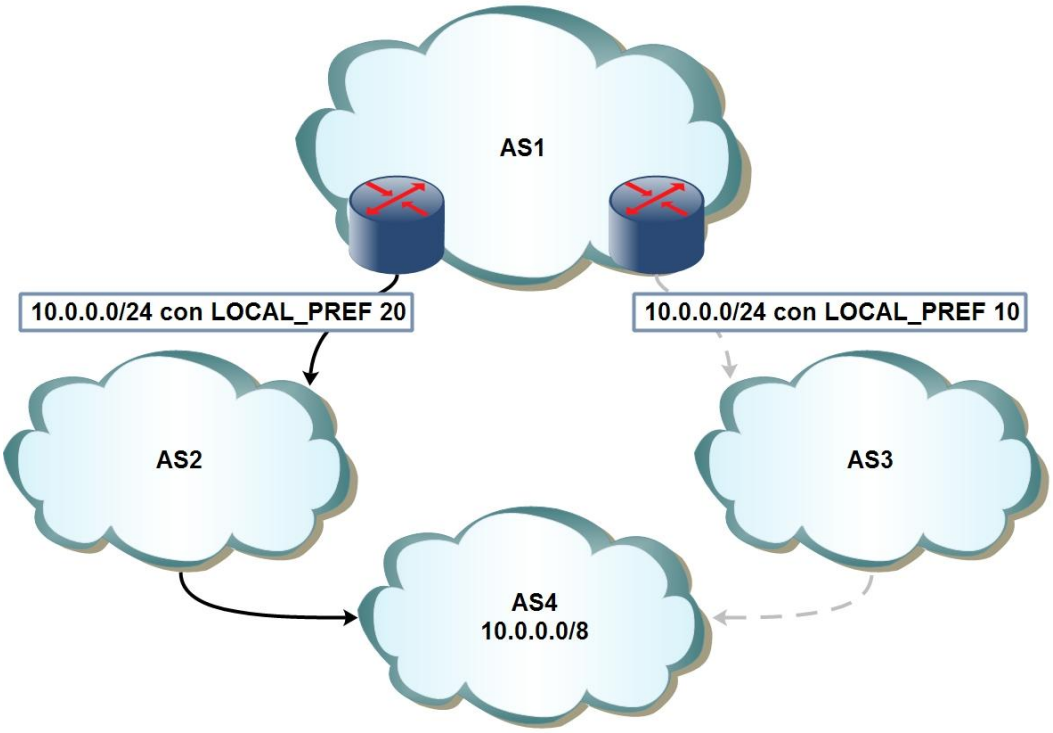

Figura 13. Attributo LOCAL_PREF di BGP

- *Next-Hop*: indirizzo IP del successivo peer BGP che deve essere preso in considerazione per raggiungere le destinazioni contenute nella NLRI;

3. *Notification* (type 3) [Tabella 8.c]: per la comunicazioni di eventuali messaggi di errore o di notifica;

4. *KeepAlive* (type 4) [Tabella 8.d]: ogni 60 secondi ciascun router invia questo messaggio ai propri vicini per segnalare lo stato della propria attività. Nel caso nessun messaggio di questo tipo sia ricevuto, la connessione verso quel determinato peer BGP inattivo viene chiusa.

La routing table di un peer BGP è logicamente suddivisa in:

- *Adj-RIBs-In*: contiene tutte le informazioni acquisite dagli annunci ricevuti. Può essere visto come il database delle informazioni da mandare in ingresso al processo decisionale;

- *Loc-RIB*: contiene le vere e proprie informazioni usate per l'instradamento, selezionate mediante il processo decisionale e le policy adottate. Un'entry per la Loc-RIB è, in generale, così costituita:

  - *Destinazione*: <length, IP prefix>;

- *BGP Next-Hop*: può essere all'interno dello stesso AS nel caso si tratti di iBGP mentre può essere all'esterno se i due peer sono connessi tramite eBGP;
- *Metric*: conosciuto in letteratura come MED;
- *Local-Pref*;
- *AS-Path*;

- *Adj-RIBs-Out*: contiene le informazioni da annunciare. Non sono necessariamente tutte quelle contenute nella Loc-RIB, in quanto possono essere state selezionate da eventuali filtri in uscita.

Nella configurazione delle politiche è indispensabile tener presente il costo (monetario) di transito verso i vari AS vicini.

Le politiche vengono attuate con l'applicazione di appositi filtri. Essi possono essere specificati in fase di progettazione e possono essere:

- Filtro in entrata: specifica quali route possono essere accettate dai router di bordo fra tutte quelle ricevute dagli altri AS ad esso collegati;
- Filtro in uscita: specifica quali route possono essere annunciate ad un vicino AS.

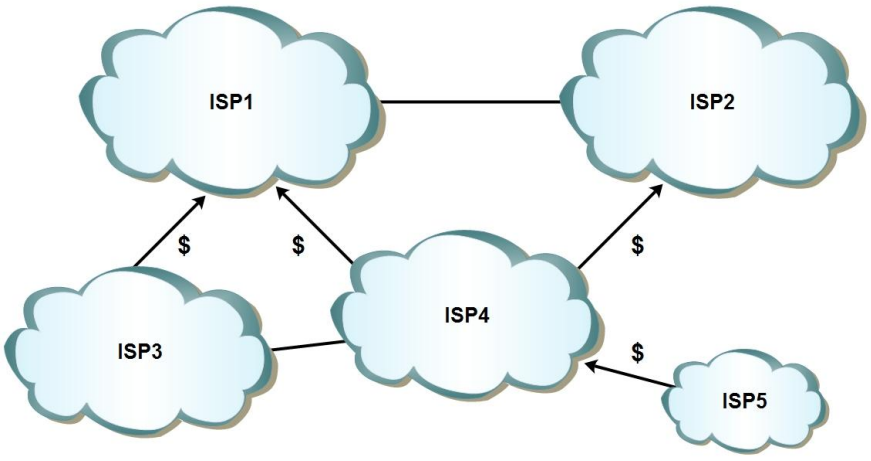

Figura 14. Esempio di interazione tra ISP

Nell'esempio in [Figura 14.], si suppone che ISP1 e ISP2 siano provider per ISP3 e ISP4 e a che a sua volta ISP4 sia provider per ISP5. I filtri per ISP4 [Tabella 9.a] e ISP5 [Tabella 9.b], potrebbero essere così impostati:

(a) **ISP4 import:**

```
Import: from ISP3 accept ISP3
Import: from ISP5 accept ISP5
Import: from ISP1 accept ANY
Import: from ISP2 accept ANY
```

ISP4 export:

```
Export: to ISP3 announce ISP4 ISP5
Export: to ISP5 announce ANY
Export: to ISP1 announce ISP4 ISP5
Export: to ISP2 announce ISP4 ISP5
```

(b) **ISP5 import:**

```
Import: from ISP4 accept ANY
```

ISP5 export:

```
Export: to ISP4 announce ISP5
```

Tabella 9. Esempio di configurazione dei filtri in entrata/uscita di (a) ISP41 (b) ISP5

Vediamo un possibile esempio [Figura 15.] di policy per la seguente configurazione di due Autonomous System, collegati attraverso un link principale (preferito) ed uno di backup. Tali politiche possono essere definite, per AS1 [Tabella 10.a] e [Tabella 10.b] AS2, in un linguaggio del tutto analogo al Routing Policy Specification Language (RPSL) [30], comunemente utilizzato dagli ISP. I valori di Local-Pref vengono aggiunti dall'AS ricevente, subito dopo l'importazione delle route ricevute.

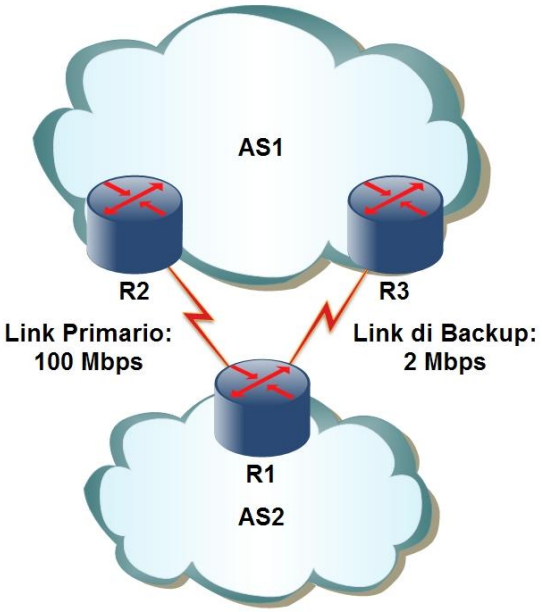

Figura 15. AS multihomed (AS2) con una connessione primaria e una di backup

(a) Aut-num: AS1

```
Import:
       from AS2 R1 at R2 set local-pref=200
       from AS2 R1 at R3 set local-pref=100
       accept AS2
Export:
       to AS2 R1 at R2 announce ANY
       to AS1 R1 at R3 announce ANY
```

(b) Aut-num: AS2

```
Import:
       from AS1 R2 at R1 set local-pref=200
       from AS1 R3 at R1 set local-pref=100
       accept ANY
Export:
       to AS1 R2 at R1 announce AS2
       to AS1 R3 at R1 announce AS2
```

Tabella 10. Configurazione dei filtri in entrata/uscita di (a) AS1 (b) AS2

Funzionamento di BGP

La seguente procedura è una sintesi dell'*algoritmo di funzionamento di BGP*, ed è eseguita non appena un router di bordo BGP riceve un messaggio di update da un vicino. Consiste dei seguenti tre passi, per ognuno dei quali vengono specificati ulteriori sotto-casi:

1. *Processo Decisionale*: può essere formalizzato come una funzione che prende in ingresso gli attributi ricevuti per una certa route e che restituisce o un intero non negativo che denota il grado di preferenza o un valore indicante l'esclusione di quella route dalla Loc-RIB. Esso è richiesto nei seguenti casi di ricezione di un annuncio:

   - Se la stessa destinazione si trova nella Adj-RIBs-In con attributi diversi, allora la nuova route (in quanto trasporta nuovi attributi di raggiungibilità) rimpiazza la vecchia e viene eseguito il processo decisionale;

   - Se la destinazione non è presente nella Adj-RIBs-In, allora la nuova route è aggiunta e viene eseguito il processo decisionale;

   - Se la destinazione ricevuta è meno specifica di una esistente nell'Adj-RIBs-In, allora viene eseguito il processo decisionale.

   Alcuni dei fattori secondo cui l'algoritmo di decisione sceglie un'unica route per diverse destinazioni sono impostati manualmente dagli amministratori BGP, sono detti politiche o policy e sono contenuti nell'apposita Policy Information Base (PIB).

La scelta del percorso migliore è compiuta esaminando di volta in volta le caratteristiche delle route ricevute, utilizzando nell'ordine i seguenti criteri, fino a quando non si ottiene una sola route per prefisso di destinazione:

1. scegli route con valore Local-Pref più alto;
2. scegli route con AS-Path più corto: questa decisione tenta di selezionare il percorso migliore. La conseguenza è che le route risultano generalmente limitate a 3-5 AS con convergenza in almeno un ISP di livello 1;
3. scegli route con MED più basso (facendo riferimento al caso che una stessa destinazione sia raggiungibile attraverso più punti d'entrata allo stesso AS);
4. le route apprese tramite eBGP vanno preferite a quelle ottenute tramite iBGP: si introduce questa regola per seguire il principio di base secondo il quale è maggiormente conveniente inviare il traffico non destinato al proprio AS, agli AS esterni il prima possibile;
5. scegli route con il più vicino (costo IGP minore) next-hop BGP: analogamente al precedente caso, questo cerca di assicurare che il traffico esca al più presto dall'AS;
6. Tie-breaking rule: seleziona la route appresa dal router con ID più basso, questo garantisce che la route selezionata sia unica.

2. *Update*: nel caso di aggiunta di una nuova route o eliminazione di una route non più raggiungibile, la Loc-RIB viene aggiornata;
3. *Propagazione*: le nuove informazioni elaborate dal router vengono aggiunte alla Adj-RIBs-Out e propagate ai vicini sotto forma di messaggio di update in base ai filtri in uscita impostati precedentemente per questa tabella. I peer destinatari, a loro volta, eseguiranno l'algoritmo non appena il messaggio sarà ricevuto.

L'algoritmo di funzionamento di BGP è riassunto in [Figura 16.].

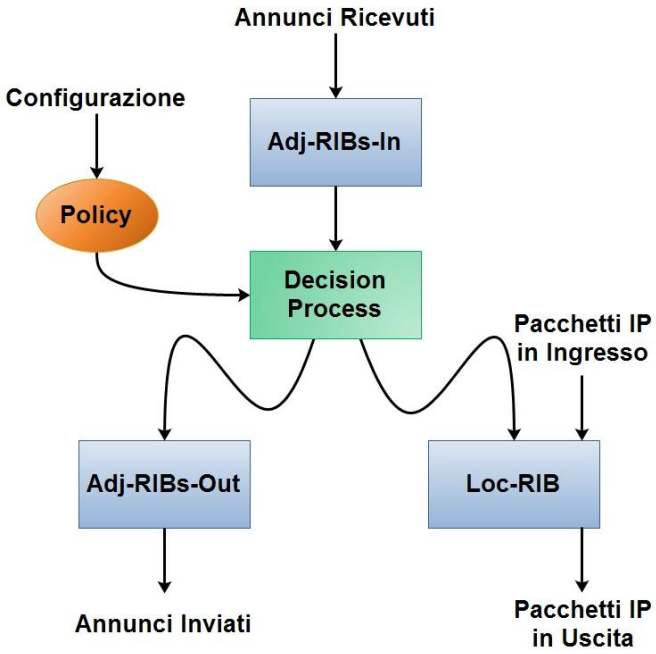

Figura 16. Processo decisionale di BGP

E' indispensabile precisare che BGP opera in sincronizzazione con IGP, in quanto esso non annuncia una route se questa prima non è nota attraverso, ad esempio, OSPF. Inoltre, i router che eseguono IGP devono sapere dove instradare i pacchetti con destinazioni BGP, cioè devono saper come raggiungere gli indirizzi IP relativi ai next-hop BGP.

Load sharing

In OSPF è possibile parlare di *Equal Cost Multi Path* (ECMP) [26], nel caso in cui una destinazione sia raggiungibile attraverso più percorsi, tutti di costo minimo. In questo caso, si può attuare il load balancing in quanto i dati possono essere equamente suddivisi tra i due o più cammini disponibili. Eseguire un partizionamento dei flussi in uscita ad un AS multihomed con BGP, invece, non risulta così automatizzato. Infatti, nessuna tipologia di load sharing è esplicitamente prevista dallo standard in materia, anche se è lecito pensare che esso sia implementabile attraverso un'appropriato "fine tuning" degli attributi ricevuti.

A tal proposito, esiste una tecnica denominata “path prepending” [31], che permette di preferire una certa route con percorso più lungo rispetto a quella teoricamente migliore. Questo è possibile aggiungendo in coda all’attributo AS-Path dell’annuncio ricevuto, un numero arbitrario di volte l’ultimo AS della lista. Ciò viene effettuato dal router di bordo ricevente l’annuncio stesso.

Questo ragionamento vale per il traffico in uscita da un AS in quanto esso rappresenta, nella pratica, il valore fatturabile a carico degli AS che forniscono contenuti (i quali pagano il transito verso AS di livello superiore). Tecniche per il load sharing in ingresso, adottate più comunemente dagli ISP di primo e secondo livello, risultano per forza di cose meno precise, in quanto dipendono principalmente dal comportamento degli AS in upstream.

Per facilitare un così complicato meccanismo per il partizionamento dei flussi (*load sharing*), vendor specifici implementano soluzioni proprietarie diverse. Ad esempio, Cisco Systems [33] propone per i suoi router il cosiddetto “BGP Multipath” [32] ovvero l’installazione nelle proprie tabelle BGP, di percorsi multipli per la stessa destinazione. Questi ultimi non sono pensati in sostituzione alla route risultante dal processo decisionale, ma in aggiunta, e contribuiscono al bilanciamento del carico.

2.5. Interazione e Redistribuzione delle Route

E’ necessario capire come avviene la redistribuzione OSPF-BGP, così come è definita dalla RFC 1403.

L’esportazione delle route OSPF in BGP è il caso generale in cui un dato prefisso IGP, appartenente all’AS di riferimento, deve essere annunciato esternamente via BGP:

1. Di default, tale esportazione non avviene ma deve essere esplicitata dall’amministratore. Le route esterne di tipo 1 e 2 non vengono di norma mai ridistribuite. L’esportazione delle route desiderate deve essere impostata dall’amministratore secondo un appropriato meccanismo di filtri;

2. Le route esportate da OSPF devono essere raggiungibili. Nel caso le route diventino irraggiungibili in seguito, la propagazione di questa nuova informazione deve avvenire immediatamente;

3. Di default, la metrica IGP può essere tradotta con l’attributo MED di BGP. Ciò deve essere esplicitamente gestito dall’amministratore, in quanto essa rappresenta un attributo opzionale in ambito BGP;

4. L'integrazione di BGP e OSPF su un ASBR deve prevedere il minor tempo possibile tra l'apprendimento di una nuova route via OSPF e il successivo annuncio della route stessa via eBGP. Di default questo valore è impostato per avere una propagazione immediata.

L'importazione delle route BGP in OSPF è il caso in cui una route esterna annunciata via BGP deve essere resa nota ai router OSPF interni all'AS di riferimento:

1. Di default, questa importazione non avviene ma deve essere esplicitata da management. L'importazione delle route desiderate è eseguita dall'amministratore verificando di volta in volta le route da ridistribuire;

2. Inoltre, l'amministratore deve essere in grado di configurare il costo e il tipo di metrica per ogni route esterna importata in OSPF. Di default, il costo è 1 e la tipologia è external route type 2 per tutte le route importate. Nel caso in cui tutte le route esterne siano di quest'ultima tipologia, viene utilizzato come discriminante il costo interno associato;

3. Route apprese via iBGP non devono essere importate in OSPF;

4. Un ASBR non deve mai generare una default route per il dominio OSPF, a meno che non sia esplicitamente configurato per per farlo.

Oltre al meccanismo di esportazione, la RFC specifica anche la corrispondenza tra i campi <External_Route-cost, External_Route-Tag, IGP_Next-Hop> di OSPF rispettivamente con gli attributi BGP: <MED, Origin/AS_Path, BGP_Next-Hop>. LA RFC 1403 dà un'idea di come il ruolo dell'amministratore risulti cruciale per l'importazione/esportazione delle route e di come questo sia scarsamente automatizzato nella pratica. Si rimanda allo standard per ulteriori dettagli tecnici.

3. Multi Protocol Label Switching

Multi Protocol Label Switching (MPLS) [1] è una tecnologia di rete che utilizza un meccanismo di inoltro basato sulle etichette, o *label*. Il meccanismo delle etichette fu originariamente introdotto dai protocolli Frame Relay [8] e ATM [7]. Entrambi fanno riferimento ad un *Virtual Circuit* (VC) in cui, ad ogni hop attraverso la rete, un'etichetta viene opportunamente modificata. Con l'espandersi di Internet, il protocollo IP divenne estremamente popolare ma l'integrazione tra questo e le diffusissime reti ATM fu sempre notevolmente complessa da gestire. Furono molte le proposte in merito, ad esempio la RFC 1483 "Multiprotocol Encapsulation over ATM Adaptation Layer 5" secondo cui, però, tutti i circuiti ATM dovevano essere stabiliti manualmente. Successivamente, si è giunti a quella che per molto tempo è stata considerata l'alternativa migliore, ovvero il Multi Protocol over ATM (MPoA), voluto e specificato dall'ATM Forum. In questo scenario, MPLS fu originariamente proposto (con scarso successo) da Ipsilon Networks e definitivamente sviluppato da Cisco Systems, che poco tempo dopo collaborò con IETF per la realizzazione di uno standard aperto.

E' necessario specificare che d'ora in poi, il termine "router" viene utilizzato, così come nel resto della letteratura sull'argomento, per ragioni storiche. E' improprio definire un "router MPLS", in quanto esso instrada sì pacchetti IP, ma gestisce anche pacchetti etichettati. Lo shim header MPLS viene aggiunto tra il pacchetto IP ed il frame di livello inferiore, e si colloca tra i livelli 2 e 3 del modello ISO/OSI.

3.1. Vantaggi di MPLS

Le principali ragioni che portano all'adozione di MPLS sono le seguenti [34]:

- *Infrastruttura di rete unificata*: quando un datagram IP entra in una rete MPLS, viene etichettato inizialmente in base all'indirizzo di destinazione e tutto il traffico all'interno della rete viene inoltrato mediante lo scambio di etichette. Inizialmente, MPLS era stato pensato per fornire una miglior integrazione di IP su ATM, anche se con il tempo una delle sue applicazioni principali è divenuta la realizzazione di VPN [35] di livello 2, dette anche Virtual Private LAN Service (VPLS) [4]. Da questo punto di partenza, è possibile pensare a MPLS come la struttura portante nell'ambito delle reti di trasporto, per la creazione di un'infrastruttura di nuova generazione, che mantiene tutti i vantaggi di una rete a commutazione di circuito su una più flessibile rete a commutazione di pacchetto;

- *Traffic Engineering* [5]: è una tecnica di ottimizzazione del traffico, il cui scopo principale è fornire un efficiente utilizzo delle risorse e un incremento della performance di rete. L'idea principale dietro a questo concetto è quella di riuscire ad impiegare efficientemente la parte di rete non rientrante nei cammini minimi calcolati dal protocollo IGP, che risulterebbe altrimenti sottoutilizzata. Inoltre, tramite questo meccanismo è stato possibile implementare il cosiddetto *Fast ReRoute* (FRR), che permette di re-indirizzare il traffico, in caso di guasti o indisponibilità di un router, lungo un percorso alternativo, in meno di 50 ms;

- *BGP-free core*: in MPLS, un'etichetta viene solitamente associata ad un router BGP in uscita dall'AS piuttosto che all'indirizzo IP di destinazione. Pertanto, i nodi interni non necessitano più di mantenere nelle proprie tabelle di routing tutti i prefissi per le destinazioni esterne ma solo quelli interni utilizzati dal protocollo IGP. Tali router si limitano ad inoltrare i pacchetti in ingresso secondo le informazioni contenute nella label.

Uno dei primi obiettivi di MPLS era strettamente legato alla necessità di velocità. Eseguire il *longest prefix match* per i pacchetti IP, da parte della CPU, era considerato più dispendioso rispetto ad un semplice scambio di etichette. Sebbene si possa pensare che le label permettano una consultazione più veloce, i progressi tecnologici ottenuti in tale ambito rendono quest'argomentazione non più attuale [34]. In particolare, l'utilizzo di *Application-Specific Integrated Circuits* (ASIC) per la gestione

del Data Plane IP via hardware, ha permesso ai pacchetti IP di essere gestiti tanto velocemente quanto le etichette [34].

3.2. Architettura

Innanzitutto, è utile capire quali sono i componenti di MPLS [1]:

- *Dominio MPLS*: è un insieme di nodi MPLS contigui che appartengono, solitamente, anche ad uno stesso dominio amministrativo;
- *Label*: è un'etichetta di lunghezza fissa, utilizzata per identificare univocamente la FEC di appartenenza;
- *Forwarding Equivalence Class* (FEC): è una classe di equivalenza a cui appartengono tutti i pacchetti IP che: possiedono lo stesso next-hop, utilizzano la stessa interfaccia d'uscita e hanno la stessa priorità. E' pratica comune che ogni label, aggiunta all'ingresso (LER in entrata) del dominio MPLS, sia in corrispondenza biunivoca con il next-hop BGP (LER in uscita) a cui fa riferimento la sottorete di destinazione. Ne deriva che destinazioni esterne diverse, con il medesimo next-hop BGP, potrebbero subire lo stesso trattamento di inoltro all'interno del dominio MPLS e appartenere alla stessa FEC. Si può asserire che pacchetti IP etichettati nello stesso modo appartengano tutti alla stessa FEC, anche se non necessariamente vale il contrario, in quanto potrebbero avere la stessa label ma essere inoltrati con priorità diverse. Il punto cruciale di MPLS sta nel capire perché il concetto di FEC è così importante. Analogamente, nel routing IP è definibile una classe di equivalenza del tutto simile (l'insieme di next-hop, interfaccia d'uscita e priorità), ma diversamente da MPLS, essa viene determinata ad ogni hop dell'instradamento. In MPLS la FEC è elaborata solo dall'Ingress-LSR di un LSP, delegando ai router successivi solo un semplice scambio di etichette;
- *Label Switching Router* (LSR): è un nodo MPLS capace di inoltrare pacchetti etichettati. Le operazioni che può eseguire sono *pop*, *push* e *swap*. Si può considerare definito da un componente di Control Plane e da uno di Data (o Forwarding) Plane [Figura 17.]; le due parti sono logicamente indipendenti ma di fatto sono integrate all'interno di un router IP, che diventa IP/MPLS;

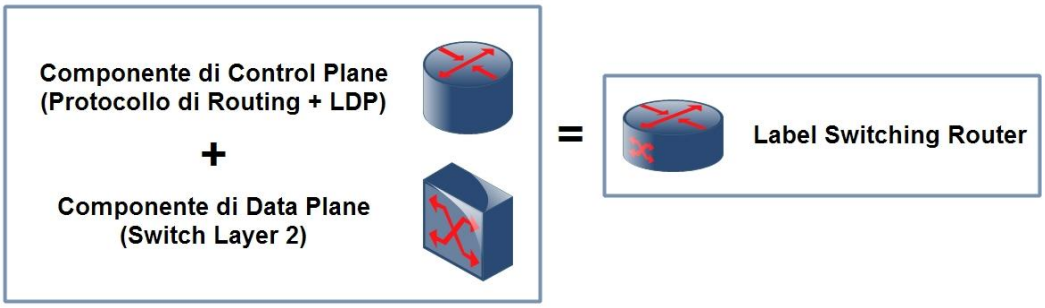

Figura 17. Componenti di un LSR

- *Label Edge Router* (LER): è un LSR che opera ai margini del dominio MPLS. Esso riceve un pacchetto dall'esterno, vi aggiunge un'etichetta basandosi sulla FEC più appropriata, e lo inoltra al successivo LSR del suo percorso;
- *Label Switching Path* (LSP): è un percorso attraverso uno o più LSR. Esso inizia da un nodo d'ingresso che appone un'etichetta, prosegue per uno o più LSR intermedi in cui avviene lo scambio di label, e termina su un nodo d'uscita che toglie la label e instrada il pacchetto IP al successivo next-hop;
- *Ingress-LSR*: è il ruolo che assume un LSR in entrata ad un LSP;
- *Egress-LSR*: è il ruolo che assume un LSR in uscita da un LSP;
- *Label Distribution Protocol* (LDP): è l'insieme di procedure che un LSR deve seguire per informare il successivo, relativamente alla mappatura label/FEC che ha eseguito.

In [Figura 18.] è riportato un esempio in cui vengono collocati gli elementi costitutivi di MPLS.

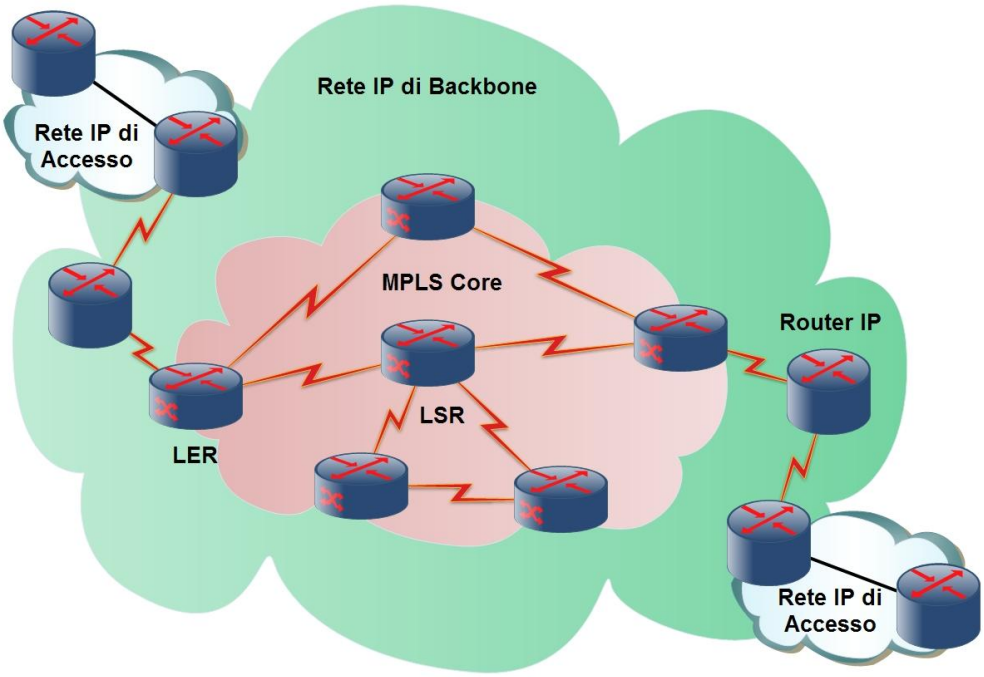

Figura 18. Esempio di rete IP/MPLS

Lo shim header MPLS [Tabella 11.] è un campo di 32 bit che presenta la seguente struttura:

- i primi 20 bit sono il valore dell'etichetta;
- i bit da 20 a 22 rappresentano 3 bit sperimentali (Exp), normalmente usati per la gestione della qualità del servizio;
- il bit 23 è il cosiddetto "Bottom of Stack" (BoS), e indica se quell'etichetta è, o meno, l'ultima della pila. MPLS prevede la possibilità di annidare un numero di etichette pari al numero di LSP incapsulati uno dentro l'altro;
- i bit da 23 a 31 sono utilizzati per il Time To Live (TTL).

bit:	20	3	1	8
	Label	Exp	BoS	TTL

Tabella 11. Shim Header MPLS

Le label possono essere distribuite in due modalità:

- eseguendo l'integrazione, o *piggyback*, su un protocollo di routing già esistente (MP-BGP) [36];
- usando un protocollo separato, che può essere ad esempio LDP [37], RSVP [38] ecc.

Un annuncio BGP può includere NLRI e label allo stesso tempo, e tale tipo di distribuzione è largamente diffusa per reti di tipo VPN (MP-BGP). Il protocollo più adottato dai vendor, però, è senza dubbio LDP, mentre RSVP è in larga parte utilizzato in MPLS-TE (vedi capitolo 4).

Per gestire le informazioni necessarie al Data Plane è necessario mantenere in ogni LSR un insieme di strutture dati, tutte collegate tra loro.

- *Routing Information Base* (RIB): è un database contenente la topologia della rete, popolato dai protocolli di routing adottati dalla rete stessa (OSPF, BGP, ecc.). Ciascuna riga della RIB presenta almeno tre campi:
  - Prefisso di destinazione;
  - Costo associato al percorso per quella destinazione;
  - Next-hop a cui inoltrare il pacchetto per raggiungere la destinazione;
- *Label Information Base* (LIB): per ogni entry della RIB viene associata una label locale (in ingresso) e una o più label remote ricevute dai propri vicini (label in uscita). Questa tabella è popolata da LDP;
- *Next-Hop Label Forwarding Entry* (NHLFE): è utilizzata per effettuare l'inoltro di un pacchetto per cui è stata trovata una corrispondenza nella FTN o nella ILM. Contiene le seguenti informazioni:
  - Next-hop;
  - Porta di uscita;
  - Label in uscita;
  - Action: l'azione da effettuare sul pacchetto. Può essere una delle seguenti:
    - *Push*: un Ingress-LSR aggiunge un'etichetta al pacchetto IP (o al label-stack, se precedenti etichette erano già presenti);

- *Swap*: un LSR scambia la label in entrata con quella in uscita;
- *Pop*: un Egress-LSR elimina un'etichetta dal pacchetto etichettato (o dal label-stack);

- *FEC-To-NHLFE* (FTN): associa ogni FEC ad un'entry della NHLFE. E' utilizzata quando un Ingress-LSR riceve un pacchetto senza shim header, che necessita di essere associato ad una FEC secondo una data regola (ad esempio, il LER/next-hop BGP di uscita dall'AS). Contiene le seguenti informazioni:
  - FEC: è un prefisso IP che identifica la classe di equivalenza;
  - Puntatore alla NHLFE;
- *Incoming Label Map* (ILM): gestisce i pacchetti che arrivano già etichettati in ingresso ad un LSR; questa tabella associa ad una label in entrata un'entry della tabella NHLFE utilizzata per l'inoltro. Contiene le seguenti informazioni:
  - Label in ingresso;
  - Porta d'ingresso;
  - Puntatore alla NHLFE;
- *Label Forwarding Information Base* (LFIB): rappresenta le informazioni necessarie per l'inoltro MPLS ed è costituita dall'insieme di FTN, ILM e NHLFE.

Nel routing IP è possibile individuare anche una cosiddetta Forwarding Information Base (FIB), analoga alla LFIB in ambito MPLS, e contenente tutte le informazioni necessarie per l'inoltro.

Riassumendo [Figura 19.], all'arrivo di un nuovo pacchetto l'LSR esamina se esso contiene uno shim header: in caso positivo, esso consulta la tabella ILM identificando l'etichetta trasportata dal pacchetto; in caso negativo, l'LSR analizza l'intestazione IP e consulta la tabella FTN cercando una FEC da associare al pacchetto. Passando attraverso la ILM o la FTN, il pacchetto viene associato ad un'entry della NHLFE che specificherà quale azione eseguire sulla label (inserirne una nuova, sostituirla o eliminarla) ed il next-hop per l'inoltro.

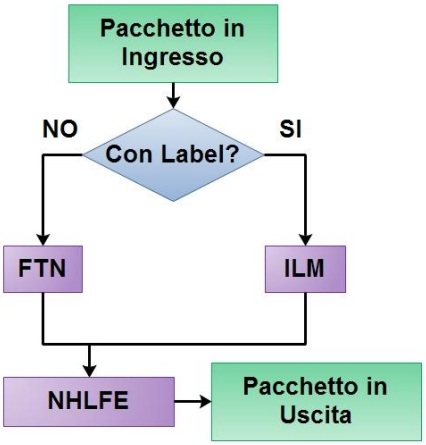

Figura 19. Inoltro di un pacchetto MPLS

3.3. Label Distribution Protocol

Nello standard RFC 3031, viene definito un protocollo di distribuzione delle etichette, un insieme di procedure grazie a cui un LSR informa un LSR vicino delle label localmente utilizzate per l'inoltro del traffico.

Partendo da questa definizione, sono stati standardizzati alcuni protocolli di distribuzione delle etichette, distinguendo i diversi casi di impiego. Il Label Distribution Protocol (LDP), trattato dalla RFC 5036, raccoglie l'insieme delle procedure necessarie ad una coppia di Ingress/Egress-LSR per stabilire un LSP. Due LSR che adottano LDP per scambiarsi informazioni sulle label sono identificati come vicini LDP mentre le informazioni vengono scambiate attraverso una cosiddetta sessione LDP.

Esistono quattro categorie di messaggi in LDP:

- *Discovery*: utilizzati per annunciare la presenza degli LSR nella rete. il meccanismo di discovery consente ad un LSR di trovare dei potenziali nodi LDP vicini. Ogni LSR invia periodicamente da ogni interfaccia un messaggio di Hello incapsulato in un pacchetto UDP alla porta 646 all'indirizzo multicast 224.0.0.2, che corrisponde a tutti i router di quella subnet. Il messaggio trasporta anche l'identificativo dell'LSR: il router-ID in ambito LDP è solitamente l'indirizzo IP più alto tra le interfacce di loopback;

- *Session*: impiegati per mantenere, stabilire e terminare una sessione tra nodi LDP. Essa consiste di due fasi:

    - Stabilire una connessione TCP sulla porta 646;

- Negoziare i parametri della sessione;

- *Advertisement*: essenziali per annunciare al vicino la creazione, modifica o cancellazione di una label;

- *Notification*: utilizzati per fornire informazioni di avviso o per segnalare condizioni di errore.

I messaggi Discovery forniscono agli LSR un meccanismo per annunciare la propria presenza nella rete. Periodicamente, vengono inviati dei messaggi Hello come pacchetti UDP all'indirizzo di multicast 224.0.0.2, porta UDP 646. Quando un LSR decide di stabilire una sessione con un vicino, esso la inizializza, sulla porta TCP 646. Se la procedura termina con successo, i due nodi LDP diventano adiacenti e può cominciare così lo scambio di messaggi. L'utilizzo di una sessione TCP è dovuto alla necessità di stabilire una comunicazione affidabile, che garantisca la consegna dei messaggi tra i due nodi; mentre per le operazioni di Discovery è sufficiente una comunicazione veloce e best-effort basata su UDP.

Per stabilire una sessione, i peer si accordano su chi deve effettuare la connessione, facendo da client, e su chi deve rimanere in ascolto, facendo da server. Rispettivamente, questi due ruoli prendono il nome di "active" e "passive". La scelta è basata sul router-ID: il router con identificativo più alto svolgerà il ruolo attivo e avvierà la connessione TCP, mentre la controparte si limiterà al ruolo passivo. Negoziati i parametri e stabilita la connessione, questa viene mantenuta attiva tramite l'invio periodico di messaggi KeepAlive. Una volta mappata un'etichetta (label in ingresso o locale) ad ogni prefisso IGP della propria tabella di routing, ogni peer LDP può annunciare ai nodi adiacenti le proprie mappature, ricevendo a sua volta quelle (label in uscita, o remote) assegnate dai vicini peer LDP per quel dato prefisso.

La nozione di "*label space*" è utile a descrivere la gestione delle etichette. Si utilizza la definizione di "*per-interface label space"* quando l'inoltro è basato sulla coppia label/interfaccia in entrata mentre più in generale, il "*per-platform label space"* associa a ciascuna piattaforma uno spazio univoco per la numerazione delle etichette, del tutto indipendente dall'interfaccia d'arrivo.

La distribuzione delle etichette dipende da chi esegue la richiesta per una certa label:

- *Downstream-on-Demand* (DoD): in cui ciascun LSR richiede la mappatura dal proprio next-hop, fino ad arrivare all'Egress-LSR;
- *Unsolicited Downstream* (UD): senza alcuna richiesta, ciascun LSR invia la propria mappatura a tutti i peer LDP vicini.

La distribuzione può essere anche:

- *Indipendente*: se è possibile inviare la label locale richiesta, per un determinato prefisso IP, immediatamente. In altre parole, è consentito annunciare una mappatura in upstream prima ancora di averla ricevuta in downstream.
- *Ordinata*: l'annuncio della mappatura richiesta è possibile, solo se si è già ricevuta la relativa mappatura remota dall'LSR in downstream. In questa modalità gli LSP vengono sempre costituiti a ritroso, dall'Egress all'Ingress LSR.

Le tipologie di conservazione delle etichette possono essere due:

- *Liberal Label Retention* (LLR): un LSR mantiene tutte le mappature remote. Questo meccanismo ben si abbina al caso in cui l'LSR riceva annunci DoD. Nel caso il link in downstream dovesse subire delle variazioni, l'LSR coinvolto può rapidamente adattarsi al cambio di topologia;
- *Conservative Label Retention* (CLR): mantiene solo le mappature effettivamente utilizzate. In una modalità UD, tutti i nodi LDP ricevono le label remote per tutti i prefissi noti della rete, e ciò potrebbe popolare inutilmente la LIB. Adottando il CLR si ha quindi il vantaggio di una maggior memoria disponibile ma di una peggiore capacità di adattamento ai cambiamenti della rete.

In [Figura 20.] si può vedere un esempio di distribuzione LDP, ricordando che ad ogni prefisso IGP (in questo caso solamente 10.0.0.0) di ogni LSR corrisponde un'etichetta e supponendo che R4 sia un LER:

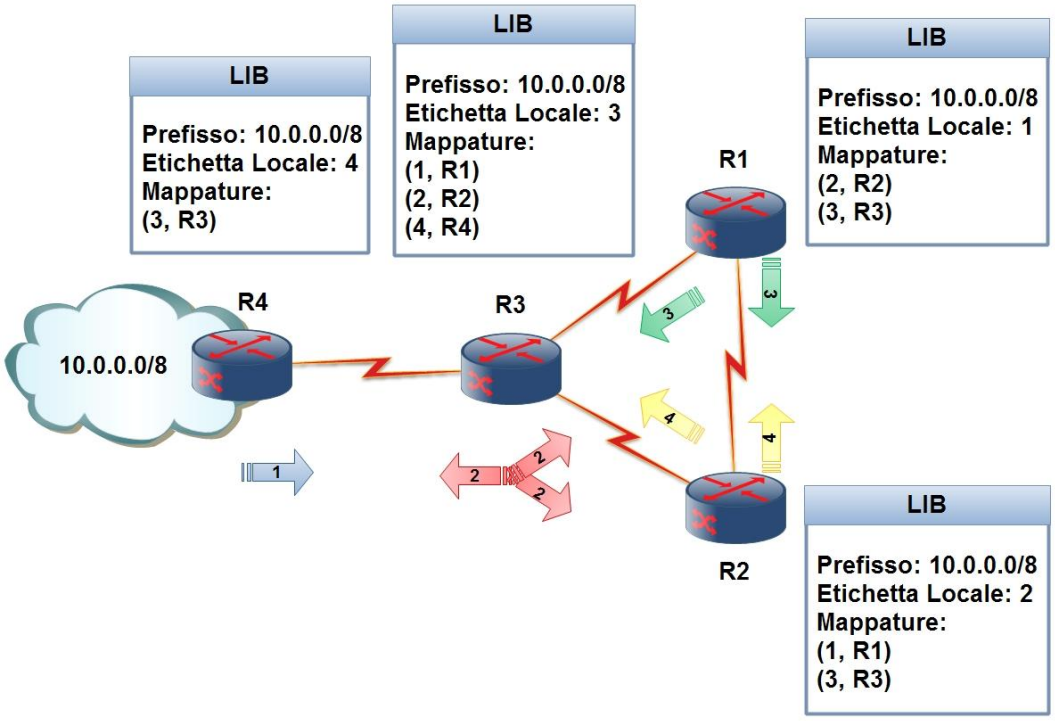

Figura 20. Distribuzione delle etichette con LDP (caso generale)

1. R4 sceglie la label locale 4 per il prefisso 10.0.0.0/8 e la annuncia ai proprio vicino (R3);
2. R3 sceglie la label locale 3 per il prefisso 10.0.0.0/8 e la annuncia ai propri vicini (R1, R2, R4);
3. R1 sceglie la label locale 1 per il prefisso 10.0.0.0/8 e la annuncia ai propri vicini (R2, R3);
4. R2 sceglie la label locale 2 per il prefisso 10.0.0.0/8 e la annuncia ai propri vicini (R1, R3);
5. La LIB di R1 mantiene le proprie mappature locali e quelle remote (1, R1; 3, R3) ricevute dai vicini per il prefisso 10.0.0.0/8;
6. La LIB di R2 mantiene le proprie mappature locali e quelle remote (2, R2; 3, R3) ricevute dai vicini per il prefisso 10.0.0.0/8;
7. La LIB di R3 mantiene le proprie mappature locali e quelle remote (1, R1; 2, R2; 4, R4) ricevute dai vicini per il prefisso 10.0.0.0/8;

8. La LIB di R4 mantiene le proprie mappature locali e quelle remote (3, R3) ricevute dal vicino per il prefisso 10.0.0.0/8.

Volendo instradare un pacchetto verso la rete 10.0.0.0/8, R1 (o R2) etichetterà (eseguendo un push) un pacchetto con label 3 e lo inoltrerà verso R3. Esso scambierà (con uno swap) la label 3 in ingresso con la label 4 in uscita per R4 e quest'ultimo eliminerà (effettuerà un pop) l'etichetta, per instradare il pacchetto alla corretta destinazione.

Ci sono situazioni in cui pacchetti appartenenti a differenti FEC vengono inoltrati tutti nello stesso LSP, arrivando al relativo Egress-LSR, per poi proseguire verso le rispettive destinazioni. In tal caso, l'instradamento/inoltro efficiente può essere ottenuto adottando un'unica label per tutte le precedenti FEC, identificando così un singolo punto d'uscita per tutti quei pacchetti. Quindi, non è più necessario mantenere etichette distinte per ogni FEC e si può adottare la tecnica chiamata "*Egress-Targeted Label Assignment"* (ETLA) [1]. Chiamando Ri ed Re rispettivamente l'lngress e l'Egress-LSR di un LSP, ciò è possibile se e solo se:

1. L'indirizzo dell'LSR Re è presente nella RIB di Ri, come "host route", ovvero come indirizzo IP specifico per Re, non come generico prefisso della sottorete associata;

2. Esiste un modo per Ri di determinare che Re è l'egress-LSR per tutti i pacchetti di un dato insieme di FEC.

Ri può determinare che un dato LSR Re è il punto d'uscita per un insieme di FEC in diversi modi:

- Se la rete adotta un protocollo IGP e tutti i nodi dell'area supportano MPLS, allora l'algoritmo di routing fornisce sufficienti informazioni per determinare il percorso di costo minimo che i pacchetti di una determinata FEC devono attraversare per abbandonare il dominio di routing o una sua sottoarea;

- Se la rete adotta BGP, Ri deve essere in grado di determinare che i pacchetti di una determinata FEC devono passare tutti per lo stesso next-hop BGP per abbandonare l'AS;

- E' possibile, inoltre, utilizzare LDP per fornire informazioni su quali prefissi IP siano "associati" ad un determinato Egress-LSR/next-hop BGP.

Utilizzando l'ETLA il numero di label necessarie per il routing nel dominio MPLS risulta notevolmente ridotto.

Funzionamento di MPLS

In un LSP, utilizzeremo il termine "*downstream*" per indicare tutti i nodi successivi ad un LSR di riferimento, fino ad arrivare all'Egress-LSR; useremo, invece, il termine "*upstream*" per individuare tutti i nodi precedenti ad un dato LSR, fino ad arrivare all'Ingress-LSR.

Si vuole riportare l'*algoritmo di funzionamento di MPLS*, riassumendo i passi fondamentali di una rete IP/MPLS (integrata con IGP/BGP) così come sono stati introdotti in questo capitolo:

1. All'interno della rete viene eseguito un dato protocollo IGP (ad esempio OSPF), necessario a popolare la RIB di ciascun LSR con gli indirizzi IP dei vicini;

2. Per ogni prefisso IGP (FEC) presente nella RIB, ogni LSR mappa un'etichetta locale ad ogni entry della RIB. Ciascuna *mappatura locale* è resa nota ai propri vicini. Allo stesso tempo sono ricevute le *mappature remote* annunciate dai vicini stessi. Nella LIB di ogni LSR viene eseguita l'associazione tra label locali e remote ricevute per ogni prefisso IGP;

3. Per un LER, il procedimento è del tutto analogo ma in più (abilitando l' ETLA) la sua tabella di routing è integrata con le informazioni ricevute via BGP dagli altri LER. In tal modo, se un pacchetto non etichettato arriva ad un router di bordo, esso analizza l'indirizzo IP di destinazione, ne individua la relativa sottorete, lo etichetta e lo inoltra così nell'LSP relativo al LER/next-hop BGP d'uscita.

4. In base a FTN, ILM e NHLFE, ogni router ha a disposizione tutte le informazioni per l'inoltro, quali: label locale, label in uscita, FEC corrispondente, operazione da eseguire e interfaccia d'uscita. *Un generico LSR, durante il suo normale funzionamento*:

    a. Se riceve un pacchetto non etichettato, determina a quale FEC appartiene l'indirizzo IP di destinazione e consulta la FTN, da cui viene indirizzato alla NHLFE. Da essa estrapola l'operazione da eseguire (push), viene aggiunta l'etichetta, e inoltra il pacchetto etichettato al successivo LSR dell'LSP;

    b. Se riceve un pacchetto etichettato, consulta la ILM dalla quale viene indirizzato alla NHLFE:

        i. Se l'operazione da eseguire è uno swap, le etichette vengono scambiate, ed inoltra il pacchetto con la nuova label all'LSR successivo;

ii. Se l'operazione da eseguire è un pop, l'etichetta viene tolta, ed esegue un longest prefix match sull'indirizzo IP di destinazione. Il pacchetto viene gestito come nel routing IP tradizionale ed instradato.

L'algoritmo di funzionamento di MPLS è riassunto dal diagramma in [Figura 21.]:

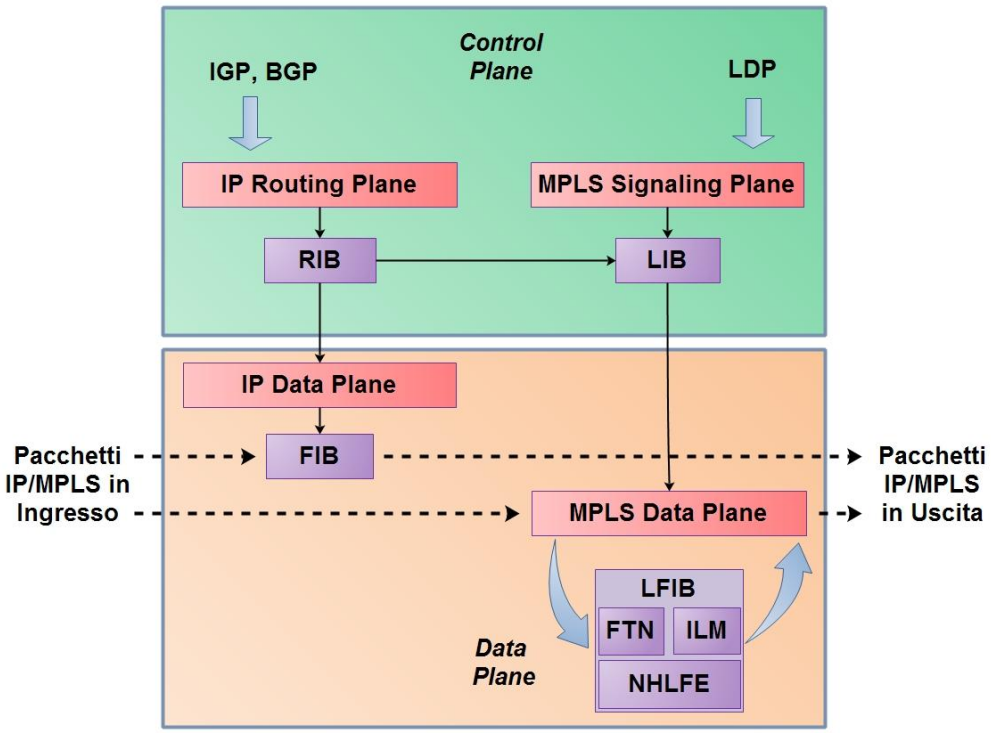

Figura 21. Interazione tra Control e Data Plane in IP/MPLS

Non è necessario che un LSR conosca il payload MPLS, è sufficiente che in generale sia effettuato solo lo swap delle etichette. Al contrario conoscere la natura del payload si rende indispensabile per gli Egress-LSR che inoltrano il frame in uscita dalla rete

Presentiamo ora un esempio pratico di funzionamento di una rete IP/MPLS, i cui LER sono anche peer BGP, e al cui interno sono implementati OSPF e LDP:

1. Un nuovo sito esterno viene aggiunto alla rete e collegato a R2 [Figura 22.]. Esso rappresenta un dominio amministrativo diverso da quello del provider. R1 annuncia via eBGP la propria sottorete a R2, LER del dominio MPLS, specificando che per la sottorete 1.0.0.0/8, R1 stesso rappresenta il next-hop BGP;

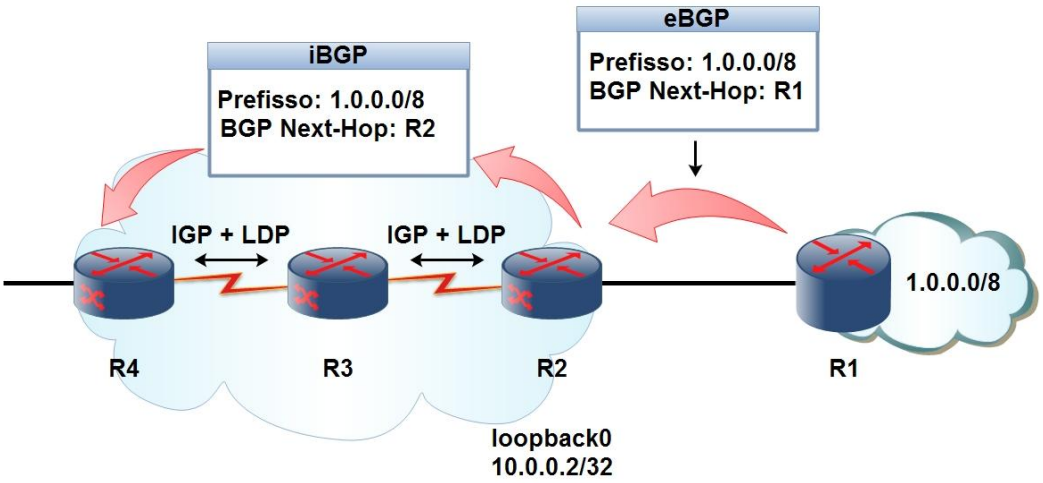

Figura 22. Esempio di annuncio BGP relativo ad un nuovo sito

2. Ricevuto il messaggio da R1, R2 esegue a sua volta un annuncio iBGP a tutti i vicini del suo Autonomous System, in questo caso solo R4 [Figura 23.], indicando che per il prefisso 1.0.0.0/8 esso rappresenta il nuovo next-hop BGP. Ciò è reso possibile dal fatto che all'interno dell'AS la topologia di rete è già nota grazie ad un protocollo IGP attivo a livello IP, e che l'annuncio stesso utilizza la connessione via TCP tra LER, di norma, incapsulata nel tunnel MPLS. Supponiamo di aver configurato un'interfaccia di loopback per R2, con indirizzo 10.0.0.2. La rete su questa interfaccia virtuale, che rappresenta in realtà il prefisso di un host (R2) la cui sottomaschera è semplicemente 255.255.255.255, viene annunciato via OSPF. A questo punto, sia R3 che R4 hanno un'entry nella loro RIB per questa destinazione. R2 esegue una distribuzione della sua etichetta locale 2 relativa al prefisso 10.0.0.2/32 ai propri vicini, con modalità UD. 10.0.0.2 è anche il router-ID di R2 per i protocolli: OSPF, LDP e BGP. Le mappature remote sono ricevute di conseguenza;

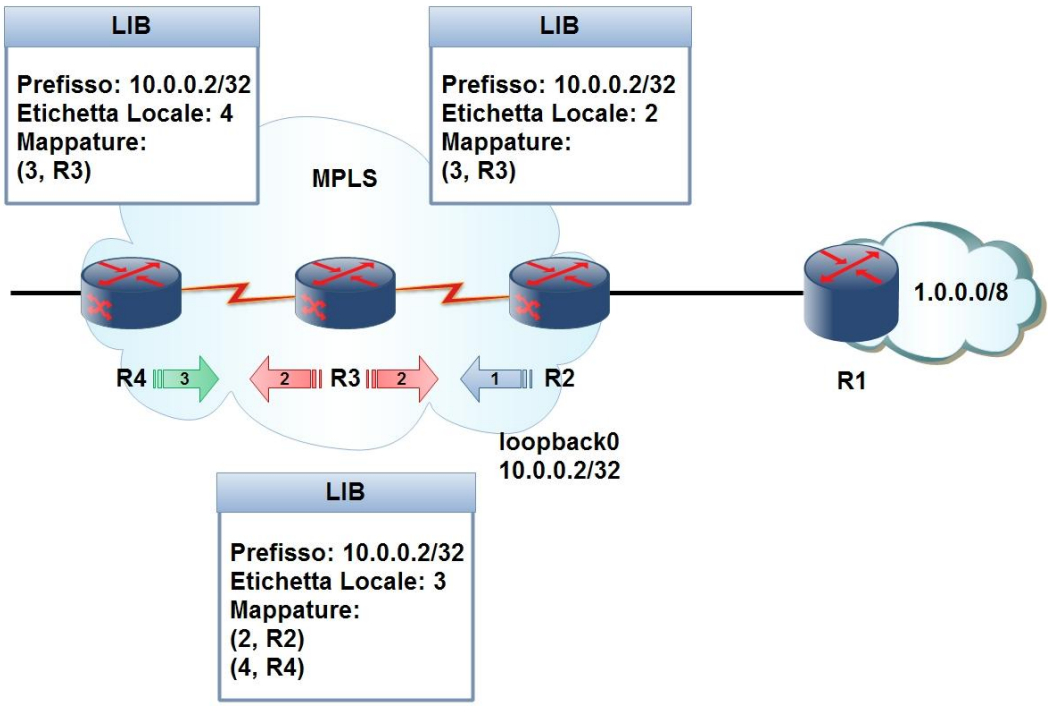

Figura 23. Esempio di distribuzione delle etichette

3. Una volta aggiornate le proprie strutture dati, il LER/Ingress-LSR R4 è in grado di determinare una FEC (utilizzando l'ETLA) per il pacchetto con destinazione 1.255.255.254 [Figura 24.]. La FEC corrispondente è 10.0.0.2/32, ovvero il router-ID del next-hop BGP che gli ha effettuato l'annuncio, R2. Ora è possibile eseguire l'inoltro, facendo un push, ovvero aggiungendo al pacchetto la label 3, ed inviandolo a R3. Quest'ultimo esegue lo swap, scambiando l'etichetta 3 con la 2, e invia il messaggio a R2, Egress-LSR di questo LSP. R2 elimina l'etichetta, tramite un pop, e dopo un longest prefix match determina che il pacchetto con destinazione 1.255.255.254 appartiene alla rete 1.0.0.0/8 di un AS a lui connesso, e lo inoltra a R1.

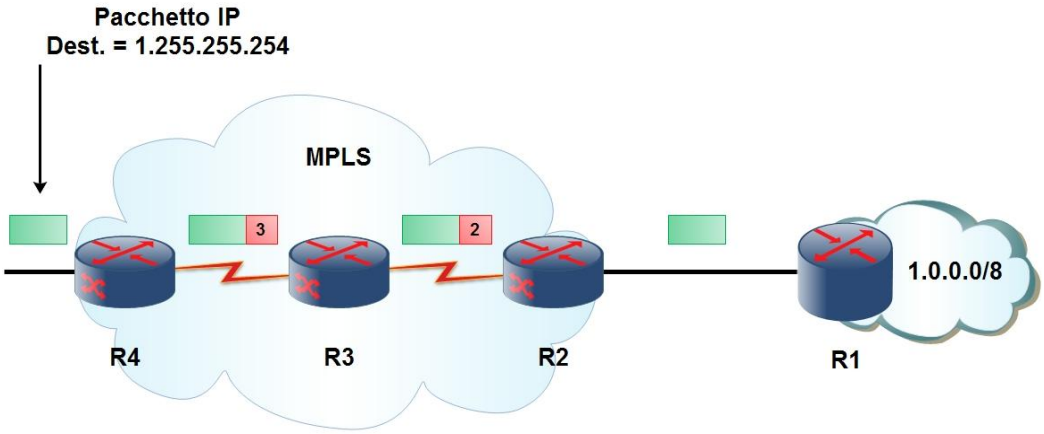

Figura 24. Esempio di inoltro di un pacchetto MPLS

Penultimate Hop Popping

Il *Penultimate Hop Popping* [1] è un'ottimizzazione del tradizionale inoltro MPLS e consiste nel distribuire sul penultimo LSR di un LSP, il carico computazionale che tradizionalmente grava soltanto sull'Egress-LSR. Esso consiste in una doppia azione di pop eseguita dal penultimo hop di un LSP. Di norma, l'Egress-LSR esegue una doppia scansione delle proprie tabelle, che può risultare onerosa in ambito di reti piuttosto grandi. La prima riguarda ILM e NHLFE: il pacchetto dotato di shim header necessita di un riscontro in queste tabelle per identificare l'azione da eseguire. Trattandosi di un Egress-LSR l'azione sarà un pop; supponendo che non ci siano etichette annidate, dopo aver tolto lo shim header, si ottiene un pacchetto con una destinazione IP da processare tramite longest prefix match. Queste due operazioni possono essere separate, eseguendo la scansione della ILM e il relativo pop sul penultimo nodo; dato che il next-hop è noto dall'etichetta in ingresso, il pacchetto IP può essere inviato all'Egress-LSR. Quest'ultimo, dovrà solamente eseguire il longest prefix match e inoltrare il pacchetto a destinazione.

Time-To-Live e Maximum Trasmission Unit

Analogamente ai datagram IP, anche per i pacchetti etichettati esiste una misura di *Time To Live* (TTL). In particolare, essa viene propagata dall'IP allo shim header e viceversa, ed impedisce al pacchetto entrante nella rete MPLS di essere inoltrato nel caso esso entri in un loop. Il TTL è decrementato di una unità nei seguenti casi: entrata e uscita dal dominio MPLS, swap, push e pop. Nel caso di arrivo di un pacchetto con TTL uguale a uno, l'LSR ricevente eliminerà il pacchetto e invierà un messaggio ICMP di

tipo "time exceeded" (type 11, code 0) alla sorgente del pacchetto IP, seguendo a ritroso l'LSP.

Un altro parametro fondamentale è il *Maximum Trasmission Unit* (MTU). Esso indica la massima dimensione che un pacchetto può assumere prima di venire frammentato. Tale parametro, che esiste anche nel contesto MPLS, differisce da quello utilizzato in ambito IP per il fatto che ogni shim header ha una dimensione di 4 byte. Perciò al datagram IP entrante nel dominio MPLS saranno aggiunti N × 4 byte, dove N è il numero di intestazioni con cui il pacchetto è stato etichettato. Nel caso di Ethernet l'MTU, di default, è 1500 byte pertanto anche con l'aggiunta di sole due label, si avrà la necessità di frammentare il pacchetto etichettato. Questo fenomeno dovrebbe essere sempre evitato, in quanto causa un indesiderato calo della performance della rete, ed è possibile configurare gli LSR perché di default non lo eseguano.

4. MPLS Transport Profile

MPLS Transport Profile, o MPLS-TP [14], è l'insieme di caratteristiche delle tecnologie MPLS che soddisfano i requisiti della RFC 5654, opportunamente esteso con ulteriori funzionalità tipiche delle reti di trasporto. MPLS-TP è un nuovo profilo tecnologico basato principalmente su MLPS-TE [5], GMPLS [11] e PseudoWire [2] a cui sono state integrate avanzate funzioni per garantire specialmente Quality of Service (QoS) [6] e Operations, Administrations and Maintenance (OAM) [15]. MPLS-TP viene introdotto allo scopo di costituire una nuova tecnologia di rete a commutazione di pacchetto, con le stesse caratteristiche di una a commutazione di circuito, sia in termini architetturali che in termini di intervento.

4.1. MPLS Traffic Engineering

Nel routing IP, ad ogni link della rete è associato un costo e il cammino ottimo è rappresentato da quello di costo minimo complessivo. Tale costo può essere assegnato in base ad una metrica singola (Open Shorthest Path First [26] e Intermediate System to Intermediate System [39] utilizzano la capacità nominale del link), multipla (Interior Gateway Routing Protocol [40] e Enhanced Interior Gateway Protocol [41] prendono in considerazione, oltre alla capacità nominale del link, anche il delay, l'affidabilità, ecc.) o essere un semplice conteggio di hop (Routing Information Protocol [42]). Il paradigma di inoltro adottato da IP, quindi, si basa solamente sul cammino di costo minimo e sull'indirizzo di destinazione, senza tener conto, per esempio, della banda disponibile sul link. Il risultato che ne consegue è il sovrautilizzo di alcuni cammini e il sottoutilizzo di altri. L'upgrade della rete potrebbe essere una soluzione non sempre ottimale, e spesso non conveniente, quindi l'esigenza di convogliare traffico dai link più congestionati a quelli sottoutilizzati, si traduce con l'introduzione di MPLS Traffic Engineering (MPLS-TE) [5].

MPLS-TE, è la variante di MPLS opportunamente estesa per supportare il Traffic Engineering. Il suo corretto funzionamento richiede la conoscenza di paramenti aggiuntivi di collegamento, come la capacità residua sul canale, che i normali protocolli LS non forniscono. MPLS-TE implementa il *source-based routing*, in quanto la selezione del cammino ottimo è effettuato all'ingresso del tunnel. Ciò introduce l'esigenza, da parte dell'LSR sorgente, di conoscere la banda rimanente sui singoli link e, non ultima, la possibilità di stabilire un LSP tra due nodi MPLS-TE, definiti come *head-end* (sorgente) e *tail-end* (destinazione) [5]. E' richiesta, quindi, l'introduzione o la modifica di un protocollo di tipo LS per il supporto del TE, grazie a cui ogni router può disporre e propagare le caratteristiche aggiuntive relative ai propri link. La creazione di un LSP, o *tunnel TE*, può avvenire in maniera preventiva, creando a priori i percorsi preferenziali attraverso ogni coppia di router (full-mesh di tunnel TE), o abilitando il Traffic Engineering, senza creare i relativi LSP finché essi non siano effettivamente richiesti. Il miglior percorso su cui instaurare un LSP può essere stabilito: dinamicamente, da un algoritmo per il calcolo dell'albero di costo minimo, vincolato dalle specifiche del tunnel TE; o staticamente, configurato da management.

Riassumiamo i concetti chiave di MPLS-TE:

- Caratteristiche di collegamento aggiuntive: capacità nominale e residua sul link, gruppo amministrativo a cui quel link appartiene;
- Distribuzione delle informazioni TE: un protocollo LS (ad esempio OSPF) opportunamente modificato per integrare le informazioni aggiuntive;
- Un algoritmo per il calcolo del miglior percorso dall'head-end al tail-end, che soddisfa le specifiche richieste dal nodo sorgente del tunnel TE;
- Un protocollo per prenotare le risorse necessarie al tunnel TE, attraverso la rete.

LSA opachi

L'estensione di OSPF in ambito di Traffic Engineering è chiamata *OSPF-TE* e prevede l'utilizzo di appositi LSA aggiuntivi, detti "opachi" [43]. Il flooding di tali informazioni avviene se si verifica un cambiamento dello stato dei link (ad esempio, una variazione della capacità disponibile) o periodicamente ogni 30 minuti. OSPF-TE prevede LSA opachi di tipo 9, 10 e 11. Nello specifico, la validità degli LSA di tipo 9 è locale ad un singolo segmento, quella degli LSA di tipo 10 ad una singola area OSPF-TE mentre gli LSA di tipo 11 sono propagati nell'intero AS. Inoltre, uno speciale bit viene utilizzato nel

campo “Options” (presente nei messaggi Hello scambiati tra vicini) e indica che il router mittente è in grado di mandare e ricevere LSA di tipo opaco.

Il payload di un LSA opaco è suddiviso in oggetti chiamati *TLV* [43], che contengono i seguenti campi: *<Type, Length, Value>*. I TLV sono fondamentali per un corretto funzionamento di OSPF-TE in quanto sono i veri responsabili del trasporto delle informazioni estese. Nello specifico, ogni LSA contiene un *Router-Address-TLV* (type 1) contenente il router-ID del mittente e un *Link-TLV* (type 2) che descrive lo stato del collegamento attraverso ulteriori nove *Sub-TLV* [Tabella 12.].

Sub-TLV	Nome	Numero di Ottetti
1	Link Type	1
2	Link ID	4
3	Local Interface IP Address	4
4	Remote Interface IP Address	4
5	Traffic Engineering Metric	4
6	Maximum Bandwidth	4
7	Maximum Reservable Bandwidth	4
8	Unreserved Bandwidth	32
9	Administrative Group	4

Tabella 12. Informazioni aggiuntive propagate da OSPF-TE

Ciascun Sub-TLV corrisponde ad una delle caratteristiche riportate in seguito:

1. *Link Type*: indica se il collegamento è di tipo point-to-point o multiaccess;
2. *Link ID*: definisce il router-ID all’estremità opposta del collegamento. E’ il router-ID del nodo adiacente, se il collegamento è di tipo point-to-point, o del DR, nel caso di rete multiaccess;
3. *Local Interface IP Address*: è l’indirizzo IP relativo all’estremità locale del link;
4. *Remote Interface IP Address*: è l’indirizzo IP relativo all’estremità opposta del link;
5. *Traffic Engineering Metric*: è il costo associato al link;
6. *Maximum Bandwidth*: indica la capacità nominale del collegamento;
7. *Maximum Reservable Bandwidth*: è la banda massima riservabile per il TE su quel link;

8. *Unreserved Bandwidth*: è la banda che in un dato momento non è utilizzata da altri tunnel;

9. *Administrative Group*: denominato anche "link color", indica a quale gruppo amministrativo appartiene il link.

L'insieme delle informazioni raccolte da OSPF-TE forma il *Traffic Engineering DataBase* (TE-DB).

Constrained Shorthest Path First

Il calcolo del miglior percorso dall'head-end al tail-end viene effettuato da un algoritmo denominato *Constrained Shorthest Path First* (CSPF) [44], del tutto simile a quello utilizzato in ambito OSPF. Esso riceve in input i requisiti del tunnel TE ed il TE-DB, e restituisce l'albero di costo minimo verso il tail-end, vincolato dalle specifiche iniziali. CSPF identifica sempre un solo ed unico percorso migliore, anche se possono essere presenti più cammini potenzialmente ottimi. In tal caso l'algoritmo applica sequenzialmente le seguenti tre regole:

1. Viene scelto il percorso con banda disponibile maggiore;

2. Viene scelto il percorso con il minor numero di hop;

3. Viene scelto un percorso casuale tra i rimanenti.

L'output della procedura è un oggetto, l'*Explicit Route Object* (ERO), essenziale per il setup dell'LSP. L'ERO è la lista ordinata degli hop che costituiscono il tunnel TE, computata alla sorgente da CSPF. Ogni singolo hop può essere di tipo "*strict*" o "*loose*". Un hop *strict* deve essere direttamente connesso al nodo precedente mentre un hop *loose* può essere raggiunto dal nodo precedente tramite il percorso che quest'ultimo ha calcolato con l'algoritmo di Dijkstra. L'ERO può anche essere configurata manualmente a livello amministrativo.

La prenotazione delle risorse lungo il percorso individuato da CSPF avviene tramite Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [38]. RSVP-TE prevede l'invio, lungo il percorso ottimo, precedentemente identificato da CSPF, di un messaggio RSVP-PATH contenente la richiesta per un LSP che soddisfi le specifiche del tunnel TE. Se le risorse sono disponibili sul link, queste vengono prenotate temporaneamente e il messaggio è inoltrato in downstream, fino al tail-end. Se le risorse, invece, non sono disponibili, il router coinvolto ha il compito di segnalare il problema all'head-end, il quale utilizzerà CSPF per ricalcolare una nuova route candidata. Nel momento in cui RSVP-PATH giunge all'uscita del tunnel TE e che la

prenotazione temporanea lungo il percorso è avvenuta, il tail-end risponde con un messaggio di tipo RSVP-RESV, con destinatario il router d'ingresso, il quale riattraversa tutti i nodi precedenti in upstream. RSVP-RESV si occupa di riservare le risorse già prenotate da RSVP-PATH e di distribuire le etichette con un procedimento analogo all'Unsolicited Downstream (UD) di LDP (vedi capitolo 3).

E' possibile suddividere il Control Plane in *Routing Plane* e *Signaling Plane* (vedi Paragrafo 4.2). Il primo raccoglie le informazioni necessarie per l'instradamento, il secondo si occupa di prenotare le risorse relative ai tunnel TE e di segnalarne il relativo cambio di disponibilità sui link, agli altri nodi della rete. OSPF-TE è un protocollo di Routing Plane mentre RSVP-TE è un componente di Signaling Plane.

Prendendo come esempio la [Figura 25.], supponiamo che ogni link abbia metrica di valore 1 e capacità nominale di 100 Mbps. In figura, la capacità residua è indicata singolarmente per ogni collegamento. Viene richiesto all'head-end R1, un tunnel TE con tail-end R8 e con banda minima garantita di 60 Mbps. Prendendo in input queste specifiche e il TE-DB popolato da OSPF-TE, CSPF restituisce l'ERO: <R1, R2, R3, R6, R7, R8>. L'head-end, R1, procede alla prenotazione delle risorse lungo il percorso, inviando il messaggio RSVP-PATH verso R2, con destinazione R8. Dopo che le risorse sono state temporaneamente riservate con successo lungo il percorso, il tail-end, conferma la prenotazione a R1 con un messaggio RSVP-RESV, che risale il percorso in upstream e distribuisce le etichette necessarie per instaurare l'LSP. Al termine, la capacità residua sui link interessati dal nuovo tunnel TE è di 40 Mbps.

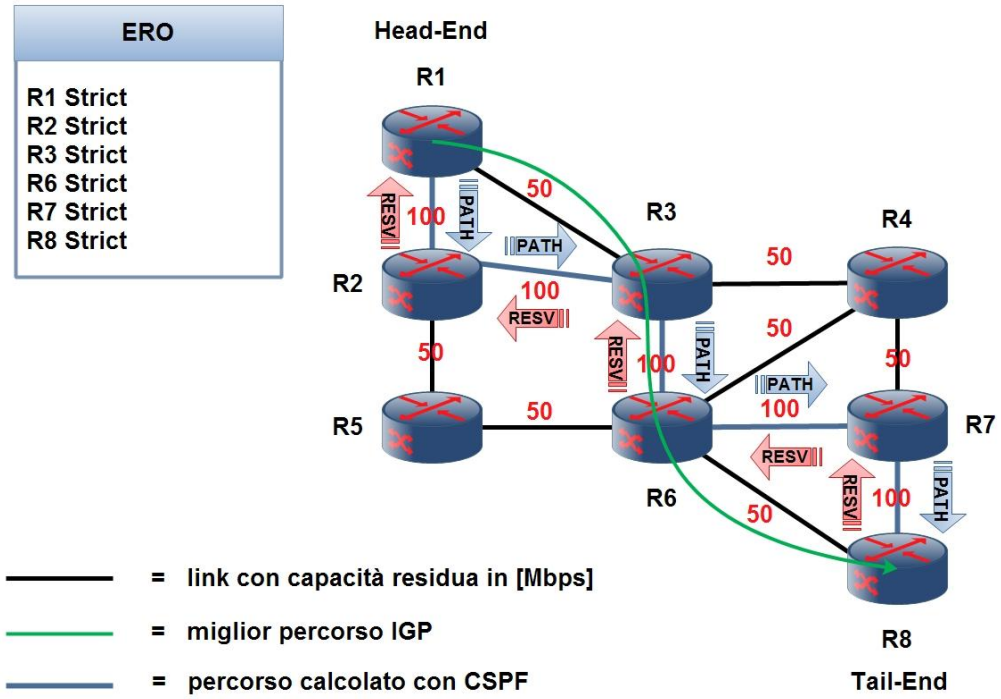

Figura 25. Esempio di calcolo del percorso di costo minimo con CSPF

Funzionamento di MPLS-TE

Per concretizzare quanto introdotto finora dalla parte di studio teorico del protocollo, si vuole presentare un possibile *algoritmo di funzionamento per MPLS-TE*. Esso inizia dalla raccolta delle informazioni atte a popolare il TE-DB e ha la finalità di instaurare gli LSP tramite RSVP-TE:

1. Per ogni nodo del dominio, vengono raccolte le informazioni sulla topologia di rete tramite un protocollo IGP LS, esteso per supportare il Traffic Engineering. A tali scopi viene utilizzato OSPF-TE che introduce appositi LSA opachi, il cui scopo è quello di propagare informazioni LS aggiuntive, come la capacità disponibile sui link o il gruppo amministrativo di appartenenza;

2. OSPF-TE crea il tradizionale LS-DB, sul quale è basata la RIB, e un addizionale TE-DB, contenente le informazioni relative agli LSA opachi. Ciascun nodo MPLS-TE ha a disposizione tutte le specifiche di collegamento necessarie per computare il percorso migliore verso gli altri nodi dell'area;

3. All'head-end vengono inserite da management le specifiche del tunnel TE che si vuole creare. Tramite l'algoritmo CSPF viene calcolato il percorso di costo minimo (ERO), con destinazione il tail-end, che soddisfa tali richieste;

4. Un LSP deve essere stabilito tra l'head-end e il tail-end. Questo viene segnalato ai nodi intermedi tramite il protocollo RSVP. Un messaggio RSVP-PATH viene inviato in downstream lungo il percorso. Ogni nodo attraversato da tale messaggio, verifica che le risorse richieste dalla sorgente siano disponibili, le prenota temporaneamente e inoltra il messaggio all'hop successivo. Nel caso in cui tali risorse non siano disponibili, il nodo in questione ha il compito di segnalare il problema al router di ingresso, il quale utilizzerà CSPF per ricalcolare una nuova route candidata per il tunnel TE;

5. Una volta che RSVP-PATH giunge al tail-end, rispettando integralmente i requisiti del tunnel TE, il nodo risponde con un messaggio di tipo RSVP-RESV, con destinatario l'head-end, risalendo in upstream tutti i router già visitati precedentemente da RSVP-PATH. RSVP-RESV si occupa di riservare le risorse prenotate e di distribuire le etichette necessarie per instaurare l'LSP;

6. Una volta che le etichette sono state distribuite e le risorse prenotate, è possibile inoltrare il traffico lungo il tunnel TE, con le modalità di inoltro previste dal tradizionale MPLS.

L'algoritmo di funzionamento di MPLS-TE può essere schematicamente rappresentato dal diagramma in [Figura 26.]:

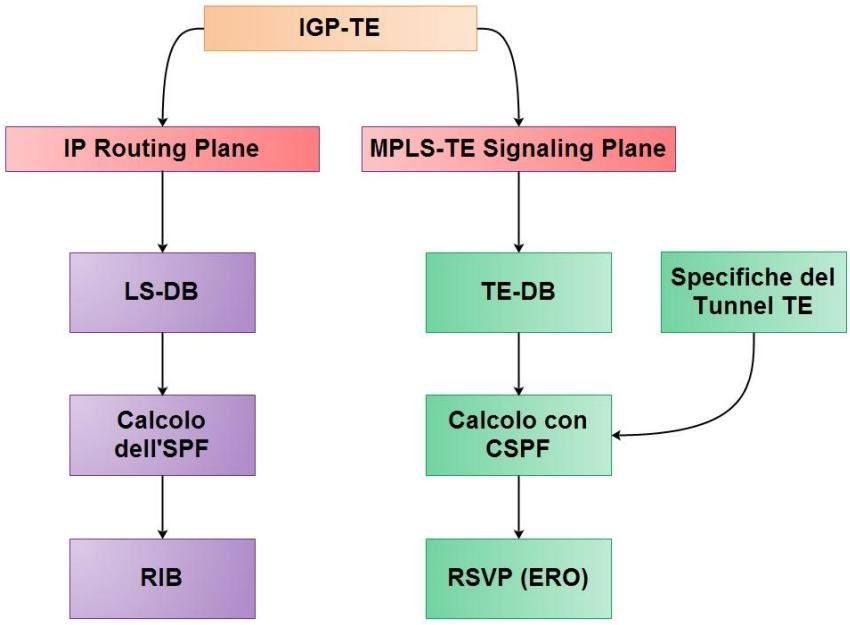

Figura 26. Relazione tra il Routing Plane IP e il Signaling Plane MPLS-TE

Fast Re-Route

Il Fast ReRoute (FRR) [45] è il meccanismo di fault tolerance di MPLS-TE, basato sull'instaurazione preventiva di un LSP di backup. Quest'ultimo può essere realizzato staticamente, dall'amministratore di rete che specifica anche l'intero percorso hop-by-hop, oppure dinamicamente, tramite CSPF. Un LSP che dispone di un percorso di backup è detto "LSP protetto" o "LSP primario". L'LSP di backup è sempre incapsulato in un LSP protetto. Questa tecnologia permette che il traffico mission-critical non subisca rallentamenti superiori ai 50 ms, dovuti al guasto di un router o all'interruzione di un link lungo l'LSP primario. Esistono due tipologie di protezione:

- Link-Protection [Figura 27.]: se l'interruzione avviene su un singolo link. Il traffico che si trova a dover passare per l'LSP protetto è rediretto attraverso l'LSP di backup fino al next-hop successivo al guasto. Questo tipo di LSP è detto anche "next-hop backup tunnel" (NHOP). Il router a cui è collegato il link interrotto è chiamato "Point of Local Repair" (PLR) ed è l'LSR responsabile della rilevazione del guasto e del push della label relativa all'LSP di backup. Quest'ultimo, passa attraverso un nodo intermedio e termina all'LSR "Merge Point" (MP), che si occupa del pop della label imposta dal PLR e dell'inoltro del traffico sull'LSP primario. Il pop può essere eseguito, in alternativa, anche dal penultimo nodo dell'LSP di backup. In tal modo, l'MP riceverà comunque un pacchetto etichettato con la stessa label in ingresso che avrebbe ricevuto prima del guasto, rendendo di fatto questa tecnica completamente trasparente;

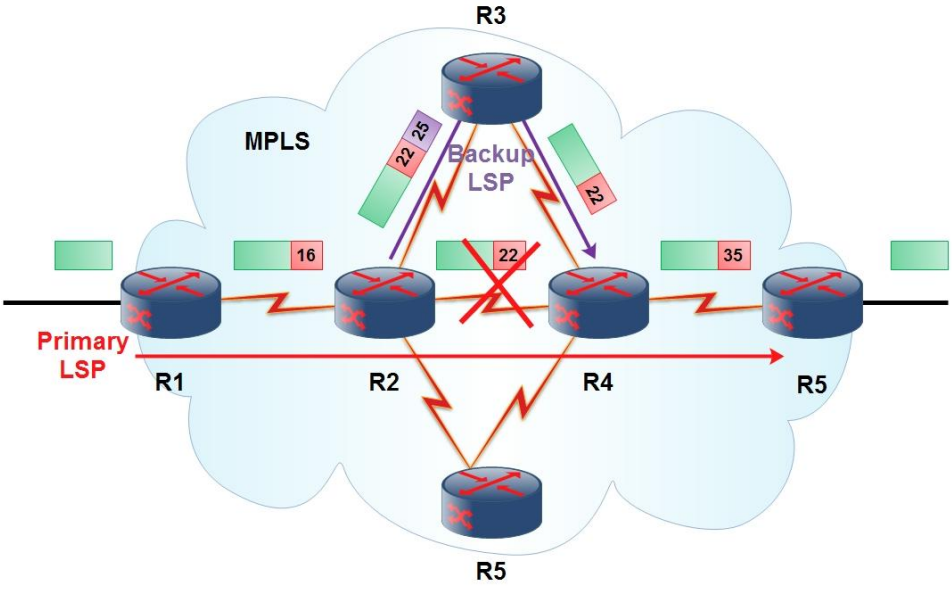

Figura 27. Fast ReRoute Link-Protection

- Node-Protection [Figura 28.]: se il guasto coinvolge un router lungo l'LSP protetto. In questo caso, il tunnel di backup bypassa il nodo irraggiungibile arrivando direttamente all'hop successivo al guasto. Per tali motivi, l'LSP di backup è chiamato anche "next-next-hop backup tunnel" (NNHOP). La procedura di inoltro lungo il tunnel di backup è del tutto analoga al caso di link-protection, anche se Il tempo di ripristino in questa modalità potrebbe essere leggermente maggiore a quanto visto nel caso precedente, in quanto dipende dal tempo impiegato a verificare che il nodo risulti effettivamente irraggiungibile.

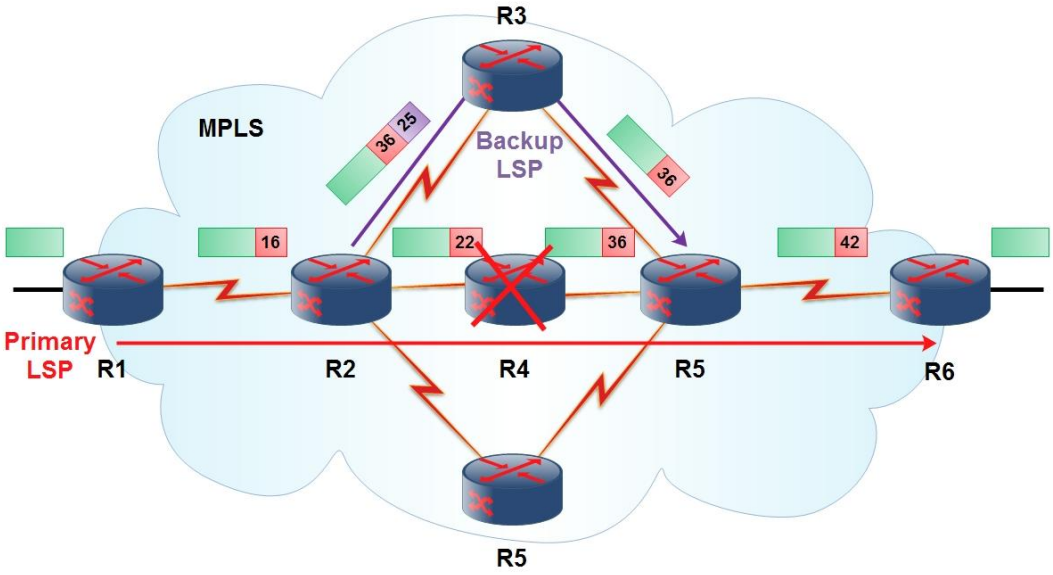

Figura 28. Fast ReRoute Node-Protection

4.2. Generalized MPLS

Con l'affermarsi di tecnologie che commutano nel dominio del tempo, in quello della lunghezza d'onda e in quello dello spazio, risultava sempre più evidente una spiccata eterogeneità tra i diversi nodi appartenenti ad un rete di trasporto, "Transport Network Elements" (TNE), che inevitabilmente portava a crescenti difficoltà di gestione [46]. GMPLS è una generalizzazione dei concetti di MPLS-TE, allo scopo di fornire un Control Plane comune per tutti i nodi una rete di trasporto.

Architettura di GMPLS

Generalized MPLS (GMPLS) [11] è un'estensione dell'architettura MPLS-TE che mira a fornire una migliore integrazione tra alcune tecnologie di trasporto, quali:

- *Time Division Multiplexing* (TDM) [12]: un segnale trasmissivo viene condiviso da molteplici flussi di dati, ad ognuno dei quali spetta una ben definita e regolare porzione o "timeslot" di segnale;
- *Wavelength Division Multiplexing* (WDM) [13]: consiste di molteplici segnali ottici all'interno di una stessa fibra. Ciascun segnale è chiamato anche colore o "*lambda*" e viene codificato utilizzando la relativa frequenza a cui quella lunghezza d'onda, o *wavelength*, appartiene. Come standard di riferimento, ITU-T ha pubblicato alcune liste di frequenze accettabili. Ognuna di queste fornisce un insieme di wavelength uniformemente spaziate, secondo una modalità grossolana detta "Coarse WDM" (CWDM), in cui le spaziature sono di 2500 GHz, oppure densa detta anche "Dense WDM" (DWDM), in cui esse si riducono a 100, 50 o 25 GHz;
- *Fiber* [46]: una fibra convoglia al suo interno molteplici segnali di tipo CWDM o DWDM, e tratta il segnale complessivo come una singola "white light". Uno switch, in questo contesto, prende tutti i dati trasportati da una singola fibra e li replica su un'altra di queste;
- *Altre tecnologie di layer 2*: come Ethernet, Frame Relay [8] e ATM [7].

Cercando di definire le componenti funzionali di GMPLS si rende necessario introdurre alcuni piani di lavoro, in base ai quali è possibile gestire logicamente una rete [46]:

- *Management Plane*: identifica il cammino percorso dal traffico di management, generato da richieste di configurazione, diagnostica, statistiche ecc.;
- *Control Plane*: tratta il disegno della topologia di rete attraverso la propagazione dei cosiddetti "*pacchetti di controllo"*, come possono essere gli LSA in ambito OSPF o le mappature scambiate tra vicini LDP. Questo piano può essere ulteriormente suddiviso in:
    - *Signaling Plane*: formato da tutti quei messaggi che si occupano di segnalare esplicitamente un percorso all'interno della rete. In questa definizione rientrano ad esempio: LDP (necessario per la distribuzione delle etichette e per la conseguente segnalazione degli LSP) e RSVP-TE (nell'ambito della segnalazione dei tunnel TE e alla relativa distribuzione

delle etichette ad essi associati). Esso include tutte le informazioni contenute nel TE-DB;

- *Routing Plane*: composto da tutti i protocolli di routing intra-AS necessari per il popolamento del LS-DB e per la creazione della routing table;

- *Data Plane*: rispecchia la topologia di tipo fisico/data-link, e si occupa dell'inoltro del traffico degli utenti nella rete di trasporto;

GMPLS fornisce un Control Plane comune per la gestione integrata di tutte le precedenti tecnologie trasmissive, generalizzando quanto proposto da MPLS-TE [5]. Nello specifico, questo piano di controllo permette di gestire le seguenti tipologie di interfacce in maniera del tutto interoperabile:

- *Packet Switch Capable* (PSC) [Figura 29.a]: interfacce che inoltrano il traffico in uscita basandosi sull'intestazione del pacchetto ricevuto;

- *TDM Switch Capable* (TSC) [Figura 29.b]: interfacce che inoltrano il traffico in uscita basandosi sul timeslot ricevuto;

- *Lambda Switch Capable* (LSC) [Figura 29.c]: interfacce che inoltrano il traffico in uscita basandosi sulla wavelength su cui il traffico è ricevuto;

- *Fiber Switch Capable* (FSC) [Figura 29.d]: interfacce che inoltrano il traffico in uscita basandosi sulla porta d'ingresso su cui il traffico è ricevuto.

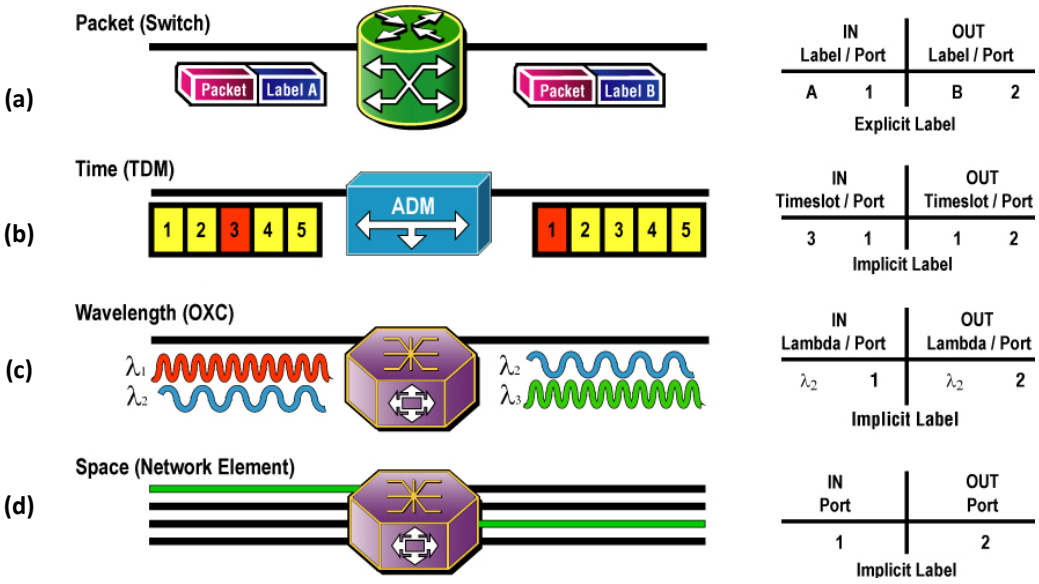

Figura 29. Interfacce GMPLS: (a) PSC (b) TSC (c) LSC (d) FSC

Nel contesto delle reti IP/MPLS, una label è un tag arbitrario al quale corrisponde un'entry della LFIB mentre nell'ottica più generalizzata di GMPLS, ogni label è associata ad un più ampio concetto di risorsa [46]. Quindi, nell'esempio in cui un tradizionale LSR mappa un'etichetta entrante ad una uscente in ambito PSC, un nodo TSC mappa uno slot entrante ad uno uscente, un nodo LSC mappa una lambda (per questo motivo, GMPLS era inizialmente chiamato MPλS [47]) entrante ad una uscente e un nodo FSC mappa una porta in entrata ad una in uscita.

In una rete IP/MPLS, i pacchetti di controllo relativi ai protocolli di routing/signaling viaggiano nello stesso canale del traffico dati (*in-band control channel*). In GMPLS ciò non è possibile in quanto, ad esempio, in una rete WDM gli LSR dovrebbero continuamente monitorare la tipologia di pacchetti in arrivo a scapito della velocità di processamento degli stessi [46]. Tale scenario sarebbe inverosimile in quanto il principale motivo per adottare queste tecnologie sta proprio nella capacità dei singoli nodi di gestire efficientemente ed indistintamente moli di dati, impensabili per una rete a commutazione di pacchetto [46]. Per questo motivo, in GMPLS si è scelto di creare un canale di controllo dedicato, denominato "*out-of-band control channel*", che può essere nel dettaglio*: in-fiber out-of-band* (uno specifico timeslot o wavelength) o *out-of-fiber out-of-band* (un collegamento fisico dedicato, nel quale passano solo informazioni di Control Plane). Ne consegue che, per garantire la connettività nel piano di controllo, ogni nodo della rete, indistintamente dalla sua natura, necessita di essere identificato da un indirizzo IP. IETF si è dedicata all'estensione del paradigma MPLS ed il gruppo di lavoro che se ne occupa attualmente è il CCAMP [48].

In GMPLS, ogni LSP instaurato deve necessariamente iniziare e terminare su interfacce omologhe. Quindi, per PSC-LSP s'intende un tunnel che inizia e termina su interfacce PSC. L'area compresa tra di esse è detta "*regione PSC*". Ciò vale anche per le tipologie TSC, LSC e FSC. L'interoperatività tra interfacce eterogenee è possibile attraverso la definizione di una gerarchia che prevede, dall'alto verso il basso: FSC, LSC, TSC, PSC. L'ordinamento gerarchico degli LSP è basato sulla capacità di multiplexing dei singoli nodi mentre i nodi di bordo di ciascuna regione sono responsabili per l'instaurazione degli LSP di livello superiore.

Gerarchia di GMPLS

Il concetto di LSP gerarchico è indispensabile per il funzionamento di GMPLS. Cerchiamo di capirlo descrivendo uno scenario [49] molto semplificato [Figura 30].

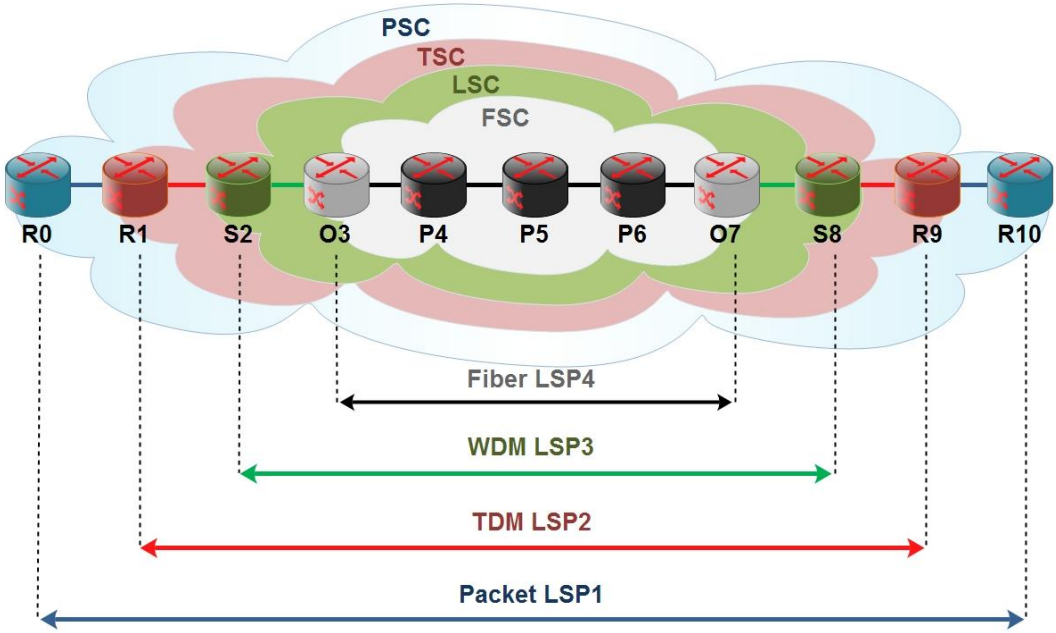

Figura 30. Esempio di rete GMPLS

Definiamo i seguenti nodi di una rete di trasporto GMPLS [Figura 31]:

- R0 e R10: possiedono solo interfacce PSC con collegamenti Gigabit Ethernet. Questo tipo di nodi sono presenti agli estremi del dominio e delimitano la regione PSC;
- R1 e R9: presentano un'interfaccia PSC e una TSC con collegamenti OC-48 (4 OC-12 da 622 Mbps). Essi delimitano la regione TSC;
- S2 e S8: possiedono un'interfaccia TSC e una LSC con collegamento OC-192. Questi nodi delimitano la regione LSC;
- O3 e O7: presentano un'interfaccia LSC e una FSC capace di contenere 16 lambda OC-192. Essi delimitano la regione FSC;
- P4, P5 e P6: possiedono solo interfacce FSC. Questo tipo di nodi sono presenti nel core della rete di trasporto, che si occupa del'inoltro del traffico ad altissime prestazioni, su lunghe distanze.

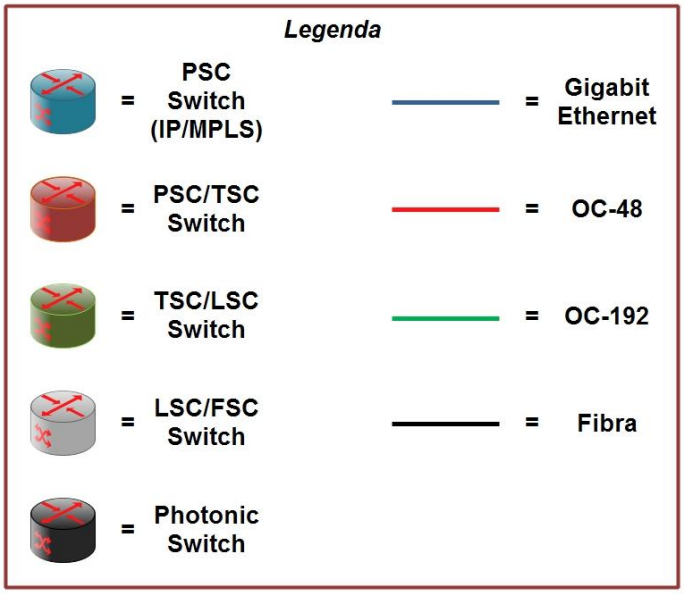

Figura 31. Legenda relativa alla Figura 30.

A livello di Control Plane, è necessario configurare un opportuno protocollo di routing LS (ad esempio OSPF-TE) in modo che ogni nodo conosca la topologia dell'intera rete. In più, devono essere disponibili anche le informazioni di Traffic Engineering utili a popolare il TE-DB di ciascun nodo come: il tipo di link (PSC, TSC, LSC, FSC), la capacità residua su di esso, ecc. Popolato il TE-DB, ogni interfaccia è in grado di instaurare un tunnel con un'altra omologa, al capo opposto della rete. Per rendere possibile questa operazione, il percorso deve essere segnalato (ad esempio con RSVP-TE) attraverso tutti i nodi visti precedentemente.

Riprendendo l'esempio precedente [Figura 30] [49], supponiamo che nessun LSP sia già stato instaurato. Viene richiesto a R0 un tunnel con banda minima disponibile di 500 Mbps verso R10. A livello di Signaling Plane, R0 è in grado di determinare in base al suo TE-DB, dopo aver eseguito CSPF, il percorso più breve che soddisfa i requisiti di banda per arrivare a R10 e ad esso invia il messaggio RSVP-Path1. Questo contiene, oltre alle caratteristiche del tunnel da instaurare, la label in downstream, l'ERO e anche una label in upstream suggerita. La distribuzione di una doppia etichetta è utile al fine di creare LSP bidirezionali [Figura 32].

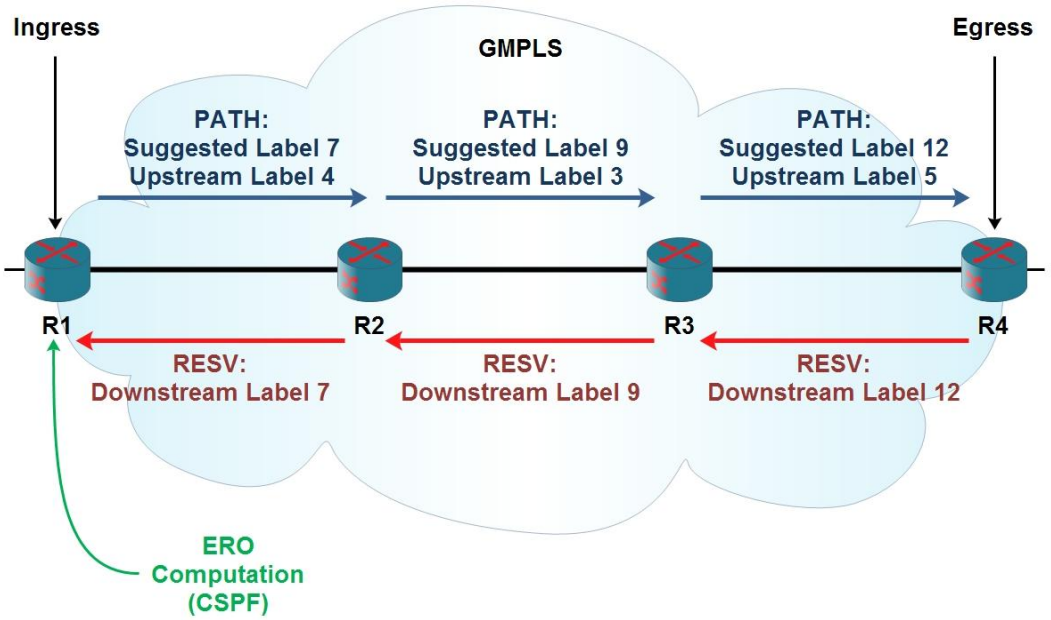

Figura 32. Setup di un LSP bidirezionale

Il messaggio viene ricevuto dal nodo di bordo che delimita la regione di livello gerarchicamente superiore (in questo caso TSC) e fa sì che venga richiesto in cascata un LSP di tipo TSC tra R1 e R9 [49]. Analogamente a quanto appena visto, il nuovo messaggio RSVP-Path2 segue il percorso più breve (contenuto nell'ERO) ed arriva al bordo della regione LSC, delimitata da S2. A sua volta, S2 invia il messaggio RSVP-Path3 fino al più vicino nodo di bordo della regione FSC, per richiedere un LSP di tipo LSC. Arrivati al livello più alto della gerarchia GMPLS, O3 è in grado di mandare direttamente il messaggio RSVP-Path4 a O7, il nodo di bordo della sua regione, il quale risponde con il messaggio RSVP-Resv4 che conferma o meno la richiesta di risorse, e fornisce la label in upstream, preferenzialmente la stessa suggerita da O3 stesso. Una volta stabilito LSP4 il messaggio RSVP-Path3 pendente può essere inoltrato in LSP4 e raggiungere il nodo di bordo della regione LSC, S8. Una volta instaurato LSP3 possono essere creati in cascata anche LSP2 (tra R1 e R9) e LSP1, fornendo così un LSP gerarchico tra R0 e R10 (chiaramente, tutti i link che vengono attraversati dall'LSP1 devono avere una banda sufficiente a soddisfare la richiesta di R0).

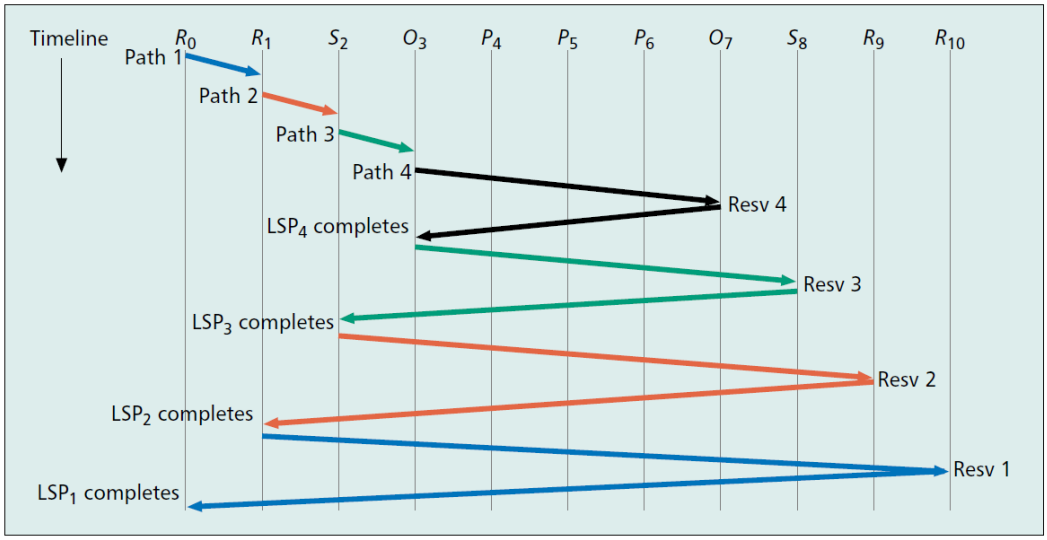

Figura 33. Instaurazione degli LSP con riferimento alla Figura 30.

Gli LSP creati sono inseriti tra le informazioni del TE-DB e compaiono come link virtuali, o *Forwarding-Adjacency* (FA), tra interfacce omologhe. Il costo attribuito a questi link è sempre inferiore alla somma delle metriche dei link fisici, in modo tale che alla successiva computazione, CSPF scelga preferenzialmente di instradare il traffico proprio sugli LSP già disponibili, garantendo un'ottimizzazione della banda sotto questo punto di vista. Le nuove informazioni relative alle FA create dagli LSP in [Figura 30.] sono le seguenti [49]:

- R1 annuncia: 622 Mbps [1 x OC-12] – 500 Mbps = 122 Mbps liberi sull'adiacenza TSC che collega direttamente R1 ad R9 e nella quale possono essere incanalati altri PSC-LSP;
- S2 annuncia: 9953 Mbps [1 x OC-192] – 622 Mbps [1 x OC-12] = 9331 Mbps liberi sull'adiacenza LSC tra S2 e S8 e nella quale possono essere incanalati altri TSC-LSP;
- O3 annuncia: [16 x OC-192] – [1 x OC-192] = [15 x OC-192] liberi sull'adiacenza FSC tra O3 e O7 nella quale possono essere incanalati altri LSC-LSP.

I dettagli per cui si è resa indispensabile l'introduzione di una gerarchia sono da ricercare nel miglioramento della gestione delle label. Infatti, il grande numero di etichette "logiche" disponibili in MPLS mal si adatta alla ben più limitata disponibilità di label "fisiche" in ambito GMPLS [46]. Inoltre, la natura discreta delle grandezze fisiche prese in considerazione avrebbe reso altamente inefficiente l'allocamento delle risorse per LSP di piccole dimensioni (in riferimento all'esempio in [Figura 30.], per soli 500

Mbps si sarebbe occupata un'intera lambda da 9953 Mbps). Da qui l'intuizione di un'organizzazione gerarchica degli LSP.

Funzionamento di GMPLS

Si vuole riassumere in linea generale, l'*algoritmo di funzionamento di GMPLS*. Esso segue i passi fondamentali di MPLS-TE con le dovute estensioni derivanti dall'eterogeneità dei nodi che ne compongono il dominio. Si suppone che l'head-end sia al bordo di una regione PSC e che non vi siano altri tunnel già configurati:

1. Per ogni nodo del dominio GMPLS, vengono raccolte le informazioni LS sulla topologia di rete e le informazioni TE sulle caratteristiche aggiuntive necessarie ad instaurare il tunnel TE (come la capacità residuta dei link). Viene popolato il TE-DB;

2. Le specifiche del tunnel TE sono configurate alla sorgente del PSC-LSP;

3. L'head-end utilizza CSPF prendendo come input il TE-DB e restituendo come output il percorso di costo minimo vincolato (ERO), fino alla destinazione del PSC-LSP;

4. La sorgente invia il messaggio RSVP-PATH lungo il percorso specificato dall'ERO, fino a raggiungere il più vicino router di bordo TSC;

5. In cascata, il router di bordo TSC sospende la creazione del PSC-LSP e inizia l'instaurazione di un TSC-LSP sino al più vicino nodo di bordo della regione LSC. Questo meccanismo continua sino al culmine della gerarchia, in cui l'head-end FSC può instaurare direttamente un FSC-LSP con il nodo di bordo diametralmente opposto, che inoltra lungo il percorso inverso un messaggio RSVP-RESV;

6. Creato l'LSP gerarchicamente più elevato (FSC), possono essere ripresi, in cascata, tutti gli altri. E' importante precisare che ogni messaggio RSVP-RESV utilizza l'LSP di livello superiore per giungere al relativo nodo di bordo sorgente, sino ad arrivare all'head-end iniziale. L'intuizione alla base di questo passo è illustrata in [Figura 33.];

7. Gli LSP creati (FSC, LSC, TSC e PSC), vengono installati nel TE-DB di ogni router come link virtuali e propagati tramite OSPF-TE agli altri nodi del dominio. Tali link, rappresentano collegamenti point-to-point con metrica inferiore alla somma delle metriche sui link fisici lungo il medesimo percorso e sono utilizzati in via preferenziale per costituire tunnel gerarchicamente inferiori.

Link Management Protocol

In ambito di Management Plane, il *Link Management Protocol* (LMP) [50] è un nuovo protocollo point-to-point adottato a livello di Control Plane da nodi vicini, che si occupa principalmente di gestire il controllo del canale fisico, introducendo le seguenti caratteristiche:

- Link Property Correlation: gestisce il passaggio dei flussi di dati da una determinata interfaccia ad un'altra potenzialmente diversa, all'interno del dominio GMPLS. Ad esempio, in un nodo di bordo TSC/LSC questa funzionalità deve garantire la corretta conversione del flusso di dati da TDM a WDM;
- Control Channel Management: si occupa di verificare l'esistenza di un'effettiva e ininterrotta connessione a livello di Control Plane;
- Link Connectivity Verification: si occupa di verificare l'esistenza di un'effettiva e ininterrotta connessione a livello di Data Plane;
- Fault Management: include la localizzazione e la notifica dei guasti a livello di Data Plane, in modo da poter ripristinare il collegamento nel minor tempo possibile.

4.3. PseudoWire

Una rete *PseudoWire* (PW) [2] è un'emulazione di un servizio orientato alla connessione e point-to-point, di livello 2, su una rete a commutazione di pacchetto. Si parla anche di servizio *Pseudowire Emulated End-to-End* (PWE3). Come suggerisce il nome stesso, questa tecnologia emula un "filo trasparente" sul quale è trasportato un *servizio nativo*, che può essere [2]: Frame Relay [8], ATM [7], Ethernet, TDM [12], SONET [51], PPP [52], HDLC [53]; mentre la rete a commutazione di pacchetto può essere implementata tramite MPLS [1] o L2TPv3 [53]. Nel 2001, IETF costituì il gruppo di lavoro PWE3 [54] al fine di sviluppare l'architettura PW per ISP mentre le prime specifiche, relative alla bozza Martini [2], sono datate 2004.

Definiamo gli elementi costitutivi di PW:

- *Servizio nativo*: è la tecnologia di livello 2 che si vuole emulare attraverso la rete PW;
- *Customer Edge* (CE): è il dispositivo di rete del cliente che richiede l'emulazione di un servizio nativo;

- *Provider Edge* (PE): è il dispositivo dell'ISP che fornisce connettività PW ai CE;
- *Attachment Circuit* (AC): è il link fisico che connette un CE al relativo PE.

Quindi, un collegamento PW è una connessione logica tra due router Provider Edge [2].

I due PE devono essere connessi tramite una rete a commutazione di pacchetto di proprietà di un ISP. L'emulazione del servizio nativo avviene incapsulando il traffico di livello 2 in tunnel di livello superiore. Supponiamo che l'infrastruttura di trasporto dell'ISP implementi IP/MPLS e che i PW vengano realizzati tramite LSP [55] [Figura 34].

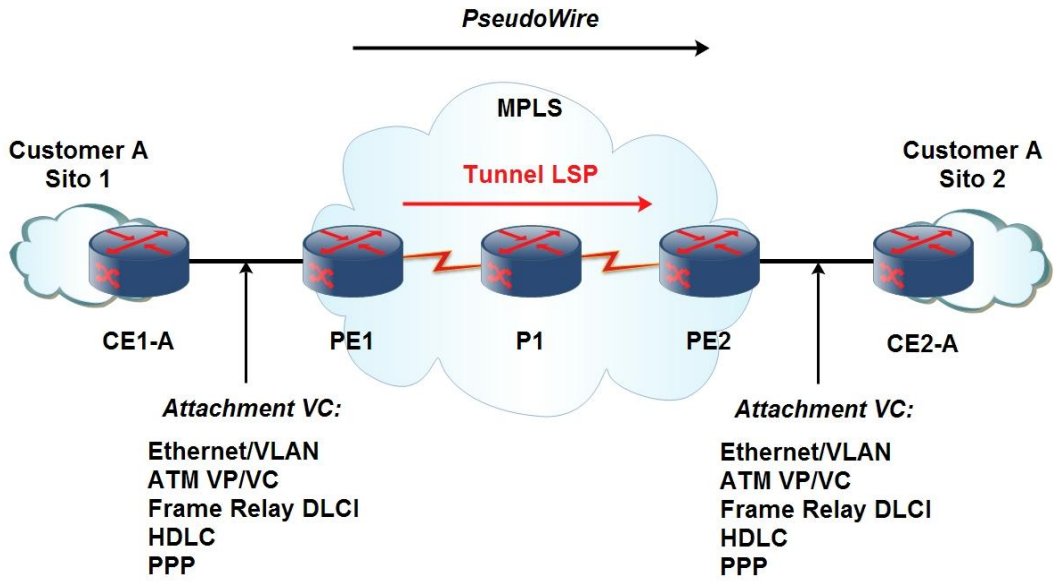

Figura 34. Architettura di una rete PseudoWire

Architettura e Funzionamento di PseudoWire

Il traffico di livello 2, che deve fluire da un sito cliente all'altro, viene inoltrato dal Customer Edge al Provider Edge attraverso l'Attachment Circuit. Al PE, vengono apposte due label per l'emulazione del servizio e il successivo inoltro:

- *PW-label* (etichetta interna): è la prima etichetta che viene associata, mappa un'interfaccia in entrata al PE ad un PW. Essa è necessaria per eseguire il multiplexing di più PW in un LSP;

- *Tunnel-label* (etichetta esterna): è la seconda etichetta che viene associata, annidata alla prima. Essa è necessaria per il tradizionale inoltro nella rete IP/MPLS dell'ISP.

Il funzionamento di PW è del tutto analogo a quanto visto per MPLS-TE, con le sole differenze che: il payload del pacchetto MPLS è un frame/cella di layer 2; avviene una doppia etichettatura, la prima (interna) è una mappatura alla corrispondente porta d'ingresso al PE sorgente, la seconda (esterna) è una tradizionale label MPLS che viene scambiata attraverso tutta la rete dell'ISP.

Dopo aver attraversato la rete di trasporto dell'ISP, il traffico del cliente viene de-etichettato, verificato il PW di riferimento, e inoltrato verso il successivo AC, per raggiungere il sito cliente di destinazione.

Ethernet over MPLS (EoMPLS) [3] è un esempio particolare di PW, in cui un collegamento Ethernet point-to-point rappresenta il servizio nativo emulato su IP/MPLS. Più generalmente, *Virtual Private LAN Services* (VPLS) [4] implementa una topologia full-mesh PW che connette ogni PE a tutti gli altri, emulando una topologia multipoint-to-multipoint.

4.4. MPLS-TP: Framework

MPLS, GMPLS e PW trovano una loro evoluzione congiunta in Multi Protocol Label Switching - Transport Profile (MPLS-TP) [14], l'insieme di caratteristiche delle tecnologie MPLS che soddisfano i requisiti della RFC 5654, opportunamente esteso con ulteriori nuove funzionalità. Tali specifiche indicano come questo nuovo paradigma, tuttora in fase di standardizzazione congiunta da parte di IETF e ITU-T, possa essere utilizzata efficacemente nell'ambito delle reti di trasporto. MPLS-TP viene introdotto allo scopo di fornire un nuovo paradigma per reti a commutazione di pacchetto, con le stesse qualità di quelle a commutazione di circuito, sia in termini architetturali che in termini di intervento.

Esigenze di un nuovo profilo di trasporto

MPLS-TP è l'infrastruttura di base (detta "*MPLS-TP layer*" [14], in quanto su di esso sono trasportati i servizi nativi dei propri clienti, chiamati "*client layer*" [14]) per la realizzazione della cosiddetta "Next Generation Network" (NGN) [56], o rete di nuova generazione, la cui definizione secondo ITU-T è la seguente:

una rete di nuova generazione è una rete a commutazione di pacchetto che può fornire servizi nell'ambito delle telecomunicazioni attraverso l'uso di tecnologie a banda larga e QoS, del tutto indipendenti da come è organizzata la struttura sottostante su cui poggiano. Supporta e offre, inoltre, un accesso senza restrizioni e potenzialmente ubiquo a tutti i suoi utenti.

L'*MPLS-TP layer* è implementato attraverso tecnologie MPLS-TE/GMPLS/PW [14].

Non tutte le caratteristiche previste da MPLS sono richieste per questo nuovo modello operazionale e, per converso, ci sono altri aspetti cruciali per le reti di trasporto attualmente non implementate in MPLS. La definizione di MPLS-TP data dalla RFC 5921 è la seguente:

il profilo di trasporto per MPLS è il sottoinsieme di funzioni presenti in MPLS che soddisfa i requisiti della RFC 5654.

Una NGN deve poter trasportare efficientemente sia traffico dati che voce, potenzialmente connettendo ciascun cliente a ciascun altro e mantenendo propri i principali vantaggi della tecnologia di trasporto precedente:

- Connettività basata su circuiti permanenti, che possono essere instaurati anche manualmente;
- Un alto livello di disponibilità e tempi di downtime ridotti;
- Qualità del Servizio;
- Estese capacità di Operations, Administrations and Maintenance (OAM) [15]: la funzionalità che permette una rapida rilevazione del guasto, monitora la performance e garantisce che i termini di Service Level Agreement (SLA) siano rispettati.

I requisiti specificati dalla RFC 5654 hanno l'obiettivo di associare alle reti MPLS un grado di prevedibilità e protezione simile a quello di una rete a commutazione di circuito:

- L'*MPLS-TP layer* supporta un *client layer* ed è supportato da un *server layer* [14] (lo strato di livello 2 che incapsula i pacchetti MPLS). L'operatività della rete MPLS-TP deve essere possibile senza alcuna dipendenza né dal *server* né dal *client layer*;
- Tutti i pacchetti generati dai *client layer*, sono trasportati sul *server layer* di MPLS-TP;

- La topologia di rete adottata dall'*MPLS-TP layer* risulta del tutto trasparente ai *client layer*, e viceversa;
- Il servizio fornito dall'*MPLS-TP layer* a un dato cliente non può essere compromesso, in termini di Service Level Agreement (SLA), dal traffico relativo ad altri clienti della rete;
- I piani di controllo e di management di ogni cliente che utilizza il proprio *client layer*, è isolato dai piani di controllo e management dell'*MPLS-TP layer*;
- L'interazione tra il *server* ed il *client layer* deve essere minima (preferibilmente assente).

Nello sviluppo del nuovo profilo vengono eliminate, inoltre, alcune caratteristiche tipiche di MPLS, non strettamente necessarie a soddisfare i requisiti visti in precedenza, ad esempio: la possibilità di avere LSP non-TE, il Penultimate Hop Popping, l'Equal Cost Multi Path (ECMP), ecc.

Architettura di MPLS-TP

L'architettura di MPLS-TP prevede due tipologie di *client layer* [14]:

1. Un PseudoWire, per l'emulazione e il trasporto dei servizi nativi di layer 2 come: Ethernet, Frame Relay [8], ATM [7], PPP [52] e HDLC [53];
2. Un LSP, in versione TE, per i servizi nativi basati su MPLS-TE/GMPLS, ad esempio le MPLS-VPN [32].

Uno strumento essenziale per l'OAM è il Generic Associated Channel (G-ACh) [57], un PW dedicato esclusivamente al trasporto dei messaggi di controllo e monitoraggio dell'*MPLS-TP layer*. Esso utilizza label riservate, in modo tale che sia sempre riconoscibile all'interno della rete.

Un forte requisito delle reti di trasporto è l'operatività attraverso un *Network Management System* (NMS) [58], una piattaforma centralizzata che permette di monitorare la performance, rilevare/notificare il guasto e di configurare sia il routing che il signaling staticamente, senza l'ausilio di un Control Plane. Tuttavia, una gestione di tipo dinamico, della rete MPLS-TP, è prevista tramite l'utilizzo di un piano di controllo comune per interfacce eterogenee, così come implementata da GMPLS.

Riassumendo, MPLS-TP è stato introdotto come variante di MPLS orientata alle reti di trasporto e ha come obiettivi:

- Estendere le principali caratteristiche di MPLS e rimuovere quelle non necessarie per il soddisfacimento dei requisiti previsti dalla RFC 5654;
- Costituire una nuova tecnologia di rete a commutazione di pacchetto con le stesse caratteristiche di una a commutazione di circuito, sia in termini architetturali che in termini di intervento (inferiori ai 50ms);
- Gestire e monitorare in modo automatico le situazioni di guasto attraverso un insieme di funzionalità OAM;
- Provvedere efficaci meccanismi di troubleshooting e di verifica dei termini di SLA;
- Fornire ai propri clienti la possibilità di trasportare efficientemente, su MPLS-TP, servizi nativi eterogenei (dal traffico Internet alla telefonia tradizionale), basati su MPLS-TE, GMPLS e PW.

5. Implementazione del testbed MPLS

Questo capitolo descrive l'attività di progettazione, implementazione e testing di una piattaforma pilota per lo studio sperimentale delle tecnologie MPLS, allo scopo di ottenere un ambiente di test degli attuali protocolli della famiglia MPLS. Inoltre, tale ambiente rappresenta una base di partenza per la sperimentazione di nuove tecnologie e altri protocolli di rete. In particolare, sono stati implementati MPLS, LDP, OSPF e BGP. Successivamente, sono stati eseguiti dei test di funzionalità del sistema i cui risultati di tali test sono qui riportati e commentati.

L'hardware impiegato per lo sviluppo del testbed è consistito in sei PC, ognuno dotato di cinque schede di rete; ciascun PC rappresenta un nodo della rete MPLS da implementare.

5.1. Selezione del Software

Per lo sviluppo del testbed, sono stati presi in considerazione alcuni prodotti open source disponibili online, scegliendo tra questi il candidato ideale per l'implementazione.

MPLS-Linux

Il primo software esaminato è il progetto "*MPLS-Linux*" [18]. Il forum relativo al gruppo di sviluppo è accessibile da [59].

MPLS-Linux è il più diffuso applicativo open source che permette di realizzare MPLS in un ambiente Linux. E' costituito da due componenti:

1. *mpls-linux* [60]: un insieme di funzionalità che implementano il Data Plane MPLS su Kernel Linux. Esso prevede pieno supporto per:
    a. Interfacce Ethernet;
    b. Label multiple annidate (vedi Capitolo 3);
    c. Associazione di un'etichetta in uscita per ogni entry della routing table di Linux.
2. *ldp-portable* [61]: rappresenta la parte di Signaling Plane di MPLS, un'implementazione della RFC 3036, che prevede principalmente:
    a. Due modalità di distribuzione delle etichette: Downstream-on-Demand (DoD) e Unsolicited Downstream (UD) (vedi capitolo 3). Quest'ultima è quella predefinita;
    b. Un'eventuale distribuzione stabilita via Policy.

Esiste la possibilità di integrare il progetto *MPLS-Linux* con la suite open source di protocolli denominata "*Quagga*" [19]. Quagga supporta una serie di algoritmi di routing quali OSPF, BGP. L'approccio utilizzato da Quagga prevede la cooperazione tra diversi demoni al fine di creare la routing table su cui basare l'inoltro. Per il testbed sono richiesti i seguenti moduli di Quagga [62]:

1. *ospfd:* il demone che rappresenta il protocollo di routing OSPF versione 2 [26];
2. *bgpd:* il demone relativo al protocollo BGP versione 4 [29];
3. *zebra:* il "Kernel routing table manager", che si occupa dell'inter-operazione tra i diversi protocolli.

Un'architettura multi-processo come questa, presenta il vantaggio di una struttura modulare a cui è possibile aggiungere altri protocolli oltre a quelli già presenti, rendendo estremamente scalabile ed espandibile l'intera suite. Questo è proprio l'approccio utilizzato per *ldp-portable*, il quale viene integrato in Quagga (con la possibilità di implementare anche IS-IS [39] e RIP [42]), attraverso una patch applicata direttamente a livello di codice sorgente. Successivamente alla sua installazione, LDP è utilizzabile tramite un demone denominato "*ldpd*". E' in sviluppo anche il supporto per RSVP [38].

Come aspetto negativo, una struttura di questo tipo presenta un numero di file di configurazione proporzionale al numero dei demoni configurati e può risultare, pertanto, oneroso dal punto di vista della gestione funzionale.

zMPLS

zMPLS [69] è un software per l'implementazione del Control/Data Plane MPLS in ambiente Linux. Esso si basa sulla suite di protocolli open source *Zebra* [70]. zMPLS, come MPLS-Linux, prevede alcune modifiche al Kernel Linux per il supporto di MPLS a livello di Data Plane e l'integrazione di questo con le componenti di Control Plane, un processo per ogni protocollo di routing/signaling:

- *zmpls*: è il demone principale, il cui scopo è quello di ricevere le informazioni dalle altre componenti di Control Plane, come l'interazione con ldpd e la gestione degli LSP;
- *ldpd*: è il demone che si occupa della distribuzione delle etichette e interagisce esclusivamente con zmpls;
- *rsvpd*: è il demone che si occupa della segnalazione dei tunnel TE;
- *MPLS Forwarding Engine*: è la componente di Data Plane. Nello specifico questo modulo è una patch utile ad abilitare l'inoltro MPLS a livello di Kernel.

La documentazione disponibile online è molto scarsa, sebbene scaricando i sorgenti del progetto sia possibile ottenere qualche informazione aggiuntiva. Purtroppo, allo stato attuale, non è supportato il protocollo BGP e la suite Zebra, sui cui si basa l'applicativo, risulta obsoleta (l'ultima release è datata 2005). Anche lo sviluppo di zMPLS stesso risulta fermo al 2006, alla sua versione 0.95-alfa, altamente instabile. Inoltre, sebbene sia previsto un demone per RSVP, l'unico protocollo di routing supportato è OSPF, mentre di IS-IS o RIP non si fa menzione.

Analisi Comparativa

Sotto il profilo dello sviluppo MPLS-Linux è ad uno stadio più avanzato. Di questo prodotto sono già state rilasciate numerose beta anche nell'ultimo anno, mentre zMPLS è fermo ad una versione alfa del 2005.

Per quanto riguarda le caratteristiche, l'offerta funzionale è molto simile, sebbene il fatto che MPLS-Linux sia basato su Quagga e non su Zebra (da cui Quagga stesso deriva), fa sì che i protocolli di routing supportati dall'applicativo siano più numerosi rispetto a quelli previsti da zMPLS. Quest'ultimo include anche RSVP, che per MPLS-Linux risulta ancora in fase di sviluppo.

Entrambi i progetti sono strutturati secondo un approccio modulare, permettendo una grande flessibilità nel caso si vogliano implementare caratteristiche addizionali. E'

previsto un insieme di moduli del Kernel aggiuntivi per il supporto del Data Plane MPLS e l'implementazione delle componenti di Control Plane, nella forma di processi Linux. Le due proposte adottano una struttura comune: presentano un demone principale, che si occupa di coordinare le informazioni ricevute dai protocolli di routing/signaling e di gestire l'inoltro al livello inferiore; inoltre, sono inclusi alcuni demoni ausiliari come ospfd e ldpd.

Il supporto per zMPLS è assente. Non esistono community di sviluppatori o di utenti al riguardo e l'unica possibilità di reperire informazioni utili è dalla scarsa documentazione allegata ai sorgenti. MPLS-Linux rende disponibile, invece, due forum: quello degli utenti e quello degli sviluppatori.

Infine, strettamente collegato al punto precedente, la documentazione di MPLS-Linux è molto più ampia, sebbene estremamente eterogenea, rispetto a quella di zMPLS. Per quest'ultimo prodotto, infatti, non esiste alcun punto di riferimento online, e gli unici documenti sono costituiti dai file/commenti incompleti, allegati al codice sorgente.

Considerati i punti precedenti, ne consegue che per motivi legati soprattutto allo stadio di sviluppo e alla notevole disponibilità di supporto/documentazione, il software candidato all'installazione sulla piattaforma sia MPLS-Linux.

5.2. Installazione

Per implementare il testbed basato su MPLS-Linux, è indispensabile installare dapprima la componente mpls-linux e successivamente una versione di Quagga, opportunamente integrata da ldp-portable.

La versione 1.971 di mpls-linux messa a disposizione degli sviluppatori [59], richiede il Kernel 2.6.32.27, mentre Quagga 0.99.15 è stato esteso con ldp-portable versione 0.900.

Collegandosi al repository GIT [63] del progetto, è possibile scaricare le tre cartelle principali di cui avremo bisogno:

- `mpls-linux-1.971:`

  ```
  git clone git://mpls-linux.git.sourceforge.net/gitroot/mpls-linux/mpls-linux
  ```

- `mpls_ldp_portable:`

  ```
  git clone git://repo.or.cz/mpls-ldp-portable.git
  ```

- `mpls_ldp_quagga:`

```
git clone git://mpls-linux.git.sourceforge.net/gitroot/mpls-linux/quagga
```

Per abilitare MPLS sui nodi del testbed è necessario dapprima eseguire una ricompilazione del Kernel, applicando la patch contenuta nella cartella `mpls-linux-1.971`. I dettagli implementativi di questa fase sono consultabili in [Appendice.A.1].

Il passo successivo è l'installazione di ldp-portable. E' fondamentale preparare l'ambiente di lavoro e avviare successivamente la compilazione dei sorgenti. I dettagli di questa fase sono riportati in [Appendice.A.2].

Infine, su ciascun PC configurato come router, è opportuno effettuare alcuni aggiustamenti, senza i quali la piattaforma non funzionerebbe correttamente:

- Disabilitare "Reverse Path Filter" [66]:

  ```
  echo 0 >/proc/sys/net/ipv4/conf/all/rp_filter
  ```

  e in aggiunta:

  ```
  sysctl -w net.ipv4.conf.mpls0.rp_filter=0
  ```

  ```
  sysctl -w net.ipv4.conf.<interface_name>.rp_filter=0
  ```

  modificando opportunamente il nome di ogni interfaccia;

- Abilitare "IP Forward" [67]:

  ```
  sysctl -w net.ipv4.ip_forward=1
  ```

- Modificare i nomi delle interfacce:

  ```
  nano /etc/udev/rules.d/70-persistent-net.rules
  ```

Infine, assicurarsi che siano attivi i seguenti moduli del Kernel, con il comando `lsmod`:

```
modprobe mpls4
```

```
modprobe mplsbr
```

```
modprobe mpls_tunnel
```

Eseguiti i passi precedenti è possibile iniziare il testing vero e proprio della rete pilota.

5.3. Test Funzionali

In questa ultima parte del lavoro di tesi si è voluto dar prova delle potenzialità del testbed implementato, eseguendo e commentando le catture del traffico e dando per ogni esempio riportato, riscontro a quanto affrontato nella parte teorica. Inoltre, è stato anche possibile testare il funzionamento approfondito di LDP. La distribuzione delle etichette è stata monitorata e comparata rispetto a quanto visto nel caso di una assegnazione da management.

Configurazione e test di una rete IP/MPLS con route statiche

Il primo scenario di test [Figura 35] prevede la realizzazione di una rete IP/MPLS in cui tutti i parametri sono configurati da management. Ciò implica che tutte le entry di NHLFE, ILM e XC debbano essere inserite manualmente. Inoltre, si è voluto emulare HOST A e HOST B connettendo le loro interfacce Dummy [68], rispettivamente a R1 e R2. I dettagli implementativi sono disponibili in [Appendice.A.3].

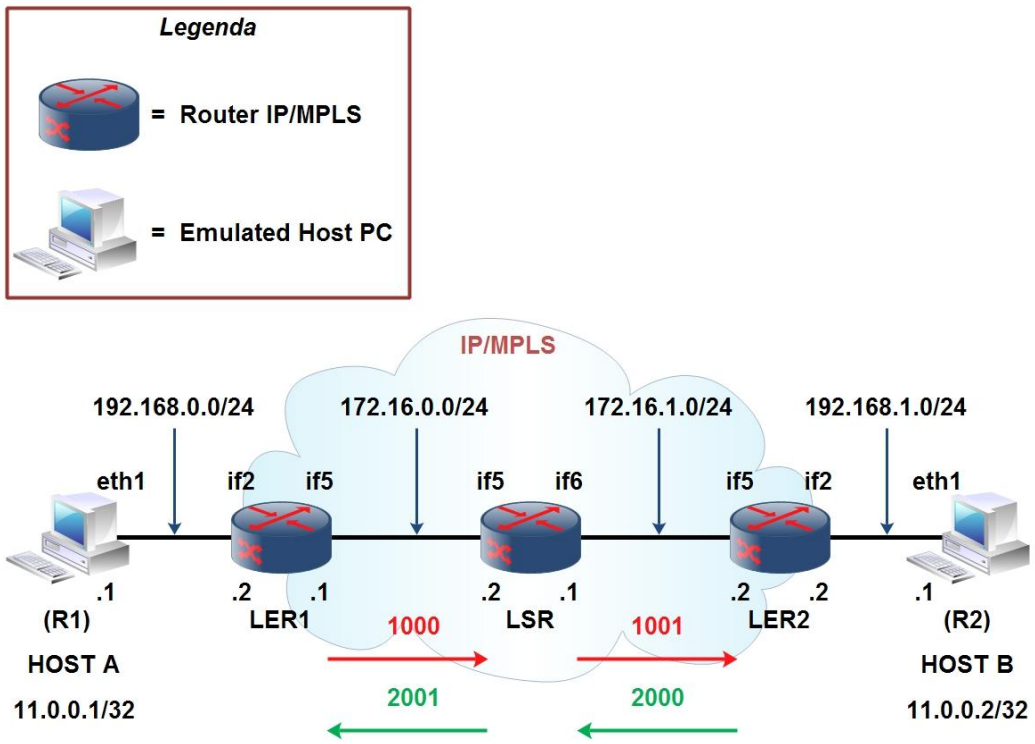

Figura 35. Configurazione di una rete IP/MPLS con route statiche

Il primo test prende in considerazione un ping tra HOST A e HOST B, in cui si vuole dimostrare la correttezza dell'assegnazione da management di tutte le etichette riportate in [Figura 35]. L'indirizzo sorgente è 11.0.0.1 mentre quello destinazione è 11.0.0.2. Quindi, viene definita una FEC per tale indirizzo da LER1, LSR e LER2.

HOST A: `ping -I 11.0.0.1 11.0.0.2`

```
[...]

64 bytes from 11.0.0.2: icmp_seq=3 ttl=61 time=0.396 ms
```

Dall'interfaccia *if2* di LER1 è possibile catturare i datagram IP, "*ICMP echo request*" e "*ICMP echo reply*", relativamente al messaggio in andata e a quello in ritorno. Da notare anche il campo *id*, che si mantiene tale in tutti i successivi segmenti della rete.

LER1-if2: `tcpdump -i if2 -nvv`

```
[...]

16:37:07.681803 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1),
length 84)

    11.0.0.1 > 11.0.0.2: ICMP echo request, id 39175, seq 3, length 64

[...]

16:37:07.682121 IP (tos 0x0, ttl 61, id 43271, offset 0, flags [none], proto ICMP
(1), length 84)

    11.0.0.2 > 11.0.0.1: ICMP echo reply, id 39175, seq 3, length 64
```

In LER1 il datagram IP viene etichettato, all'andata con label 1000 e al ritorno con label 2001. I parametri di *Exp* [1] e *BoS* [1] sono configurati ai rispettivi valori di default (in quanto il traffico non è prioritario e vi è un'unica etichetta apposta al pacchetto). Il *TTL* [1] è ereditato dal pacchetto IP, che una volta incapsulato non verrà più modificato durante il passaggio nella rete MPLS.

LSR-if5: `tcpdump -i if5 -nvv`

```
[...]

16:37:07.626360 MPLS (label 1000, exp 0, [S], ttl 63)

        IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto ICMP (1), length
84)

    11.0.0.1 > 11.0.0.2: ICMP echo request, id 39175, seq 3, length 64

[...]

16:37:07.626564 MPLS (label 2001, exp 0, [S], ttl 62)

        IP (tos 0x0, ttl 63, id 43271, offset 0, flags [none], proto ICMP (1),
length 84)

    11.0.0.2 > 11.0.0.1: ICMP echo reply, id 39175, seq 3, length 64
```

In LSR viene eseguito lo swap delle etichette: all'andata, scambiando la label 1000 in ingresso con la 1001 in uscita; al ritorno, scambiando la label 2000 in entrata con la 2001 in uscita.

LER2-if5: `tcpdump -i if5 -nvv`

```
[...]

16:37:07.594165 MPLS (label 1001, exp 0, [S], ttl 62)

        IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto ICMP (1), length
84)

    11.0.0.1 > 11.0.0.2: ICMP echo request, id 39175, seq 3, length 64

[...]

16:37:07.594252 MPLS (label 2000, exp 0, [S], ttl 63)

        IP (tos 0x0, ttl 63, id 43271, offset 0, flags [none], proto ICMP (1),
length 84)

    11.0.0.2 > 11.0.0.1: ICMP echo reply, id 39175, seq 3, length 64
```

In LER2 viene effettuato il pop dell'etichetta 1001 aggiunta dall'LSR al passo precedente, aggiornato il TTL, che passa da 62 (presente nello shim header) a 61. Quest'ultimo campo del pacchetto IP viene modificato prima di effettuare l'inoltro verso 11.0.0.2. Esaminando il traffico di ritorno si può verificare come LER2 esegua il push dell'etichetta 2000 e passi il relativo pacchetto etichettato, contente il messaggio "ICMP echo reply" a LSR.

HOSTB-eth1: `tcpdump -i eth1 -nvv`

```
[...]

16:37:07.625404 IP (tos 0x0, ttl 61, id 0, offset 0, flags [DF], proto ICMP (1),
length 84)

    11.0.0.1 > 11.0.0.2: ICMP echo request, id 39175, seq 3, length 64

[...]

16:37:07.625410 IP (tos 0x0, ttl 64, id 43271, offset 0, flags [none], proto ICMP
(1), length 84)

    11.0.0.2 > 11.0.0.1: ICMP echo reply, id 39175, seq 3, length 64
```

Infine, dall'interfaccia *eth1* di HOST B sono stati catturati i datagram IP in uscita (all'andata) e in entrata (al ritono) dalla rete MPLS.

Questo primo semplice scenario ha una duplice finalità: la prima, di verificare quanto visto nella teoria a proposito dell'inoltro basato sulle etichette MPLS, la struttura delle label e la relazione che esiste tra i parametri dello shim header e quelli dell'header IP; la seconda, testare il funzionamento della piattaforma implementata, in particolare la corretta gestione del Data Plane MPLS da parte del Kernel Linux.

Configurazione e test di una rete IP con OSPF e BGP

Col secondo test [Figura 36] si desidera introdurre alcuni dei componenti di routing plane della piattaforma di test, in particolare OSPF e BGP. Vengono presi in considerazione due Autonomous System. AS1 è un ISP con due router di bordo BGP, AS1BR1 e AS1BR2, connessi al router BGP AS2BR di AS2, un piccolo Autonomous System che necessita di rendere raggiungibile la rete 192.123.0.0/24. AS2 dispone di un link primario utilizzato per le comunicazioni in entrata/uscita e di un link di backup nel caso di problemi sul primo. Per comodità d'uso adotteremo indirizzi IP privati, sapendo che nella realtà la comunicazione tra AS avviene tramite indirizzi pubblici. I dettagli implementativi sono disponibili in [Appendice.A.4].

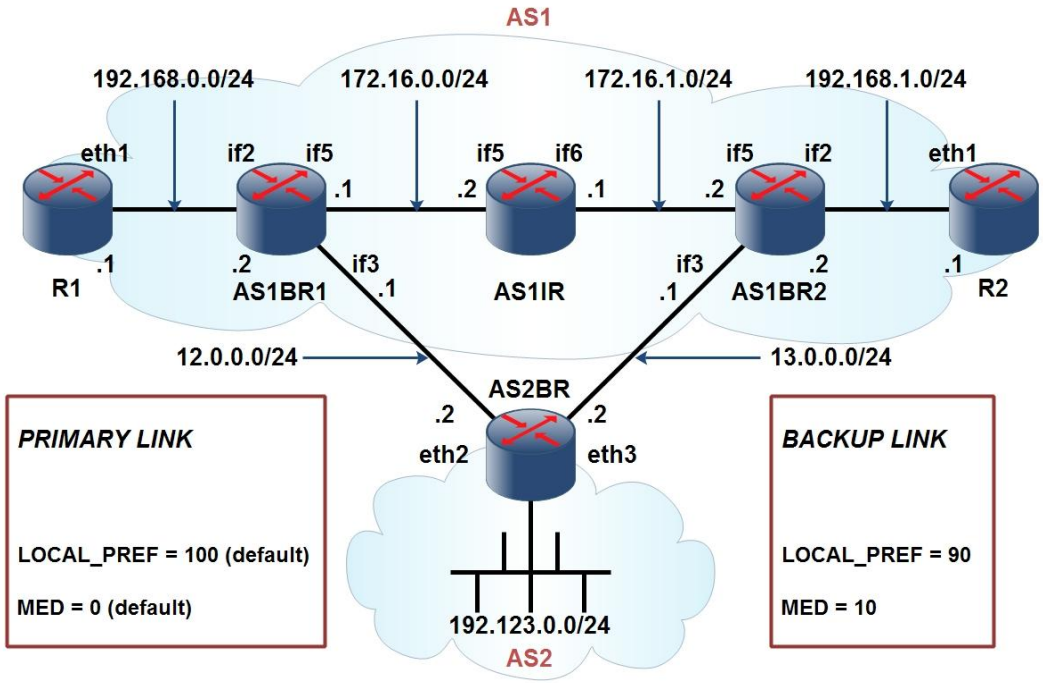

Figura 36. Configurazione di una rete IP con OSPF e BGP

Innanzitutto, viene effettuato un test della connettività tra AS2 e AS1 con il comando *tracepath* di Linux, tracciando il percorso effettuato dal traffico in uscita da AS2BR verso R2. Si può notare che il traffico viene effettivamente instradato sul link primario, 12.0.0.0/24 e che l'obiettivo perseguito dalla configurazione è raggiunto:

AS2BR: `tracepath 192.168.1.1`

```
     1:  12.0.0.2 (12.0.0.2)                                   0.100ms pmtu 1500
     1:  12.0.0.1 (12.0.0.1)                                   0.149ms
     1:  12.0.0.1 (12.0.0.1)                                   0.158ms
     2:  172.16.0.2 (172.16.0.2)                               0.272ms
     3:  172.16.1.2 (172.16.1.2)                               0.453ms
     4:  192.168.1.1 (192.168.1.1)                             2.163ms reached
         Resume: pmtu 1500 hops 4 back 61
```

Tentiamo ora di andare nel dettaglio, esaminando scrupolosamente i messaggi BGP scambiati, e verificando gli attributi delle NLRI annunciate.

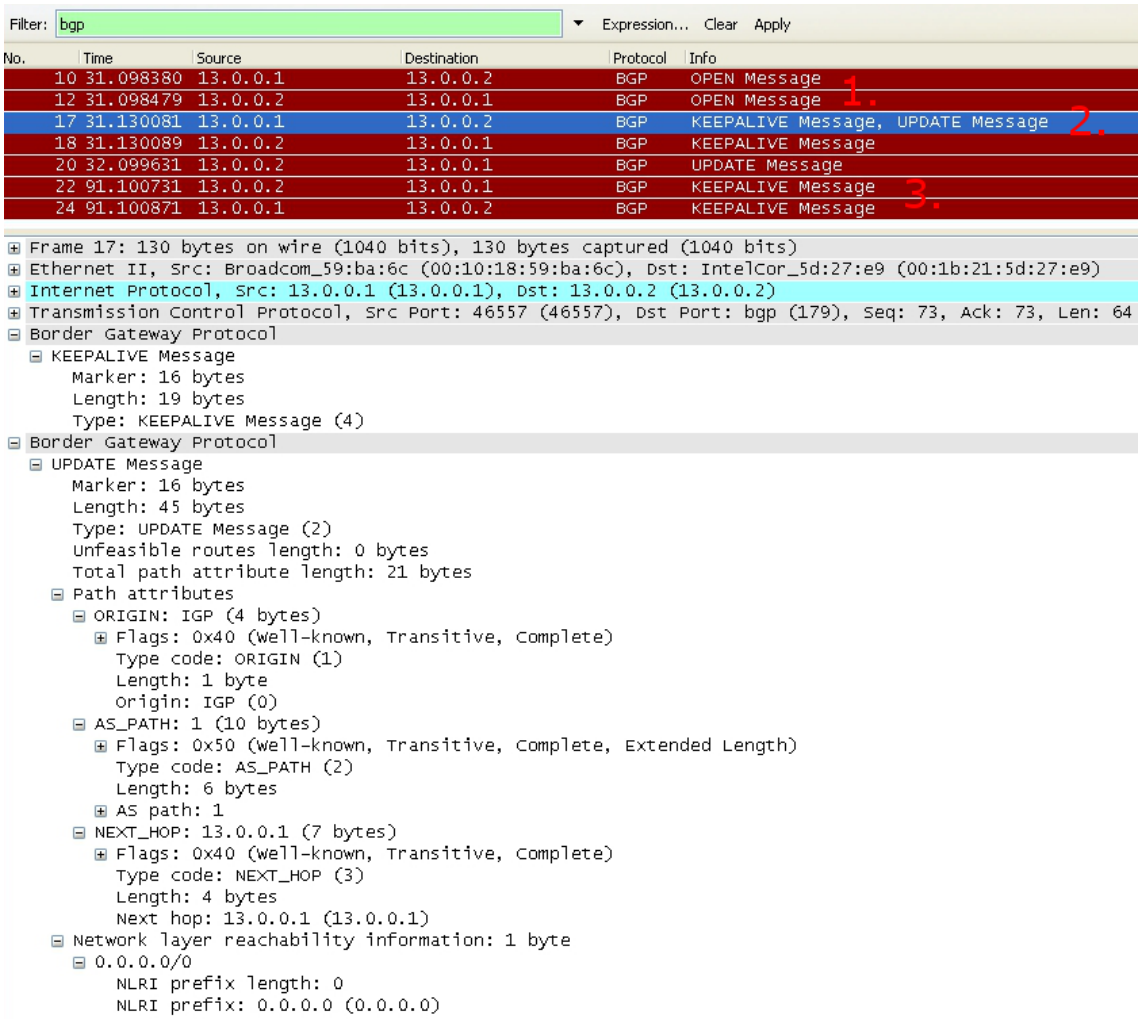

Figura 37. Sessione BGP: (1.) Instaurazione (2.) Update (3.) KeepAlive

In [Figura 37.] è possibile verificare l'instaurazione di una sessione BGP tra AS1BR2 e AS2BR:

1. La prima fase è caratterizzata dallo scambio dei messaggi Open, in cui vengono negoziati i paramentri della connessione TCP [Figura 37.1.];

2. I messaggi KeepAlive in questo caso servono a confermare l'avvio effettivo della sessione. E' inviato anche un messaggio Update [Figura 37.2.] di cui è evidenziata la struttura: viene annunciata la rotta di default (campo "NLRI prefix: 0.0.0.0 (0.0.0.0)") dall'AS 1 (attributo AS_PATH), con next-hop BGP 13.0.0.1 (attributo NEXT_HOP), AS1BR2. Le rotte del router AS2BR vengono analogamente annunciate nel messaggio Update successivo;

3. La connessione viene mantenuta attraverso l'invio di KeepAlive periodici (ogni 60 secondi).

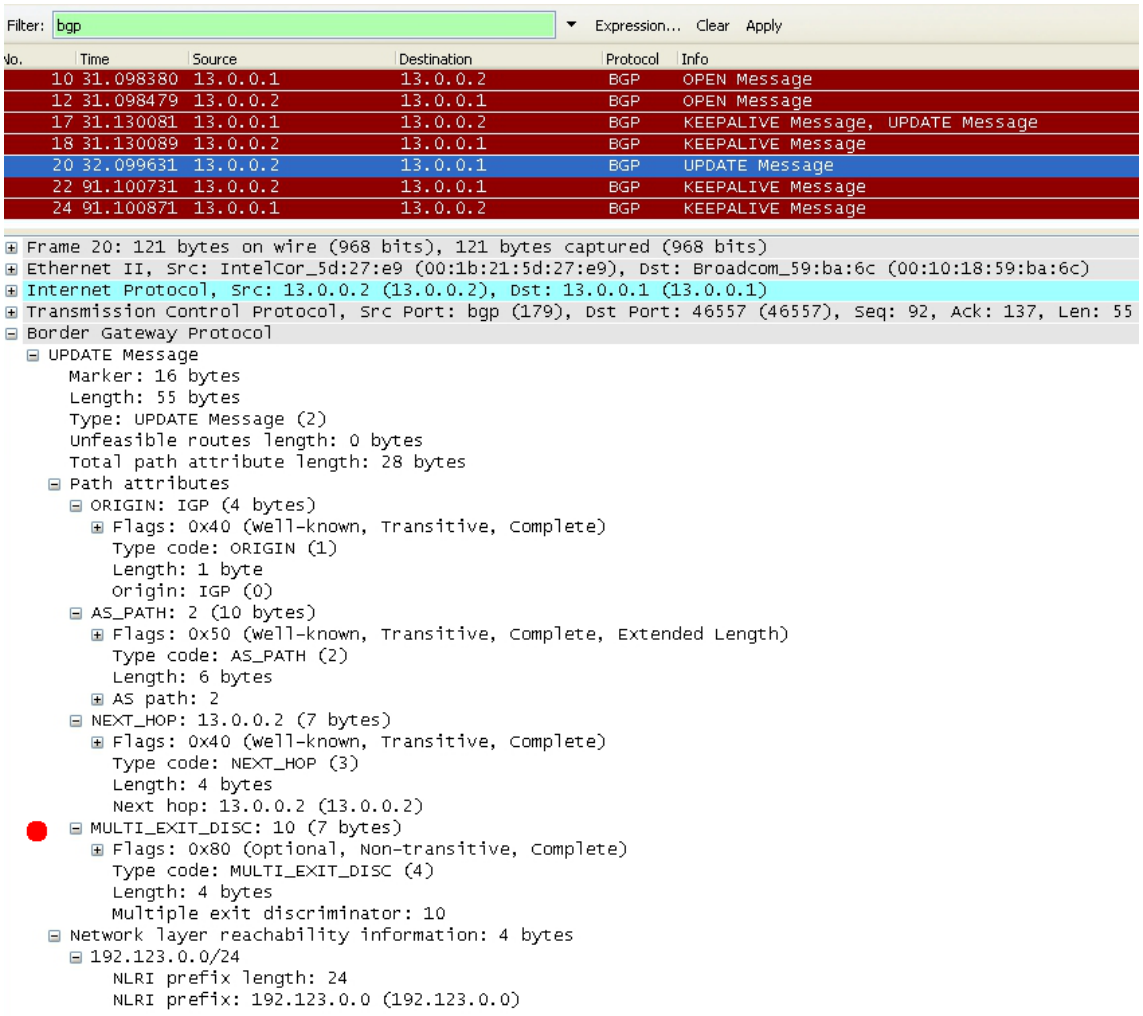

Figura 38. Struttura di un messaggio eBGP Update con attributo MED

Nel messaggio Update in [Figura 38.], inviato da AS2BR a AS1BR2, si può notare che oltre agli attributi presenti in [Figura 37.], viene aggiunto il paramentro opzionale MED che indica il costo sul link. Di default tale valore è 0. In questo caso, indicando il valore 10, AS2 invia la richiesta AS1 utilizzare come link principale il collegamento 12.0.0.0/24, riservando al link 13.0.0.0/24 un ruolo di backup, in caso di fallimento del primo collegamento. Le destinazione annunciata è contenuta nel campo NLRI e corrisponde alla rete 192.123.0.0/24.

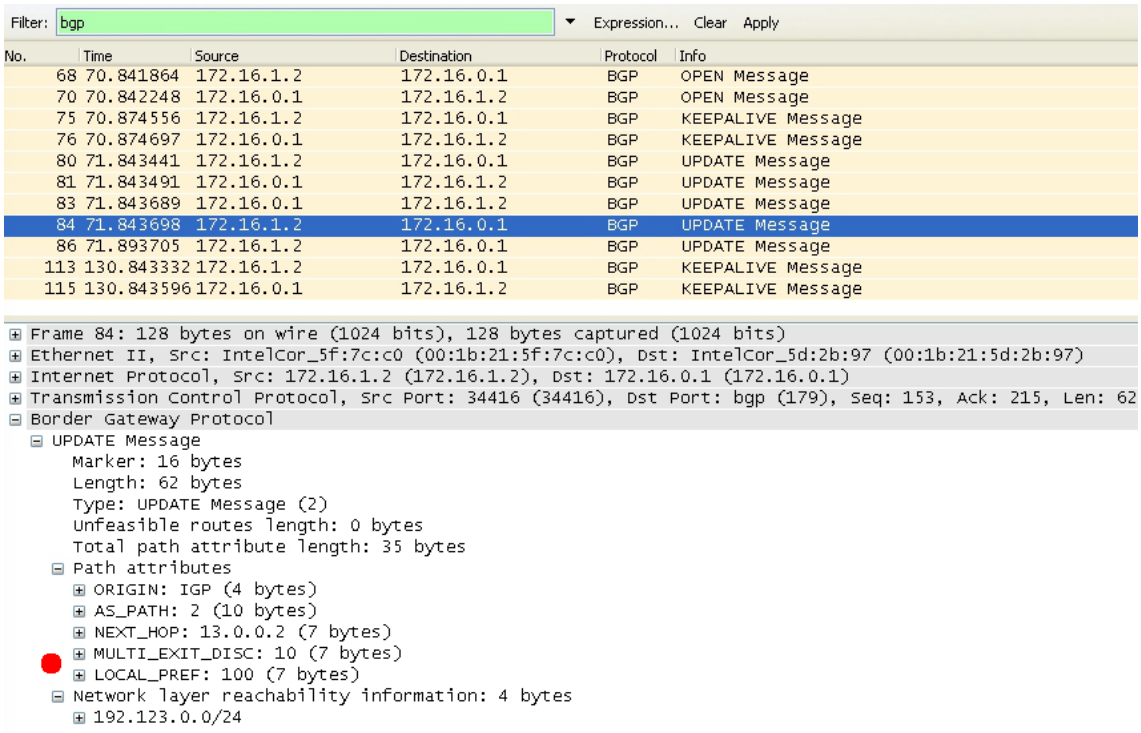

Figura 39. Struttura di un messaggio iBGP Update con attributi MED e LOCAL_PREF

In [Figura 39.] si evidenzia la struttura del messaggio iBGP inviato da AS1BR2 ad AS1BR1, relativo alla NLRI annunciata in [Figura 38.], 192.123.0.0/24. Si può verificare la propagazione dell'attributo MED (posto uguale a 10) all'interno di AS1 e dell'attributo LOCAL_PREF di default (100), che si ricorda avere solo valenza locale all'Autonomous System stesso. Sono specificati, infine, i campi NEXT_HOP e AS_PATH. Il primo è il router iBGP mittente (AS1BR2), il secondo corrisponde alla lista di Autonomous System attraversati dall'annuncio, in questo caso solo AS2.

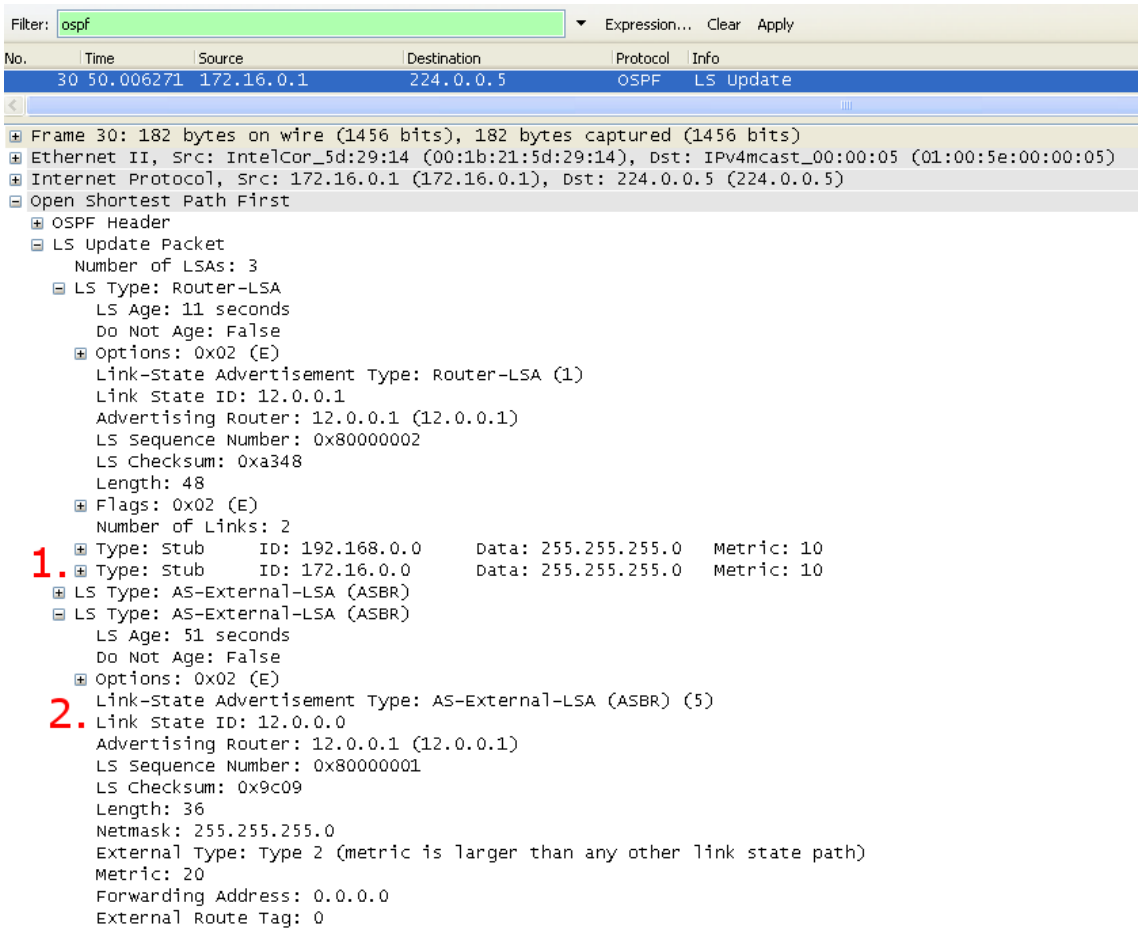

Figura 40. Struttura degli annunci OSPF: (1.) Router-LSA (2.) AS-External-LSA

In [Figura 40.] è riportata la struttura di un LS Update OSPF inviato da AS1BR1 al suo gruppo di multicast, 224.0.0.5. In particolare vengono evidenziati due LSA contenunti nell'annuncio:

1. Le reti annunciate ai vicini da AS1BR1, 192.168.0.0/24 e 172.16.0.0/24, contenute in un Router-LSA, entrambe con metrica 10 [Figura 40.1.];

2. Essendo AS1BR1 un router di bordo BGP, esso redistribuisce anche la rete esterna 12.0.0.0 (indicandone il Link-State ID 12.0.0.0 e la sottomaschera 255.255.255.0) [Figura 40.2.]. Il costo associato a questa rete è, di default, una metrica esterna di tipo 2, con valore 20 .

Al termine di questa seconda fase di testing, è stato verificato che la piattaforma sperimentale implementa correttamente OSPF e BGP così come approfondito dalla prima parte di questo lavoro di tesi. Sono stati verificati anche i parametri di efficienza della rete, notando che i tempi di convergenza sono quasi istantanei nel caso di OSPF e

leggermente superiori, nell'ordine dei minuti, per BGP. La rete pilota dispone, quindi, di uno stabile Control Plane che può essere espanso da ulteriori nuovi lavori, adottando protocolli di routing aggiuntivi disponibili nella suite Quagga, come RIP [42] e IS-IS [39].

Configurazione e test di una rete IP/MPLS con OSPF e LDP

Il terzo ed ultimo test [Figura 37] presenta una rete IP/MPLS su cui sono implementati OSPF e LDP, composta da un LSR e da due LER. A questi sono collegati HOST A e HOST B, due PC che utilizzano i router vicini come default gateway per scambiarsi traffico. Le configurazioni dettagliate sono disponibili in [Appendice.A.5].

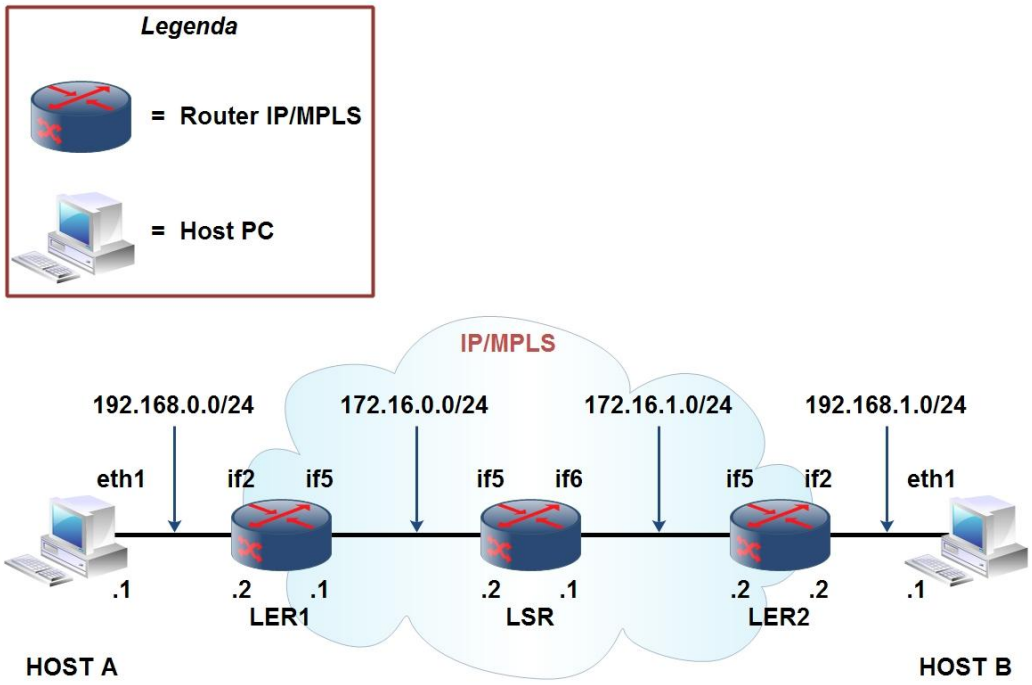

Figura 41. Configurazione di una rete IP/MPLS con OSPF e LDP

Assumendo il corretto funzionamento del Routing Plane testato nel paragrafo precedente, viene catturato e commentato il traffico di Signaling Plane, e nello specifico, relativo al protocollo LDP.

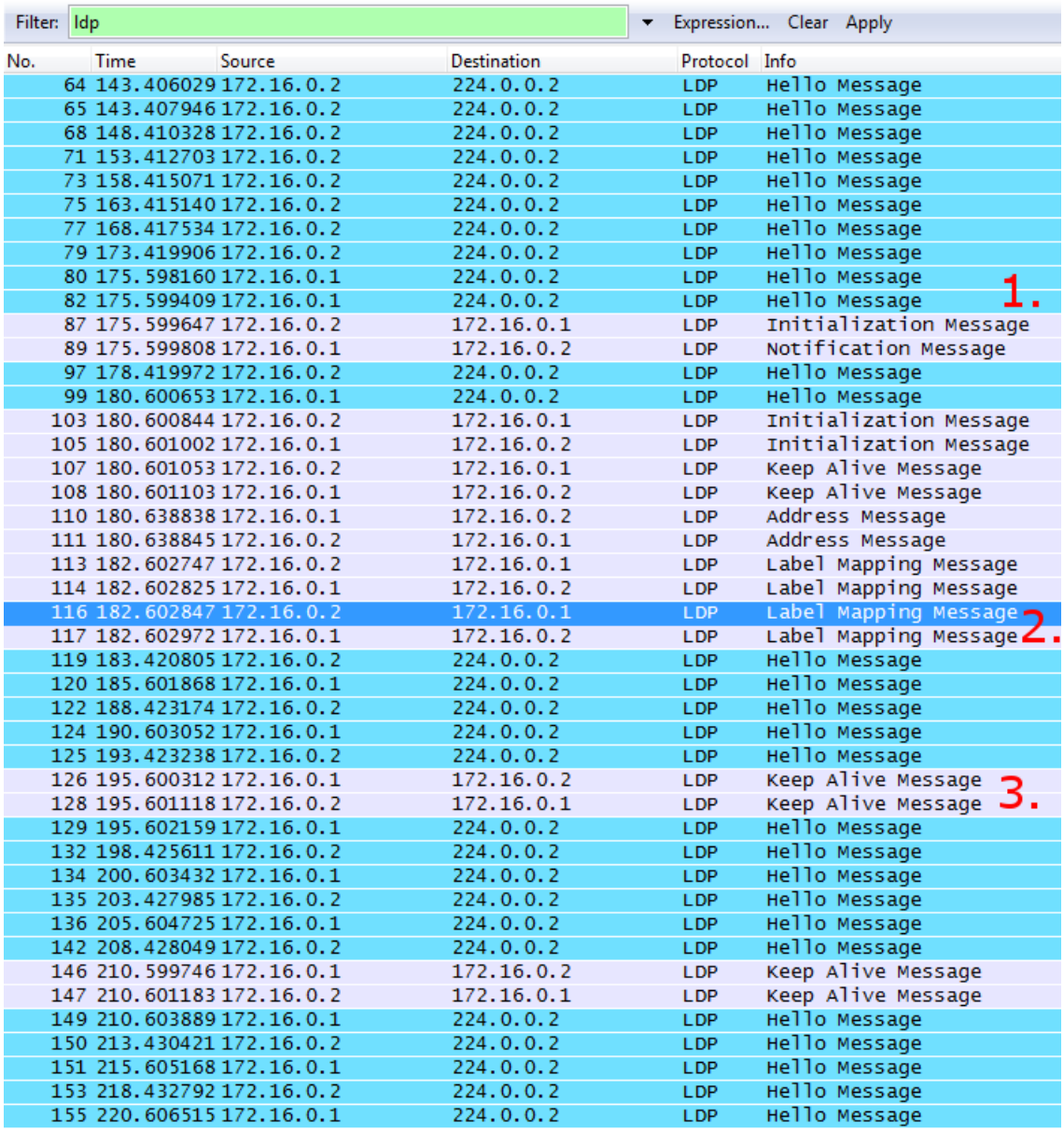

Figura 42. Sessione LDP: (1.) Avvio (2.) Sincronizzazione (3.) Mantenimento

In [Figura 42.] è verificabile l'instaurazione di una sessione LDP tra LSR e LER1. Nello specifico possono essere individuate le seguenti fasi:

1. Apertura della sessione [Figura 42.1.], seguita dalla negoziazione dei parametri TCP e lo scambio dei propri router-ID;

2. Scambio delle mappature locali [Figura 42.2.], che saranno approfondite in [Figura 43.];

3. Mantenimento della connessione con periodici messaggi KeepAlive (di default ogni 15 secondi) [Figura 42.3.].

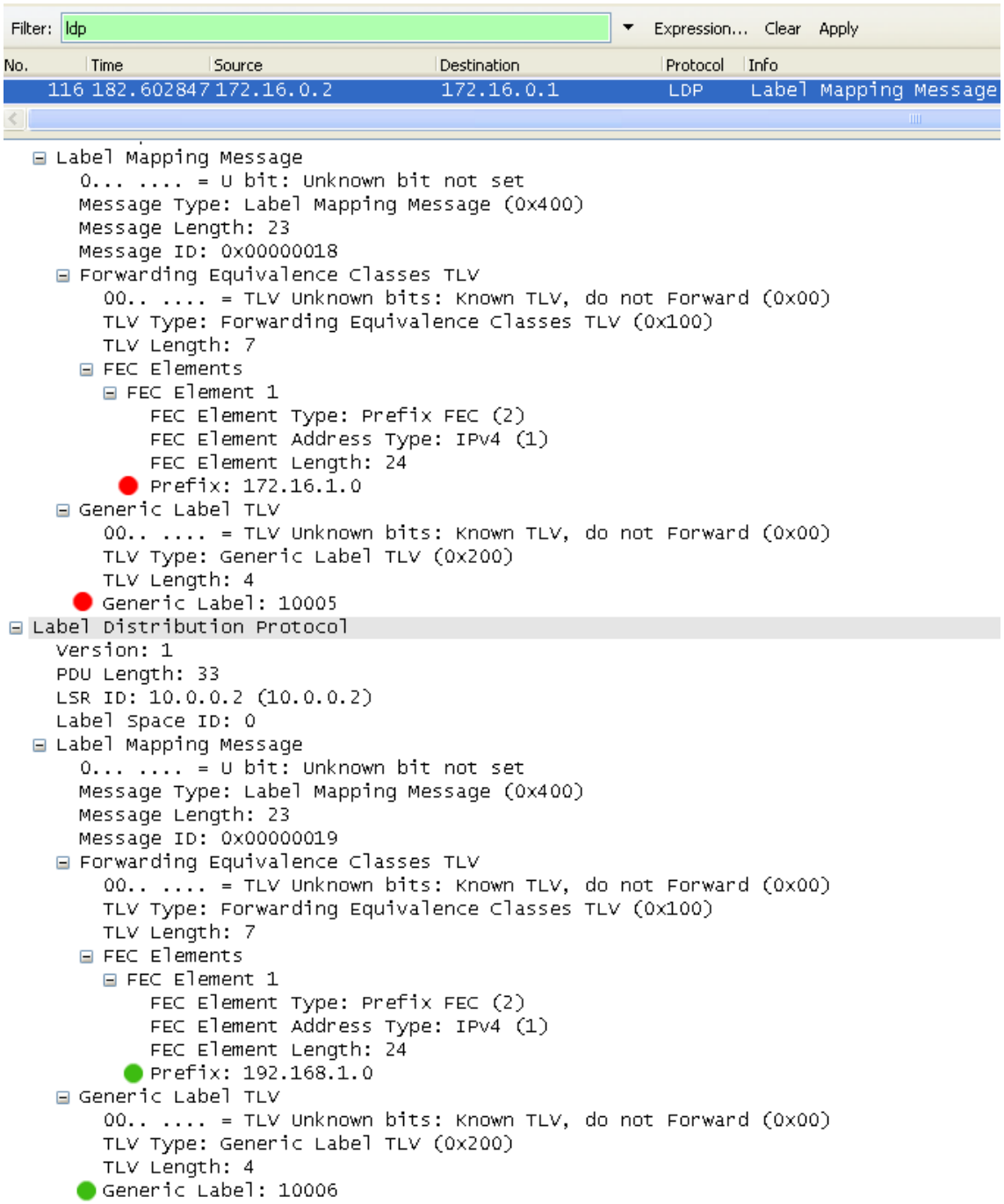

Figura 43. Struttura di un annuncio LDP

In [Figura 43.] viene approfondita la struttura di una mappatura LDP inviata da LSR a LER1. In particolare, LSR annuncia che la sua label locale (in entrata) 10005 corrisponde alla FEC 172.16.1.0/24 mentre alla label 10006 è stata abbinata la FEC 192.168.1.0/24.

Questi abbinamenti sono attribuiti dinamicamente sulla base dei prefissi contenuti nella routing table e acquisiti via Control Plane tramite OSPF, così come dimostrato dalla [Figura 44.].

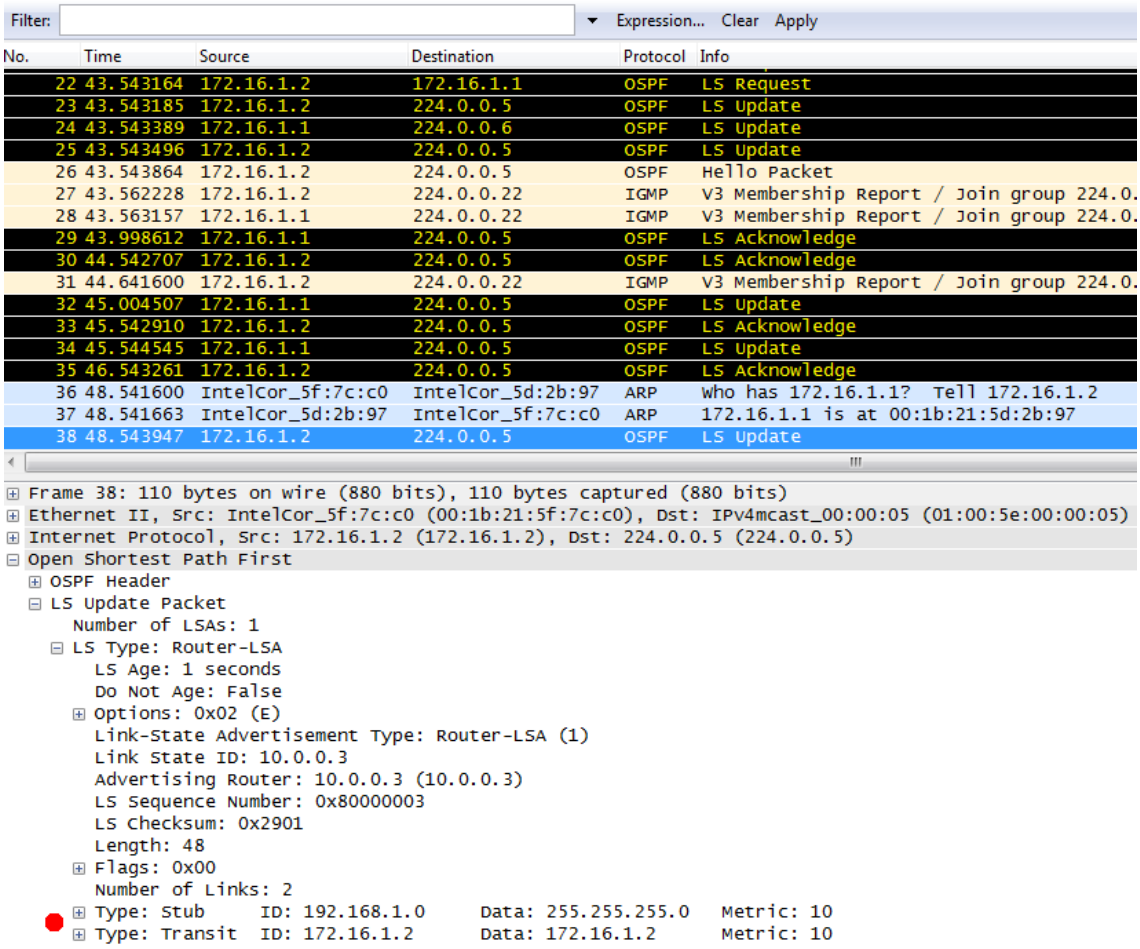

Figura 44. Annuncio OSPF relativo alle FEC utilizzate da LDP

In [Figura 44.] è presentato l'annuncio OSPF (Router-LSA) delle reti direttamente connesse a LER2. Queste sono le FEC a cui LSR, una volta ricevuto il messaggio, e installato i prefissi nella routing table, attribuirà le label in [Figura 43.]. L'annuncio ha come destinatario il gruppo Multicast di OSPF, 224.0.0.5, a cui appartengono LER1, LSR e LER2.

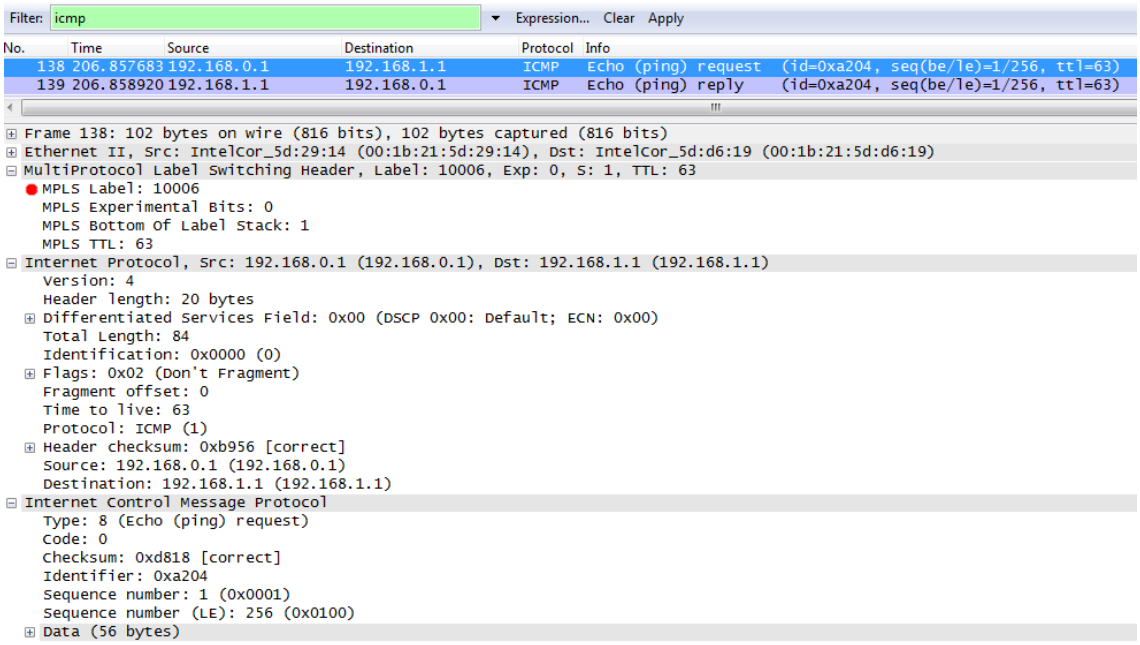

Figura 45. Shim Header MPLS in ambiente LDP

In [Figura 45.] è stata testata la connettività tra HOST A (192.168.0.1) e HOST B (192.168.1.1) e catturato il traffico MPLS nel segmento compreso tra LER1 e LSR. E' possibile apprezzare la corretta etichettatura del messaggio "ICMP echo request" con la label 10006, abbinata proprio alla FEC 192.168.1.0/24 ricevuta via OSPF [Figura 44.] mappata come illustrato in [Figura 43.].

Ciò dimostra come LDP sia stato effettivamente integrato con le primitive di Data Plane di cui il testbed è provvisto. Inoltre, anche l'interoperazione con OSPF si rivela stabile e funzionale, rispettando quanto visto in linea teorica nella prima parte di questa tesi.

Infine, è necessario segnalare le problematiche derivanti dall'utilizzo congiunto di OSPF, LDP e BGP. In questo contesto, si sono verificati dei comportamenti non deterministici da parte dei demoni *zebra* e *ldpd*. Nello specifico, ci si riferisce al crash non prevedibile di uno o di entrambi i demoni, derivante dall'operazione di mappatura di LDP. Quest'aspetto risulta documentato in [59] ma non ne sono approfondite le cause. Un possibile approccio risolutivo potrebbe riguardare il monitoraggio dei log relativi ai demoni coinvolti o l'analisi mirata del codice sorgente, argomentazioni che potrebbero costituire la base per futuri nuovi lavori.

6. Conclusioni

Il presente lavoro ha lo scopo di fornire una panoramica sulle tecnologie MPLS per le reti di trasporto, adottate ormai da tutti gli ISP. Inizialmente introdotto solo in ambito di reti a commutazione di pacchetto basate su IP, MPLS si è evoluto, dapprima integrato per soddisfare esigenze di Traffic Engineering, successivamente opportunamente esteso e generalizzato per meglio adattarsi al contesto di elevata efficienza e disponibilità delle reti di trasporto.

Questa evoluzione è dovuta alla necessità, da parte principalmente degli ISP, di un'infrastruttura di rete unificata sulla quale trasportare il traffico multi-protocollo di layer 2, come Ethernet o ATM, opportunamente etichettato ed incapsulato in pacchetti di livello superiore. Inoltre, si è visto come le tecnologie MPLS contribuiscano alla riduzione sostanziale della routing table dei nodi interni alla rete di trasporto, i quali ne possono beneficiare in termini di carico di lavoro notevolmente diminuito. In aggiunta, adottando tecniche per il Traffic Engineering, è possibile dirottare il traffico di un tratto di rete congestionato, verso un altro sottoutilizzato, contribuendo così al bisogno di ottimizzazione di applicativi real-time, altamente sensibili al ritardo.

Negli ultimi tempi la maggiore efficienza nell'inoltro basato su etichette rispetto a quello basato sul longest prefix match è venuta a mancare: i router moderni infatti, non demandano più questo compito alla CPU bensì a circuiti integrati, ASIC, opportunamente studiati a tal scopo. Ne risulta che l'inoltro di un singolo pacchetto etichettato equivale, in termini di carico computazionale, allo stesso di un pacchetto IP [34].

Sono stati trattati nel dettaglio l'architettura e il funzionamento di MPLS-TE, dedicando ampio spazio a tutte le sue successive evoluzioni, come GMPLS, PseudoWire e MPLS-TP. In ogni caso, tutta la parte teorica è stata introdotta al fine di un migliore approccio della piattaforma che si è andati a creare, considerata il punto cruciale della tesi stessa. A tal fine si è tentato di fornire, per ogni protocollo affrontato, esempi implementativi pratici e un algoritmo riepilogativo di funzionamento.

Nella parte finale del presente lavoro di tesi, si è voluto presentare un testbed basato sul progetto open source MPLS-Linux. Oltre a spiegare nel dettaglio come aggiungere il pieno supporto MPLS al Kernel Linux, sono stati presentati e documentati tre scenari; il primo relativo ad una rete IP/MPLS con route statiche; il secondo inerente l'introduzione dei protocolli OSPF e BGP, sviluppando il contesto dell'interazione tra un AS multihomed cliente ed il relativo ISP; l'ultimo esempio tratta di una rete IP/MPLS integrata con OSPF, in cui le etichette vengono distribuite dinamicamente attraverso LDP. I risultati sono stati discussi e commentati, ponendo di fatto le basi per lo sviluppo di una piattaforma di testing utilizzabile anche, e soprattutto, per la sperimentazione di nuovi scenari e protocolli, nonché per la misurazione dei relativi parametri di funzionamento. E' stato possibile valutare la stabilità del sistema, verificando le difficoltà di una coesistenza, allo stato attuale, dei demoni LDP e BGP di MPLS-Linux [18]. Ulteriori sviluppi futuri potranno partire proprio da tale considerazione, garantendo quanto più possibile una migliore integrazione proprio tra MPLS e BGP, e fornendo caratteristiche al momento non previste come l'ETLA o il supporto per MP-BGP. Procedendo in questo senso sarà fondamentale anche l'implementazione di PseudoWire sulla piattaforma proposta.

Concludendo, l'obiettivo finale raggiunto è stato l'implementazione di un testbed sperimentale per l'analisi in laboratorio delle tecnologie MPLS, ideale a fornire una base operazionale valida per il testing delle tecnologie al momento supportate e per lo sviluppo di ulteriori estensioni o di futuri nuovi lavori.

Appendice A

Appendice.A.1.

Per le macchine che implementeranno MPLS, viene fornita una procedura passo-passo per l'installazione di *mpls-linux* [59].

Step 1 - ottenere i privilegi di root per il nome utente "`user_name`" relativamente alla cartella `/usr/src` e scaricare alcune applicazioni necessarie per la compilazione/installazione dei sorgenti:

```
adduser user_name src
logout and login again as user_name
apt-get install module-init-tools kernel-package libncurses5-dev fakeroot
```

Step 2 - scaricare e decomprimere i sorgenti del nuovo Kernel:

```
cd /usr/src
wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.32.27.tar.bz2
tar xvjf linux-2.6.32.27.tar.bz2
```

Step 3 - configurare il nuovo Kernel copiando il file di configurazione corrente del sistema:

```
cp /boot/config-your_current_kernel_configuration linux-2.6.32.27/.config
make oldconfig
```

Step 4 - applicare la patch fornita da mpls-linux al nuovo Kernel:

```
cp -r ~/mpls-linux/ /usr/src/
mv /usr/src/mpls-linux /usr/src/mpls-linux-1.971
patch -p1 < /usr/src/mpls-linux-1.971/patches/linux-kernel-v2.6.32.27.patch
```

Step 5 - compilare il nuovo Kernel:

```
CONCURRENCY_LEVEL=X fakeroot make-kpkg --append-to-version -mpls-linux-1.971
kernel_image --initrd
```

Dove X – 1 è il numero di CPU utilizzate dalla macchina. Ad esempio, se si sta effettuando la compilazione con una piattaforma quadriprocessore, X è 5.

Step 6 - installare il nuovo kernel:

```
dpkg -i linux-image-2.6.32.27-mpls-linux-1.971_xxx.deb
```

Step 7 - verificare che sia stato creato il file `initrd.img-2.6.32.27-mpls-linux-1.971` in `/boot/` . In caso contrario lo si crei manualmente e si aggiorni il file di configurazione del boot manager di Linux:

```
update-initramfs -c -k 2.6.32.27-mpls-linux-1.971

nano /boot/grub/menu.lst
```

Nel caso non si utilizzi Grub Legacy [64] come boot manager ma il più recente Grub 2 [65], si dovrà scrivere attraverso il comando `nano /etc/grub.d/11_mpls-linux-1.971` uno script analogo al seguente:

```
#!/bin/sh
cat << EOF
menuentry "Ubuntu, with Linux 2.6.32.27-mpls-linux-1.971" {
recordfail
insmod ext2
set root=(hd0,1)
search --no-floppy --fs-uuid --set 399e5745-5a4e-42d1-899a-46d7295b758a
linux /boot/vmlinuz-2.6.32.27-mpls-linux-1.971 root=UUID=399e5745-5a4e-42d1-
899a-46d7295b758a ro find_preseed=/preseed.cfg noprompt quiet splash
initrd /boot/initrd.img-2.6.32.27-mpls-linux-1.971
}
EOF
```

Aggiornare di conseguenza la configurazione attraverso il seguente comando:

```
update-grub
```

E' fondamentale che la riga `initrd` si riferisca all'immagine `.img` creata precedentemente o il sistema risulterà non avviabile (Kernel-Panic).

Step 8 - riavviare il sistema:

```
reboot
```

Step 9 - una volta che il nuovo kernel è avviato, compilare i moduli aggiuntivi del Kernel contenuti in `/usr/src/mpls-linux-1.971/src/` e modificare il file `/etc/modules` :

```
cd /usr/src/mpls-linux-1.971/src/

make

nano /etc/modules
```

Aggiungendo le seguenti righe:

```
mpls
mplsbr
mpls4
mpls_tunnel
xt_mpls
```

Step 10 - creare i necessari collegamenti ai moduli nella cartella `/lib/modules/2.6.32.27-mpls-linux-1.971` :

```
cd /lib/modules/2.6.32.27-mpls-linux-1.971

ln -sf /usr/src/mpls-linux-1.971/src/mpls.ko mpls.ko

ln -sf /usr/src/mpls-linux-1.971/src/mpls4.ko mpls4.ko

ln -sf /usr/src/mpls-linux-1.971/src/mpls_tunnel.ko mpls_tunnel.ko

ln -sf /usr/src/mpls-linux-1.971/src/xt_mpls.ko xt_mpls.ko

ln -sf /usr/src/mpls-linux-1.971/src/mplsbr.ko mplsbr.ko

depmod –a
```

Step 11 - riavviare il sistema e verificare che i moduli siano in esecuzione:

```
reboot

lsmod | grep -i mpls
```

Nel caso si voglia rimuovere il Kernel appena installato è necessario riavviare la macchina con il vecchio Kernel e una volta ottenuti i privilegi di root eseguire il seguente comando:

```
dpkg -P linux-image-2.6.32.27-mpls-linux-1.971
```

Appendice.A.2.

Prima di poter utilizzare *ldp-portable* [59], è necessario che siano caricati tutti i moduli del Kernel compilati [in Appendice.A.1]. Infine, per assicurarsi che tutto il traffico possa essere scambiato, disabilitare IPtables.

Step 1 - copiare le seguenti cartelle in `/usr/src/` :

```
cp mpls_ldp_portable/ /usr/src/mpls_ldp_portable/

cp mpls_ldp_quagga/ /usr/src/mpls_ldp_quagga/
```

Step 2 - creare i link ai file sorgenti di LDP:

```
cd /usr/src/mpls_ldp_quagga/ldpd/

./create-links
```

Step 3 - installare alcune applicazioni necessari per la compilazione:

```
apt-get install build-essential gawk autoconf automake libreadline-dev libpcap-dev libtool dia-libs dia-common texinfo dia
```

Step 4 - configurare le opzioni di compilazione (la cartella contente i sorgenti del Kernel precedentemente installato deve essere rinominata "`/usr/src/linux`" perché tutto funzioni correttamente):

```
cd /usr/src/mpls_ldp_quagga/

autoreconf -i --force

./configure --sysconfdir=/usr/local/quagga --localstatedir=/usr/local/quagga --enable-vtysh --enable-netlink --enable-mpls --enable-ldpd --enable-ospfd --enable-bgpd --enable-rsvpd
```

Step 5 - eseguire l'installazione:

```
make

make install
```

Step 6 - configurare l'utente `quagga` e i relativi permessi:

```
adduser quagga

chown quagga:quagga /usr/local/quagga

cd /usr/local/quagga/

cp zebra.conf.sample zebra.conf

cp ospfd.conf.sample ospfd.conf

cp ldpd.conf.sample ldpd.conf

touch ldpd.conf
```

Step 7 - correggere alcuni bug minori:

1. Digitare `ldd /usr/local/sbin/zebra` e nel caso appaia "`libzebra.so.0 => not found`":

   ```
   cp /usr/local/lib/libzebra.* /lib/
   ```

2. Digitare `ldd /usr/local/sbin/ospfd` e nel caso appaia "`libospf.so.0 => not found`":

   ```
   cp /usr/local/lib/libospf.* /lib/

   chmod -R 777 /usr/local/quagga/
   ```

3. Creare il file di configurazione:

   ```
   nano /usr/local/quagga/ldpd.conf
   ```

   E incollare le seguenti righe:

   ```
   hostname ldpd
   password zebra
   enable password zebra
   !
   interface lo
   !
   line vty
   !
   ```

Step 8 - avviare i demoni `zebra`, `ospfd` e `ldpd`:

```
zebra -d -A 127.0.0.1
ospfd -d -A 127.0.0.1
ldpd -d -A 127.0.0.1
```

Per verificare che i demoni siano attivi:

```
ps -ef
```

Eventualmente, per terminare un demone:

```
kill -9 <PID_NUMBER>
```

Step 9 - collegarsi via Telnet ai demoni:

- zebra: `telnet localhost 2601`
- ospfd: `telnet localhost 2604`
- ldpd: `telnet localhost 2610`

Oltre ai tre demoni appena presentati è possibile implementare anche quello relativo a BGP. Dopo alcuni test preliminari, si è notato però che esso aggiunge una notevole instabilità al sistema, mandando in crash in maniera del tutto non deterministica i demoni `zebra` e `ldpd`. Per avviare il demone `bgpd` occorre eseguire le seguenti istruzioni:

```
cd /usr/local/quagga/
cp bgpd.conf.sample bgpd.conf
bgpd -d -A 127.0.0.1
telnet localhost 2605
```

Appendice.A.3.

Vengono riportati i dettagli implementativi dello scenario [Configurazione di una rete IP/MPLS con route statiche, Capitolo 5.].

Step 1 - configurare l'indirizzo IP per ogni nodo della rete:

HOST A:

```
ifconfig eth1 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255 up
modprobe dummy
ifconfig dummy0 11.0.0.1/32
```

HOST B:

```
ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
modprobe dummy
ifconfig dummy0 11.0.0.2/32
```

LER1:

```
ifconfig if2 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255 up
ifconfig if5 172.16.0.1 netmask 255.255.255.0 broadcast 172.16.0.255 up
```

LSR:

```
ifconfig if5 172.16.0.2 netmask 255.255.255.0 broadcast 172.16.0.255 up
ifconfig if6 172.16.1.1 netmask 255.255.255.0 broadcast 172.16.1.255 up
```

LER2:

```
ifconfig if5 172.16.1.2 netmask 255.255.255.0 broadcast 172.16.1.255 up
ifconfig if2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

Step 2 - popolare le tabelle previste da *mpls-linux*:

- NHLFE: indica come deve essere inoltrato il pacchetto etichettato;
- ILM: mappa ogni pacchetto etichettato entrante ad un'entry dell'NHLFE;
- XC: collega un'entry dell'ILM ad una della NHLFE.

In questa implementazione di MPLS non è presente la FTN.

Un'entry della NHLFE si inserisce con il comando `mpls nhlfe add key 0 instructions push gen <label_out_number> nexthop <interface_out_name> ipv4 <next_hop_address>` . Il corretto inserimento restituirà l'indice dell'entry di rifermento:

```
(returns key <NHLFE_entry_number>)
```

Ciascun prefisso di destinazione viene mappato staticamente all'entry dell'NHLFE tramite il comando `ip route add <destination_prefix> via <next_hop_address> mpls <NHLFE_entry_number>` , mentre LDP esegue questa mappatura automaticamente.

In un LER è sufficiente inserire l'entry della NHLFE e, restituita la chiave, configurare la route statica. Per un LSR la configurazione della route statica non è necessaria in quanto l'inoltro avviene esclusivamente via MPLS. E' necessario, però, configurare anche:

- Il label space di riferimento per ogni interfaccia: `mpls labelspace set dev <interface_in_name> labelspace 0` ;
- L'entry della ILM: `mpls ilm add label gen <label_in_number> labelspace 0` ;
- L'entry della XC, che collega ILM a NHLFE: `mpls xc add ilm_label gen <label_out_number> ilm_labelspace 0 nhlfe_key <NHLFE_entry_number>`, dove `<NHLFE_entry_number>` è restituito dell'inserimento nell'NHLFE, della label `<label_out_number>`.

Di fatto, le operazioni precedenti creano un LSP per il traffico in andata da sorgente a destinazione. Per evitare che il traffico in ritorno sia inoltrato tramite IP, è necessario configurare anche tutte le entry delle precedenti tabelle per il traffico MPLS da destinazione a sorgente.

(Configurazione del traffico da HOST A a HOST B)

HOST A:

```
ip route add 11.0.0.2/32 via 192.168.0.2 src 11.0.0.1
```

LER1:

```
mpls nhlfe add key 0 instructions push gen 1000 nexthop if5 ipv4 172.16.0.2
(returns key 0x2)
ip route add 11.0.0.2/32 via 172.16.0.2 mpls 0x2
```

LSR:

```
mpls labelspace set dev if5 labelspace 0
mpls ilm add label gen 1000 labelspace 0
mpls nhlfe add key 0 instructions push gen 1001 nexthop if6 ipv4 172.16.1.2
(returns key 0x2)
mpls xc add ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x2
```

LER2:

```
mpls labelspace set dev if5 labelspace 0
mpls ilm add label gen 1001 labelspace 0
mpls nhlfe add key 0 instructions nexthop if2 ipv4 192.168.1.1
(returns key 0x2)
mpls xc add ilm_label gen 1001 ilm_labelspace 0 nhlfe_key 0x2
```

(Configurazione del traffico da HOST B a HOST A)

HOST B:

```
ip route add 11.0.0.1/32 via 192.168.1.2 src 11.0.0.2
```

LER2:

```
mpls nhlfe add key 0 instructions push gen 2000 nexthop if5 ipv4 172.16.1.1
(returns key 0x3)
ip route add 11.0.0.1/32 via 172.16.1.1 mpls 0x3
```

LSR:

```
mpls labelspace set dev if6 labelspace 0
mpls ilm add label gen 2000 labelspace 0
mpls nhlfe add key 0 instructions push gen 2001 nexthop if5 ipv4 172.16.0.1
(returns key 0x3)
mpls xc add ilm_label gen 2000 ilm_labelspace 0 nhlfe_key 0x3
```

LER1:

```
mpls labelspace set dev if5 labelspace 0
mpls ilm add label gen 2001 labelspace 0
mpls nhlfe add key 0 instructions nexthop if2 ipv4 192.168.0.1
(returns key 0x3)
mpls xc add ilm_label gen 2001 ilm_labelspace 0 nhlfe_key 0x3
```

Step 3 - verificare che le tabelle siano state popolate correttamente:

LER1:

(Kernel IP routing table): `# route`

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
11.0.0.2        172.16.0.2      255.255.255.255 UGH   0      0        0 if5
172.16.0.0      *               255.255.255.0   U     0      0        0 if5
192.168.0.0     *               255.255.255.0   U     0      0        0 if2
localnet        *               255.0.0.0       U     0      0        0 eth0
```

(NHLFE): `# mpls nhlfe show`

```
NHLFE entry key 0x00000003 mtu 65535 propagate_ttl
        set if2 ipv4 192.168.0.1  (50988 bytes, 607 pkts)
NHLFE entry key 0x00000002 mtu 65535 propagate_ttl
        push gen 1000 set if5 ipv4 172.16.0.2  (55356 bytes, 659 pkts)
```

(ILM): `# mpls ilm show`

```
ILM entry label gen 2001 labelspace 0 proto ipv4
        pop forward key 0x00000003  (53416 bytes, 607 pkts)
```

(XC): `# mpls xc show`

```
XC entry ilm_label gen 2001 ilm_labelspace 0 nhlfe_key 0x00000003
```

LSR:

(Kernel IP routing table): `# route`

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.0.0      *               255.255.255.0   U     0      0        0 if5
172.16.1.0      *               255.255.255.0   U     0      0        0 if6
localnet        *               255.0.0.0       U     0      0        0 eth0
```

(NHLFE): `# mpls nhlfe show`

```
NHLFE entry key 0x00000003 mtu 65535 propagate_ttl
        push gen 2001 set if5 ipv4 172.16.0.1  (53004 bytes, 631 pkts)
NHLFE entry key 0x00000002 mtu 65535 propagate_ttl
        push gen 1001 set if6 ipv4 172.16.1.2  (153972 bytes, 1833 pkts)
```

(ILM): `# mpls ilm show`

```
ILM entry label gen 2000 labelspace 0 proto ipv4
        pop forward key 0x00000003  (55528 bytes, 631 pkts)
ILM entry label gen 1000 labelspace 0 proto ipv4
        pop forward key 0x00000002  (161304 bytes, 1833 pkts)
```

(XC): `# mpls xc show`

```
XC entry ilm_label gen 2000 ilm_labelspace 0 nhlfe_key 0x00000003
XC entry ilm_label gen 1000 ilm_labelspace 0 nhlfe_key 0x00000002
```

LER2:

(Kernel IP routing table): `# route`

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
11.0.0.1        172.16.1.1      255.255.255.255 UGH   0      0        0 if5
192.168.1.0     *               255.255.255.0   U     0      0        0 if2
172.16.1.0      *               255.255.255.0   U     0      0        0 if5
localnet        *               255.0.0.0       U     0      0        0 eth0
```

(NHLFE): `# mpls nhlfe show`

```
NHLFE entry key 0x00000003 mtu 65535 propagate_ttl
        push gen 2000 set if5 ipv4 172.16.1.1  (50988 bytes, 607 pkts)
NHLFE entry key 0x00000002 mtu 65535 propagate_ttl
        set if2 ipv4 192.168.1.1  (55356 bytes, 659 pkts)
```

(ILM): `# mpls ilm show`

```
ILM entry label gen 1001 labelspace 0 proto ipv4
        pop forward key 0x00000002  (57992 bytes, 659 pkts)
```

(XC): `# mpls xc show`

```
XC entry ilm_label gen 1001 ilm_labelspace 0 nhlfe_key 0x00000002
```

HOST A:

(Kernel IP routing table): `# route`

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
11.0.0.2        192.168.0.2     255.255.255.255 UGH   0      0        0 eth1
172.16.2.0      192.168.0.2     255.255.255.0   UG    0      0        0 eth1
192.168.0.0     *               255.255.255.0   U     0      0        0 eth1
localnet        *               255.0.0.0       U     0      0        0 eth0
```

HOST B:

(Kernel IP routing table): `# route`

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
11.0.0.1        192.168.1.2     255.255.255.255 UGH   0      0        0 eth1
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
localnet        *               255.0.0.0       U     0      0        0 eth0
```

Appendice.A.4.

Dettagli implementativi dello scenario [Configurazione di una rete IP con OSPF e BGP, Capitolo 5.].

Step 1 - configurare l'indirizzo IP per ogni nodo della rete:

R1:

```
ifconfig eth1 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255 up
```

R2:

```
ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

AS1BR1:

```
ifconfig if2 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255 up
ifconfig if5 172.16.0.1 netmask 255.255.255.0 broadcast 172.16.0.255 up
ifconfig if3 12.0.0.1 netmask 255.255.255.0 broadcast 12.0.0.255 up
```

AS1IR:

```
ifconfig if5 172.16.0.2 netmask 255.255.255.0 broadcast 172.16.0.255 up
ifconfig if6 172.16.1.1 netmask 255.255.255.0 broadcast 172.16.1.255 up
```

AS1BR2:

```
ifconfig if5 172.16.1.2 netmask 255.255.255.0 broadcast 172.16.1.255 up
ifconfig if2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255 up
ifconfig if3 13.0.0.1 netmask 255.255.255.0 broadcast 13.0.0.255 up
```

AS2BR:

```
ifconfig eth2 12.0.0.2 netmask 255.255.255.0 broadcast 12.0.0.255 up
ifconfig eth3 13.0.0.2 netmask 255.255.255.0 broadcast 13.0.0.255 up
ifconfig eth4 192.123.0.1 netmask 255.255.255.0 broadcast 192.123.0.255 up
```

Step 2 - avviare i demoni `zebra`, `ospfd` e `bgpd`.

```
zebra -d -A localhost
ospfd -d -A localhost
bgpd -d -A localhost
```

Step 3 - configurare le interfacce del demone `zebra` :

Dopo essersi assicurati che il demone sia attivo (`ps -ef`), è necessario collegarsi via Telnet, alla porta 2601: `telnet localhost 2601` . Verrà richiesta la password `zebra` per accedere alla CLI. In questa modalità è possibile eseguire solo comandi diagnostici. Si può accedere alla modalità privilegiata inserendo il comando `enable` e la password di enable, `zebra` . Inutile dire che un router reale necessiterà di password ben più complesse di queste.

Un comando essenziale utilizzabile in modalità privilegiata è `show running-config` indispensabile per esaminare il file di configurazione di `zebra` .

Per modificare la configurazione corrente è necessario entrare in modalità di configurazione globale con il comando `conf t` , accessibile solo dall'utente privilegiato.

Il comando `int <interface_name>` permette di accedere al menu di configurazione dell'interfaccia specificata. E' possibile, ora, configurare il relativo indirizzo IP con `ip address <IP_address>/<subnetmask>` ; `exit` e `end` vengono utilizzati per uscire

rispettivamente dalla modalità di configurazione corrente e da quella di configurazione globale. Salvare le modifiche apportate al file di configurazione con `wr mem` .

R1:

```
telnet localhost 2601

enable

conf t

int eth1

ip address 192.168.0.1/24

exit

wr mem

end
```

R2:

```
telnet localhost 2601

enable

conf t

int eth1

ip address 192.168.1.1/24

exit

wr mem

end
```

AS1BR1:

```
telnet localhost 2601
enable
conf t
int if2
ip address 192.168.0.2/24
exit
int if3
ip address 12.0.0.1/24
exit
int if5
ip address 172.16.0.1/24
exit
wr mem
end
```

AS1IR:

```
telnet localhost 2601
enable
conf t
int if5
ip address 172.16.0.2/24
exit
int if6
ip address 172.16.1.1/24
exit
wr mem
end
```

AS1BR2:

```
telnet localhost 2601
enable
conf t
int if2
ip address 192.168.1.2/24
exit
int if3
ip address 13.0.0.1/24
exit
int if5
ip address 172.16.1.2/24
exit
wr mem
end
```

AS2BR:

```
telnet localhost 2601
enable
conf t
int eth2
ip address 12.0.0.2/24
exit
int eth3
ip address 13.0.0.2/24
exit
int eth4
ip address 192.123.0.1/24
exit
wr mem
end
```

Usciti dalla modalità di configurazione globale, è possibile lanciare i comandi diagnostici `show interface` e `show ip route` per verificare che la configurazione sia corretta.

Step 4 - configurare il demone `ospfd` :

La procedura è del tutto simile alla precedente, fatta eccezione per i comandi necessari ad eseguire gli annunci ai propri vicini. Dopo aver acceduto al demone `telnet localhost 2604` ed essere entrati in modalità di configurazione globale è indispensabile attivare il processo di routing ed accedere al relativo menu di configurazione con `router ospf` . Ogni link su cui si vuole attivare l'annuncio/ricezione di messaggi OSPF deve essere specificato con il comando `network <network>/<subnetmask> area <area_number>` . Nell'esempio proposto si considererà il dominio di routing in questione come un'unica area 0.

R1:

```
telnet localhost 2604

zebra

enable

zebra

conf t

router ospf

network 192.168.0.0/24 area 0

exit

wr mem

end
```

R2:

```
telnet localhost 2604

zebra

enable

zebra

conf t

router ospf

network 192.168.1.0/24 area 0

exit

wr mem

end
```

AS1BR1:

```
telnet localhost 2604

zebra

enable

zebra

conf t

router ospf

network 192.168.0.0/24 area 0

network 172.16.0.0/24 area 0

exit

wr mem

end
```

AS1IR:

```
telnet localhost 2604
zebra
enable
zebra
conf t
router ospf
network 172.16.0.0/24 area 0
network 172.16.1.0/24 area 0
exit
wr mem
end
```

AS1BR2:

```
telnet localhost 2604
zebra
enable
zebra
conf t
router ospf
network 192.168.1.0/24 area 0
network 172.16.1.0/24 area 0
exit
wr mem
end
```

A questo punto è possibile lanciare un ping da qualsiasi nodo a qualsiasi altro. Inoltre, fuori dalla modalità di configurazione, sono disponibili i seguenti comandi diagnostici: per visualizzare lo stato del processo OSPF, `show ip ospf` ; verificare lo stato delle adiacenze coi vicini, `show ip ospf neighbor` ; consultare la routing table OSPF, `show ip ospf route` .

Step 5 - configurare il demone `bgpd`, accedendo via Telnet alla porta 2605:

Entrare in modalità di configurazione BGP col comando `router bgp <my_AS_number>` e definire le reti da annunciare `network <network>/<subnetmask>` . Il prefisso `0.0.0.0/0` rappresenta la route di default. Senza scendere troppo nel dettaglio dei comandi BGP, commentiamo sinteticamente i macroblocchi di cui la configurazione di compone.

AS1BR1:

```
telnet localhost 2605

zebra

enable

zebra

conf t

router bgp 1

network 192.168.0.0/24

network 12.0.0.0/24

network 0.0.0.0/0
```

!si imposta AS2BR come vicino eBGP e si definiscono (saranno specificate in !seguito) le liste, `prefix-list`, degli NLRI accettati in ingresso/uscita per quel !vicino

```
neighbor 12.0.0.2 remote-as 2

neighbor 12.0.0.2 description Router AS2BR

neighbor 12.0.0.2 default-originate

neighbor 12.0.0.2 prefix-list AS2In in

neighbor 12.0.0.2 prefix-list defaultOut out
```

!si imposta AS1BR2 come vicino iBGP, ponendo anche una breve descrizione

```
neighbor 172.16.1.2 remote-as 1

neighbor 172.16.1.2 description Router AS1BR2 (iBGP)
```

!la prima lista, `AS2In` , permette di acquisire l'annuncio relativo alla rete !192.123.0.0/24 mentre la seconda, `defaultOut`, permette di annunciare una !route di default verso AS2BR

```
ip prefix-list AS2In permit 192.123.0.0/24

ip prefix-list defaultOut permit 0.0.0.0/0
```

AS1BR2:

```
telnet localhost 2605

zebra

enable

zebra

conf t

router bgp 1

network 192.168.1.0/24

network 13.0.0.0/24

network 0.0.0.0/0
```

! si imposta AS2BR del tutto analogamente a quanto visto per AS1BR1

```
neighbor 13.0.0.2 remote-as 2

neighbor 13.0.0.2 description Router AS2BR

neighbor 13.0.0.2 default-originate

neighbor 13.0.0.2 prefix-list AS2In in

neighbor 13.0.0.2 prefix-list defaultOut out
```

! si imposta AS1BR1 del tutto analogamente alla configurazione precedente

```
neighbor 172.16.0.1 remote-as 1

neighbor 172.16.0.1 description Router AS1BR1 (iBGP)
```

!anche in questo caso le `prefix-lits` sono le stesse di AS1BR1

```
ip prefix-list AS2In permit 192.123.0.0/24

ip prefix-list defaultOut permit 0.0.0.0/0

!
```

AS2BR:

```
telnet localhost 2605

zebra

enable

zebra

conf t

router bgp 2

network 192.123.0.0/24
```

!imposta AS1R1 come vicino eBGP, senza configurare i valori di LOCAL_PREF e !MED che restano quelli di default, rispettivamente 100 e 0

```
neighbor 12.0.0.1 remote-as 1

neighbor 12.0.0.1 description Router AS1BR1 (primary)

neighbor 12.0.0.1 prefix-list mineOutOnly out

neighbor 12.0.0.1 prefix-list defaultIn in
```

!imposta AS1R2 come vicino eBGP e fa riferimento sia alle liste, `prefix-list`, !degli NLRI accettati in ingresso/uscita sia agli attributi configurati per il !collegamento, tramite `route-map`

```
neighbor 13.0.0.1 remote-as 1

neighbor 13.0.0.1 description Router AS1BR2 (backup)

neighbor 13.0.0.1 prefix-list mineOutOnly out

neighbor 13.0.0.1 route-map metricOut out

neighbor 13.0.0.1 prefix-list defaultIn in

neighbor 13.0.0.1 route-map localPrefIn in
```

!specifica rispettivamente le NLRI permesse in uscita e in entrata

```
ip prefix-list mineOutOnly permit 192.123.0.0/24

ip prefix-list defaultIn permit 0.0.0.0/0
```

!specifica il valore del MED in uscita, 10, valido solo per il prefisso configurato !più in basso, per `myAggregate` , 192.123.0.0/24

```
route-map metricOut permit 10

match ip address myAggregate

set metric 10
```

!specifica il valore di LOCAL_PREF sul link, 90

```
route-map localPrefIn permit 10
set local-preference 90
!
access-list myAggregate permit 192.123.0.0/24
```

Step 6 - verificare le tabelle ottenute dalla precedente configurazione:

Con il comando `show ip ospf route` è possibile visualizzare la routing table OSPF. Essa è suddivisa in tre sezioni: la prima relativa alle reti interne annunciate via OSPF, la seconda contiene le informazioni necessarie per raggiungere i router di bordo, la terza include le rotte esterne relative ai router di bordo conosciuti. Le tre colonne di cui la tabella è composta sono: tipologia di nodo (Network o Router), prefisso di destinazione (nel caso si tratti di un router, router-ID di destinazione), <metrica, area OSPF, next-hop>. Il costo, nel caso delle route esterne, è suddiviso in due componenti: la prima indica la metrica OSPF per raggiungere la rete, la seconda indica la distanza amministrativa, un valore relativo al protocollo di cui si esegue la redistribuzione. Una stessa rotta appresa tramite un altro protocollo di routing avrà distanza amministrativa diversa (e stessa metrica). BGP ha distanza amministrativa di 20.

AS1BR1:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    172.16.1.0/24         [20] area: 0.0.0.0
                           via 172.16.0.2, if5
N    192.168.0.0/24        [10] area: 0.0.0.0
                           directly attached to if2
N    192.168.1.0/24        [30] area: 0.0.0.0
                           via 172.16.0.2, if5
============ OSPF router routing table =============
R    13.0.0.1              [20] area: 0.0.0.0, ASBR
                           via 172.16.0.2, if5
============ OSPF external routing table ===========
N E2 13.0.0.0/24           [20/20] tag: 0
                           via 172.16.0.2, if5
```

Con il comando `show ip bgp` è possibile visualizzare la routing table BGP. Essa è preceduta dalla legenda dei simboli utilizzati ed è suddivisa in sei colonne: la prima indica il prefisso di destinazione, la seconda è il next-hop IP, la terza è relativa al MED, la quarta indica la LOCAL_PREF, la quinta il WEIGHT, la sesta mostra il codice dell'attributo ORIGIN (di fatto deprecato). Il next-hop 0.0.0.0 indica che la rete è direttamente connessa. E' interessante notare che la route di default preferita è, sì direttamente connessa, ma che ne è disponibile anche un'altra via iBGP (il router AS1BR2). L'attributo WEIGHT è stato introdotto da Cisco come valore di riferimento preferenziale per una route, superiore anche alla LOCAL_PREF. Esso è configurato a 32768 per le destinazioni direttamente connesse, a 0 per le altre.

(BGP):

```
bgpd> show ip bgp

BGP table version is 0, local router ID is 12.0.0.1

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

              r RIB-failure, S Stale, R Removed

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop            Metric LocPrf Weight Path

* i0.0.0.0          172.16.1.2               0    100      0 i

*>                  0.0.0.0                  0         32768 i

*> 12.0.0.0/24      0.0.0.0                  0         32768 i

*>i13.0.0.0/24      172.16.1.2               0    100      0 i

*> 192.123.0.0      12.0.0.2                 0             0 2 i

*> 192.168.0.0      0.0.0.0                  0         32768 i

*>i192.168.1.0      172.16.1.2               0    100      0 i

Total number of prefixes 6
```

AS1IR:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    172.16.1.0/24         [10] area: 0.0.0.0
                           directly attached to if6
N    192.168.0.0/24        [20] area: 0.0.0.0
                           via 172.16.0.1, if5
N    192.168.1.0/24        [20] area: 0.0.0.0
                           via 172.16.1.2, if6
============ OSPF router routing table =============
R    12.0.0.1              [10] area: 0.0.0.0, ASBR
                           via 172.16.0.1, if5
R    13.0.0.1              [10] area: 0.0.0.0, ASBR
                           via 172.16.1.2, if6
============ OSPF external routing table ===========
N E2 12.0.0.0/24           [10/20] tag: 0
                           via 172.16.0.1, if5
N E2 13.0.0.0/24           [10/20] tag: 0
                           via 172.16.1.2, if6
```

AS1BR2:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [20] area: 0.0.0.0
                           via 172.16.1.1, if5
N    172.16.1.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    192.168.0.0/24        [30] area: 0.0.0.0
                           via 172.16.1.1, if5
N    192.168.1.0/24        [10] area: 0.0.0.0
                           directly attached to if2
============ OSPF router routing table =============
R    12.0.0.1              [20] area: 0.0.0.0, ASBR
                           via 172.16.1.1, if5
============ OSPF external routing table ===========
N E2 12.0.0.0/24           [20/20] tag: 0
                           via 172.16.1.1, if5
```

Dalla routing table BGP di AS1BR2 è interessante notare che il valore di MED annunciato da AS2BR è stato correttamente acquisito per la destinazione 192.123.0.0/24 Per quest'ultima esistono due percorsi: il primo con next-hop 12.0.0.2, preferito, in quanto presenta il MED di default, il secondo con next-hop 13.0.0.2 e con MED 10, pertanto tenuto in considerazione solo se il primo non risulta più raggiungibile (ad esempio, se il collegamento primario viene interrotto).

(BGP):

```
bgpd> show ip bgp

BGP table version is 0, local router ID is 13.0.0.1

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

              r RIB-failure, S Stale, R Removed

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop            Metric LocPrf Weight Path

* i0.0.0.0          172.16.0.1               0    100      0 i

*>                  0.0.0.0                  0         32768 i

*>i12.0.0.0/24      172.16.0.1               0    100      0 i

*> 13.0.0.0/24      0.0.0.0                  0         32768 i

*>i192.123.0.0      12.0.0.2                 0    100      0 2 i

*                   13.0.0.2                10             0 2 i

*>i192.168.0.0      172.16.0.1               0    100      0 i

*> 192.168.1.0      0.0.0.0                  0         32768 i

Total number of prefixes 6
```

R1:

(OSPF):

```
ospfd> show ip ospf route

============ OSPF network routing table ============

N    172.16.0.0/24         [20] area: 0.0.0.0

                           via 192.168.0.2, eth1

N    172.16.1.0/24         [30] area: 0.0.0.0

                           via 192.168.0.2, eth1

N    192.168.0.0/24        [10] area: 0.0.0.0

                           directly attached to eth1

N    192.168.1.0/24        [40] area: 0.0.0.0

                           via 192.168.0.2, eth1

============ OSPF router routing table =============

R    12.0.0.1              [10] area: 0.0.0.0, ASBR

                           via 192.168.0.2, eth1

R    13.0.0.1              [30] area: 0.0.0.0, ASBR

                           via 192.168.0.2, eth1
```

```
============ OSPF external routing table ===========
N E2 12.0.0.0/24           [10/20] tag: 0
                           via 192.168.0.2, eth1
N E2 13.0.0.0/24           [30/20] tag: 0
                           via 192.168.0.2, eth1
```

R2:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [30] area: 0.0.0.0
                           via 192.168.1.2, eth1
N    172.16.1.0/24         [20] area: 0.0.0.0
                           via 192.168.1.2, eth1
N    192.168.0.0/24        [40] area: 0.0.0.0
                           via 192.168.1.2, eth1
N    192.168.1.0/24        [10] area: 0.0.0.0
                           directly attached to eth1
============ OSPF router routing table =============
R    12.0.0.1              [30] area: 0.0.0.0, ASBR
                           via 192.168.1.2, eth1
R    13.0.0.1              [10] area: 0.0.0.0, ASBR
                           via 192.168.1.2, eth1
============ OSPF external routing table ===========
N E2 12.0.0.0/24           [30/20] tag: 0
                           via 192.168.1.2, eth1
N E2 13.0.0.0/24           [10/20] tag: 0
                           via 192.168.1.2, eth1
```

Nella routing table BGP di AS2BR sono installati solamente due prefissi. La rete direttamente connessa 192.123.0.0/24 e la route di default, disponibile sia attraverso il collegamento primario (12.0.0.0/24) che tramite quello di backup (13.0.0.0/24). Quest'ultimo è quello effettivamente utilizzato, in quanto presenta la LOCAL_PREF di default (100) mentre il link di backup ha solo una LOCAL_PREF di 90.

AS2BR:

(BGP):

```
bgpd> show ip bgp

BGP table version is 0, local router ID is 13.0.0.2

Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,

              r RIB-failure, S Stale, R Removed

Origin codes: i - IGP, e - EGP, ? - incomplete


   Network          Next Hop            Metric LocPrf Weight Path

*  0.0.0.0          13.0.0.1                     90      0 1 i

*>                  12.0.0.1                              0 1 i

*> 192.123.0.0      0.0.0.0                  0         32768 i

Total number of prefixes 2
```

Appendice.A.5.

Sono presentati i dettagli implementativi dello scenario [Configurazione di una rete IP/MPLS con OSPF e LDP, Capitolo 5.].

Step 1 - configurare l'indirizzo IP per ogni nodo della rete. Per gli host è necessario configurare anche il default gateway con il comando `route add default gw <default_GW_IP_address>` :

HOST A:

```
ifconfig eth1 192.168.0.1 netmask 255.255.255.0 broadcast 192.168.0.255 up

route add default gw 192.168.0.2
```

HOST B:

```
ifconfig eth1 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up

route add default gw 192.168.1.2
```

LER1:

```
ifconfig if2 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255 up

ifconfig if5 172.16.0.1 netmask 255.255.255.0 broadcast 172.16.0.255 up
```

LSR:

```
ifconfig if5 172.16.0.2 netmask 255.255.255.0 broadcast 172.16.0.255 up
ifconfig if6 172.16.1.1 netmask 255.255.255.0 broadcast 172.16.1.255 up
```

LER2:

```
ifconfig if5 172.16.1.2 netmask 255.255.255.0 broadcast 172.16.1.255 up
ifconfig if2 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

Step 2 - avviare i demoni `zebra`, `ospfd` e `ldpd`.

```
zebra -d -A localhost
ospfd -d -A localhost
ldpd -d -A localhost
```

Step 3 - configurare le interfacce del demone `zebra` analogamente all'esempio precedente (vedi Appendice.A.4).

Step 4 - configurare le interfacce del demone `ospfd` analogamente all'esempio precedente (vedi Appendice.A.4).

Step 5 - configurare il label space sulle interfacce del demone `zebra` :

LER1:

```
telnet localhost 2601
zebra
enable
zebra
conf t
int if2
mpls labelspace 0
exit
int if5
mpls labelspace 0
exit
wr mem
end
```

LSR:

```
telnet localhost 2601
zebra
enable
zebra
conf t
int if5
mpls labelspace 0
exit
int if6
mpls labelspace 0
exit
wr mem
end
```

LER2:

```
telnet localhost 2601
zebra
enable
zebra
conf t
int if2
mpls labelspace 0
exit
int if5
mpls labelspace 0
exit
wr mem
end
```

Step 6 - connettersi via Telnet al demone `ldpd` , sulla porta 2610:

E' necessario abilitare LDP dalla configurazione globale con il comando `mpls ldp` e successivamente entrare in modalità di configurazione delle singole interfacce interessate, specificando il comando `mpls ip` .

LER1:

```
telnet localhost 2610
zebra
enable
zebra
conf t
mpls ldp
exit
int if5
mpls ip
exit
wr mem
end
```

LSR:

```
telnet localhost 2610
zebra
enable
zebra
conf t
mpls ldp
exit
int if5
mpls ip
exit
int if6
mpls ip
exit
wr mem
end
```

LER2:

```
telnet localhost 2610
zebra
enable
zebra
conf t
mpls ldp
exit
int if5
mpls ip
exit
wr mem
end
```

Step 7 - consultare le informazioni fornite dalla routing table di OSPF e da NHLFE, ILM e XC:

La distribuzione delle etichette fatta con LDP utilizza le tabelle di MPLS in maniera diversa rispetto alla configurazione statica. In questo caso, infatti, eseguito il pop si sbircia (peek) all'indirizzo di destinazione del datagram IP e si esegue l'inoltro, senza bisogno di mantenere traccia del next-hop in NHLFE e XC.

LER1:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    172.16.1.0/24         [20] area: 0.0.0.0
                           via 172.16.0.2, if5
N    192.168.0.0/24        [10] area: 0.0.0.0
                           directly attached to if2
N    192.168.1.0/24        [30] area: 0.0.0.0
                           via 172.16.0.2, if5
```

(NHLFE): `# mpls nhlfe show`

```
NHLFE entry key 0x00000003 mtu 65535 propagate_ttl
        push gen 10007 set if5 ipv4 172.16.0.2  (504 bytes, 6 pkts)
```

(ILM): `# mpls ilm show`

```
ILM entry label gen 10002 labelspace 0 proto ipv4
        pop peek  (528 bytes, 6 pkts)
```

LSR:

(OSPF): `ospfd> show ip ospf route`

```
============ OSPF network routing table ============
N    172.16.0.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    172.16.1.0/24         [10] area: 0.0.0.0
                           directly attached to if6
N    192.168.0.0/24        [20] area: 0.0.0.0
                           via 172.16.0.1, if5
N    192.168.1.0/24        [20] area: 0.0.0.0
                           via 172.16.1.2, if6
```

(NHLFE): `# mpls nhlfe show`

```
NHLFE entry key 0x00000003 mtu 65535 propagate_ttl
        push gen 10002 set if6 ipv4 172.16.1.2  (504 bytes, 6 pkts)
NHLFE entry key 0x00000002 mtu 65535 propagate_ttl
        push gen 10002 set if5 ipv4 172.16.0.1  (504 bytes, 6 pkts)
```

(ILM): `# mpls ilm show`

```
ILM entry label gen 10007 labelspace 0 proto ipv4
        pop forward key 0x00000003  (528 bytes, 6 pkts)
ILM entry label gen 10003 labelspace 0 proto ipv4
        pop forward key 0x00000002  (528 bytes, 6 pkts)
```

(XC): `# mpls xc show`

```
XC entry ilm_label gen 10007 ilm_labelspace 0 nhlfe_key 0x00000003
XC entry ilm_label gen 10003 ilm_labelspace 0 nhlfe_key 0x00000002
```

LER2:

(OSPF):

```
ospfd> show ip ospf route

============ OSPF network routing table ============
N    172.16.0.0/24         [20] area: 0.0.0.0
                           via 172.16.1.1, if5
N    172.16.1.0/24         [10] area: 0.0.0.0
                           directly attached to if5
N    192.168.0.0/24        [30] area: 0.0.0.0
                           via 172.16.1.1, if5
N    192.168.1.0/24        [10] area: 0.0.0.0
                           directly attached to if2
```

(NHLFE):

```
# mpls nhlfe show

NHLFE entry key 0x00000002 mtu 65535 propagate_ttl
        push gen 10003 set if5 ipv4 172.16.1.1  (504 bytes, 6 pkts)
```

(ILM):

```
# mpls ilm show

ILM entry label gen 10002 labelspace 0 proto ipv4
        pop peek  (528 bytes, 6 pkts)
```

HOST A:

(Kernel IP routing table):

```
# route

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     *               255.255.255.0   U     0      0        0 eth1
localnet        *               255.0.0.0       U     0      0        0 eth0
default         192.168.0.2     0.0.0.0         UG    0      0        0 eth1
```

HOST B:

(Kernel IP routing table):

```
# route

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
localnet        *               255.0.0.0       U     0      0        0 eth0
default         192.168.1.2     0.0.0.0         UG    0      0        0 eth1
```

Bibliografia

[1] IETF, RFC 3031
"Multiprotocol Label Switching Architecture"
http://www.ietf.org/rfc/rfc3031.txt

[2] IETF, draft-martini-pwe3-pw-switching-03
"Pseudo Wire Switching"
http://tools.ietf.org/html/draft-martini-pwe3-pw-switching-03

[3] IETF, RFC 4448
"Encapsulation Methods for Transport of Ethernet over MPLS Networks"
http://tools.ietf.org/html/rfc4448

[4] IETF, RFC 4762
"Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling"
http://tools.ietf.org/html/rfc4762

[5] IETF, RFC 2702
"Requirements for Traffic Engineering Over MPLS"
http://www.ietf.org/rfc/rfc2702.txt

[6] Microsoft
"What Is QoS?"
http://technet.microsoft.com/en-us/library/cc757120(WS.10).aspx

[7] IETF, RFC 2684
"Multiprotocol Encapsulation over ATM Adaptation Layer 5"
http://tools.ietf.org/html/rfc2684

[8] IETF, RFC 2427
"Multiprotocol Interconnect over Frame Relay"
http://tools.ietf.org/html/rfc2427

[9] Indiana University
"What is an Internet service provider (ISP)?"
http://kb.iu.edu/data/ahoz.html

[10] Harvard University
"Enhancing Transport Networks with Internet Protocols"
http://www.eecs.harvard.edu/~htk/publication/1998-comm-mag-chapman-kung.pdf

[11] IETF, RFC 3945
"Generalized Multi-Protocol Label Switching (GMPLS) Architecture"
http://www.ietf.org/rfc/rfc3945.txt

[12] IETF, RFC 5087
"Time Division Multiplexing over IP (TDMoIP)"
http://www.ietf.org/rfc/rfc5087.txt

[13] IETF, RFC 3717
"IP over Optical Networks: A Framework"
http://www.rfc-editor.org/rfc/rfc3717.txt

[14] IETF, RFC 5921
"A Framework for MPLS in Transport Networks"
http://tools.ietf.org/html/rfc5921

[15] IETF, draft-ietf-opsawg-oam-overview-03.txt
"An Overview of Operations, Administration, and Maintenance (OAM) Mechanisms"
http://tools.ietf.org/id/draft-ietf-opsawg-oam-overview-03.txt

[16] IETF, RFC 1479
"Inter-Domain Policy Routing Protocol Specification: Version 1"
http://tools.ietf.org/html/rfc1479

[17] IETF, RFC 1136
"Administrative Domains and Routing Domains, a Model for Routing in the Internet"
http://tools.ietf.org/html/rfc1136

[18] MPLS for Linux by jleu
[Online]
http://sourceforge.net/projects/mpls-linux/

[19] Quagga Routing Suite
[Online]
http://www.quagga.net/

[20] Packet Clearing House
"Survey of Characteristics of Internet Carrier Interconnection Agreements"
https://www.pch.net/resources/papers/peering-survey/PCH-Peering-Survey-2011.pdf

[21] IETF, RFC 4116
"IPv4 Multihoming Practices and Limitations"
http://www.ietf.org/rfc/rfc4116.txt

[22] IDC, International Data Corporation
"Tier 1 ISPs: What They Are and Why They Are Important"
http://www.us.ntt.net/downloads/papers/IDC_Tier1_ISPs.pdf

[23] BGP: the Border Gateway Protocol, Advanced Internet Routing Resources
"Global Internet Exchange Points/BGP Peering Points/IXP"
http://www.bgp4.as/internet-exchanges

[24] Cisco Systems
"Internetworking Technology Handbook"
http://docwiki.cisco.com/wiki/Internetworking_Technology_Hand

[25] IETF, RFC4684
"Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)"
http://tools.ietf.org/html/rfc4684

[26] IETF, RFC 2328
"OSPF Version 2"
http://www.ietf.org/rfc/rfc2328.txt

[27] Cisco Systems
"Routing Basics"
http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector

[28] IETF, RFC 1930
"Guidelines for creation, selection, and registration of an Autonomous System (AS)"
http://tools.ietf.org/html/rfc1930

[29] IETF, RFC 4271
"A Border Gateway Protocol 4 (BGP-4)"
http://www.ietf.org/rfc/rfc4271

[30] IETF, RFC 2622
"Routing Policy Specification Language (RPSL)"
http://tools.ietf.org/html/rfc2622

[31] NIL
"AS-path prepending (technical details)"
http://wiki.nil.com/AS-path_prepending_(technical_details)

[32] Cisco Systems
"BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN"
http://www.cisco.com/en/US/docs/ios/12_2t/12_2t11/feature/guide/ft11bmpl.html

[33] Cisco Systems
[Online]
http://www.cisco.com/

[34] De Ghein, L. (2006) MPLS Fundamentals, USA, Cisco Press

[35] IETF, RFC 2685
"Virtual Private Networks Identifier"
http://tools.ietf.org/html/rfc2685

[36] IETF, RFC 2858
"Multiprotocol Extensions for BGP-4"
http://www.ietf.org/rfc/rfc2858.txt

[37] IETF, RFC 5036
"LDP Specification"
http://tools.ietf.org/html//rfc5036

[38] IETF, RFC 3209
"RSVP-TE: Extensions to RSVP for LSP Tunnels"
http://www.ietf.org/rfc/rfc3209.txt

[39] Cisco Systems
"Open System Interconnection Routing Protocol"
http://docwiki.cisco.com/wiki/Open_System_Interconnection_Routing_Protocol#Intermediate_System-to-Intermediate_System

[40] Cisco Systems
"Interior Gateway Routing Protocol"
http://docwiki.cisco.com/wiki/Interior_Gateway_Routing_Protocol

[41] Cisco Systems
"Enhanced Interior Gateway Routing Protocol"
http://docwiki.cisco.com/wiki/Enhanced_Interior_Gateway_Routing_Protocol

[42] Cisco Systems
"Routing Information Protocol"
http://docwiki.cisco.com/wiki/Routing_Information_Protocol

[43] IETF, RFC 4970
"Extensions to OSPF for Advertising Optional Router Capabilities"
http://tools.ietf.org/html//rfc4970

[44] IETF, draft-manayya-constrained-shortest-path-first-01
"Constrained Shortest Path First"
http://tools.ietf.org/html//draft-manayya-constrained-shortest-path-first-01

[45] IETF, RFC 4090
"Fast Reroute Extensions to RSVP-TE for LSP Tunnels"
http://tools.ietf.org/html/rfc4090

[46] Farrel, A. & Bryskin, I. (2006) GMPLS: Architecture and Applications, USA, Morgan Kaufmann

[47] IETF, draft-basak-mpls-oxc-issues-01
"Multi-protocol Lambda Switching: Issues in Combining MPLS Traffic Engineering Control With Optical Cross-connects"
http://tools.ietf.org/html/draft-basak-mpls-oxc-issues-01

[48] CCAMP, Common Control and Measurement Plane
[Online]
http://datatracker.ietf.org/wg/ccamp/charter/

[49] Banerjee, A.; Drake, L.; Lang, L.; Turner, B.; Awduche, D.; Berger, L.; Kompella, K.; Rekhter, Y. (2001) Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques, USA, Communications Magazine, IEEE
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=933450

[50] IETF, RFC 4204
"Link Management Protocol (LMP)"
http://www.ietf.org/rfc/rfc4204.txt

[51] IETF, RFC 4842
"Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) Circuit Emulation over Packet (CEP)"
http://tools.ietf.org/html/rfc4842

[52] IETF, RFC 1661
"The Point-to-Point Protocol (PPP)"
http://www.rfc-editor.org/rfc/rfc1661.txt

[53] IETF, RFC 4349
"High-Level Data Link Control (HDLC) Frames over Layer 2 Tunneling Protocol, Version 3 (L2TPv3)"
http://tools.ietf.org/html/rfc4349

[54] IETF, Pseudowire Emulation Edge to Edge (pwe3)
[Online]
http://datatracker.ietf.org/wg/pwe3/charter/

[55] IETF, RFC 4447
"Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)"
http://tools.ietf.org/html/rfc4447

[56] ITU-T
"Definition of NGN"
http://www.itu.int/en/ITU-T/gsi/ngn/Pages/definition.aspx

[57] IETF, RFC 5586
"MPLS Generic Associated Channel"
http://tools.ietf.org/html/rfc5586

[58] IETF, RFC 5950
"Network Management Framework for MPLS-based Transport Networks"
http://tools.ietf.org/html/rfc5950

[59] Support for MPLS for Linux
[Online]
http://sourceforge.net/projects/mpls-linux/support

[60] GIT Repository for mpls-linux
[Online]
http://repo.or.cz/w/mpls-linux.git

[61] GIT Repository GIT for ldp-portable
[Online]
http://repo.or.cz/w/mpls-ldp-portable.git

[62] Quagga Documentation
[Online]
http://www.quagga.net/docs/docs-info.php

[63] GIT, the fast version control system
[Online]
http://git-scm.com/

[64] GNU
"Grub Legacy"
http://www.gnu.org/software/grub/grub-legacy.en.html

[65] Ubuntu
"Installazione e Gestione del Boot Loader di Linux"
http://wiki.ubuntu-it.org/AmministrazioneSistema/Grub

[66] Waikato Linux Users Group
"Reverse Path Filtering"
http://www.wlug.org.nz/ReversePathFiltering

[67] Linux Poison
"How to enable IP Forwarding"
http://linuxpoison.blogspot.com/2008/01/how-to-enable-ip-forwarding.html

[68] SYBASE
"Installing the Linux dummy-network interface"
http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc30119_1250/html/aseiglnx/CHDDFIEH.htm

[69] GNU zMPLS
[Online]
http://zmpls.sourceforge.net/

[70] GNU Zebra
[Online]
http://www.zebra.org/