TRABAJO ESPECIAL DE GRADO

IMPLEMENTACIÓN EN PLATAFORMA SOFTWARE ROUTER DEL PROTOCOLO ENERGY AWARE ROUTING(EAR)

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Jacob Pardi A.
para optar al título de
Ingeniero Electricista.

Caracas, Mayo de 2011

TRABAJO ESPECIAL DE GRADO

IMPLEMENTACIÓN EN PLATAFORMA SOFTWARE ROUTER DEL PROTOCOLO ENERGY AWARE ROUTING(EAR)

TUTOR ACADÉMICO: Prof. Marco Listanti

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Jacob Pardi A.
para optar al título de
Ingeniero Electricista.

Caracas, Mayo de 2011

DEDICA

Dedicato a Ulises e Vilfani

RINGRAZIAMENTI

Primero que todo quiero agradecer a mis padres, el Sr. Uli y la Sra. Fani, por haber dado todo por mi, haber sido mis maestros y enseñarme que para lograr los objetivos se debe trabajar duro, que hay que afrontar la vida siempre con humor, buena cara y con muchísima energía para poder vivir al máximo todas las experiencias. Sin la ayuda de ellos se que no estaría aquí.

Quisiera agradecer a Valeria, que ha sido de gran apoyo para mi, mostrándose siempre a disposición para cualquier problema y hasta casi un capitulo completo que me hizo.

Vorrei ringraziare al mio correlatore Antonio Cianfrani, una persona sempre disposta ad aiutarmi e un gran compagno nelle squadre di calcetto. Anche alle altre persone del 4to piano Marco, Emanuele, Andrea, Adriano... che mi hanno dato una mano con qualsiasi problema Io abbia avuto.

A mis coinquilinos Francisco, Nohelys y Renee que han sido mi segunda familia aquí, mis segundos hermanos. Nos hemos mantenido siempre unidos y pude contar siempre con ellos para lo que fuese.

A mis hermanos Samuel y Marianna por siempre dejar las cosas para lo ultimo pero al menos hacerlas y hacer mis estadías en Venezuela siempre placenteras. Quisiera agradecer también a las personas que me han ayudado de las otras casas de la comunidad de venezolanos en Roma como “Borgo Velino”, “Cibidale del Friuli”, ”Terni”, “Aquilio Manio” y “Tarquinio Prisco”.

A mis amigos de la universidad, por esos fines de semana de estudio intenso mientras estábamos en clases y las buenas idas a la playa, cine, etc para relajarnos.

Al resto de mi familia por haber estado pendiente de mi y mis amigos del Colegio San Luis que me han acompañado a lo largo de mi carrera.

Jacob Pardi Angarita

IMPLEMENTACIÓN EN PLATAFORMA SOFTWARE ROUTER DEL PROTOCOLO ENERGY AWARE ROUTING(EAR)

Tutor académico en Università La Sapienza: Prof. Marco Listanti. Tesis. Caracas. Universidad Central de Venezuela. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Mención Comunicaciones. Año 2011. Hojas 119.

Palabras claves: Energía en Internet; Protocolo de Enrutamiento-OSPF; Ahorro Energético.

Resumen. Quagga es un software que hace funcionar un computador con sistema operativo GNU como un router comercial, cambiando el código del programa se puede cambiar la forma en que un router decide que camino tomar para llegar a un destino determinado. Este cambio puede ser hecho modificando los costos de utilizar las interfaces que se comunican con otros routers dentro de una red. Cuando forzamos a un router a que tome el camino que nosotros deseamos tenemos la capacidad de dejar ciertos enlaces inutilizados, lo que significa que estas interfaces pueden ser apagadas y como conclusión se puede ahorrar energía. El protocolo en el cual fue desarrollada esta tesis es el RFC2328, llamado OSPF version 2 que se encarga de calcular las rutas hacia todos los routers de un sistema autónomo en Internet y tener un mapa completo de toda la topología de la red en la memoria del router, para luego en caso de trafico tener la capacidad de llevar los paquetes de la forma mas rápida posible a su destinación. El objetivo de la tesis se basa en cómo hacer funcionar el núcleo del cálculo de las rutas, para modificar los costos y poder forzar los caminos deseados y luego de esto crear los comandos en la interfaz del terminal para configurar el router mediante líneas de comando y de esta manera poder habilitar la estrategia de ahorro energético. Luego de las modificaciones necesarias en el código del programa Quagga se pudo implementar la estrategia Energy Aware Routing (EAR), se tiene como consecuencia la inutilización de algunas interfaces de red. Se pudo también notar que no se generaron mensajes de control entre routers que aumente el tráfico en la red y que la cantidad de tiempo para calcular los nuevos caminos dependerá de la cantidad de nodos y el nivel de conectividad entre ellos.

Jacob Pardi Angarita

IMPLEMENTAZIONE SU PIATTAFORMA SOFTWARE ROUTER DEL PROTOCOLLO ENERGY AWARE ROUTING(EAR)

Relatore nella Università La Sapienza: Prof. Marco Listanti. Tesis. Caracas. Universidad Central de Venezuela. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Mención Comunicaciones. Año 2011. Hojas 119.

Parole chiavi: Energia in Internet; Protocolli di Instradamento-OSPF; Risparmio Energetico.

Riepilogo. Quagga è un software che gestisce un computer con sistema operativo GNU, come un router commerciale, la modifica del codice del programma può cambiare il modo in cui un router decide quale strada prendere per raggiungere una determinata destinazione. Questo cambiamento può essere fatto modificando il costi di utilizzo di interfacce che comunicano con altri router all'interno di una rete. Quando abbiamo la forza di un router per prendere il sentiero che vogliamo avere la possibilità di lasciare alcuni link non utilizzati, il che significa che queste interfacce possono essere spenti e in conclusione si può risparmiare energia. Il protocollo che è stato sviluppato in questa tesi è la RFC2328, chiamato OSPF versione 2, che è responsabile per il calcolo dei percorsi per tutti i router in un sistema autonomo in Internet e avere una mappa completa della topologia della rete in memoria router, allora, il traffico dovrebbe essere in grado di trasportare i pacchetti il più velocemente possibile a destinazione. Lo scopo della tesi è basata su come far funzionare il nucleo di calcolo degli itinerari per modificare i costi e per forza i percorsi desiderati e poi ci creerà il comandi nella interfaccia terminale per configurare il router tramite riga di comando e in questo modo è possibile attivare l'energia strategia di risparmio. Dopo i necessari cambiamenti nel codice del programma Quagga potrebbe attuare la strategia energetica Aware Routing (EAR), i risultati nella disattivazione di alcune interfacce di rete. Si è anche notato che vi è generato messaggi di controllo tra i router per aumentare il traffico di rete e la quantità di tempo per calcolare le nuove strade dipenderà dal numero di nodi e la connettività tra di loro.

INDICE

CONSTANCIA DE APROBACIÓN iii

DEDICA iv

RINGRAZIAMENTI v

HOJA DE RESUMEN vii

INDICE ix

INDICE DI FIGURE xiii

INTRODUZIONE 1

CAPITOLO I 3

ENERGIA IN INTERNET 3

1.1 Internet 3

1.1.1 Le caratteristiche principali di Internet 6

1.1.2 Componenti d'Internet 7

1.1.3 Energia consumata 9

1.1.3.1 Principali contribuenti 11

1.2 Risparmio Energetico 15

1.2.1 Strategie per il risparmio energetico 16

1.2.2 Migliorando l'efficienza energetica di un Router 19

CAPITOLO II 24
ROUTING 24
2.1 Il Router 24
2.1.1 La tabella di routing 25
2.1.2 I protocolli di routing 27
2.2 OSPF 28
2.2.1 Il Flooding 29
2.2.2 La Topologia di Rete: il Link State Database 30
2.2.3 Shortest-Path-First tree 33
2.2.4 Suddivisione di un AS in aree 34
2.2.5 Il funzionamento dell'OSPF 35
2.2.6 I pacchetti OSPF 36
2.3 Router software open source basati su PC 39
2.3.1 Panoramica architettonica del PC 40
2.3.2 Quagga 42
2.3.2.1 Architettura Del Sistema 43
CAPITOLO III 46
STRATEGIA DI RISPARMIO ENERGETICO 46
3.1 Teoria dei grafi 46
3.2 Strategia ENERGY AWARE ROUTING (EAR) 48
3.2.1 Il Concetto di Mossa e Le Sue Proprietà 52

3.2.2 La soluzione del problema EAR 57

3.2.2.a L'euristica della Max_Compatibility 58

3.2.2.b Euristica della Min_Used_Links 61

3.3 Modifica dei costi per l'implementazione della strategia EAR 61

3.3.1 Dimostrazione della modifica dei costi per l'implementazione della strategia EAR 63

CAPITOLO IV 68

IMPLEMENTAZIONE IN QUAGGA 68

4.1 Ospf_spf_next() 69

4.2 Strategia EAR mediante la modifica dei costi 78

4.2.1 Funzioni e strutture 78

4.2.2 Implementazione della modifica dei costi 84

4.2.3 Correzione degli errori 87

4.3 Configurazione tramite righe di comando 87

4.3.1 Macro DEFUN 88

4.3.2 Creazione dei comandi 89

4.3.2.1 Abilitazione del modo Energy Saving 89

4.3.2.2 Disabilitazione 93

CAPITOLO V 97

TESTBED 97

5.1 Software di simulazione e acquisizione di dati 97

5.1.1 Quagga 97

5.1.2 LSA Generator 100

5.1.3 Rude&Crude 104

5.1.4 Wireshark 106

5.2 Test-bed per il controllo de la modalità “energy saving” e la valutazione del tempo di commutazione..... 108

5.2.1 Dispositivi utilizzati e caratteristiche..... 108

V.2.2 Eseguimento del test 109

V.2.2.1 Impostazioni iniziali..... 110

5.2.2.2 Descrizione del Test..... 110

5.2.2.1 Valutazione e Risultati 115

CONCLUSIONE..... 117

BIBLIOGRAFIA..... 118

INDICE DI FIGURE

Fig. 1.1 Esempio con diversi reti inter-connnesse 5

Fig. 1.2 Modello d'Internet minimalistico 8

Fig.1.3 Tendenze energetiche dei dispositivi d'Internet 14

Fig.1.4 Energia consumata dalle parti di un router 15

Fig. 2.1 Esempio molto semplice di una tabella d'instradamento 26

Fig. 2.2 Header pacchetti OSPF 37

Fig. 2.3 Pacchetti OSPF 37

Tabella 2.4 LSA Header 38

Tabella 2.5 LSA Type 39

Fig.2.6 Architettura di un PC 41

Fig.2.7 Architettura di Quagga 45

Fig.3.1 Esempio della strategia EAR con 7 router. a)topologia della rete OSPF,b)SPT di R3(importatore),c) SPTdi R1(esportatore) e d)MPT di R3. 52

Fig. 4.1 Schema di funzionamento di Dijkstra in ospfd 70

Fig. 4.2 Schema di funzionamento del calcolo del costo totale del nodo modificato 85

Fig. 5.1 Esempio di file di configurazione di ospfd 99

Fig.5.2 Schema di funzionamento di LSA Generator.....100

Fig.5.3 Esempio di un Router LSA.....101

Fig.5.4 Esempio di Network LSA.....102

Fig.5.5 Esempio di file di configurazione di RUDE.....106

Fig. 5.6 Configurazione fisica dei PC.....109

Fig. 5.7 Schema delle fasi per l'eseguimento del test.....111

Fig.5.8. Per misurare il tempo di commutazione, il DUT è collegato a PC1, PC2 e PC3. PC1 con generatore di traffico RUDE, invia i pacchetti di dati. PC2 e PC3 emulano una topologia di rete. 112

Fig. 5.9 Attivazione della modalità energy saving nella shell di LINUX 114

Fig.5.10 Andamento del tempo di switching rispetto a R,N e M. 116

INTRODUZIONE

La ricerca nel campo delle reti “green” sta avendo sempre maggiore interesse, in particolare, guidata da fini di risparmio energetico. Internet a livello mondiale e i suoi migliaia di apparecchi consumano un'enorme quantità d’energia e hanno un impatto sul riscaldamento globale. Il risparmio energetico in Internet è diventato importante in quanto le diverse applicazioni d’Internet sono in aumento esponenziale, pertanto la potenza consumata da questa struttura. In questo lavoro è stato proposto un protocollo di routing centralizzato per il risparmio energetico nelle reti core cablate chiamato Energy Aware Routing (EAR). EAR è stato pensato appoggiandosi sul protocollo OSPF e si basa sui concetti di “esportazione” e “importazione” di Shortest Path Tree (SPT), il meccanismo consiste nel condividere i SPTs tra due router. La strategia è in grado di controllare l'insieme di collegamenti da mettere in “sleep mode” attraverso il concetto di “mossa”. Questo approccio dà al gestore di rete la possibilità di controllare le prestazioni della rete e consente una degradazione contenuta della QoS durante le ore di basso traffico.

Il lavoro sviluppato in questa tesi tratta sull’implementazione di EAR in un “software router”, cioè, un PC dotato di un programma capace d’emulare un protocollo d’instradamento e far comportare un computer personale come se fosse un router commerciale. Quagga è stato il pacchetto software router utilizzato, il quale ha subìto alcune modifiche, per poter utilizzare il protocollo OSPF in modo normale, o attivare EAR tramite righe di comando, indicando l’esportatore nella shell del PC scelto come importatore.

Nel capitolo I viene offerta una breve panoramica d’Internet, le sue strutture più importanti e come questa importantissima struttura di comunicazione può essere modificata per dispendere meno energia, un problema costante oggi giorno.

Nel capitolo II si descrive il protocollo di routing OSPF utilizzato per instradare i pacchetti su Internet e il pacchetto di software routing chiamato Quagga utilizzato per testare la strategia di risparmio energetico proposta.

Nel capitolo III viene aproffondita la strategia EAR (energy aware routing), una modalità sviluppata dal gruppo di reti della Università La Sapienza nel marchio della ricerca, con grandi potenzialità di risparmio energetico. Inoltre, si presenta la proposta utilizzata per modificare il codice di Quagga.

Nel capitolo IV vengono spiegate passo a passo le modifiche che sono state fatte al demone ospfd di Quagga, per implementare la strategia EAR.

Nel capitolo V si descrive come furono realizzate le prove e i risultati ottenuti.

CAPITOLO I

ENERGIA IN INTERNET

1.1 Internet

L'idea di Internet nasce all'inizio degli anni '60. L'Advanced Research Project Agency (un'agenzia americana) vuole realizzare una rete di comunicazione tra i computer militari (ARPANET) che sia flessibile e robusta, in grado di continuare a funzionare anche in caso di spiacevoli incidenti come una guerra nucleare o altro.

Si parte nel 1969 collegando quattro computer. Nel 1977 i computer collegati sono 111 e solo dopo 7 anni si supereranno i 1000, poi la crescita diventerà inarrestabile.

L'evoluzione da rete militare a "Internet per tutti" ha richiesto tempo. Negli anni '80 la National Science Foundation diede il via a NSFNET con le stesse modalità di funzionamento di ARPANET, ma basandola su linee più veloci e con una struttura in rapida evoluzione.

Nel 1990 ARPANET ha cessato di esistere, soppiantata completamente da NSFNET. La cosa più notevole è che quasi nessuno se n'è accorto. Grazie a questa tecnologia di rete, inizialmente è stato possibile collegare i computer di una singola azienda tra di loro attraverso una rete interna che permetteva a ogni singola macchina di comunicare con le altre attraverso delle regole di comunicazione, chiamate "protocolli", e dei software che permettevano di gestire questi collegamenti.

Protocolli e software erano prodotti ad hoc per il funzionamento di queste reti, da qui il termine di "proprietari". Lo sviluppo di Internet come rete mondiale

porta a superare queste individualità specifiche di ogni azienda, mettendo a disposizione un metodo di comunicazione comune che permette a delle aziende con sistemi, hardware e software anche diversi di colloquiare tra di loro attraverso l'adozione di regole comuni di comunicazione, cioè di protocolli condivisi.

Da qui la definizione di Internet come la "Rete delle reti", poiché è diventata un sistema di trasmissioni mondiale, basato sull'utilizzo di standard di comunicazione universale. Le reti locali sono collegate a reti regionali e a loro volta connesse a dorsali (dette "backbone") ad alta velocità, che garantiscono la comunicazione a livello mondiale. Queste reti globali prendono il nome di "Autostrade dell'informazione".

Cos'è Internet?

Abbiamo detto che potremmo definire Internet come "La Rete delle reti". Una rete di calcolatori è costituita da un insieme di computer che rappresentano i "nodi" della rete stessa, dalle connessioni tra tali nodi (che possono essere costituita da cavi, collegamenti satellitari, o altro) e dal software che permette di gestire la comunicazione. Applicando questa struttura su scala mondiale abbiamo Internet. I protocolli d'inter-networking (cioè i protocolli che costituiscono l'infrastruttura di Internet) non si occupano delle singole reti, ma delle comunicazioni tra le reti. Come mostrato in figura 1.1, ogni rete è dotata di un dispositivo detto router che le connette con l'esterno. La comunicazione tra un nodo A e un nodo B (appartenenti a due sottoreti diverse) avviene in tre passi:

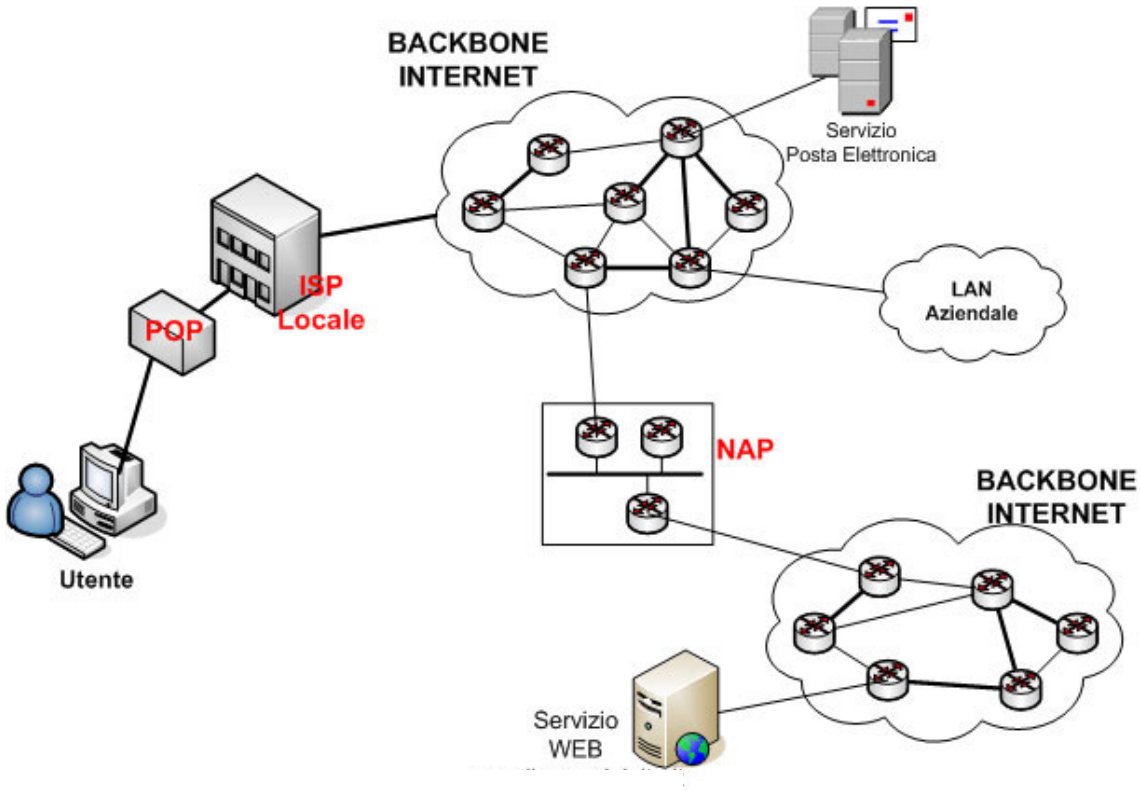


Fig. 1.1 Esempio con diversi reti inter-connnesse

• comunicazione tra il nodo sorgente e il router della sotto-rete cui esso stesso appartiene;

• comunicazione tra il router suddetto e il router della sotto-rete cui appartiene il nodo di destinazione, (eventualmente mediata da altri router);

• comunicazione tra il router e il nodo di destinazione.

I protocolli d'inter-networking si occupano principalmente del passo due, cioè della comunicazione tra i router. Il protocollo TCP/IP si occupa dell'indirizzamento dei nodi (router) e dell'instradamento dei messaggi.

Internet oltre che una rete di calcolatori è anche molto di più, grazie alle applicazioni che sfruttano la rete Internet come infrastruttura di comunicazione

sottostante: il Web (WWW), la posta elettronica (SMTP), i Newsgroup, le Chat e tutti i sistemi di scambio d'informazioni e di fornitura di servizi (dall'e-commerce all'e-banking).

1.1.1 Le caratteristiche principali di Internet

Prima di analizzare il funzionamento di questa rete occorre prima dire due parole su quelle che sono le sue caratteristiche distintive, così da evidenziare attraverso di esse il meccanismo di funzionamento della Rete stessa.

Una delle principali caratteristiche della rete è quella di essere basata su tecnologie tra di loro eterogenee (dalle infrastrutture di comunicazione - che possono essere linee telefoniche, fibre ottiche, collegamenti satellitari, ecc. - all'hardware e software dei calcolatori che costituiscono le singole sotto-reti), le quali però riescono a interagire tra di loro, attraverso delle regole che sono state concordate da tutti gli utilizzatori, che sono sintetizzati nel protocollo di comunicazione che prende il nome di protocollo TCP/IP. TCP/IP è in realtà un insieme di due protocolli distinti: IP, che definisce le regole per l'instradamento dei messaggi, e TCP, che definisce le regole per la comunicazione tra due singoli nodi.

Il protocollo IP funziona attraverso un meccanismo che viene detto di "commutazione di pacchetto", caratterizzato dalle seguenti proprietà:

• Ogni computer della rete è identificato univocamente da un numero, che viene detto "indirizzo IP", in genere il suo formato è il seguente: 114.132.12.47

• Non esiste un controllo centralizzato internet

• Ogni messaggio in partenza, non importa la sua lunghezza, viene scomposto in una serie di "pacchetti" a lunghezza fissa, che contengono le informazioni di chi li ha spediti, di chi li deve ricevere, la parte dati ed eventuali

informazioni per ricostruire il messaggio una volta che tutti i pacchetti sono arrivati a destinazione

• Il percorso che i vari pacchetti percorrono sulla rete (quali connessioni e quali nodi attraversano) non viene stabilito a priori, ma definito durante il viaggio. Di conseguenza dal punto di partenza a quello di arrivo i pacchetti possono passare anche attraverso vie diverse, purché alla fine arrivino tutti a destinazione.

Il protocollo TCP si occupa di stabilire le singole connessioni tra due nodi, secondo una modalità detta “three-way handshake” (“stretta di mano a tre vie”):

• il nodo N1 invia una richiesta di connessione al nodo N2;

• se il nodo N2 accetta la chiamata, invia un segnale di accordo e la connessione è stabilita;

• il nodo N1 può così inviare un flusso di dati (“byte stream”) sul canale appena istituito.

1.1.2 Componenti d’Internet

Internet è suddiviso in parti come mostrato in figura 1.2.

La figura 1.2 è una rappresentazione minimalista della configurazione di rete d’Internet. I principali componenti della rete sono l'accesso, la rete metro e la core network, più data center e reti di distribuzione di contenuti (ad esempio, per IPTV). Questo modello è una rappresentazione d’Internet, e, come tale, non comprende gran parte del dettaglio della vera struttura e della topologia d’Internet. Il modello tiene conto del conteggio di salti, tipico dei pacchetti che attraversano Internet.

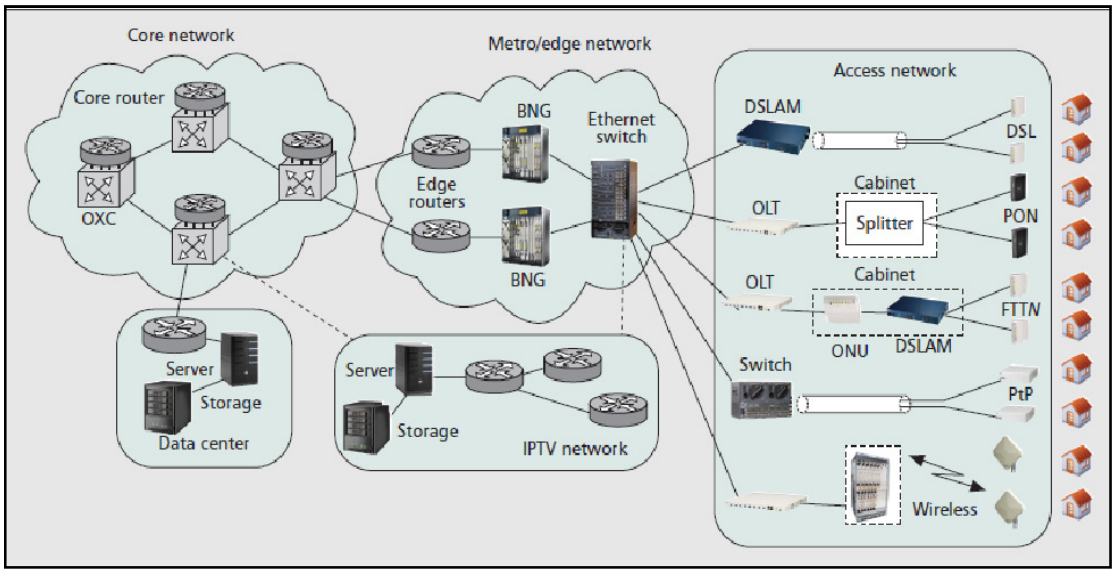


Fig. 1.2 Modello d'Internet minimalistico

La rete di accesso collega case e imprese alle sue centrali locali. Vi è una serie di tecnologie oggi in uso e in fase di sviluppo. Digital Subscriber Loop (DSL) utilizza i doppini installati originariamente per fornire servizi di telefonia fissa. Il servizio di telefono fissa, che utilizza la larghezza di banda sotto 3,4 kHz, non viene modificato, e la larghezza di banda di frequenza maggiore viene utilizzata per servizi a banda larga. Fibre che arrivano fino alla sede del cliente (FTTP) di solito utilizzano una rete ottica passiva condivisa (PON) o una connessione punto-punto (PtP) Ethernet. In una PON, una singola fibra dal nodo di rete alimenta uno o più gruppi di clienti attraverso uno splitter passivo. Un terminale ottico (OLT) si trova presso la centrale locale, e serve un numero di modem di accesso o unità di rete ottiche (ONU) collocate in ognuna delle sedi dei clienti. Le ONUs si comunicano con il OLT con multiplazione in tempo. In una rete di accesso PtP, ogni ONU è collegato direttamente alla centrale locale con una fibra dedicata allo scambio.

Nelle zone in cui i doppini telefonici sono in buone condizioni, può essere utilizzata la tecnologia fibra-to-thenode (FTTN). Questa tecnologia utilizza una fibra dedicata, dalla centrale locale a un multiplexer d'accesso DSL (DSLAM), situato in

una cabina esterna vicino a un gruppo di clienti. Una tecnologia ad alta velocità che utilizza il doppino, come DSL, è usata dalla cabina alla sede del cliente. Nelle zone in cui i doppini e la fibra non sono disponibili o fattibili, la tecnologia senza fili sarà in grado di fornire accesso a Internet. Tecnologie per quest'uso includono WiMAX, High Speed Packet Access (HSPA) e Universal Mobile Telecommunication System (UMTS). Per l'accesso wireless un modem wireless, che si trova nel terminale del cliente, si comunica con una stazione base wireless locale, che a sua volta, è collegato alla centrale locale.

Le centrali locali in una città sono collegati tra di loro e ad altre città con la rete metro/edge. Questa rete fornisce anche i punti di connessione per i fornitori di servizi Internet (ISP). La rete metro/edge funge da interfaccia tra la rete d'accesso e la core. La rete metro/edge include switches edge Ethernet, gateway di rete a banda larga (BNG), e router di bordo. Gli switch Edge Ethernet concentrano il traffico proveniente da un gran numero di nodi d'accesso uplink a due o più router BNG. Lo switch edge si collega a due o più router BNG per fornire ridondanza. I router BNG eseguono un controllo delle velocità d'accesso, autenticazione, e servizi di sicurezza, e si connettono a più router di bordo per aumentare l'affidabilità.

I router di bordo si connettono alla core network. La core network comprende un piccolo numero di router di grandi dimensioni in grandi centri abitati. Questi router del nucleo eseguono tutto l'instradamento necessario e servono anche come gateway d'ingresso agli altri router del nucleo. I router del nucleo (core network) di ogni rete hanno spesso alta connettività, ma hanno soltanto alcuni collegamenti a delle reti di altri operatori. Collegamenti (WDM) in fibra ottica interconnettono questi router con le reti di altri operatori.

1.1.3 Energia consumata

Un metodo ampiamente accettato per la modellazione del consumo energetico delle infrastrutture di Internet e la relativa infrastruttura delle tecnologie dell'informazione e delle comunicazioni (TIC), è basato sull'inventario delle attrezzature e / o dati di vendita. Utilizzando i dati storici di vendita di apparecchiature di telecomunicazione, può essere stimata un'ampia panoramica della quantità di apparecchiature nella rete. Insieme con le informazioni sul consumo energetico di queste apparecchiature, quest'approccio può fornire una buona stima. Tuttavia, non espone l'interazione tra crescita della domanda e il consumo d'energia conseguenti. Questo è importante per stimare quanto le future tendenze di crescita potrebbero far cambiare i modelli di consumo di potenza mentre nuovi servizi basati su Internet sono sviluppati.

Un approccio complementare utilizza un modello basato sui principi di disegno di reti di telecomunicazioni. Per una gamma di velocità d'accesso, il consumo energetico di ogni parte della rete è calcolato con un disegno in carta della rete combinato con l'informazione sul consumo energetico delle apparecchiature fornite dal fabbricante per una gamma di tipi di apparecchiature tipiche. Quest'approccio consente un modello complessivo di consumo di energia di rete e fornisce una piattaforma per la previsione della crescita del consumo mentre aumenta il numero d'utenti e le velocità d'accesso.

Per stimare il consumo energetico delle infrastrutture di Internet, è selezionato un bit rate di accesso (in bit al secondo). Sapendo questa velocità d'accesso e la tecnologia utilizzata (DSL asincrona [ADSL], PON, wireless, ecc), e le regole di progettazione di reti, si può calcolare la capacità necessaria che deve essere in grado di gestire le apparecchiature di telecomunicazione in quanto riguarda all'accesso, la metropolitana, e le reti di core.

1.1.3.1 Principali contribuenti

Apparecchiature di Rete

Le apparecchiature fisiche di rete sono il maggiore contributo al consumo energetico delle infrastrutture di Internet. Ciò comprende le attrezzature della rete d'accesso, metro, e core.

Tre tecnologie dominano nella rete d'accesso. Queste sono la fibra ottica (per i PON e P2P), doppini (per ADSL, VDSL, e le fibre coassiali ibride [HFC] / modem via cavo) e wireless. L'aggregazione del traffico, attraverso il multiplexing statistico, da parte degli utenti finali è una funzione importante della rete d'accesso.

La rete metro comprende una porta verso le reti core e metro. Il traffico locale richiede di routing per le zone centrali della città e periferia. Il resto viene introdotto nella core network.

La rete principale (core network) comprende router di core e un sistema di comunicazione intercity/internazionale che trasporta il traffico Internet tra i router.

Molti servizi Internet offerti agli utenti d'accesso richiedono lo scambio d'informazioni tra gli utenti finali e i punti di connessione dei fornitori di servizi a Internet (spesso chiamato un punto di presenza o POP). Il trasporto di questi dati è "backhaul" (carico di ritorno) e utilizza principalmente trasporto wireless o Ethernet.

Apparecchiature di rete devono essere alimentate e raffreddate. Ciò comprende la fornitura di corrente continua per gli scaffali che ospitano le apparecchiature e la fornitura di energia senza interruzioni (UPS) che assicuri la continuità d'alimentazione al dispositivo di rete.

Pianificazione della Capacita

Proprietari delle reti di telecomunicazione devono poter consentire picchi di traffico, crescita futura, e la protezione / ripristino dei servizi. Questo richiede un sovradimensionamento della rete che, a sua volta, aumenta il consumo d'energia della rete. Sezioni delle reti metro / edge e core possono essere il 100 per cento o più sovradimensionate secondo le politiche di progettazione della rete dei proprietari.

Servizi /Cloud Strutture

Una quantità significativa di traffico Internet nasce da una vasta gamma di servizi web-based e risorse disponibili agli utenti finali tramite Internet. Esempi di questi includono servizi cloud, trasmissione di contenuti e memoria come servizio. I centri dati che forniscono questi servizi richiedono notevoli quantità di attrezzature e di potenza per funzionare. Ad esempio, servizi di contenuti richiedono server che memorizzano i dati / contenuti e regolare l'accesso ad essi. Altri servizi richiedono server di hosting, l'elaborazione e la ricerca di dati. Le macchine che forniscono questi servizi sono di solito collegate tramite una rete LAN o WAN. Queste reti inoltre consumano energia.

Demografia

Internet è una rete intra e internazionale. La distanza fisica tra i centri di popolazione ha un impatto diretto sul consumo di energia della rete. Un altro fattore demografico importante è la densità di terminali nei centri abitati. Terminali ampiamente diffusi richiedono maggiore potenza per la connessione alla centrale locale.

Scenari dei servizi

Il consumo è fortemente influenzato dal tipo di servizio fornito. La rete renderà disponibili i seguenti tipi di servizi:

Servizi condivisi: in tempo quasi - reale o non-reale, come la posta elettronica, navigazione web, o scaricare video e audio, per i quali brevi ritardi sono accettabili. Questi servizi possono essere in eccesso di richieste, molti utenti possono condividere la banda fornita senza notare alcun degrado della velocità.

Servizi dedicati che richiedono Quality of Service (QoS): Questi includono servizi come la telefonia (voice over IP, VoIP), Internet TV, conferenze e aule virtuali. Questi servizi non possono essere richiesti in eccesso. Capacità dedicata per ogni servizio deve essere fornita attraverso le reti di accesso e backhaul al server che fornisce il contenuto o servizio.

Servizi in tempo reale consegnati a più utenti tramite multicast. Tali servizi possono includere video broadcast, near-video-on-demand e radio Internet. Una copia di ogni servizio richiesto viene trasmessa a uno switch vicino al cliente richiedente (i) e replicata a tutti i clienti collegati a questo switch che fanno la richiesta.

Tendenze Energetiche

Considerando velocità d'accesso relativamente basse, il consumo d'energia di Internet è dominato dai dispositivi di accesso (cioè, le attrezzature utilizzate per collegare la casa alla centrale locale), in particolare, l'ONU trovata a casa. Mentre siano in aumento le velocità d'accesso, la core network aumenterà il suo consumo d'energia e infine supererà il consumo energetico della rete d'accesso. Tenendo presente che il consumo di una casa o un gateway ONU è indipendente dalla sua velocità d'accesso, mentre che aumenti la velocità d'accesso, il volume del traffico crescerà nella core network. Questo, a sua volta, richiede un aumento significativo

nella quantità di dispositivi d'instradamento, perciò un maggiore consumo d'energia tale che i router di core domineranno il dispendio energetico a velocità d'accesso elevate. Senza miglioramenti tecnologici in corso, il consumo energetico delle infrastrutture di Internet crescerà in modo esponenziale verso livelli insostenibili a causa delle esigenze sui router di core. In realtà, ci saranno miglioramenti dell'efficienza energetica durante il tempo necessario per le reti a evolversi a velocità di accesso più alto.

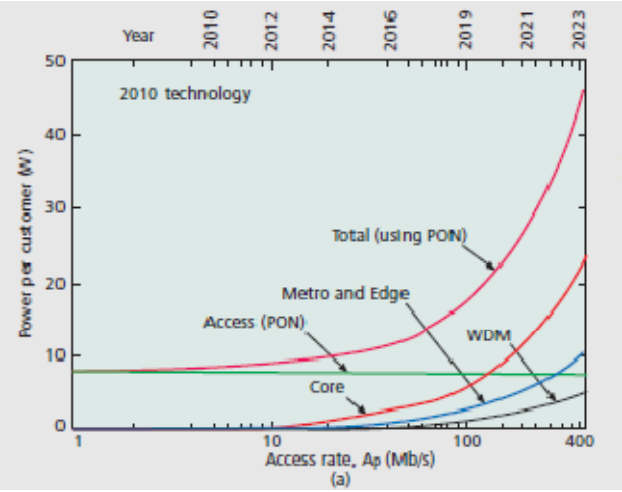


Fig.1.3 Tendenze energetiche dei dispositivi d'Internet

La rete metro/edge, così come i sistemi di comunicazioni ottiche che collegano tra i dispositivi della core network, non dominano il consumo energetico. L'attrezzatura metro/edge non deve trattare con il volume di traffico che si verifica nel nucleo. Il sistema di comunicazione ottica WDM che collegano i router lavorano relativamente ad alta efficienza energetica, in quanto possono trasportare notevole capacità a bassa potenza. Poiché i router di core domineranno il consumo di energia a velocità di trasmissione elevata, rivolgiamo ora la nostra attenzione a questi.

Il consumo di energia relativa dei sottosistemi all'interno di un core router è mostrato in fig. 1.4. Un router di core a pieno carico durante l'elaborazione di

pacchetti IP. Il motore d'inoltro, alimentazione e raffreddamento all'interno del router contribuiscono per circa il 65 per cento per il consumo di energia totale.

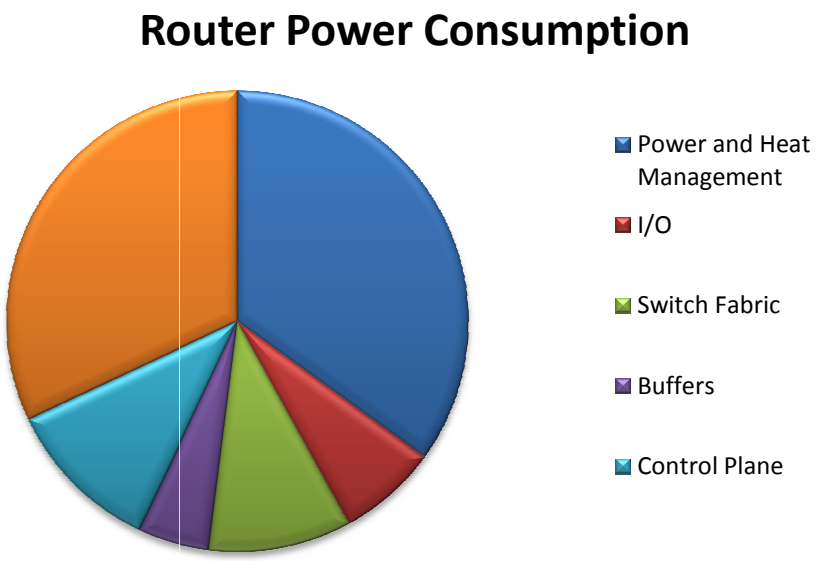


Fig.1.4 Energia consumata dalle parti di un router

1.2 Risparmio Energetico

Internet è diventato una parte integrale delle economie di tutti i paesi sviluppati e in via di sviluppo. Il ciclo virtuale di miglioramenti nel settore delle telecomunicazioni a sostegno della crescita economica, che a sua volta, sostiene la crescita nel settore delle infrastrutture delle telecomunicazioni, ha servito molte nazioni. Tuttavia, questo ciclo non può continuare senza fine, perché tutte le reti di telecomunicazioni necessitano di risorse per funzionare, in particolare di energia elettrica. Più grande diventa la rete (in termini di capacità e dimensioni fisiche), più energia elettrica viene consumata. Oggi il settore dell'informazione e delle telecomunicazioni è responsabile di circa il 5 per cento del consumo totale di energia elettrica nelle economie dei paesi sviluppati. L'infrastruttura di Internet consuma circa 1 per cento del consumo totale di energia elettrica di una nazione sviluppata. Questa

percentuale crescerà con l'aumento della velocità delle reti d'accesso a banda larga nazionali nei prossimi anni.

Il tasso di crescita di Internet, sia in termini d'aumento d'utilizzo e di capacità, significa che effettivamente ridurre il suo consumo energetico totale non è un obiettivo realistico. La rete sta crescendo troppo in fretta. Un obiettivo più pratico è quello di migliorare la "efficienza energetica " di Internet. Con l'efficienza energetica s'intende la quantità di dati che possono essere trasportati da un capo all'altro per bit quantico di energia consumata dalla rete. Questa misura d'efficienza energetica è semplicemente il reciproco dell'energia per bit di dati trasportati e / o processati.

Si noti che anche se abbiamo identificato quelle parti di Internet che dominano il suo consumo di energia (cioè, watt o watt / utente), discutiamo i metodi per migliorare l'efficienza energetica (cioè, la riduzione Joule per bit). La relazione tra queste due grandezze è che il consumo di potenza (watt) è uguale a efficienza energetica (Joule per bit) moltiplicato per il volume di traffico (bit per secondo). Adottiamo questo approccio, perché Internet è una struttura complessa d'ingegneria, e qualsiasi tentativo di migliorare l'efficienza energetica globale è meglio focalizzarla su quelle parti che consumano più energia. Perciò, un passo fondamentale in questo processo è individuare quelle parti.

1.2.1 Strategie per il risparmio energetico

Da fig. 1.3 è chiaro che le due principali aree che richiedono attenzione nel contesto del consumo totale di energia sono le reti di accesso (in particolare le attrezzature terminali a casa) e i router della core network. La sfida delle apparecchiature terminali a casa è stato affrontato dall'Unione europea (UE), che ha pubblicato gli orientamenti di consumo energetico per questi apparecchi. Questo codice di condotta volontaria è progettata per migliorare l'efficienza energetica di

tutte le apparecchiature domestiche a banda larga vendute all'interno dell'UE. Questo fa parte delle strategie sviluppate nel codice di condotta dell'Unione europea.

Tre strategie efficaci per migliorare l'efficienza energetica dei dispositivi sono:

Richiedere ai dispositivi di ridurre il suo consumo di energia quando non siano in uso. Questo stato di basso consumo è spesso definito come uno stato di "sleep" o "idle" e può essere implementato spegnendo quelle parti del dispositivo che non sono necessarie quando l'apparecchio non sta trasmettendo. L'intero dispositivo non può essere disattivato perché si perde il contatto con Internet. Una piccola quantità d'energia, deve essere usata per garantire che Internet sia consapevole della disponibilità del dispositivo e dunque in grado di risvegliarlo quando sia necessario. Poiché l'elettronica moderna è in grado di operare a velocità molto elevate, anche stati di “sleep” molto brevi (molto meno di 1 s) possono essere molto efficaci nel ridurre il consumo di energia.

Ridurre la velocità del processore di un dispositivo quando il suo carico di lavoro è basso. Molti dispositivi sono in grado di operare su una gamma di bit rate. Circuiti elettronici consumano meno energia quando si opera a una velocità inferiore. Così, quando il carico di traffico su un dispositivo è basso, il consumo di energia può essere ridotto abbassando la velocità con cui funziona.

Questi approcci vengono applicati per rendere il protocollo Ethernet più efficiente e possono essere applicati in molte parti di Internet.

Migliorare l'efficienza energetica dei core router. Ciò renderà necessario migliorare la tecnologia di elaborazione del segnale all'interno del router o cambiare la funzione del router. Inoltre, le strategie per il dimensionamento della core network per migliorare l'efficienza energetica diventeranno sempre più importanti nel futuro

quando la core network inizi a dominare il consumo energetico. Tali strategie sono state oggetto di ricerca significativa negli ultimi anni.

Implementare la tecnologia di rete di accesso più efficiente disponibile. Il predominio delle apparecchiature di accesso di rete di oggi è un chiaro orientamento per migliorare l'efficienza energetica di Internet.

Una proposta è stata quella di sostituire i circuiti elettronici all'interno di un router con circuiti fotonici. Quest'approccio è motivato dalla previsione che le tecnologie di commutazione fotoniche possono operare a velocità molto superiore a quelle dell'elettronica. Le tendenze attuali indicano che la velocità di elaborazione massima raggiungibile da dispositivi elettronici nel corso dei prossimi anni sarà di circa 100 Gb / s, mentre quelli fotonici promette raggiungere velocità di oltre 10 Tb / s. E 'stato proposto da alcuni ricercatori che molti router elettronici potrebbero essere sostituiti con una minore quantità di router fotonici, riducendo così il consumo. Purtroppo, le tendenze di consumo d'energia fino ad oggi per le tecnologie fotoniche non supportano questo scenario. Oggi Complementary Metal Oxide Semiconductor (CMOS) è di circa cinque ordini di grandezza meno dispendioso in termini energetici rispetto alle tecnologie fotoniche. Inoltre, mentre CMOS ha mostrato una tendenza di continua diminuzione di consumo di energia, le tecnologie fotoniche stanno dimostrando poco miglioramento. L'effetto netto di questo è che, ogni volta che è necessaria l'elaborazione del segnale ad alta intensità, l'elettronica è la tecnologia più efficiente disponibile.

Un'altra opzione è quella di ristrutturare le reti per ridurre il traffico processato nei router IP. In quest'approccio la rete potrebbe essere riprogettata in modo che una gran parte del traffico Internet bypassa i router della rete centrale. Il traffico Internet attraversa tra la sorgente e la destinazione, in media tra circa 14 router. Questi router non sono direttamente collegati. Essi si comunicano attraverso

sistemi di comunicazione ottica che utilizzano il protocollo SDH / SONET. Le connessioni fisiche sono basate su WDM in cui molti canali ottici indipendenti si propagano attraverso le fibre dispiegate in tutto il mondo. I router IP possono essere considerati come il livello più alto di una pila multistrato di attrezzature. Pertanto utilizzando SDH / SONET e WDM per bypassare i router si riducono le dimensioni e il consumo di potenza dei router nella core network, perché gran parte del traffico è trattato per una maggiore efficienza energetica in SDH / SONET e / o strati WDM.

1.2.2 Migliorando l'efficienza energetica di un Router

In generale, per ridurre il consumo energetico in un router, possiamo immaginare di mettere alcuni o tutti i componenti del dispositivo in uno stato di sospensione a basso consumo energetico oppure rallentando il clock dell'hardware. Guardando verso le varie architetture, possiamo identificare i componenti principali che possono essere messi in sospensione - la memoria, il processore, bus, scheda / ASIC, e lo swith di fabbrica. Si nota che la maggior parte delle memorie commerciali possono disporre di varie modalità di risparmio energetico. Quindi utilizzando questo modulo di memoria, si può ridurre il consumo energetico nei dispositivi di rete. Cerchiamo di esaminare la fattibilità di mettere in sospensione i componenti rimanenti.

Scheda di rete: vanno da quelle molto semplici a molto complesse contenenti processori di rete, interfacce, memoria, e interfacce di switch di fabbrica. Mettere una scheda di rete o qualsiasi dei suoi componenti in modo "sleep", risparmia energia, però, secondo la rapidità con cui fa il passaggio allo stato di "sveglia", ci saranno perdita di pacchetti significativi e latenza. Se assumiamo che le schede di rete si svegliano automaticamente al rilevamento di dati sulle loro porte d'ingresso (questa è una cosa ragionevole da fare in hardware poiché la logica per il rilevamento di una trasmissione su una scheda è semplice e consuma poca energia), poi, se ci vuole 10μs per la transizione allo stato sveglio, su un collegamento che funziona a 1Gbps, ci si

può aspettare di perdere circa 10kbit di dati e di aggiungere 10μs potenzialmente di ritardo end-to-end. Questo tipo di comportamento non è accettabile e devono essere sviluppate strategie per evitare che questo occorra. Si possono prevedere due possibili modi per affrontare questo problema:

- un approccio a livello di rete (chiamato sospensione coordinata) in cui il protocollo di routing aggiunge traffico ad alcuni percorsi predefiniti (durante basso carico) per consentire esplicitamente alcune interfacce di dormire o;

- un approccio a livello data link (chiamato sospensione non-coordinata) quando un'interfaccia è in sospensione basata su una decisione locale. Quest'approccio potrebbe lavorare come segue - quando l'interfaccia è quasi pronta per andare in sospensione, informa ai suoi vicini di questo fatto (in un collegamento punto-punto il vicino è ben definito, per tutti gli altri, i vicini avvisati mediante una trasmissione broadcast). Quando il vicino ha bisogno di inviare pacchetti a questa interfaccia in stato di sospensione, invia un primo pacchetto che costringe l'interfaccia a svegliarsi. Poi, dopo aver atteso il tempo necessario per la sveglia, sono inviati i pacchetti. Questo meccanismo evita la perdita di pacchetti ma aggiunge ritardo end-to-end.

Il processore di rete di una scheda può andare in modalità di risparmio energetico rallentando la sua velocità di clock quando ci sia poco traffico nella rete. Chiaramente, questo tipo di tecnologia consentirà un risparmio energetico. La sfida della ricerca è quello di determinare quando a rallentare il clock del processore.

Crossbar: Mettere il crossbar in modo "sleep", combinato con le schede di rete, non dovrebbe necessariamente comportare un'ulteriore latenza o perdita. La ragione è che quando la scheda di rete viene svegliata, ci si può svegliare contemporaneamente la crossbar.

Quindi, quando i pacchetti cominciano ad arrivare alla scheda di rete, il crossbar è pronto a indirizzarli. Se invece, la scheda di rete non "dorme", ma il

crossbar lo fa, allora le perdite possono verificarsi al punto d'ingresso fino al crossbar, salvo che, aumentino notevolmente la quantità di buffer nella scheda di rete.

Processore principali: Il processore principale è di solito a RISCprocessor (Reduced instruction set computing) che funziona a velocità dell'ordine dei GHz. Potrebbe essere possibile far funzionare al clock del processore più lento quando c'è poco traffico nella rete. Tuttavia, la questione è quando si può fare? In sintesi, sembra che diversi sottocomponenti del router possono essere messi in sospeso e poi riaccesi poco prima di arrivare ai pacchetti (utilizzando il meccanismo delineato prima). Tuttavia, come risultato, porterà un aumento del ritardo end-to-end a causa del ritardo di riaccendere le schede di rete e gli altri componenti. Adesso rivolgiamo l'attenzione sull'impatto di mettere i componenti hardware in modo "sleep" sui protocolli d'Internet, e la decisione di quando mettere l'hardware in stato di sospensione.

Come e quando avviene la sospensione?

Quando si pensa di mettere le schede di rete di un router a dormire, ci sono due questioni più importanti:

• Per quanto tempo questi componenti possono "dormire"? Quando si sveglia un dispositivo in sospensione, di solito c'è un picco nel consumo energetico. Perciò, il dispositivo dovrebbe dormire abbastanza da compensare l'energia supplementare consumata nel risvegliarsi. Inoltre, dato che un dispositivo ci mette x microsecondi a svegliarsi, il periodo in "sleep" deve essere più lungo di x.
Chiaramente questo definisce il tempo minimo per un dispositivo che può essere messo a dormire e avere un guadagno in termini energetici.

Quanto tempo può rimanere addormentato?

Questo dipende dal fatto che la sospensione sia coordinata o non-coordinata.

Se la decisione è non-coordinata, si può vedere che il periodo massimo di sospensione sarà dettato dalle esigenze di attività periodica dei protocolli d'Internet - ad esempio, nel caso di OSPF (Open Shortest Path First) la necessità d'inviare e ricevere messaggi periodici HELLO inserisce automaticamente un limite superiore a questo intervallo di "sleep". Se, invece, i periodi di sospensione sono coordinati a livello di rete (e se i protocolli d'Internet sono modificati per il supporto del modo di sospensione) questi periodi possono essere più lunghi.

• Come si decide di andare al modo "sleep"?

Quando un router sta decidendo di andare al modo "sleep" in maniera non-coordinata, la migliore cosa che può fare è monitorare il traffico su tutte le sue interfacce, determinare i tempi d'interarrivo tra pacchetti e poi mettere le interfacce a dormire sulla base di una stima dei tempi d'interarrivo. Se questa stima è troppo grande (per esempio, un pacchetto arriva mentre l'interfaccia è addormentata), quindi i pacchetti che arrivano devono svegliare l'interfaccia e la stima del tempo d'arrivo dovrà essere ricalcolata.

Se, invece, la decisione si prende in maniera coordinata a livello di un'area in un AS, poi il tempo stimato può essere ricavato misurando il traffico nella rete (di nuovo, sulla base uno schema di misura di finestra di tempo), aggregazione di esso, e poi informare ai router selezionati di dormire.

Quali router sono più plausibili per andare in sospensione?

Se guardiamo l'architettura d'Internet, si nota che secondo dove si trovano, alcuni router sono più suscettibili di altri a dormire. Se l'AS non agisce come una rete

di transito per altre reti, tutto il traffico parte da /per l'AS solo. In questi casi, è più facile prevedere il tipo di traffico ed elaborare un appropriato programma di sospensione. Per esempio, i router all'interno di un'area che instradano il traffico dagli utenti finali durante periodi di lunghi intervalli di tempo d'inattività (ad esempio di notte). All'interno dei router della dorsale dell'AS si può anche dormire nei periodi di basso carico se ci sono multipli cammini tra di loro. Inoltre è possibile mettere il router di bordo di un AS o uno stub a dormire perché essi potrebbero non vedere molto traffico (salvo che le applicazioni fornite dall'AS siano del tipo web-based, in questo caso ci può essere traffico costante).

Al fine di estendere la sospensione agli AS di transito, questi hanno bisogno di aggiungere internamente più link per consentire una migliore aggregazione di traffico durante periodi basso carico, perciò l'abilitazione di un gran numero di questi AS a dormire. Ciò andrà a beneficio degli ISP (Internet Service Provider) perché i costi di raffreddamento tendono a essere alti e la sospensione ridurrà la dissipazione del calore (oltre a ridurre l'elettricità assorbita dai router).

CAPITOLO II

ROUTING

2.1 Il Router

I router sono dei dispositivi che permettono di "scegliere" il percorso che i datagrammi prenderanno per arrivare a destinazione. Si tratta di terminali con più schede di rete ciascuna collegata a una rete differente. Quindi, nella configurazione più semplice, il router deve solo "guardare" su quale rete si trova un computer per fargli arrivare i datagrammi da parte del mittente.

Tuttavia, su Internet lo schema è molto più complicato per le ragioni seguenti:

- Il numero di reti alle quali il router è connesso è solitamente elevato
- Le reti alle quali il router è collegato possono essere connesse ad altre reti che il router non conosce direttamente

Così, i router funzionano grazie a delle tabelle di routing e dei protocolli di routing, secondo il modello seguente:

- Il router riceve una trama proveniente da un terminale connesso a una delle reti alla quale è collegato
- I datagrammi sono trasmessi al livello IP
- Il router guarda l'intestazione del datagramma

- Se l'indirizzo IP di destinazione appartiene a uno degli indirizzi cui una delle interfacce del router è collegata, l'informazione deve essere inviata al livello 4 dopo che l'intestazione IP sia stata disincapsulata (tolta)
- Se l'indirizzo IP di destinazione fa parte di una rete diversa, il router consulta la propria tabella di routing, che definisce il percorso da prendere per un indirizzo dato
- Il router invia il datagramma grazie alla scheda di rete collegata alla network sulla quale il router decide di inviare il pacchetto

Così, si hanno due scenari, sia l'emettitore sia il destinatario appartengono alla stessa rete in quel caso si parla di *rimessa diretta*, oppure vi è almeno un router tra il mittente e il destinatario, e allora si parla di *rimessa indiretta*. In caso di rimessa indiretta, il ruolo del router, soprattutto quello della tabella di routing, è molto importante. Quindi il funzionamento di un router è determinato dal modo in cui questa tabella di routing è creata.

Se la tabella di routing è inserita manualmente dall'amministratore, si parla di routing statico (utilizzabile per piccole reti)
Se il router costruisce da solo la tabella di routing in funzione delle informazioni ricevute (attraverso i protocolli di routing), si parla di routing dinamico.

2.1.1 La tabella di routing

La tabelle di routing è una tabella di corrispondenza tra l'indirizzo del terminale scelto e il nodo seguente al quale il router deve consegnare il messaggio. In realtà basta che il messaggio sia consegnato sulla rete che contiene il terminale, e non è quindi necessario stoccare l'indirizzo IP completo del terminale: solo l'identificatore di rete dell'indirizzo IP (cioè l'ID di rete) ha bisogno di essere stoccato.

Grazie a questa tabella, il router, conoscendo l'indirizzo del destinatario incapsulato nel messaggio, sarà capace di sapere a quale interfaccia inviare il messaggio (cioè quale scheda rete usare), e a quale router, direttamente accessibile sulla rete cui la carta è connessa, rimettere il diagramma.

Questo meccanismo consistente nel riconoscere solo l'indirizzo del prossimo passaggio che porta a destinazione viene detto *routing per salti successivi* (in inglese *next-hop routing*). Comunque, può succedere che il destinatario appartenga a una rete non referenziata nella tabella di routing. In questo caso, il router usa un router di default (detto anche *passerella di default*).

Ecco, in maniera semplificata, a cosa potrebbero assomigliare delle tabelle di routing:

Rete di destinazione	Scheda di rete su cui inoltrare i pacchetti	Prossimo router
A	Scheda di rete 1	NESSUNO
B	Scheda di rete 2	Router 2
C	Scheda di rete 3	Router 3
D	Scheda di rete 2	Router 2
E	Scheda di rete 3	Router 3

Fig. 2.1 Esempio molto semplice di una tabella d'instradamento

Il messaggio è così rimesso da router in router per salti successivi, fino a che il destinatario appartenga a una rete direttamente connessa a un router. Quest'ultimo allora rimetterà direttamente il messaggio al terminale scelto.

In caso di router statico, è l'amministratore che aggiorna la tabella di routing. In caso di router dinamico, invece, un protocollo detto protocollo di routing permette l'aggiornamento automatico della tabella affinché contenga in ogni momento il percorso ottimale.

2.1.2 I protocolli di routing

Internet è un insieme di reti connesse. Di conseguenza tutti i router non fanno lo stesso lavoro secondo il tipo di rete in cui si trovano. In effetti, vi sono differenti livelli di router, che funzionano quindi con protocolli diversi:

- I router nodo sono i router principali dato che sono loro a collegare le differenti reti
- I router esterni permettono un collegamento di reti autonome fra loro. Funzionano con un protocollo detto EGP (Exterior Gateway Protocol) che evolve passo dopo passo mantenendo la stessa denominazione
- I router interni permettono il routing delle informazioni all'interno di una rete autonoma. Scambiano le informazioni grazie ai protocolli detti IGP (Interior Gateway Protocol), come RIP e OSPF.

RIP significa *Routing Information Protocol* (protocollo d'informazione di routing). Si tratta di un protocollo di tipo *Vector Distance*, cioè ogni router comunica agli altri router la distanza che li separa (il numero di salti che li separa). Così, quando un router riceve uno di questi messaggi, aumenta questa distanza di 1 e comunica il messaggio ai router direttamente accessibili. I router possono quindi conservare il percorso ottimale di un messaggio stoccando l'indirizzo del router seguente nelle tabelle di routing in modo che il numero di salto possa raggiungere una rete anche minima. Tuttavia, questo protocollo prende in considerazione solo la distanza tra due terminali in termini di salto, ma non considera lo stato della connessione per scegliere la miglior banda passante possibile.

OSPF (*Open Shortest Path First*) ha più prestazioni di RIP e lo sta pian piano sostituendo. Si tratta di un protocollo di tipo *protocollo route-ling* (che si può tradurre come *Protocollo di stato dei collegamenti*), questo significa che, contrariamente al RIP, questo protocollo invia ai router adiacenti solo il numero di salti che li separa,

ma anche lo stato del collegamento che li separa. In questo modo, ogni router è capace di redigere una scheda dello stato di rete e può scegliere di conseguenza in ogni momento il percorso più appropriato per un dato messaggio.
In più, questo protocollo evita ai router intermedi di dover aumentare il numero di salti, il che si traduce con un'informazione molto meno abbondante, che permette di avere una migliore banda passante utile rispetto a RIP.

2.2 OSPF

Il protocollo OSPF, descritto nel RFC 2328, si occupa della gestione dell'instradamento del traffico dati all'interno dei Sistemi Autonomi (AS). Per effettuare tale instradamento, sulla base degli indirizzi IP di destinazione dei pacchetti, è necessario costruire e aggiornare in ogni Router all'interno dell'AS delle tabelle di routing. Queste tabelle consistono in un elenco di destinazioni, in genere raggruppate in sottoreti, e vengono associati agli indirizzi del prossimo Router ai quali inviare il traffico loro diretto (indicazione del next-hop). L'OSPF è un protocollo di routing dinamico, in grado di individuare rapidamente ogni variazione nella topologia dell'AS e di calcolare in breve tempo, nel caso si siano verificati dei guasti, un percorso alternativo. Infatti il problema principale di questi sistemi di Routing è di mantenere aggiornate tutte le tabelle d'instradamento anche in presenza di variazioni nella configurazione della rete.

La gestione dinamica dell'instradamento viene eseguita sfruttando il funzionamento link-state. Ogni Router comunica a tutti gli altri Router all'interno dell'AS lo stato dei suoi link, cioè delle sue connessioni. In questo modo tutti i Router riescono a costruire una 'immagine topologica' dell'AS, cioè una descrizione della configurazione di tutte le connessioni nel sistema. In base alla conoscenza di questa topologia, ogni Router costruirà in modo indipendente la propria tabella d'instradamento, nel modo più efficiente possibile. In assenza di anomalie tutti i Router all'interno dello stesso AS avranno la stessa identica immagine topologica

dell'AS. Discrepanze possono aversi in presenza di cambiamenti, durante un periodo detto tempo di convergenza del sistema. Ogni altra diversità stabile delle topologie è da considerarsi un errore di funzionamento del protocollo.

I pacchetti scambiati fra i diversi Router contenenti la descrizione della topologia dell'AS sono i *Link State Advertisement (LSA)*. Ogni *LSA* descrive lo stato di un particolare router, ovvero quali sono le interfacce del router attualmente attive, le reti o gli altri router ai quali è connesso, e i costi associati a queste connessioni. Ogni diverso link, infatti, avrà associato una metrica, inserita in modo da stabilire una graduatoria preferenziale nell'utilizzo dei link. Per intenderci se per andare da un nodo A ad un nodo B esistono due percorsi con metrica 5 e 10 rispettivamente, verrà utilizzato sempre quello con metrica 5. Se esiste un terzo percorso costituito da due link con costo 1 e 2, allora verrà utilizzato sempre quest'ultimo cammino, poiché il costo totale diventa 3.

La distribuzione delle informazioni contenute negli LSA all'interno dell'AS è un punto critico del funzionamento dell'OSPF. Se da un lato, infatti, è necessario che tutti i Router ricevano un LSA originato da un Router particolare, è anche impensabile che quest'ultimo invii tanti diversi pacchetti quanti sono i Router all'interno dell'AS, anche perché il Router sorgente potrebbe ancora non conoscere l'esatta topologia dell'AS, e quindi non saprebbe come raggiungere tutti gli altri Router. Viene allora utilizzata l'efficace tecnica di "*flooding*", che verrà qua brevemente descritta.

2.2.1 Il Flooding

E' uno dei possibili metodi per propagare un messaggio attraverso una rete in modo che raggiunga tutti i nodi. Nel caso dell'OSPF viene utilizzato per far sì che tutti i Router all'interno di un AS ricevano i LSA originati da ogni altro Router. Il punto cardine della tecnica consiste nello sfruttare le connessioni che ogni Router ha in

modo da ridistribuire i pacchetti ricevuti verso gli altri Router. In particolare quando un Router origina un nuovo LSA, con la sua descrizione, la invia su tutte le sue interfacce di rete, con metodologie diverse a seconda del tipo di connessione. In particolare se si tratta di connessioni che permetto l'utilizzo di messaggi broadcast, sarà utilizzato l'indirizzo di broadcasting, in modo da raggiungere ogni altro Router connesso su quella rete. Se invece si tratta di connessioni di tipo Point to Point (P2P), il destinatario sarà direttamente il partner P2P. Successivamente, quando un Router riceve l'LSA verifica, se si tratta di una nuova istanza o una replica, cioè controlla se ha già memorizzato quella particolare LSA nel suo Database. Se si tratta di un duplicato, scarta semplicemente l'LSA. Se invece si tratta di una nuova LSA, questa verrà memorizzata nel database e re-inviata su tutte le altre interfacce di rete. La tecnica di Flooding consiste in quest'ultimo passo: ogni Router propagherà la nuova informazione su tutte le sue connessioni ad eccezione di quella da cui l'ha ricevuta. Il principio di base è che, se ricevo un LSA da un'interfaccia, vuol dire che quel ramo di rete è già a conoscenza di quell'informazione, mentre tutti gli altri potrebbero non averla ancora ricevuta. Per questo motivo si rinvia l'informazione a tutti gli altri vicini, affinché loro la ripropaghino verso il resto della rete. E' chiaro che nella realtà ogni router riceverà lo stesso LSA più volte, ma essendo in grado di distinguere fra informazioni nuove e duplicate, ed evitando di propagare i duplicati, il traffico generato viene tenuto sotto controllo. La tecnica di flooding, in questo modo, propaga l'informazione a macchia d'olio attraverso l'intero AS, in tempi relativamente bassi, senza generare eccessiva congestione, e necessitando di un sistema di controllo minimo. Essenzialmente l'unico requisito necessario per effettuare il flooding è la presenza di un Database in cui memorizzare gli LSA già ricevuti, in modo da distinguere i duplicati, e questa struttura è già presente in OSPF perché necessaria per costruire il grafo topologico dell'AS, come descritto nel seguito.

2.2.2 La Topologia di Rete: il Link State Database

Abbiamo già accennato al problema della costruzione della tabella di routing all'interno di ogni Router dell'AS. Le metodologie e le tecniche per risolvere questo problema verranno affrontate nei capitoli seguenti. Per ora quello che ci interessa è che per generare correttamente la tabella d'instradamento ottima, cioè che sfrutti i percorsi all'interno dell'AS nel modo più efficiente possibile in termini di costo di attraversamento dei link, è necessario che ogni Router disponga di una immagine completa e unica dell'intera topologia dell'AS.

In generale un rete costituita da Router interconnessi attraverso link generici può essere rappresentata da un grafo orientato. Ad ogni ramo del grafo sarà associata una metrica, che costituisce il costo di attraversamento di quel ramo. I rami utilizzati nell'OSPF sono unidirezionali, il che già fa comprendere che non ci sarà una corrispondenza uno a uno fra i rami del grafo e le connessioni fra i router, che invece possono essere bidirezionali.

Vediamo allora nel dettaglio come viene costruito il grafo descrivente la topologia di un AS. Per prima cosa è necessario specificare che le connessioni fra Router possano essere di diverso tipo. Esistono connessioni Point to Point, cioè link diretti fra due Router, o connessioni attraverso Network, cioè collegamenti di rete su cui potranno essere presenti altri Router. Se su di una Network è presente almeno un altro Router, questa sarà una Transit Network, cioè potrà essere attraversata per raggiungere l'altro Router. Se invece non ci sono altri Router collegati alla Network si tratterà di una Stub Network, che verrà utilizzata unicamente per raggiungere le altre terminazioni di rete (ad esempio PC, server, etc.) connessi alla rete, ma che non effettuano funzioni di Routing. Per esempio sulle Stub Network non verrà effettuato il Flooding, in quanto non ci sarebbe nessun destinatario interessato a quelle informazioni.

Le Transit Network, a loro volta, si differenziano in Broadcast Network, che hanno un indirizzo dedicato su cui mandare i messaggi destinati a tutti gli altri utenti

della Network, e Non Broadcast Network (NBMA Non Broadcast Multiple Access). In entrambi i casi per una Transit Network verrà eletto fra i Router ad essa collegati, un Designated Router (DR) che si occuperà di gestire le funzionalità OSPF per quella rete. Nel caso di NBMA il DR emula le funzionalità di Broadcasting, ridistribuendo le informazioni a tutti i Router collegati. Il DR, in particolare, mantiene l'elenco dei Router collegati alla rete, e genera le LSA di descrizione di quella rete.

Ritorniamo ora al grafo orientato per la descrizione di un AS. Ogni router costituisce un vertice del grafo. Ogni Transit Network, a sua volta, costituisce un vertice del grafo. Ogni collegamento fra un Router ed una Network, una Network e un Router, o fra un Router ed un altro Router (P2P) è rappresentato da un ramo con associato un diverso costo. Dato che le Network hanno in realtà un costo di attraversamento, mentre nel nostro caso per attraversare una network vengono percorsi due diversi rami, il costo verrà associato solo al ramo entrante verso la Network, mentre i rami uscenti dalle Network avranno sempre costo nullo. In questo modo, Router diversi collegati alla stessa rete possono avere costi di attraversamento differenti di quella rete. Questa funzionalità è utile perché al di là dei problemi di traffico, comuni a tutti i Router connessi a una rete, diversi Router potrebbero aver stabilito accordi commerciali con i gestori della rete e usufruire di agevolazioni all'attraversamento, e quindi preferire una Network piuttosto che un'altra.

Specifichiamo che non sono presenti vertici nel grafo per i collegamenti P2P, rappresentati da semplici rami fra due Router, con associati i loro costi di attraversamento.

Vediamo adesso come i singoli Router memorizzano il grafo orientato per costruire la tabella di instradamento.

Abbiamo detto che ogni Router e ogni Network originerà un LSA con la propria descrizione. La struttura completa degli LSA verrà descritta nel seguito. Per

ora possiamo però accennare che ogni LSA comprende un identificativo del vertice a cui si riferisce (Router o Network che sia), e una descrizione di tutti i link uscenti da quel vertice. Nel caso di Router, i link specificheranno il tipo di connessione, l'identificativo del Vertice a cui si è collegati e il costo di attraversamento. Nel caso di Network, invece, i link comprenderanno solo i Router di destinazione, perché il costo uscente è sempre nullo. Apparirà chiaro, a questo punto, che i link descritti negli LSA sono semplicemente i rami del grafo orientato, e che ogni LSA corrisponde ad un Vertice del grafo. In questo modo l'intero set degli LSA costituisce una rappresentazione completa del grafo topologico dell'AS. Per capirci è sufficiente che ogni Router memorizzi l'intero set di LSA originate da ogni Router e Network all'interno dell'AS affinché conosca in maniera completa il grafo topologico dell'AS stesso. Sulla base di questo grafo, con le tecniche descritte nel seguito, verranno costruite le diverse tabelle di instradamento.

2.2.3 Shortest-Path-First tree

Ogni router deve creare una sua tabella di routing dal grafo rappresentante la topologia della rete, calcolando un albero di percorsi più brevi (shortest path first tree: SPF) avente come radice il nodo rappresentante il router stesso. L'obiettivo è di trovare il percorso con associato il costo totale minore dal router radice, cioè quello che sta costruendo la tabella di routing, e verso ogni altra destinazione, cioè ogni altro vertice del grafo (Routers e Network).

Ovviamente tale albero dipenderà dal nodo di partenza e sarà dunque specifico di ogni router nella rete.

La costruzione di questo spanning tree[1] a costi minimi viene fatta utilizzando un algoritmo particolare di calcolo in modo da ottimizzarne l'esecuzione. L'algoritmo

[1] Uno Spanning Tree è un albero che contiene tutti i vertici del Grafo di origine. Ovviamente il numero di rami dello Spanning Tree è minore o uguale del numero di rami del Grafo.

scelto è quello di Dijkstra, che verrà descritto ed analizzato dettagliatamente nel seguito. La base di partenza su cui lavora l'algoritmo di Dijkstra è il grafo topologico dell'AS, costituito in particolare dall'insieme degli LSA memorizzati nel Router.

Nel caso dell'OSPF, inoltre, è importante sottolineare che l'obiettivo finale resta la costruzione della tabella di instradamento. Quindi per ogni destinazione è sufficiente conoscere e memorizzare solamente il primo passo del cammino a costo minimo trovato (next hop), cioè il prossimo Router a cui inviare i pacchetti per quella destinazione. Questo particolare non cambia la tecnica utilizzata per calcolare lo SPF, ma solo le informazioni che saranno memorizzate alla fine dell'algoritmo.

2.2.4 Suddivisione di un AS in aree

Il protocollo OSPF permette di raggruppare una serie di reti confinanti e di host in aree, in modo da suddividere gli AS più grossi in blocchi più piccoli. Lo scopo di questa suddivisione è di rendere più flessibile il funzionamento dell'OSPF, ridurre il traffico dovuto al flooding e velocizzare il calcolo dello SPF.

Ogni area infatti esegue una copia a sé stante dell'algoritmo di routing, avrà una copia differente del set di LSA, e eseguirà separatamente le funzioni di flooding.

La topologia interna ad un'area è invisibile all'esterno e viceversa i router all'interno di un area non conoscono la topologia esterna. Il raggiungimento delle destinazioni al di fuori dell'Area è affidato a Router di confine, che smistano il traffico fra Aree.

Questa funzionalità non verrà descritta perché non interessante al fine della comprensione del lavoro svolto. Per conoscere i dettagli tecnici si rimanda all'esaustivo RFC 2328.

2.2.5 Il funzionamento dell'OSPF

Quando viene avviato, un router inizializza le strutture dati relative al protocollo di Routing e lancia il protocollo di Hello al fine di individuare i propri vicini, ovvero i router ad esso direttamente connessi.

Durante questa fase il Router invierà sulle proprie interfacce dei pacchetti di Hello contenenti l'identificativo del Router, finalizzati ad informare i Router vicini della propria presenza e riceverà sulle stesse interfacce analoghi pacchetti di Hello provenienti dai Router vicini.

A questo punto il Router cercherà di formare con essi delle adiacenze, eleggendo quando necessario un Designated Router.

Una volta stabilita l'adiacenza, i router inizieranno a scambiarsi LSA per sincronizzare i propri link-state Database.

In seguito ogni router dovrà periodicamente rispedire gli LSA descriventi il suo stato, anche se nel frattempo esso non ha subito alcun cambiamento, per far sì che, in caso di spegnimento, i vicini possano cancellarlo dal proprio database.

Gli LSA, grazie alla tecnica di flooding vengono inoltrati su tutta l'area. L'algoritmo di flooding è un algoritmo molto affidabile, e assicura che, dopo lo scambio degli LSA, tutti i Router di un'area abbiano la stessa identica copia del link-state database.

Il calcolo dello SPF tramite l'algoritmo di Dijkstra serve per mantenere aggiornate le tabelle d'instradamento. Deve essere eseguito dopo ogni modifica della topologia dell'AS, quindi ogni volta che si ricevono degli nuovi LSA, oppure quando se ne elimina qualcuno dal database, perché scaduta. In genere il calcolo dello SPF

non viene eseguito immediatamente dopo la ricezione di un nuovo LSA, perché entro breve termine si potrebbero ricevere ulteriori aggiornamenti, e quindi si dovrebbe ripetere il calcolo. Per questo l'esecuzione dello SPF viene ritardata di alcuni secondi.

Un ultima considerazione riguarda il sistema di Acknowledging implementato in OSPF. Consiste nell'invio di riscontri positivi (ACK) quando si ricevono correttamente i pacchetti LSA. Il comportamento dell'OSPF nell'invio dei riscontri è differente a seconda che l'LSA ricevuta sia nuovo o duplicato. Nel primo caso l'invio può essere ritardato, per dare la precedenza all'esecuzione delle procedure di flooding. Nel caso, invece, che il pacchetto sia un duplicato, l'invio dell'Acknowledge deve essere inviato immediatamente, in modo da evitare ritrasmissioni di informazioni inutili.

2.2.6 I pacchetti OSPF

Il protocollo OSPF viaggia direttamente incapsulato nei pacchetti IP. Se possibile, occorre dare ai pacchetti del protocollo di routing una preferenza maggiore rispetto a quella dei normali pacchetti di traffico.

Tutti i pacchetti OSPF hanno in comune lo stesso header mostrato in figura 2.2.

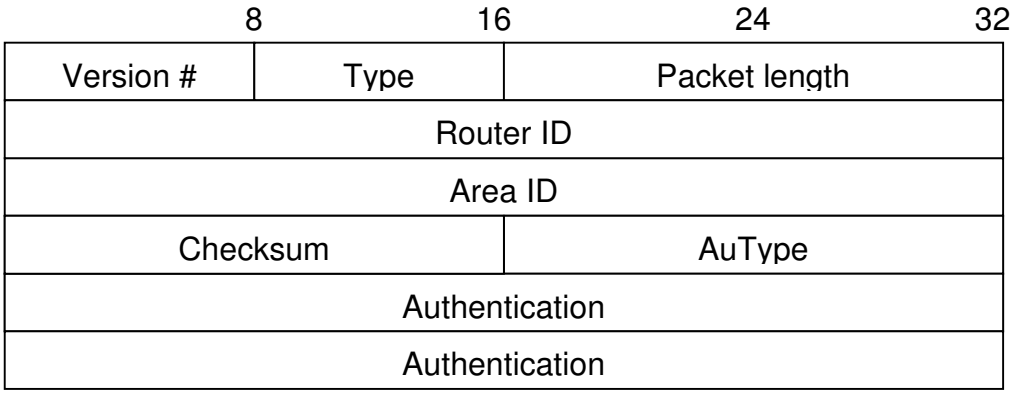


Fig. 2.2 Header pacchetti OSPF

Il campo Type contiene informazioni riguardo al tipo di pacchetto OSPF trasportato, come indicato in figura 2.3.

Type	Packet name	Protocol function
1	Hello	Scopre/mantiene i vicini
2	Database Description	Riassume il contenuto del database
3	Link State Request	Richiesta di aggiornamento sul database
4	Link State Update	Aggiornamento sul database, raggruppa e contiene le LSA (Link State Advertisement)
5	Link State Acknowlegde	Scambio avvenuto

Fig. 2.3 Pacchetti OSPF

La nostra attenzione sarà incentrata sui pacchetti Link State Update (LSU) che trasportano le informazioni topologiche dell'AS. Un singolo pacchetto di Link

State Update può contenere diversi LSA, raggruppate durante la procedura di flooding, oppure un singolo LSA appena originato da un Router o da una Network.

Ogni LSA ha un header che ne identifica il router sorgente e il tipo di LSA come schematizzato in figura 2.3 e 2.5

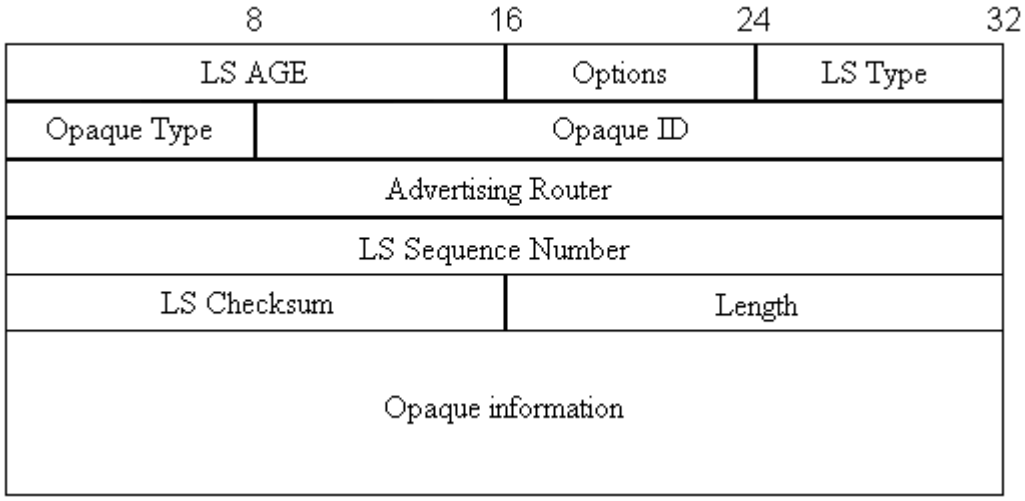


Tabella 2.4 LSA Header

LS type	LSA name	LSA description
1	Router-LSA	Originati da tutti i router, descrivono lo stato delle interfacce del router su di un'area e vengono inoltrati su tutta l'area.
2	Network-LSA	Originati nel caso di reti broadcast e NBMA dal DR, contengono una lista dei router connessi alla rete e vengono inoltrati a tutti i router di un'area.

3,4	Summary LSA	Originati dagli area border routers. Ogni summary LSA descrive un percorso ad una destinazione fuori dall'area in cui viene spedito
5	AS-external LSA	Originati dagli AS boundary routers, vengono inoltrati a tutti i router dell'AS. Ogni LSA descrive un percorso ad una destinazione in un altro AS.

Tabella 2.5 LSA Type

2.3 Router software open source basati su PC

I router sono le componenti chiave delle reti a pacchetto IP. L'invito ad apparecchiature di commutazione e trasmissione ad alte prestazioni su Internet continua a crescere a causa della crescente diffusione delle tecnologie dell'informazione e della comunicazione e la diffusione delle nuove applicazioni e servizi di grandi richieste di banda come streaming audio e video.

Finora, i router sono stati in grado di sostenere la crescita del traffico, offrendo una velocità sempre crescente di commutazione, soprattutto grazie al progresso tecnologico della microelettronica concessi dalla legge di Moore.

Contrariamente a quello che è successo per le architetture PC, dove, almeno per i componenti hardware, standard sono stati definiti, permettendo lo sviluppo di un mercato aperto multi - venditore del mercato, apparecchiature di rete in generale, e router in particolare, hanno sempre visti molti sviluppi. Architetture proprietarie sono affetti da incompatibilità di configurazione e procedure di gestione, scarsa flessibilità, e il costo è spesso molto superiore al valore effettivo dei dispositivi.

Interessanti alternative ai dispositivi di rete di proprietari sono le implementazioni di router software basati su PC hardware, che sono stati di recente a disposizione dalla comunità del software open - source, come Linux, Click e FreeBSD per il piano dati, così come Xorp e Zebra per il piano di controllo, le critiche ai router software sono focalizzati sulle prestazioni limitate, instabilità di software, mancanza di supporto del sistema, problemi di scalabilità, la mancanza di funzionalità.

I limiti delle prestazioni può essere compensato dalla naturale evoluzione delle prestazioni di l'architettura del PC. Gli attuali router e switch basati su PC hanno la potenzialità di commutazione fino a un paio di Gbit / s di traffico, che è più che sufficiente per un gran numero di applicazioni. Oggi, la maturità del software open - source supera gran parte dei problemi relativi alla stabilità e alla disponibilità di funzionalità software.

2.3.1 Panoramica architettonica del PC

Un PC è composto fondamentalmente da tre elementi principali: l'unità centrale di elaborazione (CPU), memoria ad accesso casuale (RAM), e le periferiche, incollati tra di loro dal chipset (scheda madre), che prevede l'interconnessione e le funzioni di controllo.

Come delineato nella fig. 2.6, la CPU si comunica con il chipset attraverso il sistema bus, noto anche come font side bus (FSB) in gergo di Intel. La RAM fornisce archiviazione dei dati temporanea per la CPU mentre il sistema è acceso, ed è possibile accedervi dal controller di memoria integrato sul chipset attraverso il bus di memoria (MB). Le periferiche sono collegati al chipset mediante il bus Peripheral Component Interconnect (PCI), che permette di espandere il sistema con un enorme numero di dispositivi, tra cui, ma non solo, unità di memorizzazione dati permanente, bus di espansione aggiuntivi, schede video, schede audio e schede di rete. Tutte le interconnessioni sono bidirezionali, ma, purtroppo, utilizzano diversi parallelismi, protocolli, e velocità di clock, che richiede l'implementazione di funzioni di traduzione e adattamento sul chipset.

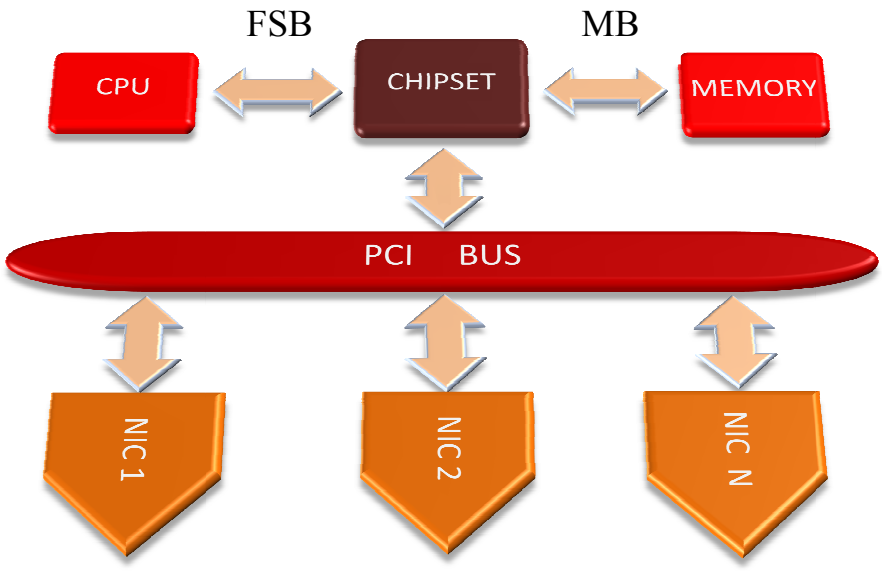


Fig.2.6 Architettura di un PC

L'hardware disponibili su un PC consente di implementare un bus condiviso, memoria condivisa del router, schede di rete dove ricevere e memorizzare i pacchetti nella RAM principale, il CPU inoltra l'informazione all'interfaccia di output corretta, e schede di rete che prendono i pacchetti dalla RAM e li trasmetti sul cavo. Pertanto, ogni pacchetto viaggia due volte sul bus PCI, dimezzando la larghezza di banda effettivamente disponibile per i flussi di pacchetti NIC-to-NIC.

2.3.2 Quagga

Quagga è un pacchetto software che fornisce servizi di routing basato su TCP / IP con supporto di protocolli di routing come RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4 e BGP-4 +. Quagga supporta anche BGP Route Reflector e il comportamento Route Server. Oltre ai tradizionali protocolli di routing IPv4, Quagga supporta anche protocolli di routing IPv6. Con il demone SNMP che supporta il protocollo SMUX, Quagga fornisce protocollo di routing MIBs (Management Information Base).

Quagga utilizza un'architettura software avanzata per offrire un motore multi server di routing di qualità elevata. Quagga dispone di un'interfaccia utente interattiva per ogni protocollo di routing e supporta i comandi client comuni. Grazie a questo disegno, è possibile aggiungere facilmente nuovi demoni di protocollo a Quagga. È possibile utilizzare la libreria di Quagga come interfaccia d'utente client del programma. Quagga è distribuito sotto la GNU General Public License.

Un sistema con Quagga installato si comporta come un router dedicato. Con Quagga, il computer scambia le informazioni d'instradamento con altri router utilizzando protocolli di routing. Quagga utilizza queste informazioni per aggiornare la tabella di routing del kernel in modo che i dati giusti vadano al posto giusto. È possibile modificare dinamicamente la configurazione e si può visualizzare le informazioni della tabella di routing dall'interfaccia terminale di Quagga.

Oltre al supporto del protocollo di routing, Quagga può impostare le “flags” dell’interfaccia, l'indirizzo dell'interfaccia, route statici e così via. Se avete una piccola rete, una stub network, o una connessione xDSL, la configurazione del software di routing Quagga è molto facile. L'unica cosa che si deve fare è impostare le interfacce e mettere alcuni comandi su percorsi statici e / o percorsi predefiniti. Se la rete è piuttosto grande, o se la struttura della network cambia frequentemente, si può sfruttare il supporto del protocollo di routing dinamico di Quagga per i protocolli come RIP, OSPF e BGP.

Tradizionalmente, la configurazione di router basata su UNIX è fatta utilizzando i comandi ifconfig e route. Lo stato delle tabelle di routing viene visualizzato con il programma d’utilità netstat. Quasi tutti questi comandi funzionano solo se l'utente ha privilegi di root. Quagga ha un diverso metodo di amministrazione del sistema. Ci sono due modi d'uso in Quagga. Uno è la modalità normale, l'altro è la modalità abilitato (enable). La modalità d’utente normale può solo visualizzare lo stato del sistema, la modalità d’utente abilitato può modificare la configurazione del sistema. Questa caratteristica indipendente “UNIX account” sarà di grande aiuto all'amministratore del router.

Attualmente, Quagga supporta protocolli di routing unicast comuni. Protocolli di routing multicast come BGMP, PIM-SM, PIM-DM possono essere sostenuti in Quagga 2.0. Supporto MPLS è in corso. In futuro, filtro TCP / IP, controllo di qualità del servizio e configurazione DiffServ verranno aggiunti a Quagga. L’obiettivo finale del progetto Quagga è di fare un software di routing libero TCP / IP produttivo e di qualità.

2.3.2.1 Architettura Del Sistema

Il software di routing tradizionale è fatto come un programma di un solo processo che fornisce tutte le funzionalità del protocollo di routing. Quagga invece adotta un approccio diverso. E' composto da una raccolta di diversi demoni che lavorano insieme per costruire la tabella di routing. Ci possono essere alcuni demoni di routing di protocollo specifici e il gestore di routing del kernel, zebra.

Il demone ripd gestisce il protocollo RIP, mentre ospfd è un demone che supporta
OSPF versione 2, bgpd supporta il protocollo BGP-4. Per modificare la tabella d'instradamento del kernel e per la ridistribuzione delle routes tra i diversi protocolli di routing, vi è il demone zebra. È facile aggiungere nuovi demoni di protocollo di routing all'intero sistema senza alterare nessun altro software. È necessario eseguire solo il demone associato al protocollo di routing in uso. Così, si potrà eseguire uno specifico demone di routing e inviare l'informazione a una centrale di routing.

Non c'è bisogno, ma è possibile che questi demoni siano in esecuzione sulla stessa macchina. Questa architettura crea nuove possibilità per il sistema di routing.

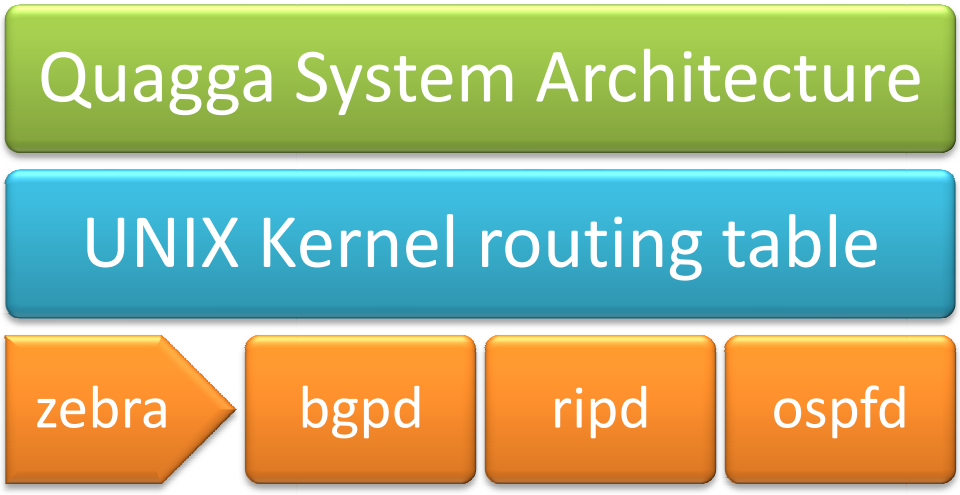


Fig.2.7 Architettura di Quagga

L'architettura multi - processo porta estensibilità, modularità e manutenzione. Allo stesso tempo, porta anche molti file di configurazione e interfacce del terminale. Quando si configura una route statica, deve essere fatto nel file di configurazione di zebra, e una rete BGP deve essere fatto nel file di configurazione bgpd. Quagga fornisce integrated user interface shell chiamata vtysh la quale connette ad ogni demone con socket di dominio UNIX e quindi funziona come un proxy per l'input dell'utente.

Quagga è stato progettato per utilizzare il meccanismo multi-threaded, quando viene eseguito con un kernel che lo supporta. Ma al momento, la libreria di thread, che viene fornito con GNU / Linux o FreeBSD, ha qualche problema con la gestione di servizi affidabili, quali il software di routing, quindi non si utilizzano threads. Invece si usa il comando del sistema select (2) per il multiplexing degli eventi.

Piattaforme supportate

Attualmente Quagga supporta GNU / Linux, BSD e Solaris. I demoni dei protocolli sono nella maggior parte dei casi di piattaforma indipendente. L'elenco delle piattaforme ufficialmente supportati sono elencati di seguito. Si noti che Quagga può funzionare correttamente su altre piattaforme, e può agire con funzionalità parziali su altre piattaforme.

• gnu/Linux 2.4.x and higher
• FreeBSD 4.x and higher
• NetBSD 1.6 and higher
• OpenBSD 2.5 and higher
• Solaris 8 and higher

CAPITOLO III

STRATEGIA DI RISPARMIO ENERGETICO

Questo capitolo riguarda alla base teorica della strategia EAR, i concetti principali e la soluzione trovata per implementare e testare la strategia proposta in un Software di Routing.

Per prima cominciamo con una breve esposizione della teoria dei grafi, molto importante per la comprensione e lo svolgimento in termini algebrici di EAR e l'inutilizzazione di un'interfaccia per il risparmio energetico.

3.1 Teoria dei grafi

I grafi sono oggetti discreti che permettono di schematizzare una grande varietà di situazioni e di processi e spesso di consentirne l'analisi in termini quantitativi e algoritmici. Un grafo è generalmente raffigurato sul piano da punti o cerchietti, che rappresentano i nodi; archi o spigoli sono rappresentati da segmenti o curve che collegano due nodi.

In questo caso, il posizionamento dei nodi e la forma degli archi o spigoli è irrilevante, dal momento che a contare, sono solo i nodi e le relazioni tra essi. In altri termini, lo stesso grafo può essere disegnato in molti modi diversi senza modificarne le proprietà.

Le strutture che possono essere rappresentate da grafi sono onnipresenti e molti problemi d'interesse pratico possono essere formulati come questioni relative a grafi. In particolare, le reti possono essere descritte in forma di grafi.

Un grafo G è una coppia (V, E) dove V è un insieme e $E \subseteq V \times V$ è un sottoinsieme del prodotto cartesiano di V. Gli elementi di V sono detti nodi e quelli di E sono detti archi. I nodi sono spesso chiamati anche "vertici". Gli archi sono detti anche "lati" o "spigoli".

Si distinguono due tipi di grafi:

- I grafi non orientati, dove la relazione E è simmetrica, quindi $(a, b) \in E \rightarrow (b, a) \in E$. In questo tipo di grafo, gli archi sono sovente denominati spigoli e i nodi vertici.
- i grafi orientati, dove la relazione E non è simmetrica ed esiste una relazione d'ordine tra i nodi.

Le definizioni di base che possono servire per la comprensione del capitolo, sono descritte sotto:

- ✓ Nodi adiacenti: Due nodi si dicono adiacenti se sono connessi da un arco.
- ✓ Archi adiacenti: Si dicono archi adiacenti quegli archi che hanno un nodo in comune.
- ✓ Vicinato: Per vicinato s'intende l'insieme di tutti i nodi adiacenti a un generico nodo.
- ✓ Archi multipli e loops: Gli archi multipli e i loops non verranno trattati, ma per completezza d'informazione diciamo che i primi sono archi differenti che collegano due stessi nodi, mentre i secondi sono archi che hanno come punto di partenza ed arrivo sempre lo stesso generico nodo.
- ✓ Grafo semplice: Si dice grafo semplice quel grafo che non contiene né archi multipli né loops.
- ✓ Cammino: È il percorso che va da un nodo ad un altro.
- ✓ Grafo aciclico: Grafo che non contiene cicli.

- ✓ Grafo connesso: Un grafo si dice connesso se esiste un cammino tra qualunque coppia di nodi.
- ✓ Grafo orientato: In un grafo orientato gli archi possono essere percorsi in un solo verso.
- ✓ Albero: Si dice albero quel grafo G con le seguenti caratteristiche:
    - ha n nodi
    - ha (n-1) archi
    - è connesso
    - è aciclico
- ✓ Foglia: Nodo di un albero con grado 1.
    - ogni albero ha almeno 2 foglie
    - due nodi di un albero sono connettibili da un solo cammino

3.2 Strategia ENERGY AWARE ROUTING (EAR)

La strategia del risparmio energetico considerata, è basata sull'abilità di mettere in'sleep mode' alcune interfacce di un router IP durante periodi di poco traffico, questo si ottiene modificando un percorso nella rete usato per indirizzare il traffico IP.

In un Sistema Autonomo (AS), vale a dire una network IP gestita da un singolo amministratore, il protocollo routing intra-AS supporta il calcolo del percorso nella rete. Il protocollo routing intra-AS più utilizzato è il "Open Shortest Path First" (OSPF) il quale sfrutta il classico algoritmo di Dijkstra per calcolare un percorso nella rete. I messaggi OSPF "Link State Advertisements" (LSAs) permettono di aggiornare continuamente il database di ogni nodo e calcolare il proprio "Shortest Path Tree" (SPT), in altre parole la serie di shortest paths di tutte le destinazioni della rete da se stesso. Il nostro scopo è di modificare questa procedura in modo tale da ridurre il consumo dell'energia nella rete, in particolare, una strategia di calcolo di un nuovo percorso definita capace di ridurre il numero delle interfacce attive nel router

IP utilizzate per inoltrare pacchetti; inoltre, la nuova strategia può essere facilmente integrata nel protocollo OSPF poiché può essere implementata utilizzando le stesse strutture di dati senza alcuna modifica.

L'idea di base della strategia per il calcolo di un nuovo percorso, chiamata "Energy Aware Routing" (EAR), è che i collegamenti IP effettivamente adoperati per indirizzare il traffico possono essere ridotti se sono usati un sottoinsieme di SPTs invece di sfruttare tutti gli SPT come nel caso classico di OSPF; in questo modo si possono mettere in 'sleep mode' le interfacce IP dei collegamenti IP non più utilizzati.

Con riferimento ai collegamenti ottici, analogamente alla soluzione applicata nello standard IEEE802.3az, è possibile mettere in sleep mode il NIC e gli elementi transponder associati ai collegamenti IP inutilizzati; lo sleep mode può essere considerato uno stato di standby, dove un numero di elementi delle interfacce IP non sono accesi. Tuttavia, si necessita un risveglio periodico dell'intera interfaccia per mantenere un'adiacenza dei protocolli di routing IP, per esempio lo scambio di Pacchetti Hello OSPF; in questo modo, il protocollo di routing mantiene la stessa topologia logica, non svolgendo una fase d'intercambio dei pacchetti per la convergenza e il controllo, e una fase di ricalcolo del percorso non eseguita nello strato WDM.

Bisogna osservare che, assumendo che tutti i router IP hanno una visuale consistente della rete IP nei loro database di LSA, ogni router è capace di calcolare lo SPT associato l'uno all'altro. Normalmente, quando l' OSPF classico è in funzione, un router calcola solo il proprio SPT attraverso l'algoritmo di Dijkstra considerando se stesso come il nodo radice; diversamente, nell'operazione EAR sono basate sul concetto di "esportazione SPT": logicamente, un set di router, chiamati esportatori, forza l'utilizzo dei loro SPTs verso un altro set di router, chiamati importatori. In sostanza, un router appartenente al set degli importatori identifica il proprio

esportatore e calcola lo SPT associato ad esso assumendo l'esportatore stesso come il nodo radice; in seguito, avendo lo SPT dell'esportatore come riferimento, il router importatore determinerà il proprio albero di percorso, chiamato "Modified Path Tree" (MPT), il quale utilizza gli stessi collegamenti IP dello SPT dell'esportatore.

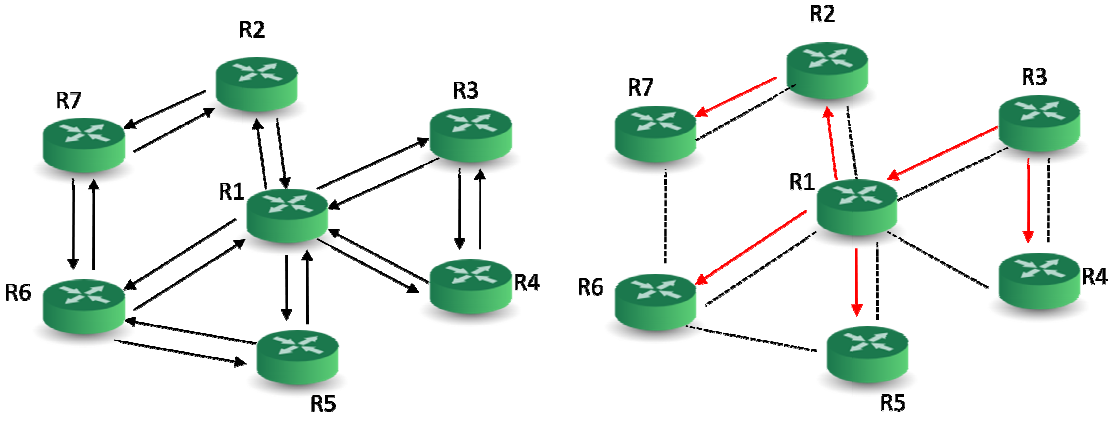
Si sposta l'attenzione dai router ai collegamenti e il ruolo del router è assegnato secondo il set dei collegamenti, chiamati *target links*, i quali sono candidati a essere spenti. Questo nuovo approccio dà la possibilità all'operatore di controllare la performance della rete e permette che sia implementata una strategia che degradi leggermente la QoS. Inoltre, nonostante la soluzione discussa qui è intrinsecamente centralizzata, sono evitati i messaggi di scambio OSPF tra i router.

Supponiamo che T è il set dei target links, cioè quelli candidati ad essere spenti, per un target link $l \in T$ deve essere identificata un'esportazione specifica, vale a dire una coppia di nodi importatori ed esportatori. Inequivocabilmente l'importatore corrisponde al router R_i essendo questo la sorgente del link l ; questo è abbastanza ovvio poiché, se il target link l deve essere messo in uno sleep mode, i percorsi della rete di R_i devono essere ricalcolati in modo tale che l non sia più usato da R_i . L'identificazione del router esportatore può essere più complessa. Un router candidato ad assumere il ruolo come esportatore del link l deve avere le due seguenti condizioni:

- il link l non deve appartenere al suo SPT
- deve far parte del set di vicino R_i.

La prima condizione è piuttosto ovvia mentre la seconda è introdotta per minimizzare l'incremento della lunghezza dei percorsi. Se più di un router soddisfa le condizioni precedenti, la scelta dipenderà sugli effetti prevedibili dell'esportazione. In seguito indichiamo con *M(l)* i set di esportatori che possono essere realizzati in modo tale da spegnere il collegamento *l*.

Come esempio nella Fig.3.1.a è mostrata una semplice topologia della rete composta di 7 router IP e 18 collegamenti diretti; per semplicità tutti i collegamenti IP hanno un peso unitario dell'OSPF. Assumiamo che il target link sia il collegamento (*R*3, *R*4). Il router *R*3 sarà l'importatore, mentre *R*1 potrebbe essere un plausibile router esportatore perché è il vicino del *R*3 e il collegamento (*R*3, *R*4) e non appartiene al suo SPT (vedi Fig.3.1.c). Il risultato dell'esportazione del MPT del *R*3 è mostrato nella Fig.3.1.d, il link (*R*3, *R*4) non è più utilizzato per instradare il traffico e può essere messo in uno stato di bassa consumazione di potenza.



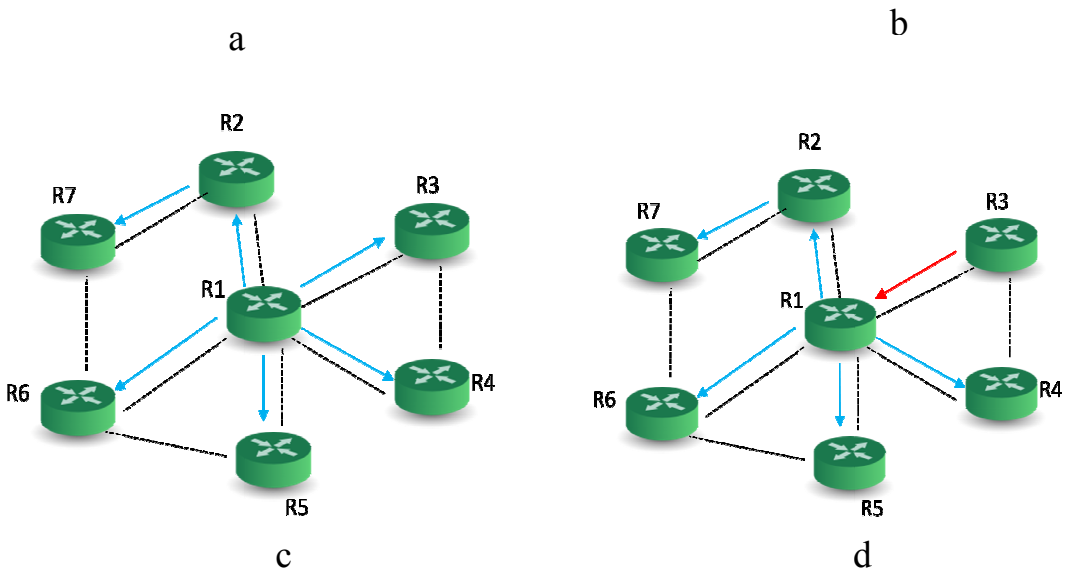


Fig.3.1 Esempio della strategia EAR con 7 router. a)topologia della rete OSPF,b)SPT di R3(importatore),c) SPTdi R1(esportatore) e d)MPT di R3.

È da considerare che, non è sempre possibile trovare un esportatore che permetta di mettere in sleep un target link, vale a dire che il set *M(l)* potrebbe essere vuoto. Questo è causato da due ragioni principali:

i) un vicino importatore non avendo il target link nel suo SPT non esiste oppure;

ii) una previa esportazione ha limitato le possibili esportazioni che possono essere considerate.

Per chiarire questo aspetto si richiede una descrizione più dettagliata del meccanismo dell'esportazione e, in particolar modo, devono essere investigate le relazioni tra tutte le possibili esportazioni. Questi aspetti hanno portato alla definizione del concetto di "mossa" il quale è discusso nella prossima sezione.

3.2.1 Il Concetto di Mossa e Le Sue Proprietà

Una rete IP può essere rappresentata per mezzo di un grafo pesato orientato *G(V,E)* dove *V* è il set di nodi che rappresentano i router IP ed *E* il set di archi orientati dei collegamenti IP.

Per ogni arco guidato $e \in E$, *S(e), E(e) e w(e)* indicano rispettivamente la fonte del nodo, la fine e il peso associato al collegamento. Diciamo che *N* e *L* sono rispettivamente le cardinalità di *V* ($N = \| V \|$) ed *E* ($L = \| E \|$). Il set di tutti gli shortest paths dal nodo v agli altri nodi, ossia lo SPT associato al nodo v , sono indicati come SPT(v).

Un nodo *x* è adiacente a un nodo *i* se esiste un collegamento diretto *e* da *i* a *x* , in altre parole *S*(e) = *i* ed *E*(e) = *x* . Un'esportazione, avendo l'effetto di mettere in modo "sleep" un collegamento target *l* uscendo da *i* , può essere eseguito avendo *i* come importatore e *x* come esportatore, così il nodo *i* può essere capace di calcolare un MPT avendo come riferimento SPT(x). Ne consegue che il collegamento *l* identifica il target link ed il collegamento e rappresenta l'esportazione capace di spegnere il collegamento l.

Vi è da notare che ci potrebbero essere diversi modi per mettere in modalità sleep il collegamento *l*; poniamo M(l) come set delle esportazioni associate al target link *l*. osserviamo che la scelta di una di queste alternative dirige verso lo stato specifico di una rete e determina diverse conseguenze sul risparmio dell'energia e la performance della rete. Per questa ragione, d'ora in poi indicheremo un'esportazione generica $m \in M(l)$ con il termine *mossa*.

Come ogni rete diretta identifica un mossa avendo il nodo sorgente come importatore e il nodo di destinazione come esportatore, si deduce immediatamente la seguente proposizione:

Il numero massimo di mosse che possono essere eseguiti in una rete descritti da un grafico *G(V,E)* è uguale a *L,* il numero totale di archi del grafo.

Investighiamo la relazione fra le mosse; in particolare, lo scopo è di definire semplici regole per stabilire se due mosse sono compatibili, cioè se loro possono essere eseguiti uno dopo l'altro o se sono mutuamente esclusivi, vale a dire se l'esecuzione di uno esclude l'applicazione dell'altro. Per fare ciò dobbiamo focalizzarci sugli effetti di una mossa il quale può essere classificato in due categorie: diretto e collaterale.

L'effetto diretto (o positivo) di un mossa consiste nella possibilità di mettere uno o più interfacce di un router in sleep così non può più essere usato per indirizzare il traffico di pacchetti. Invece, l'effetto collaterale (negativo) consiste nel bloccare l'altra/e mossa(e): quando un mossa *m* viene eseguita alcune delle altre mosse rimanenti, non potevano più essere eseguite.

Questo concetto può essere riassunto introducendo la seguente definizione di *compatibilità* tra coppie di mosse:

Condizione di compatibilità: data una coppia di mosse m_1 e m_2 , la mossa m_2 viene detto *compatibile* con la mossa m_1, e utilizziamo la notazione $m_2 \alpha m_1$, se m_2 può comunque essere eseguito dopo l'esecuzione di m_1; altrimenti la mossa m_2 viene detto *incompatibile* con la mossa m_1.

C'è da notare che la compatibilità è simmetrica, vale a dire se $m_2 \alpha m_1$ allora $m_1 \alpha m_2$, cioè che le due mosse m_1 e m_2 possono essere esecutati in qualsiasi ordine.

Data la mossa *m* siamo interessati a dare a tutte le mosse *incompatibili* con *m* il set $M_{NC}(m)$. Come primo passo, aggiusteremo il set di *condizioni* che dovranno essere verificati in modo che due mosse sono compatibili; possono essere classificati in condizioni di *procedura* e di *performance.*

Le condizioni di procedura possono derivare dalla definizione dell'esportazione stessa; in particolare se, un'esportazione avendo *i* come nodo importatore e *x* come nodo esportatore, viene eseguita, le seguenti condizioni (C1-C3) si verificano:

C1) *i* non può essere importatore di una prossima esportazione;
C2) *i* non può essere esportatore di una prossima esportazione;
C3) *x* non può essere importatore di una prossima esportazione.

La prima e la seconda condizione indicano che un router importatore può eseguire soltanto una singola esportazione; la terza condizione significa che un router esportatore non potrà mai assumere il ruolo di importatore.

Le condizioni di performance rappresentano la costrizione che si stabiliscono per controllare gli effetti dei cambiamenti del percorso del router sulle performance della rete. In particolare, introduciamo la seguente regola:

- Regola 1: un SPT associato a un nodo esportatore non può essere modificato da un'esportazione, cioè gli alberi d'instradamento dei nodi esportatori devono rimanere i più corti.

Questa regola determina un duplice effetto: i) una gran parte dei percorsi di rete sono identici a quelli più corti e ii) i percorsi re-instradati sono abbastanza vicini, in numero di salti, a quelli più corti. Se la Regola 1 regge, è possibile provare che in una rete con costi dei collegamenti uguali , il massimo incremento di percorsi della rete in numero di salti è uguale a due.

L'applicazione della Regola 1 introduce altre 2 condizioni extra:

C4) un collegamento usato dal SPT di un esportatore non può essere messo in modalità sleep;

C5) un nodo con un percorso modificato da un esportazione non può essere un esportatore.

La quarta condizione vuol dire che, dopo un'esportazione, i collegamenti del nodo esportatore utilizzati da un SPT diventano *collegamenti che non sono spenti*: tutte le possibili esportazioni causando lo spegnimento di questi collegamenti non sono più esecutabili. La quinta condizione porta alla definizione di *nodi bloccati*: dopo un'esportazione, tutti i nodi che hanno almeno un percorso che gli è stato cambiato non possono essere esportatori per prossime esportazioni. Se le condizioni C1-C5 reggono, è possibile dimostrare che l'instradamento del percorso della rete è libera da loop. Questa procedura può essere localmente eseguita dagli importatori senza il bisogno dello scambio dei messaggi del OSPF.

In sintesi, un mossa m è descritta completamente da un rapporto composto da tre campi e cioè:

$$m = \{(i, x) E_{sleep}(m), M_{NC}(m)\} \qquad (1)$$

I tre campi hanno le seguenti definizioni:

- (i , x) identifica la coppia di importatori (i) e nodi esportatori (x)
- E_{sleep} (m) è il set di collegamenti che la mossa m permette di mettere in modalità sleep
- M_{NC}(m) è il set di mosse incompatibili con m.

Una volta definito il concetto di mossa e le condizioni di compatibilità tra due mosse, il problema del risparmio energetico in una rete IP può essere formulato in maniera equivalente come il problema di trovare un set di mosse compatibili capaci di minimizzare il consumo della rete, garantendo un buon livello di QoS.

Nella prossima sezione ci focalizzeremo nello specifico problema del trovare un set di mosse compatibili garantendo il massimo risparmio energetico; in altre parole, cercheremo di dare una risposta alla seguente domanda: *data una topologia di rete, qual'è il set di mosse compatibili che permettono di mettere in modalità di bassa energia il massimo numero di reti?*. Questo verrà chiamato problema EAR.

3.2.2 La soluzione del problema EAR

Consideriamo una rete IP modellata da un grafo diretto *G(V,E)*. Definiamo un grafo indiretto $H = (M, C)$, dove ogni nodo $m \in M$ rappresenta un mossa m e gli archi $c \in C$ indicano la compatibilità di relazione tra le mosse: se esiste un arco c tra i nodi m_1 e m_2 , allora le mosse m_1 e m_2 sono compatibili. Inoltre, ogni nodo m è caratterizzato da un peso $w(m)$ il quale rappresenta il numero di collegamenti IP che la mossa m permette di mettere in sleep. In $H = (M,C)$, un set di mosse compatibili è rappresentato da un clique $K_H(n)$ dove n è il numero di nodi del clique. Una clique è un completo sub grafo di H nel quale ogni coppia di nodi è adiacente.

Ogni clique $K_H(n)$ di H è caratterizzato dal suo peso $W(K_H)$, dato dalla somma dei pesi di tutti i nodi del clique. In questo caso, dato che due mosse compatibili non possono spegnere gli stessi collegamenti IP, $W(K_H)$ è il numero di collegamenti IP che il set di mosse compatibili rappresentati da $K_H(n)$ permette di mettere in modalità sleep. Perciò il problema EAR equivale a trovare il completo sub grafo di H caratterizzato da un peso massimo. Una versione semplificata di questo problema, ottenuta se ogni nodo ha lo stesso peso, è conosciuta nella letteratura come il massimo problema del clique di un grafo, ed è *NP-hard*; come conseguenza anche il nostro problema è *NP-hard.*

In seguito, si propongono due diverse euristiche per risolvere il problema EAR: la prima, l'euristica di Max_Compatibility, che cerca di trovare il miglior set di mosse compatibili; la seconda, l'euristica del Min_Used_Links, è basata sulla

strategia della rilevazione dei collegamenti target e cerca di mettere in modalità sleep un particolare sottoinsieme di collegamenti di rete considerando una particolare proprietà dei collegamenti e non la compatibilità delle mosse.

3.2.2.a L'euristica della Max_Compatibility

Per spiegare l'euristica della Max_Compatibility, introduciamo delle ulteriori notazioni. Ogni mossa mi è caratterizzato da un vettore di compatibilità ci raffigurando la compatibilità delle relazioni fra mi e tutti gli altri mossa:

$$c_i = \{c_{ij}, 1 \leq j \leq L\},\ with\ cij = \begin{cases} 0 & m_i \propto m_j \\ 1 & m_i \propto m_j \\ 0 & i = j \end{cases} \qquad (2)$$

Il grado di compatibilità *gi* della mossa *mi* rappresenta il numero di mosse compatibili con *mi*:

$$g_i = \sum_{j=1}^{L} c_{ij} \qquad (3)$$

Data una topologia di rete G(V,E) , la relazione fra tutti i mossa possibili può essere rappresentato per mezzo della matrice di compatibilità C:

$$C = \{c_{ij}, 1 \leq i \leq L \quad 1 \leq j \leq L\}. \qquad (4)$$

Una soluzione possibile Sk , che permette di spegnere un set di interfacce IP, è rappresentato un set di mosse compatibili:

$$Sk = \{m_k, \quad k \in K\},\ c_{ij} = 1\ \forall i,\ j \in K \quad \text{con} \quad i \neq j. \qquad (5)$$

Definiamo una funzione utility $U(S_k)$ per paragonare le diverse soluzioni:

$$U(S_k) = \sum_{k \in K} w(m_k), \quad (6)$$

dove $w(m_k)$ è il numero di interfacce IP che la mossa m_k mette in modalità sleep. In questo modo ogni soluzione è caratterizzata dalla sua abilità di risparmiare energia. Ovviamente, siamo interessati in rilevare la migliore soluzione S_k^* , il quale massimizza $U(S_k)$:

$$U(S_k^*) = max\, U(S_k) \quad \forall k. \quad (7)$$

Si osserva che se si considera il problema massimo del clique, vale a dire la versione semplificata del problema EAR, caratterizzata da pesi constanti, la soluzione S_k^+ corrisponde al set di mosse con massima cardinalità. Perciò

$$\|S_k^+\| = max\, \|S_k\| \, \forall k. \quad (8)$$

L'euristica della Max_Compatibility, descritta nel pseudo codice Algoritmo 1, è stato definito considerando entrambi i problemi precedenti. Nella prima parte l'euristica cerca di risolvere il problema massimo del clique rilevando una serie di possibili soluzioni S_k caratterizzato dal massimo numero di mossa. Nella seconda parte, la migliore soluzione, nei termini di risparmio energetico, è scelta dalla valutazione della funzione $U(S_k)$.

In particolare, l'euristica sceglie immediatamente la mossa di massima compatibilità m_M, quello con il più alto grado di compatibilità g_M (Linea 1). In questo modo si forza la soluzione a contenere il mossa m_M. Il passo successivo (Linee 3-8) definisce g_M che candida soluzioni S_j che saranno valutati durante l'esecuzione

dell'algoritmo; ogni set S_j è inizialmente composto da il mossa m_M e un mossa compatibile m_j . Inoltre, due strutture extra di dati sono immessi: il set M_j il quale contiene i mossa che potrebbero essere inseriti in S_j nelle fasi successive, che è composta da tutti i mossa compatibili con sia m_M che con m_j; il vettore c_{Sj} rappresenta il vettore della compatibilità dei mossa m_M e m_j allo stesso tempo. Nelle linee 9-16 sono eseguiti i set S_j ; per ogni soluzione S_j viene aggiunta la mossa in M_j perché ha la più alta compatibilità con esso, finché non rimangono più mosse compatibili residui (cioè $M_j = \{\emptyset\}$); il set M_j e il vettore di compatibilità c_{Sj} vengono ricalcolati ogni volta un mossa nuovo viene introdotto in S_j.

Infine viene scelta (Linea 17) la migliore soluzione S_j* in termini di risparmio energetico.

Algorithm 1 Max_Compatibility Heuristic

1: **Find** $m_M \; s.t. \; g_M = max_{j \in L} \; g_j$

2: $j = 1$

3:**for all** $m_k \;\; t.c. \;\; c_{M_k} = 1$ **do**

4: $S_j = \{ m_M, \; m_k \}$

5: $j = j + 1$

6: $M_j = \{ \; m_i \;\; t.c. c_{iM} = 1 \; AND \;\; c_{ik} = 1 \; \}$

7: $c_{S_j} = c_M \;\; AND \;\; c_k$

8:**end for**

9:**for** $j = 1 \; to \; g_M$ **do**

10: **while** $M_j \neq \{\emptyset\}$ **do**

11: $m_l = \max m_k \in M_j (c_{S_j} \; AND \; c_k)$

12: $S_j = S_j \cup \{m_l\}$

13: $M_j = M_j - \{m_l\} - \{ \; m_i \; \in M_j \; s.t. \; c_{il} = 0\}$

14: $c_{S_j} = c_{S_j}\ AND\ c_l$

15: **end while**

16:**end for**

17:$S_j^* = max_{1 \le j \le g_i} U(S_j)$

3.2.2.b Euristica della Min_Used_Links

L'idea dietro questa euristica è basata sulla considerazione che, se un collegamento IP appare in alcuni SPT, il suo arresto determinerà alcune deviazioni del percorso della rete e di conseguenza l'impatto sulle prestazioni della rete dovrebbe essere limitata. L'euristica della Min_Used_Links ordina i collegamenti sulla base del numero dei percorsi che li attraversano e i collegamenti meno usati sono messi in modalità sleep.

L'euristica della Min_Used_Links è basata sulla conoscenza degli SPT associati a ogni router; infatti, per determinare l'utilizzo del collegamento, l'euristica associa a ognuno di esso l_i un valore n_i che rappresenta il numero di router che hanno il collegamento l_i nei loro SPT.

Dopo il calcolo dello SPT, i collegamenti l_i sono classificati secondo il valore di n_i. Ad ogni passo il collegamento che ha il valore più basso n_i viene estratto dalla lista e viene cercata una mossa fattibile e il collegamento relativo viene spento. Dopodiché, i nuovi SPT vengono calcolati e la lista dei collegamenti viene aggiornata.

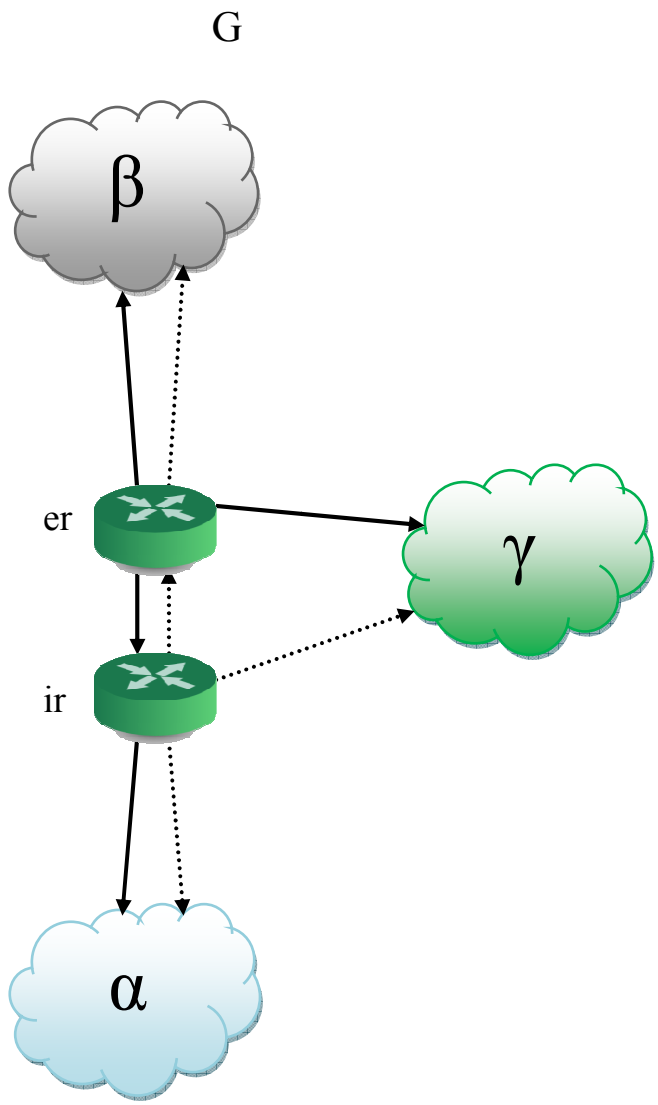
Le procedure terminano quando, considerando le condizioni di compatibilità definite nella Sezione IV, non ci sono più esportazioni fattibili da eseguire.

3.3 Modifica dei costi per l'implementazione della strategia EAR

Per poter realizzare un'esportazione a livello implementativo, cioè, modificare il SPT del nodo importatore in modo tale da utilizzare alcuni percorsi del SPT dell'esportatore, sono state proposte due soluzioni:

- ✓ La prima consiste nel calcolare il SPT dell'esportatore nell'importatore e poi invertire il percorso che va dall'esportatore all'importatore. Questo si potrebbe fare visto che tutti i nodi della rete hanno una mappa completa della topologia, dunque prendendo l'esportatore come radice si calcola l'albero utilizzando Dijkstra. Il problema si presentava al momento di cambiare la direzione del link esportatore-importatore, prima di tutto dovuto a le interfacce associate al nodo esportatore le quali sono ovviamente diverse a quelle del nodo importatore. Per il modo nel quale viene eseguito Dijkstra si è arrivata a questa soluzione molto complicata;
- ✓ La seconda si basa nel "giocare" con i pesi degli archi della topologia, per dare priorità ad un percorso desiderato. Ricordiamo che una rete può essere rappresenta con un grafo G, in cui i nodi saranno i vertex e i collegamenti sono gli archi. Ai collegamenti è associato un costo che dipende dell'interfaccia utilizzata. Si può dimostrare che modificando opportunamente i pesi del grafo, il risultato dello SPT sarà quello necessario per la strategia EAR.

3.3.1 Dimostrazione della modifica dei costi per l'implementazione della strategia EAR



G = G(V,E)

G è un grafo orientato, semplice e connesso.

α, β, γ: Insiemi di vertex.

α: Solo si può arrivare attraversando IR. Vertex di partenza non appartiene a questo insieme.

β: Solo si può arrivare attraversando ER. Vertex di partenza non appartiene a questo insieme.

γ: Si arriva attraversando ER se si parte da β, o attraversando IR se si parte da α. Vertex di partenza non appartiene all'insieme γ.

SPTer ————

SPTir ·····

SPa(b): Il percorso più conveniente in termini di costi per andare dal vertex a al vertex b, eseguendo il calcolo senza modifiche.

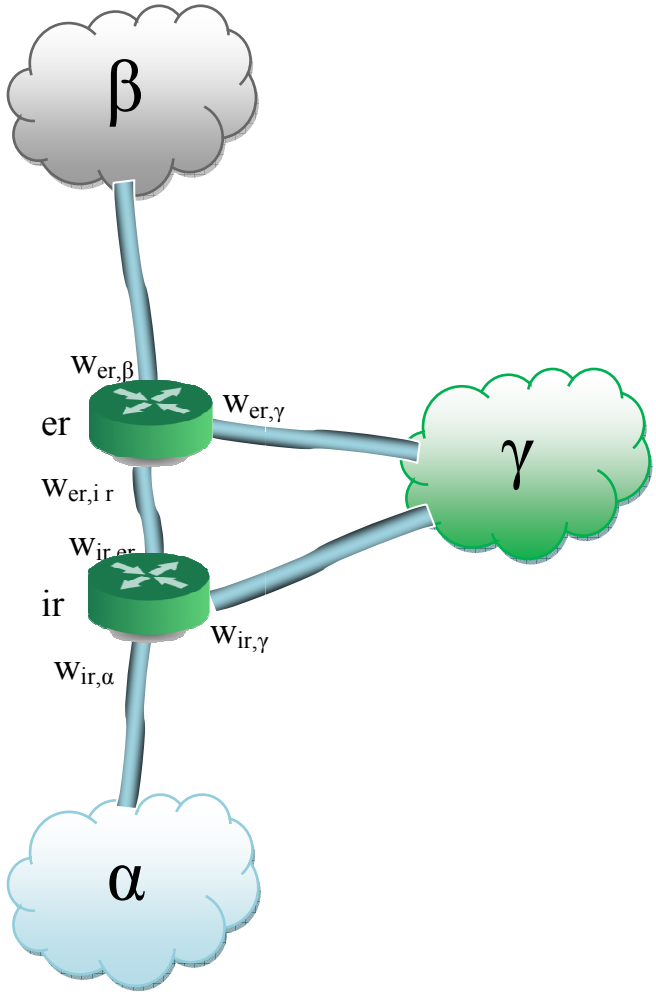
SPm,a(b): Il percorso più conveniente in termini di costi per andare dal vertex a al vertex b, eseguendo il calcolo modificando i costi degli archi.

C(): Somma totale dei costi di tutti gli archi utilizzati di un percorso.

ir: router importatore

er: router esportatore

Collegamenti a livello fisico



wa,b:costo di utilizzare la interfaccia di a che si connette con b.

w(dζ)= costo per andare dal nodo di bordo dell'insieme ζ alla destinazione d che si trova all'interno di ζ. Nel caso la destinazione sia il nodo di bordo il costo sarà zero.

$$d \in \alpha \begin{cases} C\big(SPir(d)\big) \;=\; w_{ir,\alpha} + w(d_\alpha) & (2.1) \\ C\big(SPer(d)\big) = C\big(SPir(d)\big) + C\big(SPer(ir)\big) = \; w_{er,ir} + w_{ir,\alpha} + w(d_\alpha) & (2.2) \end{cases}$$

$$d \in \gamma \quad \begin{cases} C\big(SPer(d)\big) \le C\big(SPer(ir)\big) + C\big(SPir(d)\big) & (3.1) \\ C\big(SPir(d)\big) \le C\big(SPir(er)\big) + C\big(SPer(d)\big) & (3.2) \end{cases}$$

$$d \in \gamma \quad \begin{cases} w_{er,\gamma} + w(d_\gamma) \le w_{er,ir} + w_{ir,\gamma} + \; w(d_\gamma) & (3.1) \\ w_{ir,\gamma} + w(d_\gamma) \le w_{ir,er} + w_{er,\gamma} + w(d_\gamma) & (3.2) \end{cases}$$

$$d \in \gamma \quad \begin{cases} w_{er,\gamma} \le w_{er,ir} + w_{ir,\gamma} & (3.1) \\ w_{ir,\gamma} \le w_{ir,er} + w_{er,\gamma} & (3.2) \end{cases}$$

La (3.1) vuole dire che è più conveniente andare alla destinazione d attraverso il suo collegamento diretto, invece di andare al vertex ir e poi partendo da lì, arrivare a d. La (3.2) indica il caso contrario.

Modificando i costi degli archi associati al vertex ir, si può dimostrare che

$$Cm\big(SPir(d)\big) = \; C\big(SPer(d)\big) \qquad (4)$$

Due cambi devono essere fatti:

i. $C(SPir(er)) = w_{ir,er} = 0$

Poi, si somma $w_{er,ir}$ *a tutti gli archi assocciati a ir tranne quello che si collega con er.*

ii.1. $w^*_{ir,\gamma} = w_{ir,\gamma} + w_{er,ir}$

ii.2. $w^*_{ir,\alpha} = w_{ir,\alpha} + w_{er,ir}$

Rispetto al primo cambio, la (1.1) e (1.2) risultano

$d \in \beta$

$$\begin{cases} C\big(SPer(d)\big) \;= w_{er,\beta} + w(d_\beta) & (4.1) \\ Cm\big(SPir(d)\big) = C\big(SPer(d)\big) + C\big(SPir(er)\big) = w_{ir,er} + w_{er,\beta} + w(d_\beta) & (4.2) \end{cases}$$

$$d \in \beta \quad \begin{cases} C\big(SPer(d)\big) = w_{er,\beta} + w(d_\beta) & (4.1) \\ Cm\big(SPir(d)\big) = w_{er,\beta} + w(d_\beta) & (4.2) \end{cases}$$

Rispetto al cambio ii.2, la (2.1) e (2.2)

$$d \in \alpha \begin{cases} Cm\big(SPir(d)\big) = w^*_{ir,\alpha} + w(d_\alpha) & (5.1) \\ C\big(SPer(d)\big) = C\big(SPir(d)\big) + C\big(SPer(ir)\big) = w_{er,ir} + w_{ir,\alpha} + w(d_\alpha) & (5.2) \end{cases}$$

$$d \in \alpha \quad \begin{cases} Cm\big(SPir(d)\big) = w_{ir,\alpha} + w_{er,ir} + w(d_\alpha) & (5.1) \\ C\big(SPer(d)\big) = w_{ir,\alpha} + w_{er,ir} + w(d_\alpha) & (5.2) \end{cases}$$

La (5.1) e la (5.2) sono uguali, si verifica (4).

Rispetto ai cambi i e i.1, la (3.1) e (3.2) risultano

$$d \in \gamma \quad \begin{cases} C\big(SPer(d)\big) \leq C\big(SPer(ir)\big) + Cm\big(SPir(d)\big) & (6.1) \\ Cm\big(SPir(d)\big) \nleq C\big(SPir(er)\big) + C\big(SPer(d)\big) & (6.2) \end{cases}$$

$$d \in \gamma \quad \{w_{er,\gamma} + w(d_\gamma) \leq w_{er,ir} + w^*_{ir,\gamma} + w(d_\gamma) \quad (6.1)$$

$$d \in \gamma \quad \{w_{er,\gamma} + w(d_\gamma) \leq 2w_{er,ir} + w_{ir,\gamma} + w(d_\gamma) \quad (6.1)$$

La (3.1) viene rinforzata con la modifica 2), quindi per er non cambia niente e come prima della modifica, il percorso più conveniente lo trova NON attraversando ir. Però il nostro interesse riguarda gli SP dell'importatore, pertanto si vede nel dettaglio cosa succede con i due membri della disuguaglianza (6.2).

$$d \in \gamma \qquad \begin{cases} Cm\big(SPir(d)\big) = w^{*}_{ir,\gamma} + w\big(d_\gamma\big) = w_{ir,\gamma} + w_{er,ir} + w\big(d_\gamma\big) \\ C\big(SPir(er)\big) + C\big(SPer(d)\big) = C\big(SPer(d)\big) = w_{er,\gamma} + w(d_\gamma) \end{cases}$$

Ricordando la (3.1)

$$d \in \gamma \ \{w_{er,\gamma} \leq w_{er,ir} + w_{ir,\gamma}$$

La (6.2) si può riscrivere nel seguente modo

$$d \in \gamma \qquad \{Cm\big(SPir(d)\big) \geq C\big(SPir(er)\big) + C\big(SPer(d)\big) \quad (6.2)$$

La (3.2) non si verifica più, dunque partendo da ir sarà più conveniente attraversare il vertex er tranne il caso di =, nel quale qualsiasi scelta può soddisfare il cammino più corto. Dunque per ir, i nodi del insieme γ passeranno a far parte dell'insieme β, cioè che per arrivare si deve passare per er.

CAPITOLO IV

IMPLEMENTAZIONE IN QUAGGA

Gli obbiettivi principali del lavoro svolto sono stati due:

- ✓ Mediante la modifica del codice di quagga, fare l'importazione di un percorso da un nodo predeterminato che chiamiamo esportatore, che in pratica corrisponde a cambiare i costi delle distanze del grafo di una topologia, in modo tale di dare priorità a un percorso desiderato.
- ✓ Creare una funzione, modificando il codice di Quagga, capace di abilitare la strategia EAR nello spazio di configurazione del terminale, cioè, tramite righe di comando.

Il calcolo dello shortest path viene fatto nella funzione *ospf_spf_calculate()*, questa funzione chiama altre funzione d'inizializzazione come quelle per creare le tabelle d'instradamento associate all'area nella quale si sta eseguendo il calcolo dello SPF, la lista dei candidati, il nodo radice che per il caso trattato in questa tesi è il nodo importatore, ecc.

Dopo uno studio delle diverse funzioni che vengono impiegate nella procedura, è stato scoperto che la funzione più adatta a essere utilizzata per cambiare i costi è la funzione *ospf_spf_next()* per due caratteristiche principali:

- Questa funzione viene chiamata una volta sola per ogni nodo nella topologia,

aspetto rilevante, perché vogliamo cambiare solo una volta i costi per andare al nodo successivo.

- L'LSA del nodo in questione viene analizzato in questa funzione e visto che questa "copia" l'informazione del LSA nella struttura *vertex,* saremo in grado di modificarla.

Prima di passare alla modifica del codice è importante fare un'analisi esaustiva della funzione ospf_spf_next(), dopodiché saremo in grado di comprendere gli aspetti significativi che riguardano ad essa.

4.1 Ospf_spf_next()

Come già accennato questa importante funzione si occupa di ispezionare i link uscenti da un vertice e di aggiornare la lista dei Candidates. All'interno di questa funzione è eseguita la gestione della lista dei Candidates. Cio si fa prendendo sempre come guida lo standard di OSPF version 2 specificato nel RFC2328.

Il funzionamento logico di questa funzione è presentato in figura 4.1

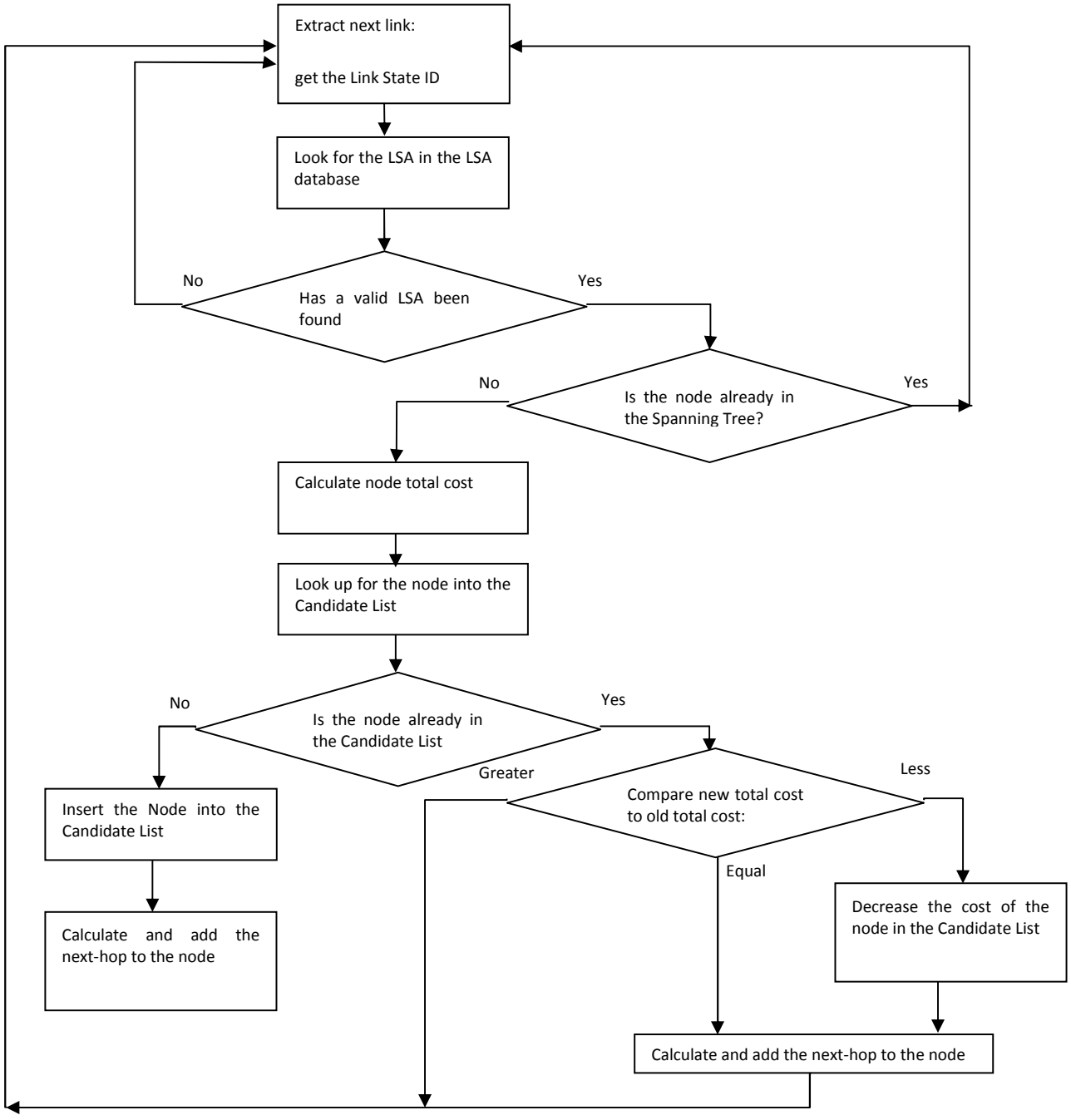


Figura 4.1 Schema di funzionamento di Dijkstra in ospfd

Vengono esaminati uno dopo l'altro tutti i link del vertice appena estratto dai candidates. Per ogni link viene ricavata la LSA corrispondente al nodo all'altra estremità del link nel Database delle LSA, e viene controllato lo stato di quest'ultimo vertice. In base allo stato trovato, vengono eseguite operazioni diverse, che verranno descritte insieme al codice nel seguito.

```
static void
ospf_spf_next (struct vertex *v, struct ospf_area *area,
                                    struct pqueue * candidate)
{
  struct ospf_lsa *w_lsa = NULL;
  u_char *p;
  u_char *lim;
  struct router_lsa_link *l = NULL;
  struct in_addr *r;
  int type = 0;
```

Sono dichiarate e inizializzate le variabili e i puntatori che saranno utili per il buon esito della funzione. Dopo di che si cominciano ad esaminare i link del vertice di partenze, quello al quale sta puntando la struttura vertex *v*. Per fare questo viene scandagliata la LSA associata al vertice, e vengono letti uno dopo l'altro i Link State ID del vertice all'altro capo del link. L'analisi è differenziata nel caso che il vertice di partenza sia un Router o una Network, perché la struttura della LSA cambia. Per il caso di studio, si ritengono d'importanza solo quei vertex di tipo ROUTER che sono connessi a vertex di tipo sia Router che Network.

```
p = ((u_char *) v->lsa) + OSPF_LSA_HEADER_SIZE + 4;
  lim = ((u_char *) v->lsa) + ntohs (v->lsa->length);

  while (p < lim)
```

```
{
  struct vertex *w;
  unsigned int distance;

  /* In case of V is Router-LSA. */
  if (v->lsa->type == OSPF_ROUTER_LSA)
    {
      l = (struct router_lsa_link *) p;

      p += (ROUTER_LSA_MIN_SIZE +
            (l->m[0].tos_count * ROUTER_LSA_TOS_SIZE));
```

A questo punto è stato estratto un link. Se ne controlla il tipo.

```
/* (b) Otherwise, W is a transit vertex (router or transit
       network).  Look up the vertex W's LSA (router-LSA or
       network-LSA) in Area A's link state database. */
      switch (type)
        {
        case LSA_LINK_TYPE_POINTOPOINT:
        case LSA_LINK_TYPE_TRANSIT:
          if (IS_DEBUG_OSPF_EVENT)
            zlog_debug ("Looking up Network LSA, ID: %s",
                        inet_ntoa (l->link_id));
          w_lsa = ospf_lsa_lookup_by_id (area, OSPF_NETWORK_LSA,
                                         l->link_id);
          if (w_lsa)
            if (IS_DEBUG_OSPF_EVENT)
              zlog_debug ("found the LSA");
          break;
```

```
    default:
      zlog_warn ("Invalid LSA link type %d", type);
      continue;
    }
  }
```

Nella LSA ad ogni link è associato il Link State ID del vertice adiacente. La funzione *ospf_lsa_lookup_by_id ()* si occupa di effettuare la ricerca nel database delle LSA, sulla base del Link State ID considerate. La variabile *w_lsa* quindi memorizza la LSA del Router o della Network all'altro capo del link.

```
else
  {
    /* In case of V is Network-LSA. */
    r = (struct in_addr *) p;
    p += sizeof (struct in_addr);

    /* Lookup the vertex W's LSA. */
    w_lsa = ospf_lsa_lookup_by_id (area, OSPF_ROUTER_LSA, *r);
    if (w_lsa)
      {
        if (IS_DEBUG_OSPF_EVENT)
          zlog_debug ("found Router LSA %s", inet_ntoa (w_lsa->data->id));
      }
  }
```

Nel caso else, il vertex di partenza è del tipo Network, i cui link andranno di conseguenza verso dei Router.

Una volta trovata la LSA del vertice adiacente, si procede a controllarne lo stato. Ci sono quattro casi in cui si scarta il link e passa al prossimo.

1. LSA non esiste
2. L'età del LSA è uguale a MaxAge
3. LSA non ha un link di ritorno a v
4. Il vertice associato alla LSA è già stato inserito nel albero.

```
    /* (b cont.) If the LSA does not exist, or its LS age is equal
  to MaxAge, or it does not have a link back to vertex V,
  examine the next link in V's LSA.[23] */
if (w_lsa == NULL)
 {
  if (IS_DEBUG_OSPF_EVENT)
    zlog_debug ("No LSA found");
  continue;
 }

if (IS_LSA_MAXAGE (w_lsa))
 {
  if (IS_DEBUG_OSPF_EVENT)
    zlog_debug ("LSA is MaxAge");
  continue;
 }

if (ospf_lsa_has_link (w_lsa->data, v->lsa) < 0 )
 {
  if (IS_DEBUG_OSPF_EVENT)
    zlog_debug ("The LSA doesn't have a link back");
  continue;
 }
```

```
/* (c) If vertex W is already on the shortest-path tree, examine
   the next link in the LSA. */
if (w_lsa->stat == LSA_SPF_IN_SPFTREE)
                        {
                          if (IS_DEBUG_OSPF_EVENT)
                            zlog_debug ("The LSA is already in SPF");
                          continue;
                        }
```

Successivamente si prosegue al calcolo del costo totale per raggiungere il nuovo vertex, sommando il costo di partenza a quello del link considerato.

```
/* (d) Calculate the link state cost D of the resulting path
       from the root to vertex W.  D is equal to the sum of the link
       state cost of the (already calculated) shortest path to
       vertex V and the advertised cost of the link between vertices
       V and W.  If D is: */

    /* calculate link cost D. */
    if (v->lsa->type == OSPF_ROUTER_LSA)
                        distance = v->distance + ntohs (l->m[0].metric);
    else /* v is not a Router-LSA */
                        distance = v->distance;
```

Il prossimo passo consiste nel controllare se l'LSA non è stato esplorato, in questo caso si crea un nuovo vertex con l'informazione dell'LSA e si esegue la ricerca di questo vertex nella lista dei Candidates, se non è presente si setta il next-hop e lo si inserisce nella coda dei Candidates con *pqueue_enqueue()*.

```
/* Is there already vertex W in candidate list? */
if (w_lsa->stat == LSA_SPF_NOT_EXPLORED)
                        {
  /* prepare vertex W. */
  w = ospf_vertex_new (w_lsa);

  /* Calculate nexthop to W. */
  if (ospf_nexthop_calculation (area, v, w, l, distance))
   pqueue_enqueue (w, candidate);
  else if (IS_DEBUG_OSPF_EVENT)
   zlog_debug ("Nexthop Calc failed");
                        }
```

Nel caso si presenti nella lista Candidates si usa un'altra procedura, nello specifico consiste nel confrontare la nuova e la vecchia metrica per raggiungere il vertice. Se la nuova metrica è maggiore di quella già presente, il link può essere scartato. Se le due metriche sono esattamente uguali, si aggiunge un nuovo next-hop a quelli già trovati per quel vertice. Altrimenti, la nuova metrica calcolata è minore di quella presente nei Candidates, questa ultima va decrementata al nuovo valore.

```
else if (w_lsa->stat >= 0)
                        {
                          /* Get the vertex from candidates. */
                          w = candidate->array[w_lsa->stat];

                          /* if D is greater than. */
                          if (w->distance < distance)
    {
```

```
          continue;
        }
      /* equal to. */
                        else if (w->distance == distance)
        {
                          /* Found an equal-cost path to W.
           * Calculate nexthop of to W from V. */
          ospf_nexthop_calculation (area, v, w, l, distance);
        }
       /* less than. */
                        else
        {
          /* Found a lower-cost path to W.
           * nexthop_calculation is conditional, if it finds
           * valid nexthop it will call spf_add_parents, which
           * will flush the old parents
           */
          if (ospf_nexthop_calculation (area, v, w, l, distance))
            /* Decrease the key of the node in the heap.
             * trickle-sort it up towards root, just in case this
             * node should now be the new root due the cost change.
             * (next pqueu_{de,en}queue will fully re-heap the queue).
             */
            trickle_up (w_lsa->stat, candidate);
        }
      } /* end W is already on the candidate list */
    } /* end loop over the links in V's LSA */
}
```

4.2 Strategia EAR mediante la modifica dei costi

Questa prima fase della implementazione può essere suddivisa in tre sottofasi.

- Creazione delle strutture e funzioni per la nostra funzione *ospf_spf_next()*.
- Implementazione della modifica dei costi.
- Correzione degli errori.

4.2.1 Funzioni e strutture

È stata creata una struttura chiamata *energy_saving* che ha la seguente forma:

```
struct energy_saving {

  struct in_addr *ex_s_addr;
  int en_sa1;

};
```

Questa struttura serve per salvare le informazioni necessarie per avviare il calcolo dello SPF modificato. L'integer *en_sa1* funziona come un flag per sapere se la modalità en_sa è attiva. Il puntatore alla struttura *in_addr* serve per conoscere quale sarà il nodo utilizzato come esportatore e modificare il calcolo dello SPF dell'importatore.

struct energy_saving si trova dentro la struttura *ospf_area* in forma di puntatore:

```
struct ospf_area{
```

```
    ....
    ....
struct energy_saving *en_sa_area;
    ....
    ....
};
```

La ragione per la quale questa struttura è stata inserita qui è che dopo l'inizializzazione di ospf_area non viene modificata più e di conseguenza se la modalità energy saving è attiva e arriva un LSA update, quando si ricalcolerà l'albero con la nuova informazione il flag *en_sa1* continuerà ad essere attivo.

Oltre a questa sono state create altre 3 funzioni:

- energy_saving_new (void);
- in_addr_new(void);
- find_link_adcost();

energy_saving_new alloca uno spazio per salvare gli elementi dentro la struttura *energy_saving*. Non ha nessun ingresso e ritorna un puntatore del tipo *struct energy_saving* allo spazio specificato oppure NULL se la richiesta non può' essere soddisfatta, e inizializza con i valori di default *en_sa* e *in_addr*.

```
/* allocate new energy_saving object */
struct energy_saving *energy_saving_new (){

struct energy_saving *new;

 new = (energy_saving *)malloc (sizeof (struct energy_saving));
 if (new == 0)
```

```
  zlog_debug ("virtual memory exhausted");
 new->en_sa1=ENERGY_SAVING_MODE_DEFAULT;
 new->ex_s_addr=in_addr_new();

 return new;
}
```

Il valore di default di *en_sa* è 0, definito in *ENERGY_SAVING_MODE_DEFAULT* nel file di tipo header *ospf_spf.h*. A sua volta questa funzione chiama *in_addr_new()*, che analogamente a *energy_saving_new()* alloca uno spazio della memoria per la *struct ex_s_addr*, che viene inizializzata con un valore qualunque tramite la funzione *inet_aton()*.

```
struct in_addr *in_addr_new(){
 struct in_addr *new;

 new = (in_addr *)malloc (sizeof (struct in_addr));
 if (new == 0)
  zlog_debug ("virtual memory exhausted");
 inet_aton();

 return new;
}
```

Le funzioni *energy_saving_new()* e *in_addr_new()* si trovano nel file *ospf_spf.c*. Successivamente è stato aggiunto il costrutto *typedef* per poter chiamare senza errori le funzioni *malloc*, questi nuovi elementi sono presenti nel file *ospf_spf.h* in questo modo:

typedef struct in_addr in_addr;
typedef struct energy_saving energy_saving;

In teoria si dovrebbero aggiungere ai typedef definiti nella cartella *lib* di quagga.

La funzione *energy_saving_new()* sarà chiamata dalla funzione *ospf_area_new()* che si incarica di allocare e inizializzare tutti gli elementi che fanno riferimento a un'area determinata, questa si troverà presso il file *ospfd.c*.

```
static struct ospf_area *
ospf_area_new (struct ospf *ospf, struct in_addr area_id)
{
  struct ospf_area *new;
  /* Allocate new config_network. */
  new = XCALLOC (MTYPE_OSPF_AREA, sizeof (struct ospf_area));

  new->ospf = ospf;
        ....
        ....
        ....

  new->en_sa_area=energy_saving_new ();
        ....
        ....
        ....
  return new;
}
```

Infine è stata creata la funzione *find_link_adcost()*, il motivo principale è stato che il costo importatore-esportatore deve essere disponibile e salvato in una variabile, dato che sarà necessario per il calcolo del costo di un nodo non esportatore.

Dobbiamo tener conto che non sappiamo a priori come sono organizzati i link states dentro ai messaggi LSA, perciò il costo imp-exp deve essere ricavato prima d'arrivare a questo punto dell'algoritmo altrimenti si potrebbe generare un errore.

Il calcolo del costo imp-exp mediante la funzione find_link_adcost è stato messo all'inizio di ospf_spf_next con un if(), dovuto a fatto che, questa funzione deve essere chiamata quando sono verificate due condizioni:

- ✓ la prima è che sia impostato a 1 il flag en_sa1 che indica che è attivo la modalità energy saving;
- ✓ La seconda è che il nodo v passato in ingresso è la radice, dato che l'importazione sarà fatta da questo vertex .

Per salvare ciò che ritorna è stata definita una variabile locale *unsigned int dist_imp_exp* dentro di *ospf_spf_next()*.

La funzione ha due ingressi:

- ✓ un puntatore a una struttura *vertex* che sarà utilizzato per trovare l'lsa a lei associata, che come già detto prima sarà il nodo radice;
- ✓ e l'altro ingresso è un puntatore a una struttura del tipo ospf_area, in cui gli elementi più importanti che si utilizzeranno sono la radice che ha come riferimento un puntatore del tipo *vertex* chiamato *spf* e il puntatore alla struttura *energy_saving* che è ritenuto importante dato che contiene l'indirizzo ip dell'esportatore.

La funzione *find_link_adcost()* in definitiva analizza riga per riga l'informazione di ciascuna interfaccia associata al nodo *v*, quando trova l'indirizzo ip dell'esportatore memorizza il costo di quella interfaccia e fa un break al ciclo *while* per non esaminare le interfacce mancante e ritorna il valore trovato.

Questa funzione è stata creata basandosi su una parte del codice di *ospf_spf_next()*, quindi gran parte di *find_link_adcost()* e uguale, la differenza è che questa funzione non modifica nessuna struttura o variabile, esamina solo le informazione contenute nei suoi ingressi e poi cerca un indirizzo ip determinato. La parte ritenuta rilevante di questa funzione viene mostrata sotto.

```
unsigned int find_link_adcost(struct vertex *v, struct ospf_area *area){

    ……
    .........
p = ((u_char *) v->lsa) + OSPF_LSA_HEADER_SIZE + 4;
  lim = ((u_char *) v->lsa) + ntohs (v->lsa->length);
  while (p < lim)
   {
     ……
     ……

  if(w_lsa->data->id.s_addr==area->en_sa_area->ex_s_addr->s_addr){
       /* calculate link cost D. */
        if (v->lsa->type == OSPF_ROUTER_LSA){
           ad_cost=ntohs (l->m[0].metric);
           break;}
        else {/* v is not a Router-LSA */
           ad_cost=0;       //questo caso non dovrebbe succedere mai
           break;
```

```
        }
      }
    } /* end loop over the links in V's LSA */
     return ad_cost;
}
```

Si potrebbe pensare ad utilizzare *find_link_adcost* con un solo ingresso visto che *spf* sta puntando alla radice e non utilizzare più l'ingresso *struct vertex *v*, però è stato previsto il caso si voglia modificare i percorsi calcolati da SPF facendo l'importazione di un nodo che non sia la radice, in cui sarà necessaria questa struttura, che per adesso è ridondante.

4.2.2 Implementazione della modifica dei costi

Visto che ormai è noto il costo tra i nodi importatore-esportatore, passiamo ad analizzare la funzione *ospf_spf_next()*. Come già accennato sopra, questa funzione analizza l'lsa update associato al nodo v in ingresso e legge una alla volta tutte le interfacce con i suoi rispettivi link, ciò viene fatto con un *while* finché non trova la fine del LSA update. Ulteriormente, ad ogni giro decide se prenderlo in considerazione per inserirlo nella lista dei Candidates.

La parte di questa funzione che viene modificata è quella dove si calcola il costo totale per arrivare al suddetto nodo. Il funzionamento logico viene mostrato nella figura 4.2

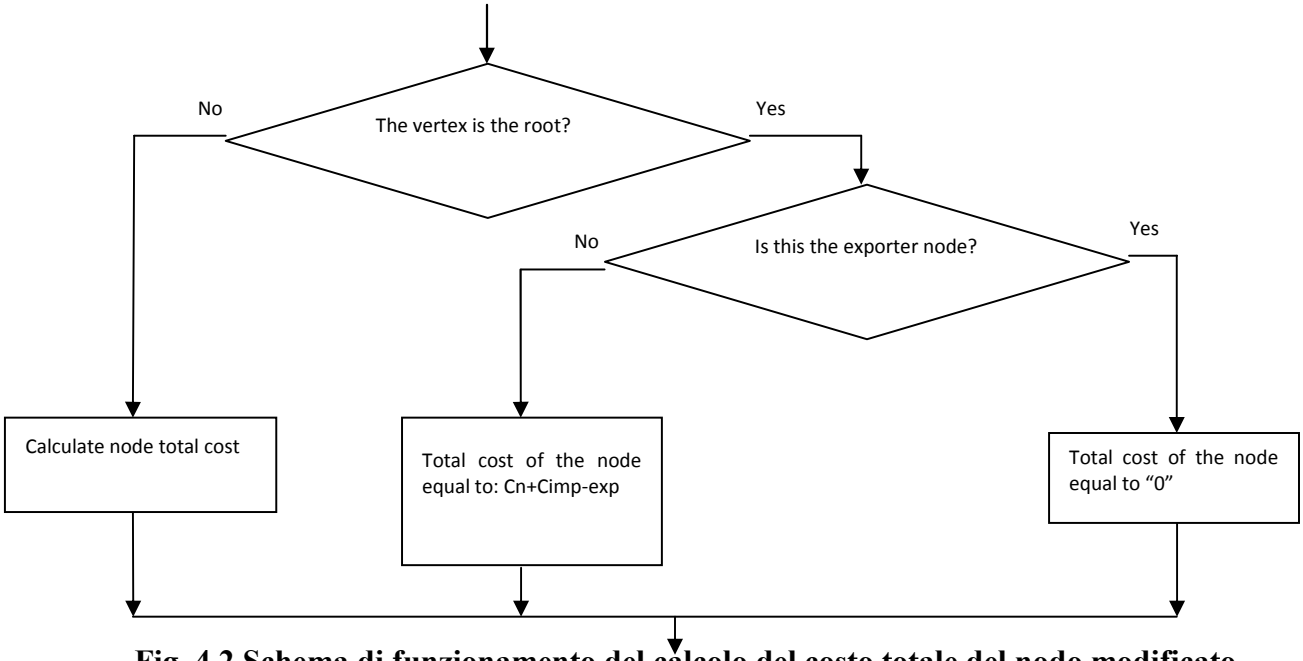


Fig. 4.2 Schema di funzionamento del calcolo del costo totale del nodo modificato

Il primo passo sarà quello di verificare il rispetto delle due condizioni:

- ✓ È attivo il modo energy_saving, la struttura en_sa_area deve avere il flag en_sa1 con il valore "1".
- ✓ Il vertex esaminato è la radice.

Se non si verificano entrambe queste condizioni, si calcola il costo totale del nodo in modo normale. Altrimenti si va al secondo passo della procedura e si controlla se il nodo all'altra estremità del link sia l'esportatore. Si possono avere due casi:

- La struttura in_addr contenuta nella struttura *en_sa*, che indica l'indirizzo dell'esportatore coincide con l'id del nodo all'altra estremità dell' interfaccia che si sta esaminando nel giro. Il costo totale del nodo esportatore viene impostato a zero.
- La struttura in_addr contenuta nella struttura *en_sa*, che indica l'indirizzo

dell'esportatore NON coincide con l'id del nodo all'altra estremità dell' interfaccia che si sta esaminando nel giro. Il costo totale del nodo non importatore viene impostato come la somma del costo per arrivarci più il costo per arrivare dal importatore al esportatore.

```
/*quì comincia la modifica */
    if(v==area->spf&&area->en_sa_area->en_sa1){

        if(w_lsa->data->id.s_addr==area->en_sa_area->ex_s_addr->s_addr){

          distance = v->distance+0;

              }
        else {
          distance = v->distance + ntohs (l->m[0].metric)+dist_imp_exp;
            }

      }
/*qui finisce */

      else{

        /* calculate link cost D. */
   if (v->lsa->type == OSPF_ROUTER_LSA)

    distance = v->distance + ntohs (l->m[0].metric);

   else /* v is not a Router-LSA */
    distance = v->distance;
```

}

4.2.3 Correzione degli errori

Uno degli errori causati per la modifica dei costi è stato impostare a 0 il valore della distanza del vertex scelto come esportatore. Questo succede dato che nel calcolo di Dijkstra in OSPF non è stato previsto che non occorra nessun cambio in un vertex qualsiasi dell'area tranne la radice.

Andando allo specifico nel file ospf_spf.c nella funzione ospf_spf_add_parent viene verificata questa condizione con il seguente macro:

assert (distance);

se "distance" ha come valore 0, il demone ospfd si blocca e genera un errore. Per farlo funzionare ho dovuto scrivere "//" così il compilatore salta questa riga e non si produce l'errore.

Un'altro modo più corretto per farlo potrebbe essere abilitare questo macro quando non si lavora con il modo energy saving.

4.3 Configurazione tramite righe di comando

VTY Virtual TeletYpe serve per creare in un router una sessione telnet che fornisce all'utente una comunicazione interattiva utilizzando una connessione al terminale virtuale per la configurazione del demone di routing. La sessione telnet è quella utilizzata per configurare tramite righe di comando un router.

Il file cambiato per aggiungere i nuovi comandi è ospf_vty.c che si trova dentro la cartella *ospfd*. In questo file si definiscono tutte le funzioni che possono

essere utilizzate per la configurazione di un router. Di particolare interesse sono le due Macro, *DEFUN* e *ALIAS*, il *DEFUN* serve per la definizione della funzione nella vty e *ALIAS* serve per definire un pseudonimo di una funzione già esistente. Il Macro è molto simile ad una funzione, viene chiamata e utilizza altre funzioni, di solito questi macro vengono definiti nei file ".h".

4.3.1 Macro DEFUN

Il Macro DEFUN è la base di tutti i comandi per la configurazione del router nella VTY, ragione per la quale bisogna fare una breve descrizione, specificando i punti che si useranno per la creazione dei nuovi comandi.

DEFUN(argomento1, argomento2, argomento3, argomento4\n argomento 5\n argomento N\n)

- Argomento1: nome della funzione.
- Argomento2: nome del comando nella vty.
- Argomento 3: stringa dove si inserisce come verrà chiamato il comando dal utente, per esempio "*cambio colore rosso*", ogni spazio differenzia un sottocomando dall'altro, e può essere visto se facciamo "list" dentro di "ospfd(config-router)#", che mostra tutti i comandi che possono essere chiamati.

  Dopo questo argomento, gli altri argomenti sono stringhe per il help.

- Argomento4: Utilizzato per la descrizione del primo sottocomando. Quando facciamo "?" dentro di "ospfd(config-router)#" ci fa vedere le opzioni principali.
- Argomento5: Utilizzato per la descrizione del secondo sottocomando. Per esempio se si fa "*cambio?*" compaiono i sottocomandi di cambio con le sue

descrizioni.

- ArgomentoN: Utilizzato per la descrizione del ennesimo sottocomando.

4.3.2 Creazione dei comandi

Per prima dobbiamo dare un nome alla funzione che sarà chiamata tramite righe di comando, che per il caso trattato, si necessitano due comandi, uno per abilitare il modo energy saving ed un'altro per disabilitarlo.

I nomi dei comandi saranno:

- *router_ospf_energy_saving_cmd*
- *no_router_ospf_energy_saving_cmd*

La prima cosa da segnalare è che il nome della funzione e del comando cominciano con router_ospf_ questo succede per indicare che l'istallazione di questo comando si fa dentro di "router ospf".

4.3.2.1 Abilitazione del modo Energy Saving

Avendo definito il nome della funzione e del comando, è stato necessario individuare come verrà chiamata l'abilitazione della strategia EAR.

energy_saving router-id_ex A.B.C.D

Il nome del comando principale è *energy_*saving però questo non può essere chiamato da solo, viene chiamato con alcuna delle sue modalità, cioè, per il caso della abilitazione "*router-id_ex*".

Successivamente deve essere inserito l'indirizzo ip del router esportatore. Le descrizioni di ogni sottocomando vengono mostrate sotto.

```
DEFUN (router_ospf_energy_saving,
      router_ospf_energy_saving_cmd,
      "energy_saving router-id_ex A.B.C.D",
      "enable or disable the energy saving mode\n"
      "enable the energy saving mode and do the importation from the router-id of the exporter\n"
      "OSPF router-id of the exporter in IP address format\n")
{
```

Nel seguito viene mostrato cosa fa il comando. Come prima cosa vengono dichiarate le variabili e i puntatori, poi prende l'indirizzo ip che è in formato stringa e lo trasforma in una struttura *in_addr* che si chiama *router_id_ex*, dopo di che lo salva in formato stringa dentro *addr*. Viene poi fatta la ricerca dell'area nella quale si vuole eseguire l'OSPF, se viene trovata, *area* punterà a questa struttura e saremo in grado di accedere ai suoi attributi, altrimenti *area* sarà NULL POINTER.

```
struct ospf *ospf = vty->index;
  struct in_addr router_id_ex;
  //router id nel caso di un collegamento punto punto,o indirizzo ip della interfaccia del DR in caso di network
  struct ospf_area *area;
  struct in_addr area_id;
  const char *addr;
  int ret;

  //int format;
```

```
ret = inet_aton (argv[0], &router_id_ex);
addr=inet_ntoa(router_id_ex);
//VTY_GET_OSPF_AREA_ID (area_id, format, i);
inet_aton("0.0.0.5", &area_id);
area = ospf_area_lookup_by_area_id (ospf, area_id);
```

Successivamente si verifica la correttezza dei dati inseriti, in modo tale da poter eseguire il calcolo dello SPT senza problemi, in particolare devono essere controllate alcune condizioni di cui se almeno una è verificata il comando non sarà eseguito fino alla fine, queste condizioni sono:

- ✓ Il puntatore *area* è un NULL POINTER
- ✓ Il formato dell'indirizzo ip e diverso da A.B.C.D
- ✓ L'indirizzo ip inserito non coincide con nessuno dei router o network collegati all'importatore.

```
if (area == NULL){
  zlog_debug("non è stata trovata nessun area ret=%d y router_id_ex=%s e unsigned long=%d",ret,argv[0],router_id_ex.s_addr);
  return CMD_WARNING;
}

if (!ret)
  {
   vty_out (vty, "Please specify Router ID of the exporter by A.B.C.D%s", VTY_NEWLINE);
   return CMD_WARNING;
  }

 if(!find_link(area, router_id_ex))
```

{

vty_out (vty, "Router ID not found, please verify the Router ID of the exporter%s", VTY_NEWLINE);

return CMD_WARNING;

}

Per il controllo dell'ultima condizione è stata creata la funzione *find_link()*, questa cerca i router o le network connessi alle interfacce del nodo importatore(radice), nel caso in cui l'indirizzo viene trovato se ne deduce che il collegamento esiste. *find_link()* come *find_link_adcost()*, è stata pensata in maniera simile a *ospf_spf_next()* per la gestione di ispezionamento degli LSA ed è stata dichiarata nel file *ospf_spf.h* come:

extern int find_link(struct ospf_area *area, struct in_addr w);

È definita in ospf_spf.c. La differenza sarà che quando la struttura in ingresso coincide con uno dei link ispezionati del LSA, controllo è impostato a "1". Le parti più rilevanti di questa funzione si mostrano sotto.

```
int find_link(struct ospf_area *area,struct in_addr w){
        .....
        .....
//attenzione
        if ((area->spf != area->spf) && l->m[0].metric >= OSPF_OUTPUT_COST_INFINITE)
          continue;
        ....
        ....
      if(w_lsa->data->id.s_addr==w.s_addr){
```

```
            controllo=1;//esiste il link!!

            break;
          }
      } /* end loop over the links in V's LSA */
        return controllo;
}
```

A questo punto si è vicino all'attivazione del modo energy_saving, prima di questo tramite un *if()* si controlla che non ci si trovi già in questo stato, se non e attivo, cioè che il flag *en_sa1* è impostato a zero. Fatto questo viene cambiato e si imposta il flag a 1 per abilitare il modo *energy_saving*, si salva l'id dell'esportatore dentro la struttura *en_sa_area* e si riesegue il calcolo dell'OSPF chiamando la funzione *ospf_spf_calculate_schedule()*, questa richiama la funzione ospf_spf_calculate che calcola l'albero dell'area tenendo conto dei valori cambiati della struttura en_sa_area.

```
if (!area->en_sa_area->en_sa1)
   {
      area->en_sa_area->en_sa1=1;
      *area->en_sa_area->ex_s_addr=router_id_ex;
      ospf_spf_calculate_schedule (ospf);
   }
 return CMD_SUCCESS;

}
```

4.3.2.2 Disabilitazione

Per un controllo totale della modalità suddetta, si deve essere in grado di "spegnere" il modo energy saving quando è abilitato. Questi due comandi sono molto simili, infatti, il comando principale *energy_saving* e il quarto argomento dove viene descritta non cambia. La funzione verrà chiamata dalla sessione telnet come descritto nel terzo argomento.

energy_saving disable

Dunque il resto degli argomenti come accennato prima servono per la descrizione dei sottocomandi di questa funzione.

```
DEFUN (no_router_ospf_energy_saving,
      no_router_ospf_energy_saving_cmd,
      "energy_saving disable",
      "enable or disable the energy saving mode\n"
      "disable the energy saving mode\n")

{
```

Il modo nel quale funziona questo comando è più semplice rispetto a quello d'abilitazione. Si inizia dichiarando le variabili e inizializzandole, successivamente si verifica la correttezza dell'area.

```
struct ospf *ospf = vty->index;
  struct ospf_area *area;
  struct in_addr area_id;

  //int format;

  //VTY_GET_OSPF_AREA_ID (area_id, format, i);
```

```
 inet_aton("0.0.0.5", &area_id);
 area = ospf_area_lookup_by_area_id (ospf, area_id);
 if (area == NULL){
   zlog_debug("non è stata trovata nessun area");
   return CMD_WARNING;
 }
```

Dopo di che, si controlla con un *if()* se il flag del modo energy_saving è attivo o meno. Se non lo è, cioè se *en_sa1*=0 non esegue nessun calcolo, se invece è 1, modifica questo flag a zero e ricalcola lo SPF chiamando la funzione *ospf_spf_calculate_schedule ()*.

```
 if (area->en_sa_area->en_sa1)
  {
      area->en_sa_area->en_sa1=0;
      ospf_spf_calculate_schedule (ospf);
  }
 return CMD_SUCCESS;

}
```

Infine è necessario installare i comandi creati nella vty in modo tale che siano disponibili ed utilizzabili in qualsiasi momento. Questo si fa tramite la funzione install_element, che mediante altre funzioni rende disponibile i comandi appena creati dentro di un altro comando che per il nostro caso è “router ospf” che abilita la configurazione di OSPF.

```
install_element (OSPF_NODE, &router_ospf_energy_saving_cmd);
install_element (OSPF_NODE, &no_router_ospf_energy_saving_cmd);
```

Questi due comandi si trovano in *ospf_vty_init (void)* dentro del file ospf_vty.c.

Per ultimo è molto importante evidenziare che sono state prese alcune considerazioni che condizionano il calcolo dello SPT. Il caso teorico tiene conto di tutte le possibili variazioni del calcolo del MPT. A livello pratico si è cercata una soluzione semplice del tipo implementativo che non raggruppa tutti i casi, però dimostra che è possibile implementare la strategia EAR modificando i costi delle interfacce quando si calcola il costo del nodo analizzato in *ospf_spf_next()*.

Le considerazioni si mostrano in seguito:

- ✓ Diamo per scontato che esiste il backlink, cioè che la comunicazione sarà bidirezionale.

- ✓ Il costo da utilizzare le interfacce in un medesimo link sono uguali, perciò la ricerca dei costi può essere fatta sia nel LSA dell'importatore, che nel LSA dell'esportatore.

- ✓ Siccome si lavora nel caso di una sola area, nella configurazione tramite righe di comando non viene richiesto di specificare l'area.

CAPITOLO V

TESTBED

Lo scopo di questo capitolo è quello di fornire la descrizione del Testbed e il software utilizzato per testare il protocollo OSPF modificato in un router. Tutto il software impiegato è open source e ognuno svolge un compito diverso nell'eseguimento delle prove. Ci sono due tipi di software:

Software di simulazione: Quagga, Lsa Generator, Rude.
Software di acquisizione: Wireshark.

5.1 Software di simulazione e acquisizione di dati

5.1.1 Quagga

Quagga è già stato descritto brevemente nel capitolo II, dunque adesso si vuole analizzare il suo funzionamento.

Una volta lanciato zebra e il demone del protocollo di routing, che per il caso di studio è ospfd, il PC invierà pacchetti di Hello e LS Database al fine di stabilire l'adiacenza con i router vicini.

Per un corretto funzionamento entrambi demoni, prima di lanciarli, richiedono la scrittura di un file di configurazione. Tali file di configurazione saranno conservati nella directory usr/local/etc, se Quagga è installato in modo "default".

Per poter utilizzare il demone ospfd è prima necessario lanciare zebra, per far ciò è sufficiente digitare nel prompt dei comandi:

[root ...]# zebra -f [file di configurazione] &

L'introduzione del carattere & alla fine della riga di comandi è consigliata al fine di eseguire zebra in background in modi che non si sia costretto ad aprire una seconda shell di comandi per poter successivamente lanciare il demone di routing che ci interessa.

Per lanciare ospfd si digita il comando:

[root ...]# ospfd -f [file di configurazione] &

A questo punto il PC si comporterà come un router e sarà possibile monitorare il suo stato aprendo una sessione di telnet sul demone:

[root ...]#telnet localhost 2604

Il punto più importante del funzionamento del demone ospfd, è la scrittura del file di configurazione.

Un esempio viene mostrato in figura 5.1

```
! -*- ospf -*-
!
! OSPFd sample configuration file
!
hostname ospfd
password euro
enable password euro
! Setto i parametri dell'interfaccia 0
interface eth0
 ip ospf network point-to-point
 ip ospf hello-interval 10
 ip ospf dead-interval 40
! Settando questo valore a 0 non è elegibile come DR
  ip ospf priority 0
router ospf
  compatible rfc1583
! setto il router-id con il nome dell'interfaccia
  router-id 151.100.37.2
! Annulla i timer per il calcolo dello SPF
! timers spf 0 0
  network 151.100.37.1/32 area 0.0.0.5
  network 151.100.38.7/32 area 0.0.0.5
  network 151.100.36.7/32 area 0.0.0.5
! Abilito il terminale
!
line vty
!
! Setto la durata della connessione telnet a infinito
!
no exec-timeout
!
! File di log
!
!log file /usr/local/etc/ospfd.log
!no login
```

Fig. 5.1 Esempio di file di configurazione di ospfd

Si può osservare che è possibile settare varie opzioni quali il tempo di Dead, il timer che regola il calcolo dello Shortest Path e l'intervallo di Hello, si osservi

inoltre che è possibile definire il router id, e indicare le reti su cui s'interfaccia il nodo in questione.

5.1.2 LSA Generator

È un software open source capace di generare e inviare i pacchetti di aggiornamento del protocollo OSPF: i LS Update. LSA Generator è in grado di generare tutti i tipi di pacchetti previsti dal protocollo OSPF, ed offre all'utente una serie di opzioni per la gestione del flusso di pacchetti generati.

Fig.5.2 Schema di funzionamento di LSA Generator

Il software accetta in ingresso un file di configurazione, nel quale sono descritti i campi dell'Header e del Payload delle LSA che dovranno essere generate ed in uscita fornisce tali LSA inviandoli all'interfaccia di rete.

La struttura del LSA è descritta nel RFC 2328.

Nella figura 5.3 si presenta un esempio di un router LSA update.

```
0 2 192.168.0.33 192.168.0.33 0x80000011 0 3
192.168.0.34 192.168.0.33  1  0  10
192.168.0.25 192.168.0.25  2  0  10
```

Fig.5.3 Esempio di un Router LSA

Si noti che la prima riga contiene l'intestazione del'LSA mentre le righe successive contengono l'informazione dell'interfacce del nodo in questione.

La prima riga contiene:

- ✓ Età dell'LSA
- ✓ Options
- ✓ Link State ID
- ✓ Advertising Router
- ✓ LS Sequence Number
- ✓ Option
- ✓ Number of Links

Le righe successive contengono le seguenti informazioni:

- ✓ Link ID
- ✓ Link Data
- ✓ Type
- ✓ Number of TOS
- ✓ Metric

Il campo Link Data indica l'indirizzo IP dell'interfaccia di rete del nodo. Type il tipo di collegamento, nell'esempio di figura 5.4 ce ne sono due tipi:

a) type 1, il collegamento è punto-punto
b) type 2, il collegamento è verso una transit network.

Per il caso del collegamento punto-punto, il campo Link ID indica il router id a cui il nodo in questione è connesso tramite l'interfaccia. Per il caso del collegamento type 2, il link ID indica l'indirizzo del Designated Router della transit network a cui il nodo è connesso.

Per simulare la rete è pero necessario inviare al DUT l'informazione sulle transit network riportate nei router LSA.

Tale informazione è necessaria giacché è utilizzata dal DUT per redigere lo Shortest Path relativo ad ogni singolo nodo. La suddetta informazione è inviata mediante la generazione di una network LSA.

La fig 5.4 si mostra un esempio di network LSA

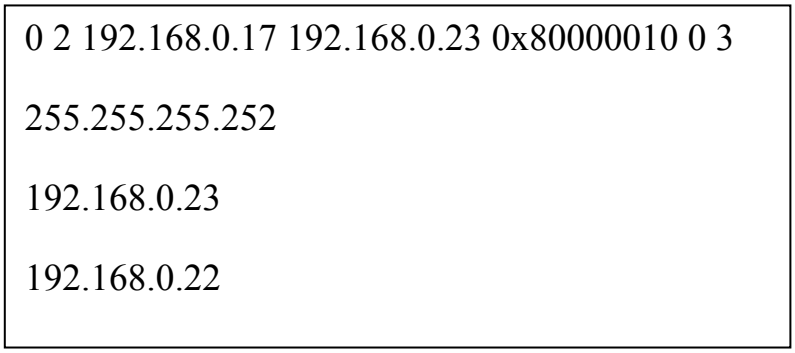


Fig.5.4 Esempio di Network LSA

Anche seguendo le regole del RFC 2328, la prima riga contiene i campi relativi all'header del pacchetto LSA, mentre le successive righe contengono le informazioni relative al suo payload, che nel caso delle Network LSA è costituito da:

1. La Network Mask della rete
2. I Router Id dei nodi direttamente connessi alla rete

È importante notare che i campi dell'header di una LSA Update hanno un significato diverso secondo il tipo di LSA. In particolare i campi Link State ID ed Advertising Router di una Router LSA indicano entrambi il Router Id del nodo sorgente, mentre quelli di una Network LSA indicano rispettivamente l'indirizzo dell'interfaccia con cui il Designated Router si connette alla rete(l'Id del DR) e il Router Id del DR.

Inviando quindi a un Router pacchetti LSA contenenti le informazioni lette dai due file di configurazione si riesce a simulare la presenza di una rete.

Una volta scritto il file di configurazione, è possibile decidere se inviare le LSA Update in esso descritte introducendo un ritardo di trasmissione tra ognuna di esse e la successiva, se inviare più di una volta il treno di LSA contenuto nel file, incrementando ad ogni invio il numero di sequenza delle LSA in modo che esse siano sempre riconosciute dal Router di destinazione come delle New LSA e se introdurre tra l'invio di un treno ed il suo successivo, un ulteriore ritardo.

Altra opzione è quella relativa alla variazione della metrica dei collegamenti di una delle LSA inviate. Tale opzione può essere utilizzata solo nel caso in cui sia attiva anche l'opzione relativa all'invio di più treni di pacchetti LSA, in tali condizioni LSA Generator è in grado di apportare una variazione della metrica dei collegamenti relativi ad una delle LSA del treno, rispetto alla metrica dei collegamenti della stessa LSA contenuta nel treno precedente.

Per lanciare Lsa Gen e le sue capacità opzionali descritte è necessario portarsi nella cartella "lsa_gen" e digitare al prompt dei comandi:

- [.../lsa_gen]#./lsa_gen -s[sorgente] -d[destinazione] -r[router id] -t[tipo di LSA] -f[file di configurazione][opzione -i, -b, -m] -n[numero di ripetizioni del treno di LSA]

- -s [sorgente]: Indica l'indirizzo ip dell'interfaccia da cui dovranno essere inviati i pacchetti LSA Update.
- -d [destinazione]: Indica l'indirizzo ip di destinazione dei pacchetti LSA Update inviati.
- -r [router id]: Indica l'indirizzo ip del router sorgente da cui si stanno inviando i LSA Update.
- -t [tipo di LSA]: Indica il tipo di LSA contenute nel file di configurazione. Router LSA=1 e Network LSA=2.
- -f [file di configurazione]:Indica il file di configurazione da utilizzare.
- -i [Ritardo tra una LSA e la successiva]: Specifica il ritardo in microsecondi tra l'invio di una LSA e la successiva.
- -b [Ritardo tra un treno di LSA e il successivo]: Specifica il ritardo in sec tra l'invio di un treno di LSA e il successivo. Ha senso unicamente se il campo -n indica che si invieranno più treni di LSA Update.
- -m [Incremento della metrica]:Indica l'incremento della metrica che subiranno i collegamenti descritti in una delle LSA Update.
- -n [numero di ripetizioni del treno di LSA]:Indica il numero di volte che il treno di LSA Update, descritto nel file di configurazione, dovrà essere inviato.

Dunque inviando al router pacchetti LSA contenenti le informazioni lette dai due file di configurazione, si riesce a simulare la presenza di una rete.

5.1.3 Rude&Crude

RUDE è l'acronimo di Real-time UDP Data Emitter e CRUDE for Collector for RUDE. RUDE è un programma piccolo e flessibile che genera traffico verso la rete, e che può ricevere e registrare al altro capo della rete con CRUDE. Attualmente questi programmi possono generare e misurare solo il traffico UDP.

Questi programmi sono un "prodotto collaterale" del progetto Faster 2000 (ex Faster Pro) che si concentra su diversi meccanismi di QoS in reti IP. Ciò significa che questi strumenti sono stati progettati e sviluppati per uno scopo specifico, perciò la documentazione non è ancora completa e le operazioni sono abbastanza limitate.

Il funzionamento e la configurazione potrebbero essere simili al tool disponibili in un altro generatore di traffico chiamato MGEN ma questi programmi non condividono nessun codice. In realtà questi strumenti sono stati progettati e codificati per le limitazioni del programma MGEN. MGEN opera con timer di sistema e ad esempio nel kernel di Linux su piattaforme PC, la risoluzione del timer è solo 10ms, che è piuttosto scarsa, così i creatori hanno deciso di fare il suo proprio generatore di traffico che non ha questa limitazione.

Per le simulazioni è di particolare interesse il software RUDE, per lanciarlo si digita il comando:

[root ...]# ./rude -s [file di configurazione] &

Come succedeva per il caso di ospfd la stesura del file di configurazione è la cosa più importante. Un esempio di file di configurazione di RUDE è mostrato in figura 5.5.

```
START NOW

1000 0030 ON 3002 10.1.1.1:10001 CONSTANT 200 250

3000 0030 MODIFY CONSTANT 400 500

4000 0030 MODIFY CONSTANT 1000 1000

5000 0030 OFF
```

Fig.5.5 Esempio di file di configurazione di RUDE.

Le righe vuote o che iniziano con '# 'sono ignorate.

- START: imposta l'inizio del tempo assoluto per il programma rude.
- NOW: se nessun ritardo è necessario
- Stime:1000. Tempo relativo al START TIME in millisecondi. Indica quando il flusso sarà attivato.
- id ON:0030, identificativo del flusso.
- sport:3002, la porta UDP per il flusso. Deve essere maggiore a 1024.
- dst.add:10.1.1.1, l'indirizzo della destinazione in formato d'indirizzo ip. Deve essere seguito di un ":" e il numero di porta della destinazione
- dst.port:10001, porta UDP della destinazione.
- type:CONSTANT, invio di pacchetti senza interruzioni.
- rate:200, pacchetti per secondi.
- psize:250, lunghezza del pacchetto.

Le due righe succesive servono per modificare il rate e il packet size.L'ultima riga ferma l'invio del flusso.

5.1.4 Wireshark

Wireshark (precedentemente chiamato Ethereal) è un software per analisi di protocollo, o packet sniffer (letteralmente *annusa-pacchetti*) utilizzato per la soluzione di problemi di rete, per l'analisi e lo sviluppo di protocolli o di software di comunicazione, per la didattica. Wireshark possiede tutte le caratteristiche di un analizzatore di protocollo standard.

Le funzionalità di Wireshark sono molto simili a quelle di tcpdump, ma con un'interfaccia grafica, e maggiori funzionalità di ordinamento e filtraggio. Permette all'utente di osservare tutto il traffico presente sulla rete utilizzando la modalità *promiscua* dell'adattatore di rete. Tipicamente si riferisce alle reti Ethernet, ma è possibile analizzare altri tipi di rete fisica.

Wireshark è rilasciato sotto una licenza Open Source; gira sulla maggior parte dei sistemi Unix e compatibili (inclusi GNU/Linux, Sun Solaris, FreeBSD, NetBSD, OpenBSD e Mac OS X) e sui sistemi Microsoft Windows appoggiandosi al toolkit di grafica multipiattaforma GTK+ (GTK+ necessita del server grafico X11 per essere eseguito su Mac OS X, l'utente Mac deve eseguire anche un X Server come X11.app).

Wireshark riesce a "comprendere" la struttura di diversi protocolli di rete, è in grado di individuare eventuali incapsulamenti, riconosce i singoli campi e permette di interpretarne il significato.

Per la cattura dei pacchetti Wireshark non dispone di proprio codice, ma utilizza libpcap/WinPcap, quindi può funzionare solo su reti supportate da libpcap o WinPcap.

Caratteristiche e Funzioni

È possibile analizzare dati acquisiti in tempo reale su una rete attiva ("from the wire"), come pure analizzare dati salvati precedentemente su file di cattura. I dati possono essere acquisiti dal vivo su reti Ethernet, FDDI, PPP, Token Ring, IEEE 802.11, IP classico su ATM, e interfacce di loopback (non tutti i tipi sono supportati su tutte le piattaforme). È possibile analizzare i dati sia tramite interfaccia grafica che da riga di comando, con il programma "tshark".I dati catturati su file possono essere facilmente modificati, convertiti o filtrati, tramite opzioni su riga di comando del programma "editcap". È possibile filtrare i dati da visualizzare, e utilizzare filtri di visualizzazione per colorare o evidenziare selettivamente le informazioni sommarie sui pacchetti.

È possibile scomporre e analizzare centinaia di protocolli di comunicazione. Il software di cattura WinPcap, che in passato doveva essere scaricato separatamente, è ora compreso nel pacchetto.

La versione a riga di comando, Ywireshark, permette di lavorare comodamente su sistemi Unix e Unix-like, ma è comunque disponibile anche su Windows.

5.2 Test-bed per il controllo de la modalità “energy saving” e la valutazione del tempo di commutazione.

5.2.1 Dispositivi utilizzati e caratteristiche

Per le prove realizzate sono stati utilizzati 3 computer fissi e 1 portatile. Tutti e 4 dotati con il sistema operativo GNU/linux Mandriva nelle versioni 2008 e 2010, il Device Under Test(DUT) con un processore dual core ognuno con le seguenti caratteristiche:

- ✓ Intel(R) Pentium(R) D CPU 2.80GHz
- ✓ Dimensione cache: 1024 KB

- ✓ memoria RAM:2048 KB.

La versione di Quagga utilizzata è Quagga 0.99.17. I computer hanno tutti adattatori di rete Fast Ethernet 10/100 e sono collegati secondo la configurazione mostrata sotto.

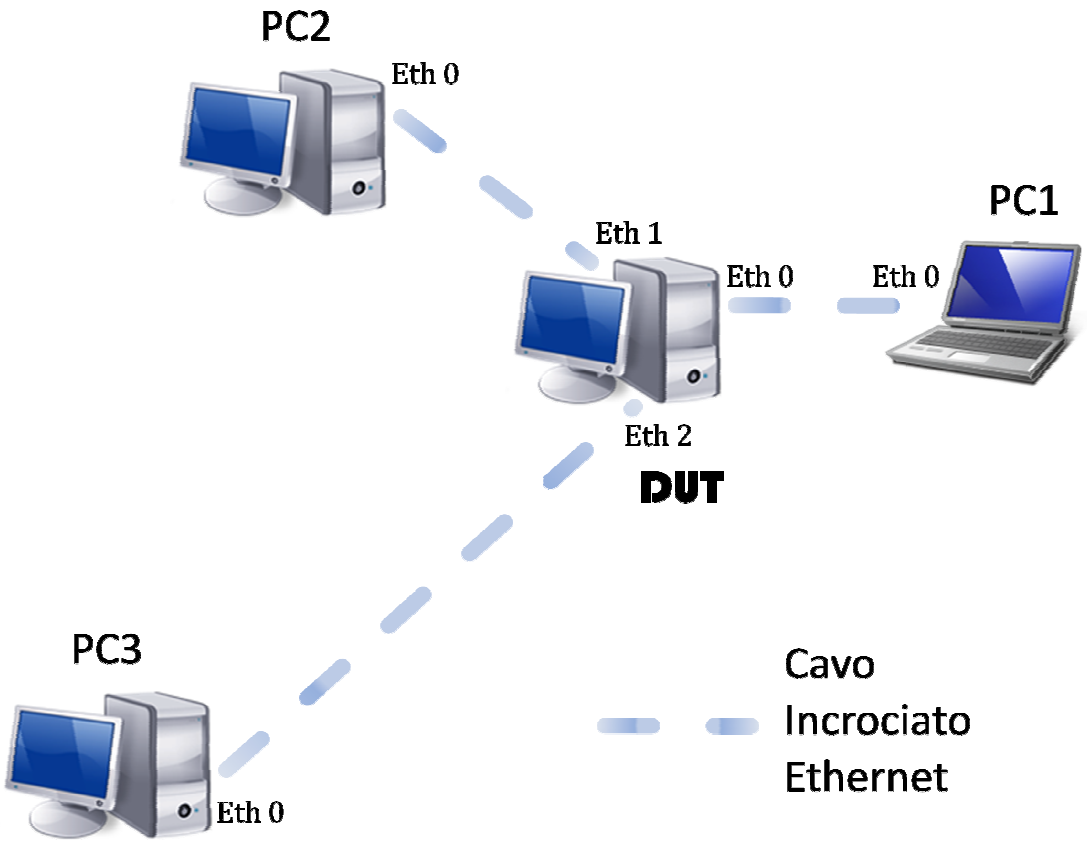


Fig. 5.6 Configurazione fisica dei PC

V.2.2 Eseguimento del test

V.2.2.1 Impostazioni iniziali

I parametri di ingresso del test sono i seguenti:

a) Fp, velocità alla quale sono trasmessi i pacchetti da PC1; 5000 pacchetti per secondo.

b) Lp, la lunghezza dei pacchetti inviati da PC1;500 bytes.

c) N e M, il numero di vertici e spigoli del grafo orientato che rappresentano la topologia di rete emulata rispettivamente. Si noti che i parametri N e M dipendono dal numero di router e delle reti nella topologia di rete emulata.

5.2.2.2 Descrizione del Test

Questo test determina il tempo di commutazione quando è attivata la modalità energy saving, cioè dopo che viene digitato il commando per abilitare la strategia EAR. Un router OSPF impiega un tempo determinato per far riconvergere la tabella di routing e di reindirizzare il traffico dati al nuovo percorso. Nella figura 5.7 vengono mostrati le fasi per l'eseguimento del test.

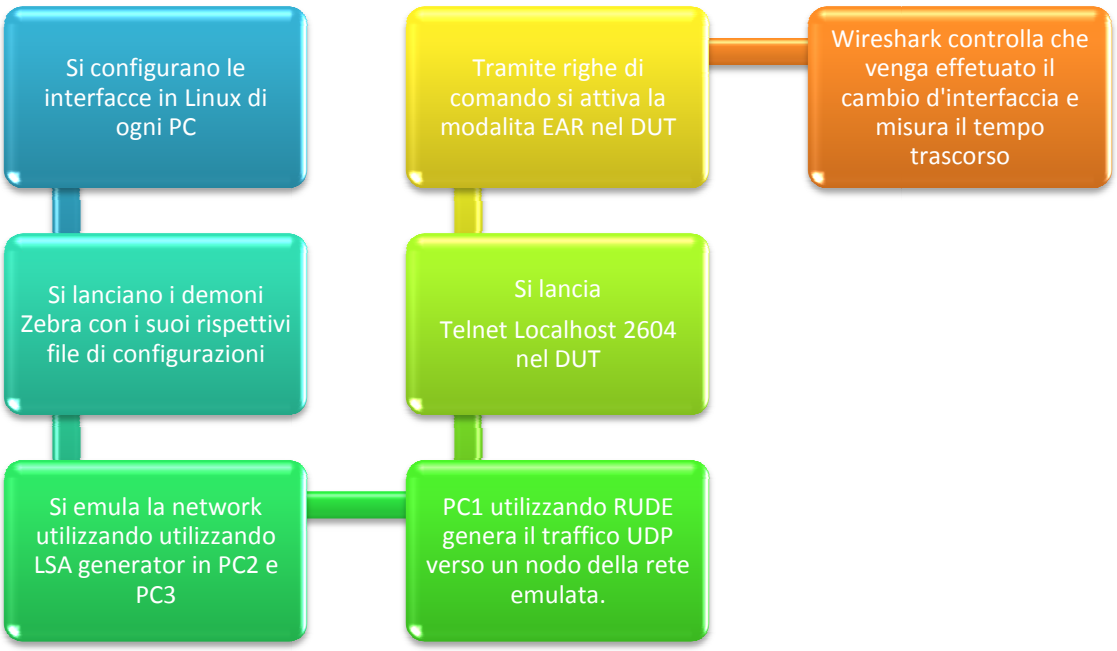


Fig. 5.7 Schema delle fasi per l'eseguimento del test.

E 'importante sottolineare che quando si rappresenta la rete emulata come un grafo orientato pesato, ogni router e ogni rete di transito della topologia di rete emulata diventa un vertice del grafo, e ogni collegamento di rete diventa un arco. Ogni arco deve assumere un costo che rappresenti il costo dell'interfaccia che connette il router con una "network" o altro router.

Usiamo la configurazione di prova riportata in fig.5.6, per determinare le prestazioni di convergenza di percorso del router Device Under Test (DUT).

Il DUT è collegato a tre PC di prova, chiamati PC1 PC2 e PC3. PC1 è connesso al DUT con una sola interfaccia di rete Fast Ethernet. La sua funzione è di generare il traffico dati che il DUT instraderà utilizzando la miglior via disponibile. PC1 genera traffico UDP per mezzo del generatore di traffico RUDE. PC2 e PC3 sono connessi al DUT con tre interfacce di rete Fast Ethernet. La loro funzione è

quella di emulare una topologia di rete complessa, questo si fa generando qualche particolare Link State Advertisements (LSA) il quale notifica al DUT che ci sono nuovi dispositivi e quindi nuovi percorsi disponibili. In particolare, PC2 e PC3 emulano la topologia riportata in fig. 5.8. La topologia "Network" in figura due è composta di un numero variabile di router e alti livelli di connettività tra di loro. Il DUT deve trovare i percorsi più breve per ogni transit network e router. La topologia di rete emulata è generata utilizzando il software LSA GENERATOR per inviare al DUT i LSAs appropriati con le descrizioni della topologia.

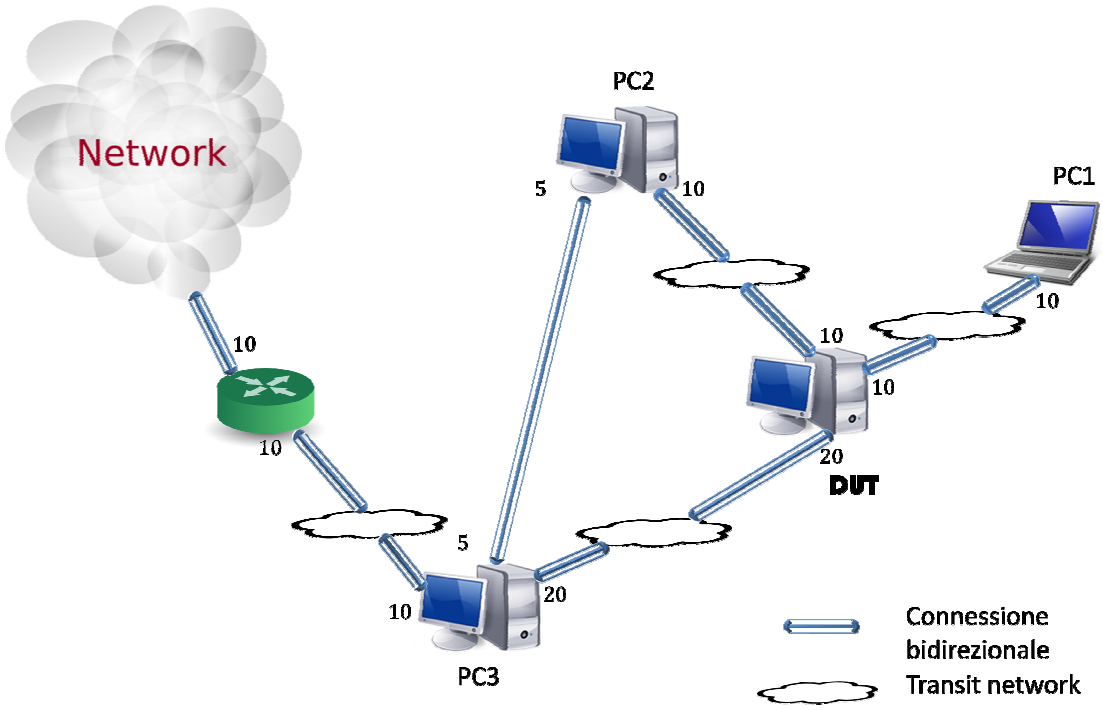


Fig.5.8. Per misurare il tempo di commutazione, il DUT è collegato a PC1, PC2 e PC3. PC1 con generatore di traffico RUDE, invia i pacchetti di dati. PC2 e PC3 emulano una topologia di rete.

Come illustrato nella fig.5.8, il traffico di dati inviato da PC1 può raggiungere qualsiasi destinazione della topologia emulata attraverso due percorsi diversi che coinvolgono ciascuna delle due interfacce di rete tra il DUT e PC2 e il DUT e PC3. In particolare, si è stabilito inizialmente tutti i costi, quindi il percorso attraverso PC2 sarà sempre il meno costoso pertanto il più conveniente, ciò significa che tutto il traffico dati che arriva da PC1 va diretto al PC2.

In un momento determinato, andando all'interfaccia di configurazione del terminale del DUT, si abilita la modalità energy saving e viene scelto l'esportatore, come mostra la figura 5.9. Questa cambio modifica il costo delle interfacce dell'DUT e rende più conveniente il percorso attraverso l'esportatore. Dopodiché i nuovi percorsi sono calcolati mediante l'algoritmo di Dijkstra e si aggiorna la tabella d'instradamento. Di conseguenza, il traffico dati viene inviato sulla interfaccia che va verso l'esportatore. Il tempo di questa operazione di cambiare da una interfaccia all'altra, sarà chiamato tempo di commutazione. Il test può essere effettuato variando la posizione del router o transit network di destinazione.

```
[root@localhost  +]# telnet localhost 2604
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
2011/04/18 11:43:40 OSPF: Vty connection from 127.0.0.1


Hello, this is Quagga (version 0.99.17).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification


Password:
ospfd> enable
ospfd# configure terminal
ospfd(config)# router ospf
ospfd(config-router)# energy_saving ?
  disable       disable the energy saving mode
  router-id_ex  enable the energy saving mode and do the importation from the
```

Fig. 5.9 Attivazione della modalità energy saving nella shell di LINUX

Il tempo di commutazione può essere calcolato come la differenza di tempo tra l'avvio della modalità energy saving e l'invio del primo pacchetto UDP sull'interfaccia che va verso il router esportatore.

$$Tsw = Tudp_{ex} - Ten\_sa \quad (7)$$

5.2.2.1 Valutazione e Risultati

Il tempo di commutazione dipende dal numero di vertici N e M archi nel grafo, che rappresentano la rete emulata. Ciò è dovuto al fatto che, come già detto, uno dei componenti del tempo di commutazione è il tempo di calcolo SPF la cui valutazione dipende sia da N e M. In particolare, il calcolo SPF inizia quando il DUT riceve il comando *energy saving*.

Se consideriamo una topologia di rete emulata composta da R router , abbiamo:

$$N = \frac{\mathrm{R(R-1)}}{2} + R + 4 \quad (8)$$

$$M = 2R(R-1) + 8 \quad (9)$$

E' importante notare che, nel caso di studio considerato, il numero di archi M (collegamenti) è proporzionale al numero di vertici N, cioè M = O (N).

Nella tabella 5.1 vengono mostrate le diverse configurazioni utilizzate e i risultati ottenuti.

Quantità di Router (R)	**4**	**17**	**50**	**100**
Numero di Vertici (N)	14	157	1279	5054

Numero di Collegamenti (M)	32	552	4908	19808
Tsw	2	2	4	14

Tabella 5.10 Tempo di commutazione a seconda dei router

Nel grafico 5.10 viene mostrato l'andamento che descrive la crescita del tempo di calcolo a seconda del numero di Router, Vertex e Collegamenti emulati.

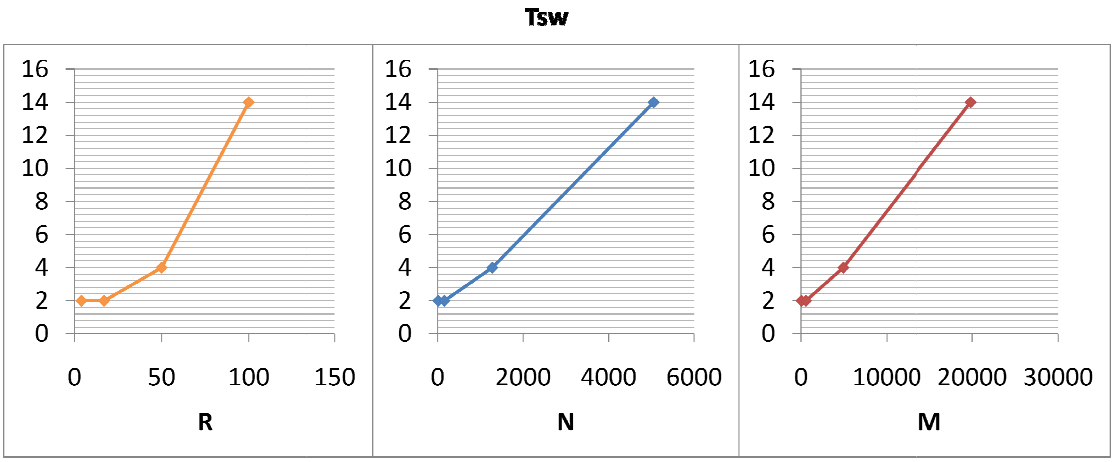


Fig.5.10 Andamento del tempo di switching rispetto a R,N e M.

Si può vedere che il tempo di switching ha un rapporto quasi lineare con la quantità di vertex e collegamenti. Questo è dovuto all'algoritmo di Dijkstra, che in termini di costo computazionale, dipende del numero di vertex e la connettività tra di loro.

CONCLUSIONE

Oggi giorno si stanno presentando molte possibilità di risparmio energetico nella core network, considerata la parte d'Internet con la tendenza ad avere il consumo d'energia maggiore nel futuro. Perciò si ritiene molto importante utilizzare le strategie proposte e avanzare verso un Internet più "green". Nel lavoro fatto è stato verificato che EAR è implementabile nel protocollo di routing OSPF e che tramite questa strategia si ha la capacità di lasciare inutilizzati alcuni collegamenti. Lo studio della topologia di rete è necessaria per poter mettere in modalità sleep la maggior quantità di collegamenti e massimizzare il risparmio energetico.

Per trovare la maggior quantità di collegamenti plausibili a uno spegnimento, è fondamentale dare un passo avanti, il quale potrebbe essere creare un nodo di controllo che decida quando avviare EAR. Questa unità deve essere in grado di calcolare quali saranno le coppie "importatore, esportatore", con il maggior risultato di collegamenti spenti dipendendo della topologia di rete. Il nodo centrale deve anche elaborare un tipo di messaggio specifico sia modificando gli LSA previsti in OSPF, sia creando un nuovo tipo di messaggio per comunicare all'importatore chi é l'esportatore.

Per ultimo sarà necessario andare al livello fisico studiando l'architettura del PC, con la finalità di spegnere o mettere in modo "stand-by" le NIC (schede di rete) con cui si instradano i pacchetti. Dopodiché sarà possibile misurare e valutare la quantità d'energia risparmiata.

BIBLIOGRAFIA

[1] Kevin H. Liu, Changdong Liu, Jorge Pastor, Arunendu Roy, and John Y. Wei "Experimental Study of Dynamic IP Topology Reconfiguration in IP/WDM Networks" Network Operations and Management Research Department, Telcordia Technologies 331 Newman Springs Road Red Bank, NJ 07701-5699, USA.

[2]A. Cianfrani et al., "An Energy Saving Routing Algorithm for A Green OSPF protocol," Proc. IEEE INFOCOM '10, San Diego, CA, Mar. 2010.

[3]Claudio Gerbino, A.A. 2004/2005, "T E C N O L O G I E W E B".

[4] Andrea Rescigno, A.A. 2009/20010, " Esperimenti di Networking per un uso didattico di una piattaforma di emulazione cluster based".

[8] GNU Quagga Routing Software, http://www.quagga.net/.

[9] http://openmaniak.com/quagga_tutorial.php.

[10] J. Baliga et al., "Energy Consumption in Optical IP Networks," J. Lightwave Tech., vol. 27, no. 13, 2009, pp. 2391–403.

[11] R. Bolla et al., "DROP: An Open-Source Project towards Distributed SW Router Architectures," Proc. IEEE GLOBECOM '09, Honolulu, HI, Dec. 2009.

[12]Andrea Bianco et al. "Open-Source PC-Based Software Routers: A Viable Approach to High-Performance Packet Switching".

[13] M. Gupta, S. Singh, "Greening of the Internet", Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003.

[14] RUDE/CRUDE Traffic Generator, http://rude.sourceforge.net/.

[15]LSA Generator Software SPOOF,http://www.cs.ucsb.edu/~rsg/Routing/download.html.

[16]V. Eramo, M. Listanti e A. Cianfrani,"SWITCHING TIME MEASUREMENT AND OPTIMIZATION ISSUES IN GNU QUAGGA ROUTING SOFTWARE", University of Roma "La Sapienza", INFOCOM Dept..

[17]IEEE P802.3az Energy Efficient Ethernet Task Force Available at: http://grouper.ieee.org/groups/802/3/az/public/index.html.

[18]J. Moy. ”OSPF Version 2” , Request for Comments 2328, April 1998.

[19]Antonio Cianfrani, Vincenzo Eramo, Marco Listanti, Marco Polverini ,“An OSPF Enhancement for energy saving in IP Networks”, SUBMITTED TO IEEE INFOCOM 2011 Workshop on Green Communications and Networking.

[20]Raffaele Bolla, Roberto Bruschi, Antonio Cianfrani e Marco Listanti “Enabling Backbone Networks to Sleep”, IEEE Network March/April 2011.

[21] Kerry Hinton, Jayant Baliga, Michael Feng, Robert Ayre, and Rodney S. Tucker, “Power Consumption and Energy Efficiency in the Internet”, IEEE Network March/April 2011.

[22] D. Kilper et al., “Power Trends in Communications Networks,” IEEE J. Sel. Topics in Quantum Elect., to appear, 2011.

[23]European Commission, “Code of Conduct on Power consumption of Broadband Equipment,” v. 3, Nov. 2008.

[24]K. Hinton et al., “The Future Internet — an Energy Consumption Perspective” invited paper FT1, OECC 2009.

[25]S. Nedevschi et al., “Reducing Network Energy Consumption via Sleep and Rate-Adaptation,” Proc. 5th USENIX Symp. Network System Design and Implementation, 2008, pp. 232–26.