UNIVERSITÀ DI PISA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria delle Telecomunicazioni

Tesi di Laurea

Gestione della QoS in WMN: progetto di un'architettura basata sul modello MPLS e definizione di algoritmi di "QoS Path Selection"

Relatori

Prof. Stefano Giordano

Ing. Rosario Garroppo

Ing. Davide Adami

Candidato

Fabio Del Vasto

Anno accademico 2007/2008

Indice

Introduzione

L'evoluzione delle reti wireless, in particolare dei sistemi WLAN 802.11, ha portato alla nascita di nuove applicazioni e nuovi scenari d'utilizzo. L'interesse della comunità scientifica è concentrata ora sulle reti wireless mesh (WMN), che costituiscono una naturale evoluzione delle precedenti reti ad hoc (MANET). Le WMN presentano una infrastruttura di rete più affidabile, costituita da nodi fissi, detti mesh point, connessi tra loro per formare un sistema di distribuzione wireless. Alcuni di questi nodi forniscono in modo aggiuntivo anche servizi di Access Point e di connettività verso le reti esterne.

Poiché le WMN stanno diventando una tecnologia di rete sempre più diffusa, esse devono essere in grado supportare una nuova generazione di applicazioni multimediali, come ad esempio quelle real-time. Queste applicazioni richiedono garanzie di Qualità del Servizio (QoS), in termini ad esempio di massimo ritardo end-to-end oppure di massima percentuale di perdita dei pacchetti. Nelle reti cablate e ottiche la soluzione è stata ricorrere a tecniche di Traffic Engineering (TE) e utilizzare il paradigma MPLS.

Il primo obiettivo di questa tesi è stato quindi quello di proporre un'architettura capace di fornire supporto di QoS anche in WMN. La

soluzione proposta utilizza sempre tecniche di TE e MPLS, tenendo conto però, rispetto allo scenario precedente, della diversa natura del mezzo wireless. In particolare ci si è resi conto che era necessario ricorrere a nuove metriche (di tipo radio – aware), nonché considerare fattori tipici del canale wireless come la mobilità dei nodi e la trasmissione in broadcast delle trame,

L'architettura per il supporto di tecniche di TE in WMN ricalca quella che viene utilizzata nelle reti cablate: essa è suddivisa in diversi blocchi funzionali, appartenenti al piano di controllo o al piano dati. Il primo comprende i blocchi (protocollo di routing, TE database, Path Computation Element e protocollo di segnalazione) che si occupano dei meccanismi di segnalazione, instaurazione e gestione dei path su cui saranno trasportati i flussi dati. Il piano dati comprende i blocchi (MPLS e altre strutture dati) che intervengono direttamente sui frame dati che viaggiano sui path instaurati. E' necessario però riprogettare ad hoc questi blocchi, in modo da affrontare le problematiche (o in altri casi, sfruttare le caratteristiche) del canale wireless già citate prima.

Il risultato finale sarà un'architettura trasparente alle stazioni client (solo i nodi mesh implementeranno le funzionalità introdotte) e collocabile a livello 2.5 nella pila ISO/OSI. Ulteriori vantaggi possono essere introdotti intervenendo anche a livello MAC, ad esempio utilizzando un nuovo protocollo di accesso al mezzo per reti wireless label switched come DCMA, oppure semplificando i campi d'indirizzamento (cosa resa possibile dall'uso delle label).

Visto che il progetto di tutti i blocchi comportava tempi decisamente superiori a quelli di una tesi, si è deciso di progettare un solo blocco, quello

del Path Computation Element. In particolare il secondo obiettivo della tesi è stato l'implementazione di quattro algoritmi di calcolo e selezione dei percorsi soddisfacenti le richieste di QoS che provengono dai flussi, da inserire poi nel PCE. L'idea di base è quella di generare più percorsi possibili tra un nodo mesh di partenza e uno destinatario, in modo da aumentare la probabilità di trovarne almeno uno soddisfacente tutte le richieste di QoS di un flusso. Questi algoritmi derivano da alcune intuizioni tratte dall'osservazione del funzionamento di algoritmi esistenti, oppure rappresentano implementazioni, opportunamente modificate, di pseudo-codici di algoritmi utilizzati nella Teoria dei Grafi.

Dall'analisi delle prestazioni, effettuata per via simulativa, si è visto come sia possibile, mediante piccole modifiche al comune algoritmo di Dijkstra, ottenere risultati paragonabili a quelli di algoritmi ben più complessi, ma con tempi di calcolo decisamente inferiori.

La tesi è strutturata come segue: nel capitolo 1 sono riassunti l'architettura delle WMN e le funzionalità principali delle tecniche di Traffic Engineering e di MPLS. Segue una panoramica di alcune proposte di architetture di rete wireless basate sul concetto di label switching.

Nel capitolo 2 viene descritta l'architettura TE-WMN, spiegando le funzionalità principali dei blocchi funzionali. Vengono poi descritte le metriche di tipo radio-aware introdotte per il supporto alla QoS.

Nel capitolo 3 vengono descritti il protocollo di routing che si è deciso di utilizzare, RA-OLSR, e la struttura del Traffic Engineering Database.

Nel capitolo 4 sono inizialmente spiegate le caratteristiche principali di un Path Computation Element. Successivamente, vengono descritti gli algoritmi più conosciuti utilizzati all'interno dei PCE, mostrando i loro

limiti e le problematiche affrontate per lo sviluppo di altri alternativi. Infine, sono descritti i quattro algoritmi sviluppati all'interno di questa tesi (Dijkstra modificato, Iwata modificato, Mittal e Martin-Santos).

Nel capitolo 5 sono descritti lo scenario di simulazione, le modalità con cui sono state effettuate le prove sperimentali e sono presentati i risultati ottenuti.

Cap. 1 Introduzione

La rapida diffusione delle tecnologie wireless nell'ultimo decennio, in particolare dei sistemi WLAN (Wireless LAN) 802.11, ha portato allo sviluppo di nuove applicazioni e scenari di utilizzo. L'attenzione della comunità scientifica si è concentrata dapprima sulle reti infrastrutturate, basate su un access point (AP) centrale collegato ad una LAN cablata che funge da unico tramite per il traffico dei dispositivi wireless che si trovano nel range di copertura dell'AP. Successivamente hanno suscitato enorme interesse le reti MANET (Mobile Ad hoc Network), in cui è possibile collegare in modo indipendente più postazioni wireless tra loro senza nessun dispositivo centrale che funga da tramite. Tuttavia un sistema di questo tipo non è adatto ad una rete numerosa e concentrata, a causa dei problemi di scalabilità, della sovrapposizione dei segnali e del conseguente calo di affidabilità. Per questo motivo le reti ad hoc hanno trovato poco interesse da parte del mercato.

Le WMN (Wireless Mesh Network) sono nate come una naturale evoluzione delle MANET. Rispetto alle reti ad hoc, le WMN differiscono da queste per la presenza di un'infrastruttura di rete più affidabile, rappresentata tipicamente da nodi fissi connessi tra loro per costituire un sistema di distribuzione wireless. In questo scenario ogni nodo della rete può fungere allo stesso tempo da ricevitore, trasmettitore e ripetitore. Il crescente interesse nello sviluppo delle WMN ha portato alla nascita nel mondo accademico ed industriale di numerose soluzioni eterogenee ed

incompatibili: per questo motivo la IEEE ha iniziato un'attività di standardizzazione. È nato così un Task Group in ambito IEEE denominato 802.11s, che ha come obiettivo quello di redigere uno standard per le WMN basato su 802.11.

Dato che le WMN stanno diventando una tecnologia di rete sempre più diffusa, esse devono essere anche in grado di supportare una nuova generazione di applicazioni multimediali come Voice over IP (VoIP) e Video On-Demand (VOD). Queste applicazioni richiedono garanzie di Qualità del Servizio (QoS) in termini di alcuni parametri, come banda minima garantita e massimo ritardo end-to-end. In questo ambito può essere utile (e in taluni casi necessario) ricorrere a tecniche di TE (Traffic Engineering), principalmente load balancing ma anche connection admission control, traffic policing, priorità di traffico, eccetera. Infine, è necessario introdurre flessibilità per quanto riguarda il routing, ossia implementare meccanismi per il calcolo dei percorsi tra due nodi che tengano conto di fattori non considerati nelle tecniche di routing tradizionali.

Nelle reti cablate e ottiche, MPLS (Multi-Protocol Label Switching) ha già mostrato grandi vantaggi come flessibilità e semplicità di integrazione con le altre tecnologie di trasporto. È anche vero che però MPLS è stato progettato per reti 'wired' e non wireless, per cui non tiene conto di alcuni fattori tipici delle reti senza fili come ad esempio l'elevata mobilità dei nodi, specie nelle reti ad hoc, e la trasmissione in broadcast delle trame verso tutti i vicini di un nodo.

Diversi sono stati i tentativi per poter implementare meccanismi di QoS e di TE nelle reti wireless (alcuni riportati nel paragrafo 1.4), usando principi

generali di MPLS come ad esempio il concetto di label switching o di resource reservation. Le soluzioni proposte tuttavia presentano diverse lacune o comunque lasciano aperti altri campi di ricerca. Il motivo sta nel fatto che solo recentemente l'attenzione della comunità scientifica si è spostata su queste problematiche. Inoltre questi lavori si riferiscono quasi esclusivamente ad ambienti MANET e non mesh, per cui si immagina come la ricerca sia ancor di più aperta se consideriamo le WMN.

Organizzazione del capitolo

Prima di passare alla descrizione dell'architettura di rete proposta in questa tesi, in questo capitolo si è voluto puntualizzare i concetti appena esposti.

Il paragrafo 1.1 affronta le WMN 802.11s. Su questo argomento esiste una vasta letteratura, per cui l'obiettivo preposto è stato quello di descrivere solamente gli elementi che compongono l'architettura ed il formato dei frame e la loro classificazione, visto che le informazioni di controllo dei protocolli sviluppati nel seguito della tesi sono incapsulate in queste trame. Aspetti come la sicurezza, la sequenza di avvio di un mesh point sono tralasciati, mentre il discorso sui protocolli di routing è affrontato successivamente.

Il paragrafo 1.2 introduce il concetto di Traffic Engineering: viene data la definizione e vengono descritti i meccanismi necessari all'implementazione di queste funzionalità.

Il paragrafo 1.3 è dedicato a MPLS. Come ben si sa, la letteratura trattante quest'argomento è ancora più ampia, dato che esistono numerose RFC della IETF e molti libri ed articoli a riguardo. Si è quindi deciso di sintetizzare i concetti principali che saranno poi utilizzati all'interno di questa tesi, ossia

gli obiettivi di MPLS, il meccanismo di label switching, gli elementi che compongono ogni nodo e i meccanismi utilizzati per l'applicazione delle funzionalità di TE. Non si descrive invece l'architettura, visto che essa sarà alla base di quella sviluppata nell'ambito di questa tesi: quest'ultima sarà infatti descritta a partire dal capitolo successivo.
Il paragrafo 1.4 effettua un panorama di alcuni articoli in cui sono realizzate architetture di reti o protocolli basati sul label switching. Sono inizialmente descritte le problematiche che ci si prefigge di risolvere, le modalità con cui sono applicate le tecniche di label switching, gli svantaggi presentati. Si è deciso di dare spazio a questa parte perché si ritiene interessante far presente i vari tentativi di applicazione del principio di commutazione d'etichetta alle reti wireless, un ambito come detto ancora inesplorato (ancor di più per le reti mesh).

1.1 Reti wireless mesh IEEE 802.11s

Lo standard IEEE 802.11s [1] descrive un framework completo per la realizzazione di una rete wireless mesh con tecnologia IEEE 802.11 possedente tutti i requisiti minimi di funzionalità. Nato nel maggio 2004, il Task Group S ha redatto da allora diversi documenti, alcuni anche di difficile reperibilità, e "redline", ossia documenti ufficiali che riportano solo le differenze rispetto alle versioni precedenti delle decisioni prese nei vari incontri. L'ultimo documento approvato e a disposizione è il draft P802.11s™/D1.01 del marzo 2007 [2].
Lo standard distingue i dispositivi wireless in due categorie: nodi mesh, che

sono dispositivi capaci di supportare servizi di tipo mesh, e nodi che non supportano i servizi mesh, come le tradizionali stazioni 802.11. I primi sono chiamati Mesh Point (MP), e partecipano alla formazione e all'operatività della rete mesh. Un MP può essere sia un dispositivo dedicato di una certa infrastruttura o un dispositivo end-user capace di partecipare completamente alla formazione e alle operazioni della rete mesh. I nodi che forniscono i servizi di AP in modo aggiuntivo a quelli di un MP sono chiamati Mesh Access Point (MAP). Le stazioni standard si connettono ai MAP per ottenere l'accesso alla rete e non partecipano alla fornitura dei servizi mesh, come ad esempio il meccanismo di routing (Figura 1.1).

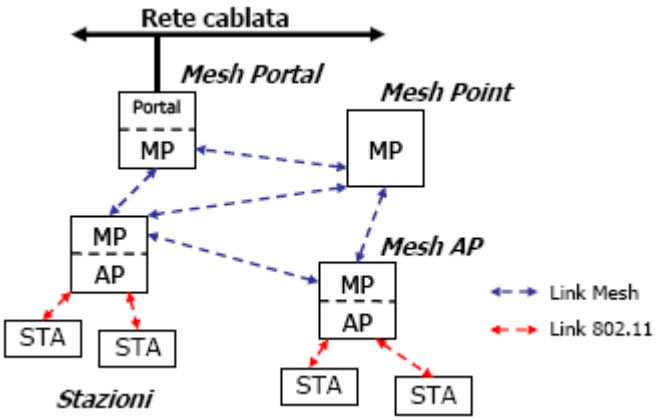


Figura 1.1 Architettura e componenti di una wireless mesh network secondo lo standard IEEE 802.11s.

Inoltre nella rete mesh spesso è presente almeno un nodo che fornisce la connettività verso reti esterne, chiamato Mesh Point Portal (MPP), il quale

integra sia le funzionalità di un MP sia quelle di un Portal. Nel caso esistano più MPP ciascun MP è tenuto ad identificare come Portal solo uno di essi; il protocollo definisce in modo basilare un meccanismo per la selezione di tale MPP.

1.1.1 Formato dei frame

Il formato generale di una trama di livello MAC comprende una serie di campi che compaiono in un ordine prefissato in tutti i tipi di frame (Figura 1.2).

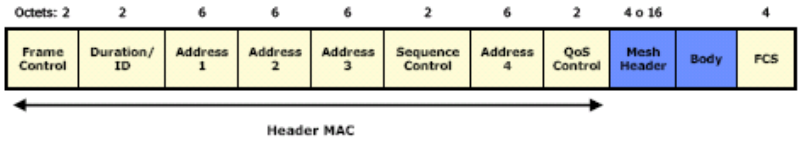


Figura 1.2 Formato generale della trama 802.11s

I primi tre campi (Frame Control, Duration/ID e Address 1) e l'ultimo campo (FCS) sono presenti in tutti i tipi di frame, mentre i campi Address 2, Address 3, Sequence Control, Address 4, QoS Control, Mesh Header e Frame Body sono presenti solo in determinati frame.

Il campo Frame Control specifica attraverso opportuni flag il contenuto effettivo degli indirizzi trasportati ed il Type (Tipo) ed il relativo Subtype (Sottotipo) di frame trasportato.

In particolare i frame utilizzati sono classificati in due Type: Management, che sono i frame classici dello standard IEEE 802.11 opportunamente

modificati con l'aggiunta di alcuni campi, e Extended, che sono invece i frame introdotti dallo standard (Figura 1.3).

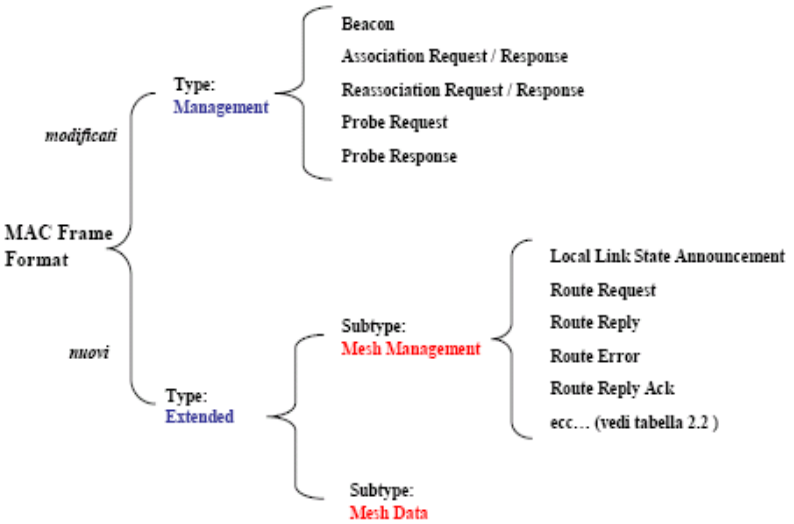


Figura 1.3 Schema dei frame utilizzati

Ritornando alla descrizione dei campi del formato generale, il campo Duration/ID, e quelli Sequence Control e FCS, hanno lo stesso formato ed utilità dei frame classici dello standard IEEE 802.11. Il campo Address 1 (RA) contiene l'indirizzo del MP che deve ricevere il frame. Il campo Address 2 (TA) è l'indirizzo del MP che ha trasmesso il frame. Il campo Address 3 (SA) contiene l'indirizzo del MP che ha originato il frame. Il campo Address 4 (DA) tipicamente rappresenta il MP destinatario del frame. Il campo QoS Control è un campo ancora non specificato, introdotto per la QoS. Il campo Mesh Header rappresenta la principale introduzione

dello standard IEEE 802.11s al formato classico delle trame IEEE 802.11: esso fornisce due indirizzi MAC addizionali, utilizzati principalmente per la comunicazione tra entità non mesh, e i campi TTL e Sequence Number.
Nei frame di tipo Management il campo Body contiene a sua volta diversi campi contenenti informazioni specifiche a seconda del frame. Poiché i frame di Management rappresentano delle modifiche ai frame classici, molte di queste modifiche si presentano sotto forma di aggiunta di determinati elementi detti Information Element (IE), che a loro volta contengono altri campi. Facendo chiarezza con un esempio, il frame di Management Beacon ha il formato generale descritto precedentemente mentre il campo Body ha il formato in Figura 1.4:

Figura 1.4 Campi contenuti nel campo body in un frame Beacon

Oltre i campi standard, è presente un IE, Mesh Capability (peraltro presente in tutti i frame di Management), il quale contiene a sua volta i seguenti campi (Figura 1.5):

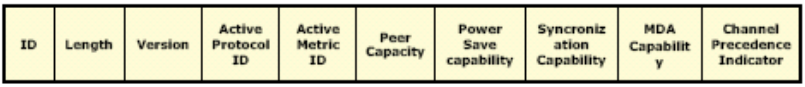


Figura 1.5 Formato dell'IE Mesh Capability

I frame Extended di Subtype Data (Mesh Data) presentano il formato generale; il campo Body contiene il payload.

Stessa cosa vale per i frame Extended di Subtype Management (Mesh Management), a cui manca però il campo QoS Control; il campo Body contiene, oltre un eventuale header per la gestione della sicurezza, un MMPDU (MAC Management Protocol Data Unit) chiamato Mesh Action Data Unit. Quest'ultimo contiene tre campi: il principale è il campo Mesh Action. Il Mesh Action Field è costituito a sua volta da tre campi, Category (posto sempre pari a 5), Action (che individua il tipo di messaggio specifico) e Mesh Action Details. Quest'ultimo contiene uno o più IE diversi a seconda del tipo di Mesh Action, specifico per ogni trama. I vari livelli di "incapsulamento" sono mostrati in Figura 1.6.

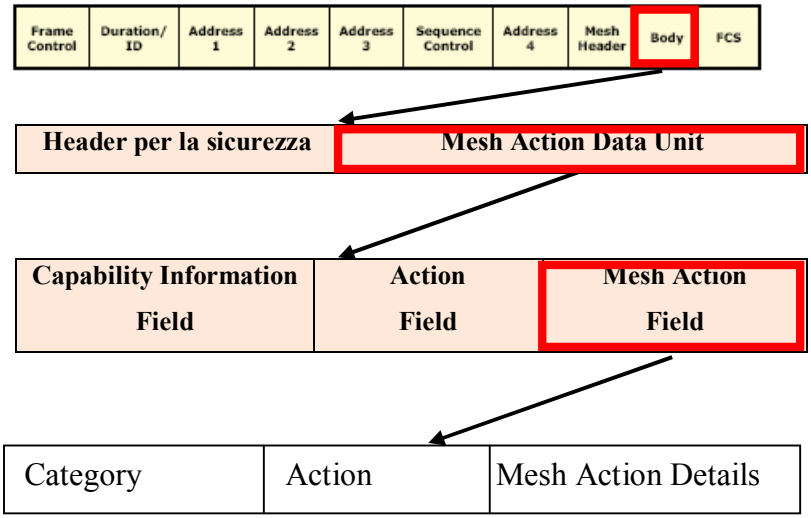


Figura 1.6 Frame di Mesh Management

1.2 Traffic Engineering

Il Traffic Engineering studia e applica quel complesso di tecniche che permettono un corretto controllo e distribuzione dei flussi di traffico in rete, con l'obiettivo di ottimizzarne l'uso delle risorse e di migliorarne le prestazioni. Applicato alle reti IP permette di superare i limiti dei protocolli di routing sulla gestione del traffico, apportando notevoli benefici sia in termini di prestazioni complessive che di economicità di gestione.

Per inserire in una rete le funzionalità di TE è necessario definire una serie di meccanismi.

Come primo passo è necessario definire e caratterizzare i flussi di traffico ai quali applicare le funzionalità di TE. Tali flussi risultano dalla sovrapposizione di più microflussi di traffico tutti appartenenti alla medesima CoS (Class of Service), ossia microflussi che la rete deve trattare allo stesso modo. Di tali flussi è necessario conoscerne gli attributi, ossia parametri che influenzano il modo con cui un flusso di traffico viene instradato. Gli attributi fondamentali possono essere parametri di traffico (come la banda media, l'ampiezza massima del burst), livello di priorità, modalità di recupero in caso di guasti sul percorso.

Il secondo ingrediente fondamentale è la conoscenza dettagliata della topologia fisica della rete. Essa è definita dall'insieme dei nodi (router, host) e dagli attributi dei collegamenti fisici (ad esempio banda trasmissiva, percentuale di banda utilizzabile). La topologia fisica della rete e gli attributi associati ai link devono essere noti ai nodi che implementano TE, i quali memorizzano queste informazioni nel TED (TE Database). Lo

scambio delle informazioni avviene utilizzando i messaggi dei protocolli di routing convenzionali di tipo Link State, opportunamente estesi.
Infine bisogna definire i meccanismi per la definizione e la realizzazione dei percorsi su cui far transitare i flussi di traffico. Per definire i percorsi (dinamici) è necessario un algoritmo di ricerca vincolata dei cammini che prende il nome di PCE (Path Computation Element). Per la realizzazione dei percorsi è necessario un protocollo di segnalazione tramite il quale informare i nodi appartenenti al path calcolato sulle caratteristiche dei flussi di traffico e scambiare altre informazioni che permettano la realizzazione del percorso.

1.3 MPLS

Come specificato dalla RFC 3031 [3], gli obiettivi di MPLS sono:

- funzionare con diverse tecnologie di livello 2;
- incrementare le prestazioni dei router;
- far evolvere il routing IP verso funzionalità di Traffic Engineering, svincolandosi dal paradigma del destination-based forwarding;
- rendere le reti IP più scalabili;
- supportare i modelli di QoS sviluppati in ambito IETF.

L'idea alla base di MPLS è introdurre nelle reti IP il concetto di "commutazione di etichetta", tipico delle reti connection-oriented, e di inserire in un mondo connection-less come quello IP il concetto di connessione virtuale. Ciò avviene associando a tutti i pacchetti un breve

identificativo di lunghezza fissa, la label (etichetta), che gli apparati di rete possono utilizzare per effettuare un instradamento veloce.

Una rete MPLS opera principalmente nel seguente modo (Figura 1.7):

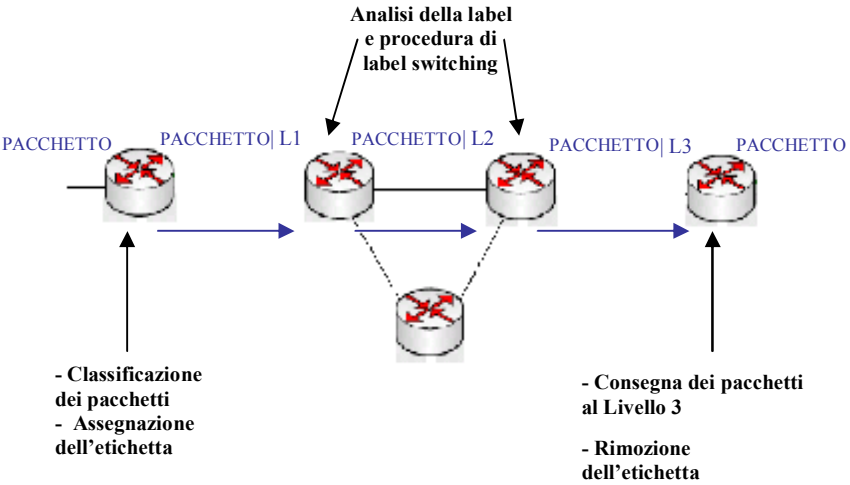


Figura 1.7 Schema di funzionamento di una rete MPLS

1. Il pacchetto ricevuto dal router di ingresso della rete MPLS viene dapprima classificato, ovvero assegnato ad una FEC (Forwarding Equivalence Class). Quindi viene etichettato e inoltrato verso il router successivo.
2. Sul router successivo, l'inoltro avviene sulla base del solo contenuto dell'etichetta: esso consulta una tabella che associa al valore della label le 'istruzioni' per l'inoltro; se il router non è l'ultimo del cammino, allora viene cambiato il valore

dell'etichetta secondo le istruzioni contenute nella tabella e quindi il pacchetto viene inoltrato al router successivo.

3. Quando il pacchetto arriva all'ultimo router del percorso, questo toglie l'etichetta ed inoltra il pacchetto sulla base dell'indirizzo IP di destinazione.

L'architettura dei router MPLS, chiamati LSR (Label Switching Router), segue la tradizionale separazione funzionale dei router convenzionali, in componente di controllo e componente dati.

La componente di controllo comprende sia i protocolli di routing, presenti in un router convenzionale, sia alcuni protocolli e meccanismi tipici di MPLS come un meccanismo per la definizione delle associazioni FEC-etichette (Label binding) e un protocollo di segnalazione per la distribuzione di tale associazioni (Figura 1.8).

La componente dati degli LSR è molto più semplice rispetto a quella di un router convenzionale in quanto la procedura di inoltro è basata sulla semplice commutazione di etichetta.

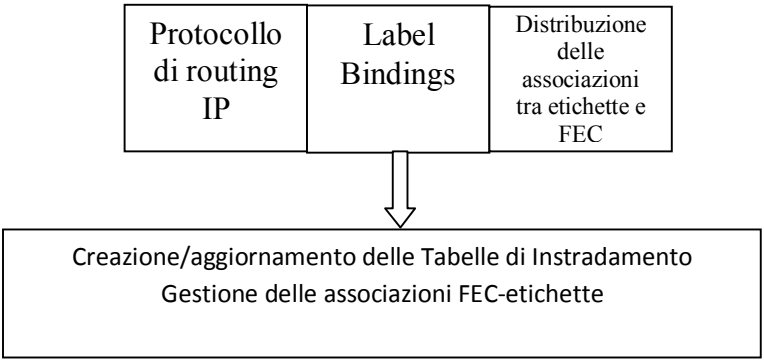


Figura 1.8 Componente di controllo di un LSR

In un LSR l'inoltro dei pacchetti avviene per mezzo di alcune tabelle:

- FTN (FEC-To-NHLFE): è la tabella presente sugli LSR d'ingresso della rete MPLS; associa ad ogni pacchetto entrante inserito in una determinata FEC un insieme di istruzioni (NHLFE, Next-Hop Label Forwarding Entry) per l'inoltro del pacchetto all'LSR successivo;
- ILM (Incoming Label Map): è una tabella presente negli LSR intermedi; associa ad un'etichetta entrante un insieme di NHLFE per l'inoltro del pacchetto al'LSR successivo;
- LIB (Label Information Base): è un database che contiene per ciascuna FEC le associazioni FEC-etichette necessarie alla costruzione delle tabelle FTN e ILM.

La connessione virtuale stabilita associando delle etichette ad un determinato percorso tra un LSR di ingresso e uno di uscita viene chiamata LSP (Label Switched Path).

1.3.1 Il Traffic Engineering con MPLS

L'entrata in campo di MPLS ha permesso di creare gli strumenti necessari per realizzare metodologie di TE efficienti, flessibili e scalabili, che trovano applicazione nelle reti reali.

In particolare, gli LSP MPLS verso una particolare destinazione:

- sono regolati dai nodi origine degli LSP, che hanno il compito di segnalarli e di determinarli;
- seguono un percorso tra origine e destinazione che rispetti eventuali vincoli assegnati;

- possono avere associati dei parametri di traffico.

In una rete MPLS le funzionalità di TE descritte nel paragrafo precedente sono implementate in questo modo:

1) I flussi di traffico sono definiti, classificati in FEC e caratterizzati dagli LSR d'ingresso.
2) La topologia fisica della rete comprende gli LSR e i link che li connettono. Le informazioni sulla topologia sono scambiate dal protocollo di routing utilizzato e memorizzate nel TED.
3) I percorsi sono definiti dinamicamente dal PCE dell'LSR d'ingresso. Per la realizzazione dei percorsi è necessario un protocollo di segnalazione tramite il quale informare gli LSR del percorso sulle caratteristiche dei flussi di traffico e scambiare le associazioni FEC-etichette. Sono stati standardizzati due protocolli di segnalazione: RSVP-TE [4] (ReSerVation Protocol – Traffic Engineering) e CR-LDP [5] (ConstRaint based Label Distribution Protocol). Entrambi i protocolli hanno le stesse funzionalità e possono essere considerati equivalenti: tuttavia la RFC 3468 [6] ha stabilito, dopo un ampio sondaggio fra i costruttori di apparati, che quello che avrà un seguito sarà solo il primo.

1.4 Applicazioni del Label Switching alle reti wireless

Sono riportati in seguito gli articoli più interessanti, che descrivono alcune semplici architetture in cui è applicato il concetto di label switching.

1.4.1 ICF e DCMA

Questo paper [7] è stato presentato nel novembre 2006 da A. Acharya, S. Ganu e A. Misra. Esso si focalizza principalmente su un aspetto che degrada le prestazioni di una rete wireless multihop, ovvero l'inefficienza del packet forwarding. In un nodo wireless quest'operazione avviene generalmente a livello di rete, come è mostrato in Figura 1.9, che mostra un'implementazione convenzionale del software di base per il forwarding.

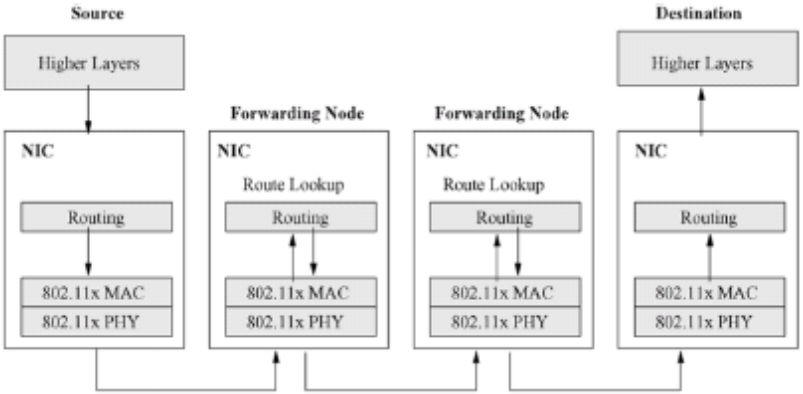


Figura 1.9 Tipico packet forwarding in una rete wireless multihop

Quest'approccio implica la ricezione di un pacchetto sull'interfaccia wireless, l'estrazione e il trasferimento del pacchetto al layer IP, il lookup della tabella di routing nella CPU dell'host per determinare gli indirizzi IP e MAC del next hop e il reincapsulamento del pacchetto in una nuova trama, operazione condotta nuovamente nella NIC dell'host, per essere trasmessa al next hop. Questo meccanismo introduce due diverse latenze. La prima è

legata al fatto che bisogna effettuare due tentativi di accesso al mezzo per la trasmissione del pacchetto. La seconda dipende dal tempo necessario per effettuare il lookup della tabella e dai tempi strettamente connessi alla CPU dell'host che esegue l'operazione.

L'obiettivo è velocizzare le operazioni di forwarding dei pacchetti attraverso l'impiego di due tecniche.

La prima tecnica consiste nel trasferire le funzionalità di forwarding a livello MAC, ossia compiere le operazioni di lookup della tabella direttamente nella NIC dell'host: tale tecnica prende il nome di ICF (Interface Contained Forwarding). Ciò avviene attraverso l'uso di label di lunghezza prefissata anteposte alle trame MAC: in ogni nodo la commutazione è effettuata in base al valore della label e del MAC Address dell'interfaccia che invia il pacchetto, le quali determinano la nuova etichetta uscente ed il MAC Address del next hop. Questo è proprio il concetto di commutazione di etichetta usato in MPLS; ogni NIC dovrà solo memorizzare una tabella contenenti queste quattro entry.

La seconda tecnica consiste nel definire un nuovo protocollo di accesso al mezzo che prende il nome di DCMA (Driven Cut-trough Multiple Access). Il DCMA consiste nel far susseguire direttamente alla ricezione di un pacchetto da un nodo in upstream la trasmissione verso il nodo in downstream. In questo modo si evita il duplice tentativo di accesso al canale previsto dal protocollo di accesso al mezzo 802.11, il DCF. Come si vede in Figura 1.10 (riferendosi al nodo B), nel meccanismo DCF la ritrasmissione del pacchetto comporta il ripetersi del "four-way handshake", che consiste nel susseguirsi di RTS/CTS/DATA/ACK. Invece nel DCMA attraverso un unico pacchetto ACK/RTS si conferma sia l'avvenuta ricezione dei dati, sia

la ‘prenotazione’ per la trasmissione del pacchetto verso il nodo in downstream.

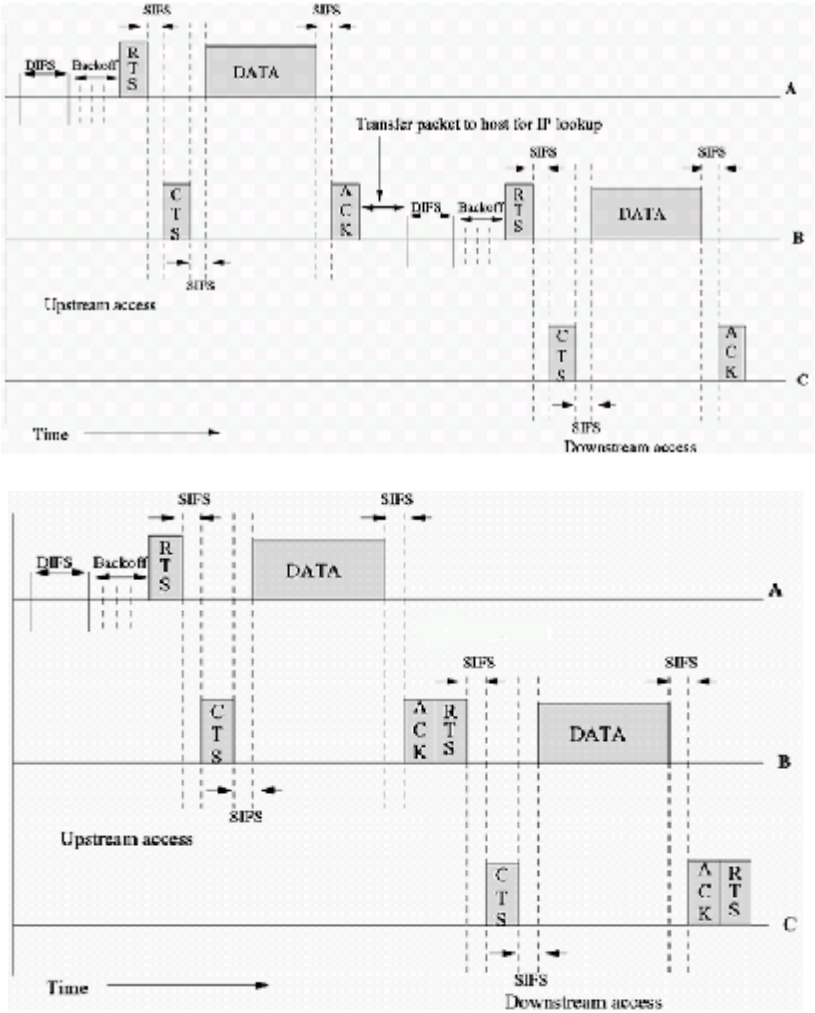


Figura 1.10 Multihop forwarding nel DCF (sopra) e nel DCMA (sotto)

Il paper tuttavia non descrive il protocollo di segnalazione da adottare per la distribuzione delle label (si consiglia di adattare quelli usati in MPLS), mostrando così una grave carenza. Inoltre la modifica al protocollo di accesso al mezzo comporta complesse modifiche dei driver della NIC dei nodi. Infine, non sono contemplati meccanismi per l'implementazione della QoS e di TE.

1.4.2 WALS

Questo paper [8] è stato presentato nel 2005 da H. Liu e D. Raychaudhuri. Esso si focalizza su una tecnica che viene utilizzata nelle reti wireless multihop per aumentare la probabilità che le trame trasmesse da un nodo siano correttamente ricevute da un altro nodo distante più hop. Questa tecnica prende il nome di multipath routing, e può essere utilizzata anche nelle reti mesh. Il multipath routing consiste nell'utilizzare più path tra due nodi A e J per la trasmissione delle trame dati: una trama viene così trasmessa sia sul percorso deciso inizialmente, che prende il nome di primary path, sia sugli altri path determinati (secondary path), aumentando la probabilità di corretta ricezione delle trame al nodo B. In alternativa si possono trasmettere le trame solamente sul primary path; gli altri path saranno di backup e verranno utilizzati qualora, per qualunque motivo, il path principale necessita di essere sostituito.
L'obiettivo è realizzare quindi un'architettura di rete in cui è utilizzato il multipath routing (ci si limita a soli due path, il primary e l'altro di backup). Essa sfrutta il concetto di label switching che è proprio di MPLS, e prende il nome di WALS (Wireless Ad-hoc Label Switching). Nelle WALS i path

vengono instaurati direttamente dal primo pacchetto di un flusso, identificato da tutti i pacchetti che hanno la stessa sorgente, lo stesso destinatario e una stessa Classe di Servizio (CoS). Esso trasporta principalmente:

- gli indirizzi MAC delle interfacce dei nodi, distanti un hop, che dovranno ricevere il pacchetto, sia sul primary che sul backup path; tali indirizzi prendono il nome di Primary MAC e Backup MAC.
- un identificativo unico del flusso stesso che costituisce la label.

A differenza di MPLS non viene quindi utilizzato un protocollo di segnalazione per la distribuzione delle etichette. La motivazione è nei frequenti cambiamenti della topologia di una rete wireless multihop dovuti all'alta mobilità dei nodi: l'adozione di un protocollo di segnalazione provocherebbe un forte overhead in termini di traffico dovuto ai continui setup e abbattimenti dei path. Con questo metodo inoltre si riduce il traffico di segnalazione e si abbassa il path setup delay.

Per creare il multipath si procede sostanzialmente in questo modo. Ogni nodo che riceve il primo pacchetto di un flusso proveniente da un nodo A e diretto a J esamina se l'indirizzo di una delle sue interfacce si trova nei due indirizzi trasportati. Se un'interfaccia figura come Primary MAC reinoltra il pacchetto verso due nuovi Primary MAC e Backup MAC, determinati dal protocollo di routing (che può essere qualsiasi) e crea una entry per quel flusso. Se invece figura come Backup MAC reinoltra il pacchetto solo verso il migliore next hop per raggiungere B, specificando attraverso un flag che proviene da un Backup MAC, e crea sempre una entry. Quest'ultimo

pacchetto instaurerà un Backup path che terminerà non appena si raggiunge un nodo che si trova sul Primary path. Un multipath di questo tipo prende il nome di locally disjoint multipath: come si nota,viene sfruttata la natura broadcast delle comunicazioni wireless. Un esempio è illustrato in Figura 1.11.

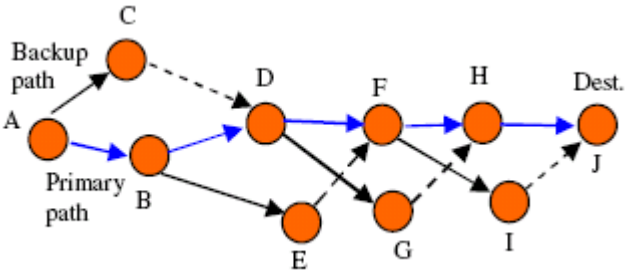


Figura 1.11 Locally disjoint multipath.

Ogni nodo si comporta da LSR e ha una label switching table. Notiamo, a differenza di MPLS, che la label è unica per tutti gli LSP, visto che essa è considerata come un ID del flusso.

Anche quest'architettura presenta degli svantaggi. Anche se implementa il concetto di CoS, manca di un meccanismo di controllo della QoS per i path stabiliti, basate ad esempio su soglie quali il tasso di perdita dei pacchetti oppure ritardo. Cioè, è previsto un meccanismo di recovery solo se un LSP "si rompe", ma non se subisce una degradazione. Inoltre quest'architettura può essere adottata solo per reti di ridotte dimensioni, visto che presenta problemi di scalabilità.

1.4.3 LSMR

Questo paper [9] è stato presentato nel 2007 da R. Wei, M.Wu e T. Yu.
Anche in questo articolo si cerca di realizzare un protocollo di routing multipath basandosi sul concetto di label switching, seguendo tuttavia un'altra strada. In particolare per la distribuzione delle label vengono utilizzati i messaggi del protocollo di routing: questo modo di distribuzione delle etichette è chiamato label implicit distribution. Poiché i protocolli di segnalazione delle label usati in MPLS instaurano i path on-demand, è necessario utilizzare un protocollo di routing anch'esso di tipo on-demand, come ad esempio AODV. I messaggi di AODV vengono così sia modificati per il trasporto delle etichette (RREQ e RREP), sia ridefiniti per poter trasportare le ulteriori informazioni di segnalazione. Il protocollo è chiamato LSMR (Label Switching Multipath Routing).
Più per le modalità attraverso cui viene istaurato il multipath, questo paper è stato citato per l'idea dell'utilizzo di un label implicit distribution. Infatti in questa tesi inizialmente si era proposta un'architettura in cui veniva utilizzato il protocollo HWMP [10], che è il protocollo mandatario definito dal TGs IEEE 802.11 per le reti mesh, per poter instaurare gli LSP su un path e per distribuire le etichette. In sostanza, quando un MP A voleva trovare un percorso per poter raggiungere un MP destinatario B di cui non conosceva la raggiungibilità, inviava un messaggio RREQ in broadcast annunciando la destinazione B. I MP a cui giungevano tali RREQ creavano degli stati, perché potevano essere possibili LSR del path, e reinoltravano in broadcast la richiesta. Solo la destinazione B rispondeva con un messaggio RREP emesso in unicast verso A. Il messaggio RREP conteneva le label: i

nodi che ricevevano questo messaggio completavano le entry precedentemente create, per cui la procedura di creazione dell'LSP poteva dirsi completata. Fra due MP potevano poi essere instaurati diversi path, ognuno dei quali relativi ad una CoS specifica.
Quest'idea è stata tuttavia abbandonata per diversi motivi. Il problema principale riguarda il fatto che questa tecnica è applicabile in modo preventivo, cioè ipotizzando che all'avvio si ha già un'idea di come potrebbe essere il traffico. In questo modo non è considerata l'evoluzione dinamica della rete. Inoltre non è possibile usare tecniche specifiche di TE, visto che HWMP non le implementa: per tale protocollo di routing i percorsi sono 'statici', cioè una volta decisi non sono modificati a meno della rottura di uno dei link costituenti il percorso.
Come si vede, per poter garantire il monitoraggio della QoS a determinati flussi è necessario usare delle tecniche che dinamicamente informano gli LSR della rete MPLS della situazione topologica dell'intera rete, cosa che nei paper descritti non avviene.
L'architettura proposta viene descritta nel prossimo capitolo.

1.4.4 Altri paper

Un protocollo che provvede al supporto della QoS end-to-end ad un determinato flusso è descritto in [11], e prende il nome di LSPQS (Label Switch Path with guaranteed QoS). La QoS è supportata in termini di banda, delay end-to-end e link loss ratio. LSPQS ha una fase di controllo che distribuisce le label, stima le risorse disponibili per hop e alloca determinate risorse per assicurare la QoS come richiesto dalla sorgente. Dopo la fase di

controllo segue la fase dati in cui i pacchetti sono classificati, assegnati ad una label (unica per tutti gli LSP del path) e instradati secondo il valore della label, registrata nei nodi intermedi in un'apposita struttura dati, la Label Forwarding Table. A causa della natura wireless del canale gli LSP possono rompersi oppure la QoS può essere violata: è previsto a tal scopo un meccanismo di recovery. L'architettura realizzata presenta tuttavia forti problemi di scalabilità.

In [12] è invece descritto un Internet draft, in cui è presentato il framework della tecnologia wireless MPLS (WMPLS). WMPLS rappresenta l'evoluzione di WATM (Wireless ATM), e descrive un modo per poter applicare MPLS alle reti MANET ed alle reti cellulari, in modo da supportare la QoS e servizi di TE. Tale draft non ha comunque avuto seguito. In [13] è riportata anche un'analisi prestazionale delle reti WMPLS.

Cap. 2 Architettura proposta

2.1 Obiettivi ed ipotesi

In questo capitolo verrà descritta l'architettura utilizzata per l'applicazione del MultiProtocol Label Switching alle reti Wireless Mesh. Ci si riferirà a quest'architettura come **TE - WMN**.

Come accennato precedentemente, gli obiettivi prefissati sono:

1. l'applicazione di tecniche di Ingegneria del Traffico (TE) nelle WMN, che permetteranno un corretto controllo e distribuzione dei flussi di traffico all'interno della mesh in modo da ottimizzare l'uso delle risorse e migliorare le prestazioni;
2. il supporto di garanzie di QoS (Qualità del Servizio) per ciascuna CoS (Classe di Servizio), sviluppate possibilmente in ambito 802.11e. I modelli introdotti hanno lo scopo di garantire dei valori minimi di qualità come ad esempio in termini di ritardo end-to-end o packet loss rate ai flussi di traffico appartenenti ad una delle CoS;
3. la velocizzazione dell'instradamento dei frame attraverso il concetto di label switching, che consisterà nell'associare a tutti i frame un breve identificativo di lunghezza fissa, la label (etichetta), che tutti i nodi mesh potranno utilizzare per poter effettuare una commutazione veloce.

Prima di passare alla descrizione dei blocchi che compongono l'architettura, è necessario descrivere le ipotesi e le scelte di progetto che hanno accompagnato il progetto di tale sistema.

La prima ipotesi consiste nel riferirsi a livello di piattaforma wireless allo standard IEEE 802.11. Ciò rappresenta ovviamente la soluzione più logica, poiché ci si riferisce allo standard più comune in ambito di reti wireless. Inoltre si considererà anche la possibilità di integrare 11s sviluppato da IEEE per la tecnologia wireless mesh.

La seconda ipotesi riguarda la dimensione del numero di nodi mesh costituenti la rete: si suppone che la WMN sia composta al massimo da 40-50 MP. In queste condizioni è possibile utilizzare un approccio di tipo distribuito, ovvero tutti i nodi mesh avranno lo stesso ruolo nei compiti 'affidati', che vanno dall'elaborazione delle informazioni sullo stato dei link all'instaurazione dei path. La presenza del Portal sarà sfruttata solo per compiti più marginali, rispetto al caso dell'adozione di un approccio di tipo centralizzato.

Descriviamo ora, entrando nei dettagli, le scelte di progetto, su cui ci si è basati.

Si è deciso di utilizzare un meccanismo che deve essere trasparente alle stazioni client 802.11 associate alla WMN. La rete sarà vista sostanzialmente come una mesh network su cui è applicato MPLS: i MP hanno le stesse funzionalità degli Label Switched Router (LSR). Ogni MP gestisce gli inoltri in base ai valori delle etichette, a loro volta determinate attraverso un meccanismo di segnalazione. I nodi mesh (MAP, MP, MPP) costituiscono così la rete MPLS; le STA invece si collegano ai MAP attraverso link standard 802.11.

I nodi mesh saranno implementati con una nuova interfaccia logica MAC indipendente dal tradizionale 802.11; i servizi invece che questi dispositivi ricoprivano all'interno delle normali mesh rimangono gli stessi. Le stazioni client che si connettono ai Proxy non dovranno installare driver o software aggiuntivi per potersi collegare: questo è il significato dato di trasparenza. Solo le STA intraprendono le comunicazioni end-to-end (i nodi mesh hanno solo funzionalità di "ripetitori" e di gestione del routing). La casistica delle possibili trasmissioni unicast comprende i casi "STA – STA" appartenenti alla stessa rete e "STA – nodo esterno alla rete mesh" (e viceversa).

Sorge spontanea la domanda di identificare a quale livello del modello a strati ISO/OSI si collocherà la rete TE - WMN. Si sa che l'architettura MPLS (quella definita dalla RFC 3031) non è collocabile a livello 3 in quanto mancano a MPLS le funzionalità di routing e indirizzamento che il modello ISO/OSI assegna a questo livello; d'altra parte non può collocarsi nemmeno a livello 2 poiché è indipendente dal protocollo a livello 2 utilizzato. Il discorso è simile anche per la rete Mesh MPLS, con evidenti modifiche. Innanzitutto questa nuova architettura è progettata ad hoc per reti wireless mesh: alcune funzionalità sono state introdotte sulla base delle caratteristiche delle WMN (ad esempio la modifica dei campi d'indirizzamento dei frame dati, come sarà visto successivamente). Tuttavia non si interviene ad esempio sul meccanismo di accesso al mezzo dell'802.11 (CSMA/CA) e su altre funzionalità MAC: in conclusione, la rete Mesh MPLS si avvicina di più al livello 2 rispetto ad MPLS. Per quanto riguarda la vicinanza al livello 3, dall'altra parte a quest'architettura mancano le funzionalità di routing e indirizzamento che il modello ISO/OSI assegna a livello 3. Inoltre, dall'esterno questa rete sarà caratterizzata da un

unico prefisso di rete poiché appartenente ad un unico dominio amministrativo. L'MPP ha così funzionalità di router, ed ha le stesse funzionalità di un LSR d'uscita nel caso in cui un nodo della rete mesh in figura voglia comunicare con un nodo esterno. Perciò l'architettura TE – WMN non è nemmeno collocabile a livello 3: possiamo concludere collocando la rete Mesh MPLS in una sorta di livello 2.5 (Figura 2.1):

IP	Livello 3
TE - WMN	**Livello 2.5**
IEEE 802.11s/e	Livello 2

Figura 2.1. Pila Protocollare

2.2 Architettura TE - WMN

In figura 2.2 è possibile visualizzare l'architettura di un Mesh Point che utilizza la soluzione MPLS over WMN.

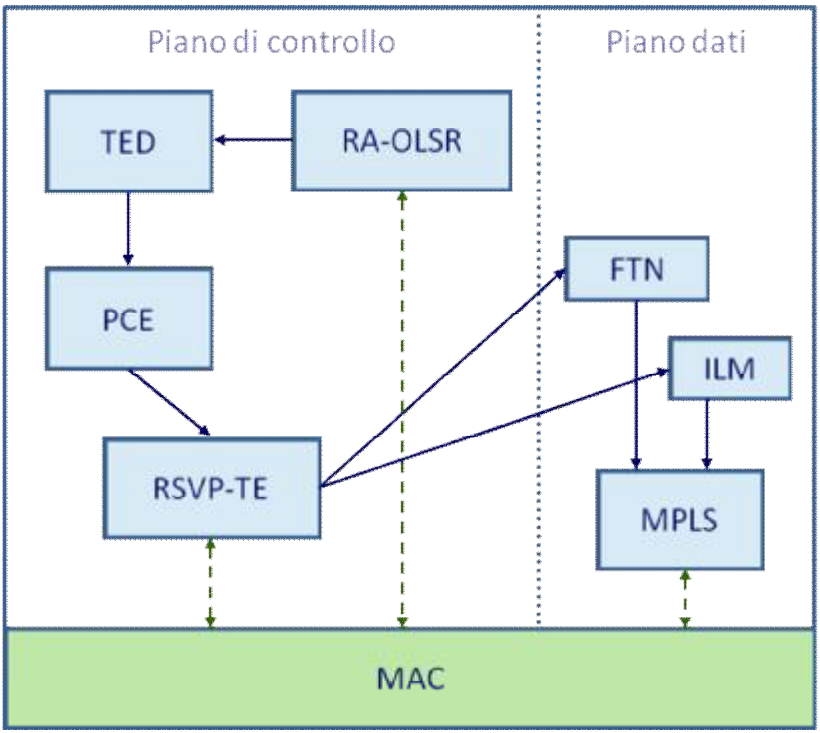


Figura 2.2. Architettura di un MP

E' possibile distinguere i blocchi che compongono l'architettura in quelli appartenenti al piano di controllo e quelli appartenenti al piano dati. Il piano di controllo comprende i blocchi che si occupano dei meccanismi di segnalazione, instaurazione e gestione dei path su cui saranno trasportati i flussi dati. Il piano dati comprende invece i blocchi che intervengono direttamente sui frame dati che viaggiano sui path instaurati.

Gli elementi che compongono il piano di controllo sono tre:

1. *Protocollo di routing*

2. *Path Computation Element*
3. *Protocollo di segnalazione*

Saranno ora introdotti i blocchi che costituiscono tali elementi, alcuni dei quali verranno poi descritti dettagliatamente nei capitoli successivi della tesi.

2.2.1 Protocollo di routing

Il protocollo di routing ha il compito di diffondere all'interno della rete informazioni su:

- Topologia della rete, ovvero fornire un elenco dei MP che compongono la rete e dei nodi vicini;
- Stato dei collegamenti;
- Metriche dei collegamenti.

Il protocollo di routing si appoggia ai servizi del livello MAC, visto che queste informazioni saranno trasportate nei frame di management dell'802.11s.

Sono stati definiti numerosi protocolli di routing per reti wireless mesh. Senza entrare nei dettagli, i vari algoritmi in circolazione possono essere classificati [14] in protocolli di tipo link-state o distance-vector, di tipo gerarchico o flat (o distribuito), di tipo proactive, on-demand oppure ancora ibrido.

Il protocollo di routing che si è scelto di usare è una versione di OLSR detta **RA-OLSR** (Radio Aware - Optimized Link State Routing). Il motivo di questa scelta risiede nelle caratteristiche base di OLSR (ideato per reti wireless ad hoc). Esso è un protocollo di tipo link-state, flat, proactive.

Queste caratteristiche sono quelle giuste per poter adottare un approccio di tipo distribuito, ipotesi di lavoro descritta nel paragrafo precedente; inoltre, questo protocollo offre un meccanismo per la riduzione del traffico di controllo all'interno della WMN. La versione Radio Aware differisce da quella base principalmente per l'introduzione di meccanismi per la gestione delle STA e per la possibilità di trasportare metriche di tipo radio aware.
Ovviamente questo protocollo di routing sarà opportunamente modificato e riadattato per poter offrire funzionalità alle reti TE – WMN.
Le informazioni trasportate dal protocollo di routing saranno memorizzate nel **TED** (Traffic Engineering Database), che contiene in particolare le informazioni di Traffic Engineering che saranno utilizzate poi dal Path Computation Algorithm.
RA-OLSR e TED saranno descritti nel capitolo 3.

2.2.2 Path Computation Element

Il **PCE** path computation element è il componente che effettua il calcolo del percorso (o dei percorsi) tra un nodo mesh sorgente ed un nodo mesh destinatario, in funzione delle specifiche di Traffic Engineering. Il PCE è presente in ogni nodo mesh (approccio distribuito) ed utilizza le informazioni del TED. I percorsi calcolati saranno poi instaurati dal protocollo di segnalazione.
Esistono diversi Path Computation Algorithm utilizzati nelle reti cablate: tuttavia, per poter essere utilizzati nelle WMN, può essere necessario un adattamento di tali algoritmi alle caratteristiche tipiche di questa tecnologia.

In questa tesi si sono implementati quattro Path Computation Algorithm, che saranno descritti a partire nel capitolo 4.

2.2.3 Protocollo di segnalazione

Il protocollo di segnalazione è l'elemento utilizzato per l'instaurazione dei percorsi, denominati LSP (Label Switched Path) tra un nodo mesh d'ingresso ed uno di uscita. Il compito del protocollo è quello di distribuire le associazioni fra etichetta d'ingresso ed etichetta di uscita per ogni MP del percorso.
Come citato nel capitolo precedente, il protocollo di segnalazione adottato in ambito IETF è RSVP-TE. Tuttavia questo protocollo è stato progettato per poter lavorare a livello IP: è necessario perciò un adattamento per poter lavorare a livello 2, in conformità alla seconda scelta progettuale descritta nel paragrafo precedente. L'adattamento consiste nel modificare gli indirizzi ed il formato dei messaggi di segnalazione.
Gli LSP sono instaurati sui percorsi calcolati dal PCE; i messaggi di segnalazione RSVP-TE sono trasportati nei frame di management dell'802.11s.

2.2.4 Piano dati

Il piano dati comprende il blocco **MPLS,** con le strutture dati associate **ILM** (Incoming Label Map) e **FTN** (FEC To NHLFE).

MPLS consente di inoltrare in maniera rapida le trame grazie a commutazioni basate su etichette anziché sugli indirizzi: in questo modo si evita il richiamo dei classici algoritmi di inoltro che prevedono il lookup delle tabelle di routing nei singoli nodi. Le etichette sono distribuite dal protocollo di segnalazione (RSVP-TE) ed hanno valenza locale. Ognuna di loro individua un flusso dati omogeneo appartenente ad una stessa FEC (Forwarding Equivalence Class). Ogni FEC comprenderà flussi di traffico aventi gli stessi:

- indirizzo MP sorgente;
- indirizzo MP destinatario;
- Classe di Servizio.

Per eseguire gli inoltri sono utilizzate la ILM e la FTN, che vengono popolate dal protocollo RSVP-TE. Nella ILM sono riportate le associazioni FEC – etichetta d'ingresso per ogni LSP instaurato. Nella FTN sono riportate le associazioni FEC – regole, le quali descrivono le procedure per l'elaborazione della trama (ad esempio l'assegnazione della label d'uscita). Tali regole sono chiamate NHLFE (Next Hop Label Forwarding Entry). Si sottolinea di nuovo come queste due tabelle siano diverse da quelle usate nell'MPLS tradizionale, visto che nel nostro contesto si sta lavorando a livello 2: ad esempio gli indirizzi memorizzati saranno di tipo MAC e non IP.

2.3 Formato delle trame

Le trame utilizzate dalla rete Mesh MPLS seguono la stessa classificazione utilizzata nell'802.11s (Figura 2.3):

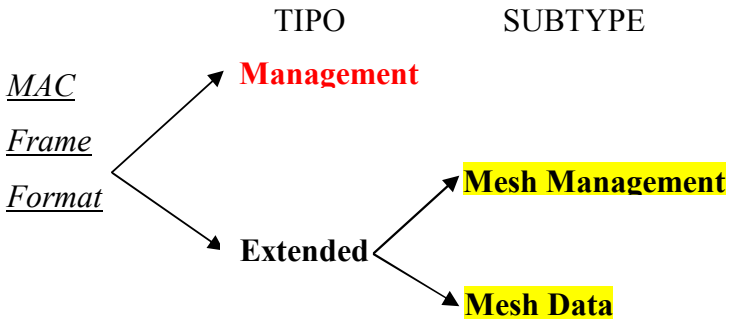


Figura 2.3 Classificazione delle trame

Le informazioni del protocollo di routing e del protocollo di segnalazione sono inserite nelle trame MAC di tipo Extended e di subtype Management, in particolare nel campo *Body* come Information Element: tali tipi di trame non subiscono quindi modifiche nel loro formato.

Le trame di tipo Extended e di subtype Data invece sono modificate. Queste nuove trame (che costituiscono le trame dati) hanno il seguente formato (Figura 2.4):

Figura 2.4 Nuova trama dati

L'header fisico comprende campi di controllo come FCS e durata della trama mentre il campo dati contiene il payload.
I campi RA, Label e DA sostituiscono i campi Address 1, Address 2, Address 3, Address 4 e QoS Control della trama 802.11s ed hanno i seguenti significati:

- il campo RA rappresenta l'indirizzo MAC dell'interfaccia del nodo mesh verso cui è trasmessa la trama dati;
- il campo Label rappresenta l'etichetta associata all'LSP;
- il campo DA rappresenta l'indirizzo MAC della stazione client destinataria della trama end-to-end.

2.4 Metriche dei collegamenti

Affinché siano fornite garanzie di QoS ai flussi di traffico dati è necessario definire delle metriche. Queste sono di tipo radio-aware, ovvero tengono conto delle caratteristiche radio del canale wireless di ogni link tra nodi mesh; tali informazioni sono trasportate dal protocollo di routing attraverso opportuni messaggi di segnalazione, come sarà visto nel capitolo successivo.
Nel definire le metriche sono stato effettuate due ipotesi:

1) lo stato della rete è temporaneamente statico: ciò vuol dire ipotizzare che i valori delle metriche (come ad esempio il ritardo medio del collegamento) rimangano costanti per tempi sufficientemente grandi in modo da escludere continue operazioni di ricalcalo dei percorsi.

2) flussi di traffico di tipo CBR (ipotesi molto semplificativa)

Le informazioni utili sullo stato dei collegamenti sono:

1) Bit Rate Fisico: rappresenta il rate a cui possono essere spedite le trame fisiche.
2) Banda Riservata per CoS: rappresenta la banda occupata per CoS su un determinato link. Questa rappresenta la metrica più critica, perché da questo valore ogni nodo mesh dovrà ricavare la banda allocabile per ciascuna CoS. La banda occupata può essere espressa in termini di bit al secondo oppure come tempo di occupazione del canale, e dipende dal tempo di accesso al canale e dalla banda riservata alle altre Cos.
3) Ritardo medio di trasmissione per CoS: rappresenta il ritardo medio a cui sono sottoposti i frame appartenenti ad una determinata CoS. Questa metrica potrebbe essere ricavata attraverso una stima del RTT, ed è importante per applicazioni di tipo real-time.
4) Jitter per Cos: rappresenta la variazione del ritardo sperimentato dai flussi appartenenti alla CoS specificata. Questo parametro è importante per applicazioni di tipo VoIP, e può essere ricavato sempre dalla stima del RTT.
5) Link Loss Ratio per CoS: rappresenta il tasso di perdita dei pacchetti su quel determinato link. Un semplice algoritmo per il calcolo del Link Loss Ratio potrebbe essere il rapporto (inverso) fra il numero di pacchetti unicast trasmessi in un certo intervallo di tempo e il numero di ACK relativi ai pacchetti trasmessi giunti al nodo stesso su quel determinato link. Una metrica più efficiente è

l'ETX (Expected Transmission Count) [8], definita come il numero medio di tentativi di trasmissione a livello MAC necessari per una corretta consegna di una trama su un link.

Tutte queste metriche saranno memorizzate nel TED; affinché vengano poi utilizzate dal PCE per il calcolo del path migliore per una determinata CoS, è necessario capire in che modo queste metriche devono essere combinate.

Come descritto in [15], dati la metrica *d(i,j)* associata al link *(i,j)*, e il path *p=(i,j,k,..,l,m)*, la metrica *d* è:

- additiva, se $d(p)= d(i,j)+ d(j,k)+\ldots+ d(l,m);$
- concava, se $d(p)= min\ [d(i,j);\ d(j,k);\ldots;\ d(l,m)];$
- moltiplicativa, se $d(p)= d(i,j)* d(j,k)*\ldots* d(l,m);$

Se la metrica *d* non rientra in questa classificazione, allora $d(p)= F\ [d(i,j);\ d(j,k);\ldots;\ d(l,m)]$. Assecondando uno schema abbastanza diffuso, possiamo supporre che le metriche relative ad un link siano indipendenti da quelle relative ad un altro link.

Bisogna capire adesso a quale categorie appartengono le cinque metriche introdotte. Il bit rate fisico e la banda disponibile (che è la differenza fra il bit rate fisico di un collegamento e la banda allocata) sono delle metriche concave: infatti definiamo la banda di un path come il minimo delle bande residue di tutti i link del path. Il ritardo medio di trasmissione è una metrica additiva, dato che il ritardo di un path è pari alla somma dei ritardi dei suoi link. La probabilità di corretta ricezione del pacchetto è una metrica invece moltiplicativa: la probabilità di ricevere correttamente un pacchetto al nodo destinatario è pari al prodotto delle probabilità di corretta ricezione dei singoli link (quest'ultime si ricavano dalla differenza fra l'unità e il link loss ratio). Infine bisogna caratterizzare il jitter. Questa è sicuramente la metrica

più difficile da caratterizzare. L'idea è quella di identificare il jitter come la deviazione standard del ritardo con cui un pacchetto può essere trasmesso. Con questa assunzione, poiché i jitter dei vari link sono fra loro indipendenti, il jitter totale può essere calcolato come la radice quadrata della somma dei quadrati dei jitter dei link che appartengono al percorso.

Quando un PCE dovrà calcolare un path tra due nodi, la prima operazione compiuta sarà il prune dei link che non rispettano i requisiti sulle metriche concave (per esempio la banda): tale operazione è compiuta allo stesso modo da qualunque tipo di path computation algorithm. Effettuata quest'operazione, il PCE calcolerà il miglior path verso un nodo destinatario tendendo conto delle altre metriche non di tipo concavo (ritardo, jitter e perdite).

In ogni messaggio di segnalazione riguardante la topologia della rete dovranno quindi essere trasportate le informazioni relative a tutte le metriche. Nel caso di metriche distinte per le quattro CoS definite in ambito 802.11e avremo fino a 17 metriche, una per il bit rate fisico e quattro per ciascuna CoS..

Ogni MP dall'altra parte dovrà eseguire le operazioni di classificazione dei flussi provenienti dalle stazioni, assegnando i flussi ad una delle quattro CoS. Ogni CoS sarà caratterizzata da determinate **FlowSpec** (specifiche del flusso), uguali in tutta la rete mesh. In base ai FlowSpec e alle metriche memorizzate nel TED il PCE calcolerà per un flusso appartenente ad una determinata FEC il best path per raggiungere un certo MAP destinatario.

Cap. 3 RA-OLSR e TED

In questo capitolo verranno descritti i blocchi che costituiscono la parte di routing dell'architettura mesh MPLS, ovvero il protocollo **RA-OLSR** e il database **TED.**

Lo studio di RA-OLSR è affrontato seguendo il documento P802.11s™/D1.01 della IEEE [2], citando nel corso della tesi le modifiche apportate. In particolare nel draft esiste una sezione, intitolata "Information Repositories" (clausola 11A.7.4) dove sono descritte le strutture dati in cui sono memorizzate le informazioni trasportate. Si è deciso in questa tesi di utilizzare l'insieme dei set di informazioni come TED, modificando le tabelle originarie per poter memorizzare le metriche introdotte nel capitolo precedente.

3.1 Introduzione su RA-OLSR

RA-OLSR (Radio Aware Optimized Link State Routing) è un protocollo di routing utilizzato nelle reti wireless mesh per la selezione dei percorsi. Esso deriva dall'OLSR (Optimized Link State Routing), descritto dalla RFC 3626 [16], con estensioni (opzionali) derivanti dal FSR (Fisheye State Routing).

Il protocollo è di tipo proactive, distributed e link-state, e ha il compito di stabilire e mantenere le route ottime calcolate basandosi su una metrica

predefinita; ogni MP ha un meccanismo per determinare la metrica di ogni link con i suoi vicini.

Per ridurre il flooding delle informazioni sulla topologia della rete e delle informazioni di controllo, RA-OLSR adotta il seguente approccio: solo i MP settati come MPR (Multipoint Relays) da un nodo ritrasmettono in broadcast i messaggi provenienti dal nodo stesso. Inoltre sono adottati anche meccanismi di association discovery e mantenaince per supportare STA non mesh sia interne (associate ai MAP) che esterne (comunicate dal MPP). Nelle comunicazioni tra STA, RA-OLSR setta un path tra i MAP Proxy a cui quelle stazioni sono associate attraverso informazioni di routing complementari.

Come in OLSR, RA-OLSR cerca di minimizzare il flooding: solo un insieme dei neighbors (vicini) del MP in considerazione è usato dal MP per il forwarding degli information elements. Gli MPR sono selezionati in modo che un messaggio di broadcast, inoltrato da questi MPR possa raggiungere tutti i 2-hop neighbors (vicini a due salti) dell'MP in considerazione, che prende il nome di MPR Selector. Le informazioni richieste per eseguire la selezione dgli MPR sono acquisite attraverso lo scambio periodico dei messaggi di HELLO. Inoltre, RA-OLSR richiede solo un parziale flooding dello stato dei link per provvedere alla determinazione delle route. Il minimo set di informazioni link state richiesto è che tutti i MP selezionati come MPR dichiarino i link ai loro MPR Selector.

Diversamente dalle tradizionali reti mobili ad hoc dove tutti i MP partecipano direttamente nelle procedure di routing, nelle WMN, le STA che non supportano servizi mesh partecipano indirettamente alle procedure

di routing attraverso i loro MAP o MPP associati. Per il supporto delle associazioni delle STA ai MAP, RA-OLSR rispetto ad OLSR prevede due ulteriori strutture dati - Local Association Base (LAB) e Global Association Base (GAB) – che intervengono nel processo di costruzione della tabella di routing.

Il protocollo non richiede trasmissioni affidabili degli Information Element: ogni MP manda IE periodicamente. Ogni IE contiene un sequence number che è incrementato ad ogni emissione, in modo da poter distinguere messaggi recenti da quelli obsoleti.

3.1.1 Terminologia

TERMINOLOGIA	**DESCRIZIONE**
Main Address	MAC Address identificativo di un MP, se quest'ultimo ha più interfacce radio
Originator MP	MP che spedisce il messaggio
Originator Address	Main address del MP che ha spedito il messaggio
Sender Interface	Interfaccia su cui il messaggio è stato ultimamente trasmesso
Receveing Interface	Interfaccia su cui il messaggio è stato ricevuto

Tabella 3.1. Terminologia adottata

3.2 Formato dei frame e degli IE

Gli RA-OLSR Information Element sono trasportati nei frame RA-OLSR. Il formato dei frame RA-OLSR usa il formato Action frame body; questi frame sono trasmessi da un MP verso un altro MP in maniera unicast (individually addressed) o in broadcast verso i vicini (mannergroup addressed). Il formato dei frame è riportato in figura 3.1.
I campi Category ed Action sono settati in valori definiti dal draft D1.01.

Octets:1	1	Variable
Category	*Action*	*One or more RA-OLSR elements*

Figura 3.1. Formato dei frame

Gli RA-OLSR elements hanno il formato riportato in figura 3.2.

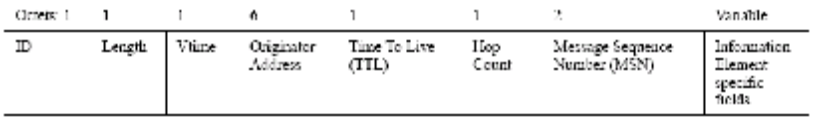


Figura 3.2. Formato degli Information Elements RA-OLSR

Il campo ID è settato secondo i valori definiti in [2] per gli IE.
Il campo Length indica la lunghezza dell'IE.
Il campo Vtime indica il tempo di validità (in secondi) durante il quale un MP considera valida l'informazione contenuta nell'IE dopo la sua ricezione.
Il campo Originator Address è il MAC Address dell'MP che ha originato questo IE.

Il campo TTL indica il massimo numero di volte che un IE può essere inoltrato. Prima che un IE venga inoltrato, il TTL è decrementato di uno. Un IE con TTL uguale a 0 o 1 non viene inoltrato in nessuna circostanza.
Il campo Hop Count indica il numero di hop che un IE ha attraversato. Prima che un IE venga inoltrato, l'Hop Count è incrementato di uno. L'Hop Count è inizializzato a zero.
Il campo MSN è un sequence number assegnato dall'originator MP, che assicura che ogni IE possa essere univocamente identificato all'interno della rete. L'MSN è incrementato di uno per ogni IE avente origine dal MP. Il wrap-around è trattato appositamente da un'apposita procedura .
Gli IE Specific Fields sono specifici per ogni IE e saranno descritti successivamente.

3.3 Processing e forwarding dei messaggi

Un MP può ricevere lo stesso messaggio (ovvero IE) diverse volte in una WMN. Perciò, ogni MP mantiene un Duplicate Set dove l'MP registra le informazioni sui messaggi ricevuti più recenti, con l'obiettivo di scartare i messaggi duplicati. Per ogni messaggio un MP registra un'entry (Duplicate Tuple) in questo modo (tabella 3.2):

(D_addr, D_seq_num, D_forwarded, D_iface_list, D_time)

CAMPI	DESCRIZIONE
D_addr	Originator Address del messaggio
D_seq_Num	MSN del messaggio
D_forwarded	Booleano che indica se il messaggio è stato già forwardato
D_iface_List	Lista degli indirizzi delle interfacce su cui il messaggio è stato ricevuto
D_time	Lifetime della entry

Tabella 3.2. Duplicate Set

Alla ricezione di un frame RA-OLSR, un MP esamina ognuno degli IE inclusi ed esegue inoltre le seguenti task:

a. Se il frame RA-OLSR non contiene messaggi (ovvero IE) il frame sarà scartato.
b. Se il TTL dell'IE è minore o uguale a zero, o se l'Originator Address dell'IE è il main address del MP ricevente, quest'IE sarà scartato.
c. Condizione di Processing:
    1) Se esiste un'entry in Duplicate Set dove:
        D_addr = Originator Address AND
        D_seq_num = MSN
        il messaggio è già stato processato.
    2) Altrimenti il messaggio viene processato a seconda del valore del campo ID.
d. Condizione di Forwarding
    3) Se esiste un'entry in Duplicate Set dove:

D_addr = Originator Address, AND
D_seq_num = MSN, AND
l'indirizzo della Receveing Interface è in D_iface_list

Allora questo messaggio è già stato inoltrato.

4) Altrimenti il messaggio sarà considerato da inoltrare secondo le modalità specifiche del tipo di messaggio.

3.3.1 Default Forwarding Algorithm

Molti messaggi seguono una stessa procedura di forwarding che prende il nome di Default Forwarding Algorithm, che è il seguente:

1. Se la sender interface del messaggio non è in vicinanza simmetrica con l'MP, il messaggio non è inoltrato.
2. Se esiste in Duplicate Set un'entry dove:
   D_addr = Originator Address AND
   D_seq_num = MSN
   allora il messaggio sarà considerato per il forwarding se e solo se
   D_forwarded è False, AND
   la receveing interface non è inclusa in D_iface_list.
3. Altrimenti, se tale entry non esiste, il messaggio sarà considerato per il forwarding.

Se il messaggio non è considerato per il forwarding, l'algoritmo si ferma, altrimenti continua con questi step successivi:

4. Se la sender interface è un'interfaccia di un MPR Selector dei questo MP e il campo TTL è maggiore di 1, il messaggio sarà

inoltrato: la prima condizione implica in pratica che un nodo ritrasmette un messaggio se e solo se il nodo stesso è un MPR dell'Originator MP.

5. Se è verificato lo step 2, l'entry in Duplicate Set è aggiornata come segue:

   D_time = current time + DUP_HOLD_TIME

   Il MAC Address della receveing interface è aggiunta in D_iface_list

   D_forwarded è settato a TRUE se e solo se il messaggio è inoltrato secondo lo step 4.

   Altrimenti viene creata una nuova entry in Duplicate Set con:

   D_addr = Originator MAC Address

   D_seq_num = MSN

   D_time = current time + DUP_HOLD_TIME

   D_iface_list contiene l'indirizzo della receveing interface

   D_forwarded è settato a TRUE se e solo se il messaggio è inoltrato secondo lo step 4.

Se e solo se è verificato lo step 4, il messaggio sarà inoltrato e:

6. Il campo TTL del messaggio è decrementato di uno.
7. Il campo Hop Count del messaggio è incrementato di uno.
8. Il messaggio è inoltrato in broadcast su tutte le interfacce.

Il meccanismo di forwarding di RA_OLSR provvede anche ad evitare l'emissione sincronizzata degli information elements. Una trasmissione simultanea del traffico di controllo porterebbe a collisioni e conseguentemente alla perdita dei messaggi. Per evitare tale sincronizzazione, un MP sottrae all'intervallo di tempo dopo il quale un

messaggio dovrebbe essere generato un valore temporale pseudo-causale detto jitter.

3.4 TED

Attraverso lo scambio degli information elements RA-OLSR, ogni MP registra informazioni sullo stato della rete in apposite strutture dati, chiamate nel draft Information Repositories (set di informazioni), e che costituiscono nell'architettura mesh MPLS il TED.

3.4.1 Link Set

Un MP mantiene un Link Set dove sono registrate le informazioni sullo stato dei link. Ogni entry (Link Tuple) ha questa forma (tabella 3.3):

(L_local_iface_addr, L_neighb_iface_addr, L_time,
L_link_metric_1,…, L_link_metric_17)

CAMPI	DESCRIZIONE
L_local_iface_addr	Interfaccia sul MP locale
L_neighb_iface_addr	Interfaccia sull'MP remoto, con cui esiste un link simmetrico
L_time	Lifetime della entry
L_link_metric_1, …, L_link_metric_17	Costo del link relativo ad ognuna delle metriche introdotte

Tabella 3.3. Link Set

3.4.2 Neighbor Set

Un MP mantiene un Neighbor Set dove sono registrate le informazioni sullo stato dei vicini. Ogni entry (Neighbor Tuple) ha questa forma (tabella 3.4):

(N_neighb_main_addr, N_willingness)

CAMPI	DESCRIZIONE
N_neighb_main_addr	Main address del vicino
N_willingness	Intero, compreso tra 0 e 7, che specifica la willingness del vicino

Tabella 3.4. Neighbor Set

Successivamente sarà chiarito il significato di willingness.

3.4.3 Interface Association Set

Un MP mantiene un Interface Association Set dove sono registrate le informazioni sullo stato delle interfacce degli MP della rete. Ogni entry (Interface Association Tuple) ha questa forma (tabella 3.5):

(I_iface_addr, I_main_addr, I_time)

CAMPI	DESCRIZIONE
I_iface_addr	Indirizzo MAC dell'interfacciadi un MP
I_main_addr	Main address dell'MP
I_time	Lifetime della entry

Tabella 3.5. Interface Association Set

3.4.4 2-hop Neighbor Set

Un MP mantiene un 2-hop Neighbor Set dove sono registrate le informazioni sui 2-hop neighbor. Ogni entry (2-hop Tuple) ha questa forma (tabella 3.6):

(N_neighb_main_addr, N_2hop_addr, N_time)

CAMPI	DESCRIZIONE
N_neighb_main_addr	Main address di un vicino
N_2hop_addr	Main address di un vicino, che ha un link simmetrico verso N_neighbor_main_addr
N_time	Lifetime della entry

Tabella 3.6. 2-hop Neighbor Set

3.4.5 MPR Set

Un MP mantiene un MPR Set dove sono registrati i vicini che sono stati selezionati come MPR. Ogni entry (MPR Tuple) ha questa forma:

(MPR_main_addr)

ed è indicato il main address dell'MPR selezionato.

3.4.6 MPR Selector Set

Un MP mantiene un MPR Selector Set dove sono registrate le informazioni sui MPR Selector del MP. Ogni entry (MPR-selector Tuple) ha questa forma (tabella 3.7):

(MS_main_addr, MS_time)

CAMPI	DESCRIZIONE
MS_main_addr	Main address del MPR Selector
MS_time	Lifetime della entry

Tabella 3.7. MPR Selector Set

3.4.7 Topology Set

Un MP mantiene un Topology Set dove sono registrate le informazioni sulla topologia della rete. Ogni entry (Topology Tuple) ha questa forma (tabella 3.8):

(T_dest_addr, T_last_addr, T_seq, T_time,
T_link_metric_1,…,T_link_metric_17)

CAMPI	DESCRIZIONE
T_dest_addr	Main address di un MP, che può essere raggiunto in un hop dal MP con main address T_last_addr
T_last_addr	Un MP che può raggiungere T_dest_addr in un hop
T_seq	Sequence Number
T_time	Lifetime della entry
T_link_metric_1,…, T_link_metric_17	Costo del link relativo ad ognuna delle metriche introdotte. Se esistono più link, sarà scelto il link con il più alto bit rate fisico

Tabella 3.8. Topology Set

3.4.8 Local Association Base (LAB)

Ogni MP mantiene un Local Association Base (LAB) dove sono registrate le stazioni associate localmente a quel MP. Inoltre, per supportare un numero alto di stazioni, il LAB è suddiviso in sottostrutture dati denominate blocks: il LAB è l'unione di tutti i blocks. I blocks sono indicizzati tramite un intero (block index). Ogni entry (Local Association Tuple) ha questa forma (tabella 3.9):

(LAB_block_index, LAB_address_station, LAB_station_sequence_number)

CAMPI	DESCRIZIONE
LAB_block_index	Block index a cui appartiene la Local Association Tuple
LAB_station_address	MAC Address della stazione associata
LAB_station_sequence_number	Sequence number del frame di management che la STA ha mandato al MAP quando la STA si è associata al MAP stesso

Tabella 3.9. LAB

3.4.9 Global Association Base (GAB)

Ogni MP mantiene un Global Association Base (GAB) dove sono registrate le stazioni associate ai MP del'intera rete mesh. Il GAB è l'unione di tutti i

LAB di ogni MP della rete. Come per il LAB, per supportare un numero alto di stazioni, il GAB è suddiviso in sottostrutture dati denominate blocks: il GAB è l'unione di tutti i blocks, i quali sono indicizzati tramite un intero (block index).

Ogni entry (Global Association Tuple) ha questa forma (tabella 3.10):

(GAB_block_index, GAB_AP_address, GAB_station_address, GAB_station_sequence_number, GAB_expiration_time)

CAMPI	DESCRIZIONE
GAB_block_index	Block index a cui appartiene la Global Association Tuple
GAB_AP_address	MAC Address del MAP
GAB_station_address	MAC Address della STA associata a quel MAP
GAB_station_sequence_number	Sequence number del frame di management che la STA ha mandato al MAP quando la STA si è associata al MAP stesso
LAB_expiration_time	Lifetime della entry

Tabella 3.10. GAB

3.5 Multiple Interfaces

La relazione tra gli indirizzi delle interfacce e i main address dei MP a cui appartengono è definita attraverso lo scambio dei messaggi Multiple

Interface Declaration (MID). Ogni MP annuncia tramite questi messaggi periodicamente la lista delle sue interfacce agli altri MP della rete; questa informazione è utilizzata da ogni MP per il calcolo della routing table. Un MP che ha solo un'interfaccia non genererà messaggi MID. I campi Vtime e TTL saranno settati rispettivamente a MID_HOLD_TIME e 255.

I messaggi MID sono inoltrati dagli MPR in broadcast per essere diffusi in tutta la rete; la procedura di forwarding avviene secondo il default forwarding algorithm.

I messaggi MID servono a popolare le entry della Interface Association Set. Gli IE hanno il seguente formato (Figura 3.3):

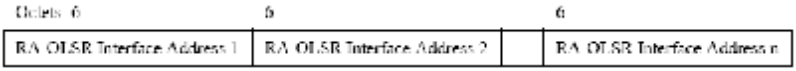


Figura 3.3 Formato del MID Information Element

I campi RA-OLSR Interface Address indicano gli indirizzi delle interfacce RA-OLSR del MP, mentre il main address è già indicato nel campo Originator Address.

Alla ricezione di un messaggio MID, sarà calcolato il tempo di validità dal campo Vtime del message header. Quindi, l'Interface Association Set sarà aggiornato in questo modo:

1. Se la sender interface del messaggio non è in vicinanza simmetrica con l'MP che ha ricevuto il messaggio, quest'ultimo non viene inoltrato.
2. Per ogni Interface Address contenuta nel MID:
    a. Se c'è un'entry in Interface Association Set dove:
        I_iface_addr = Interface Address AND

I_main_addr = Originator Address

Allora sarà aggiornata la entry:

I_time = current time + tempo di validità.

b. Altrimenti viene creata una nuova entry in Interface Association Set, in cui:

I_iface_addr = Interface Address

I_main_addr = Originator Address

I_time = current time + tempo di validità.

3.6 Messaggi di HELLO

I messaggi di HELLO sono scambiati solo tra MP vicini (TTL = 1), e servono a popolare il Neigbor set, il 2-hop Neighbor set, l'MPR set, l'MPR selector set, oltre che a trasportare informazioni di segnalazione per gli MPR. La lista delle interfacce dichiarate in un messaggio di HELLO è una lista di indirizzi di interfacce appartenenti ai vicini elencate in questo modo: per ogni entry in Link Set viene annunciata l' interfaccia contenuta nel campo L_neighb_iface_addr. Viene inoltre specificato se il MP, a cui appartiene l'interfaccia in L_neighb_iface_addr, è un MPR. L'Originator Address per un messaggio di HELLO è pari al main address del MP che genera il messaggio.

L'RA-OLSR HELLO Information Element ha il seguente formato (Figura 3.4):

Octets:1	1	1	2	6	4	…	4
Htime	Will-ingness	Link Code	Link Message Size	Neighbor Interface Address	Link Metric #1	…	Link Metric #17

…	6	4	…	4	…	1	2
…	Neighbor Interface Address	Link Metric #1	…	Link Metric #17	…	Link Code	Link Message Size

6	4	…	4	…	6	4	…	4
Neighbor Interface Address	Link Metric #1	…	Link Metric #17	…	Neighbor Interface Address	Link Metric #1	…	Link Metric #17

Figura 3.4 Formato dell'HELLO Information Element

Il campo Htime indica l'intervallo temporale di emissione degli HELLO usato dall'Originator MP su una particolare interfaccia.

Il campo Willingness indica la "volontà" dell'Originator MP di trasportare e inoltrare traffico di controllo per altri MP, ovvero la "volontà" del MP di essere selezionato come MPR. Un MP con willingness WILL_NEVER non sarà mai selezionato come MPR da qualunque MP. Invece un MP con willingness WILL_ALWAYS è sempre selezionato come MPR. Di default, un MP annuncia una willingness pari a WILL_DEFAULT. I valori associati a queste willingness sono riportati in [2].

Vengono quindi annunciate, per ogni vicino, le interfacce che hanno stabilito un link con l'Originator MP: l'insieme delle interfacce di un vicino annunciate nel messaggio di HELLO prende il nome di link message.
Il campo Link Code indica il link state. Un ulteriore link state, MPR_NEIGH, indica che il neighbor ha almeno un link simmetrico ed è stato selezionato come MPR dall'Originator MP.
Il campo Link Message Size indica la lunghezza in byte di ogni link message e misurato dall'inizio del precedente campo Link Code fino al successivo campo Link Code.
I campi Neighbor Interface Address indicano i MAC Address delle interfacce di un vicino MP.
I campi Link Metric indicano le metriche di ciascun link e rappresentano una modifica nel messaggio originario, che prevede invece solamente una metrica per ogni link.

3.6.1 Popolamento del Neighbor Set

Alla ricezione di un HELLO, un MP aggiorna il Neighbor Set come segue: se l'Originator Address è il N_neighb_main_addr di un entry del Neighbor Set, allora l'entry sarà aggiornata in questo modo: N_willingness = Willingness del messaggio di HELLO.

3.6.2 Popolamento del 2-hop Neighbor Set

Alla ricezione di un HELLO, sarà calcolato il tempo di validità dal campo Vtime del message header. Il 2-hop Neighbor Set sarà aggiornato in questo modo:

a) Per ogni indirizzo Neighbor Interface Address contenuto nell'HELLO (che rappresentano quindi gli indirizzi dei 2-hop neighbor):
    1) Se il main address del Neighbor Interface Address è pari al main address del MP che ha ricevuto l'HELLO, la Neighbor Interface Address sarà scartata (un MP non può infatti essere il 2-hop neighbor di sé stesso).
    2) Altrimenti, una nuova entry è creata:
        N_neighb_main_addr = Originator Address;
        N_2hop_addr = main address del 2-hop neighbor;
        N_time = current time + tempo di validità.

    Questa entry sostituisce una entry obsoleta con gli stessi N_neighb_main_addr e N_2hop_addr.

3.6.3 Popolamento del MPR Set

Come accennato precedentemente, ogni MP seleziona i suoi MPR tra il neighborood (vicinanza), ovvero l'insieme dei vicini ad un hop. I vicini che sono stati selezionati come MPR sono avvertiti con link code = MPR_NEIGH nei messaggi di HELLO.

Un MP ricava i suoi MPR in modo che attraverso loro è possibile raggiungere tutti gli strict 2-hop neighbor con la minima metrica; per strict 2-hop neighbor si intende un 2-hop neighbor che ha le seguenti caratteristiche:

- non è il nodo stesso;
- non è un vicino ad un hop del nodo stesso;
- è il vicino di un vicino del nodo stesso;
- ha una willingness diversa da WILL_NEVER.

L'MPR Set va ricalcolato ogni volta che sono rilevati cambiamenti nei vicini 'simmetrici', negli strict 2-hop neighbor 'simmetrici' oppure quando la metrica radio-aware supera una soglia prefissata.

Mentre non è essenziale che l'MPR Set sia minimo, è necessario che tutti gli strict 2-hop neighbor possano essere raggiunti attraverso gli MPR selezionati. Soddisfatta questa condizione, avere un MPR Set piccolo assicura basso overhead del traffico protocollare; se invece l'MPR Set coincide con l'intero neighbor set, ci si riduce all'avere un classico protocollo di routing link-state.

L'algoritmo di selezione degli MPR previsto in OLSR non tiene conto di metriche di tipo radio-aware. Esso calcola l'MPR Set con la minima cardinalità e perciò link con basse metriche radio-aware (che corrispondono ai link 'migliori') possono essere omessi. In RA-OLSR invece nell'algoritmo di selezione degli MPR vengono considerate le metriche radio-aware dei link: nella selezione degli MPR, sono preferiti i link con migliori metriche radio-aware.

Il draft D1.01 descrive un algoritmo di selezione degli MPR [2] che non viene riportato: è possibile adottare altri algoritmi o miglioramenti, come citato nello stesso draft .

3.6.4 Popolamento del MPR Selector Set

L'MPR Selector Set di un MP è popolato dai main address degli MP che hanno selezionato l'MP stesso come MPR. La selezione degli MPR è segnalata attraverso i messaggi di HELLO:
Alla ricezione di un messaggio di HELLO, se un MP trova indirizzi di proprie interfacce nella lista con Link Code = MPR_NEIGH, sarà calcolato il tempo di validità dal campo Vtime del message header. Quindi l'MPR Selector Set sarà aggiornato come segue:

a) Se non esiste un'entry in MPR Selector Set con:

   MS_main_addr = Originator Address

   allora una nuova entry è creata con:

   MS_main_addr = Originator Address

b) L'entry (vecchia o nuova) con:

   MS_main_addr = Originator Address

   è modificata come segue:

   MS_time = current time + tempo di validità.

3.6.5 Cambiamenti del neighborhood e del 2-hop neighborhood

Un cambiamento nel neighborhood di un MP si ha quando:

- l'L_time di un'entry in Link Set scade: ciò è considerato come perdita di vicinanza se il link descritto dalla entry scaduta era l'unico link con il MP con interfaccia riportata in L_neighb_iface_addr;
- una nuova entry viene inserita in Link Set: ciò è considerato come nuova vicinanza se precedentemente non c'era nessuna entry in Link Set che descriveva un link con MP con interfaccia riportata in L_neighb_iface_addr.

Un cambiamento nel 2-hop neighborhood si ottiene quando una entry in 2-hop Neighbor Set scade o si cancella come descritto nel sottoparagrafo 3.6.2.

Quando avvengono cambiamenti nel neighborhood o nel 2-hop neighborhood, è effettuata la seguente procedura:

- in caso di perdita di vicinanza, tutte le entry in 2-hop Neighbor Set con N_neighb_main_addr = main address del vicino sono cancellate;
- in caso di perdita di vicinanza, tutte le entry in MPR Selector con MS_main_addr = main address del vicino sono cancellate;
- l'MPR Set è ricalcolato quando un nuovo vicino appare o si cancella oppure quando avviene un cambiamento nel 2-hop neighborhood;

- un nuovo messaggio di HELLO è mandato quando cambia l'MPR Set.

3.7 Topology discovery

Il link sensing ed il neighbor detection del protocollo di base offrono ad ogni MP l'elenco dei vicini con cui poter comunicare direttamente. Parte delle informazioni date dal link sensing e dal neighbor detection sono inondate da ogni MP nell'intera rete attraverso il flooding controllato dagli MPR e usate per la costruzione delle route.

In ogni MP le route sono costruite (in mesh MPLS dal PCE) attraverso la conoscenza dei link con i vicini e dei link annunciati dagli altri MP, che prendono il nome di advertised link; i neighbor annunciati prendono il nome di advertised neighbor. Un MP deve annunciare almeno i link tra sé stesso e tutti i suoi MPR, che costituiscono le informazioni sufficienti ad elaborare poi il routing. I link advertised sono annunciati attraverso i messaggi Topology Control (TC).

L'RA-OLSR TC Information Element ha il seguente formato (Figura 3.5):

Il campo ANSN indica un sequence number associato all'advertised neighbor set. Ogni volta che un MP rileva un cambiamento nel suo advertised neighbor set, alla emissione del TC esso incrementa l'ANSN.

Il campo Advertised Neighbor Main Address indica il main address di un vicino MP. Tutti i main address degli advertised neighbor dell'originator MP sono inseriti nel messaggio TC.

Octets: 2	6	4	…	4	…
Advertised Neighbor Sequence Number	Advertised Neighbor Main Address	Link Metric #1	…	Link Metric #17	…

6	4	…	4
Advertised Neighbor Main Address	Link Metric #1	…	Link Metric #17

Figura 3.5. Formato del TC Information Element

I campi Link Metric indicano le metriche di ciascun link e rappresentano una modifica nel messaggio originario, che prevede invece solamente una metrica per ogni link. Se un advertised neighbor è raggiungibile attraverso più link, è avvertito il link con bit rate fisico più alto.

Per costruire il Topology Set, ogni MP, che è stato selezionato come MPR, inoltra in broadcast dei messaggi TC. Quando il set degli advertised link diventa vuoto per varie ragioni, questo MP emette un TC vuoto alla scadenza del tempo di validità (fissato a TOP_HOLD_TIME) del precedente TC emesso. Il MP ferma l'emissione dei TC fino a quando non sono inseriti MP nel neighbor set.

Alla ricezione di un messaggio TC, sarà calcolato il tempo di validità dal campo Vtime del message header. Il Topology Set è aggiornato come segue:

a) Se la sender interface del messaggio non è in vicinanza simmetrica con l'MP che ha ricevuto il messaggio, quest'ultimo viene scartato.

b) Se esiste un'entry in Topology Set in cui:

   T_last_addr = Originator Address AND

   T_seq > ANSN

   il messaggiosarà scartato.

c) Tutte le entry in Topology Set in cui:

   T_last_addr = Originator Address AND

   T_seq < ANSN

   saranno rimosse dal Topology Set.

d) Per ogni Advertised Neighbor Main Address ricevuto nel messaggio TC:

   1) Se esiste un'entry in Topology Set in cui:

      T_dest_addr = Advertised Neighbor Main Address AND

      T_last_addr = Originator Address,

      allora sarà eseguito l'aggiornamento della entry:

      T_time = current time + tempo di validità.

   2) Altrimenti, una nuova entry è registrata in Topology Set in cui:

      T_dest_addr = Advertised Neighbor Main Address

      T_last_addr = Originator Address

      T_seq =ANSN

      T_time = current time+ tempo di validità.

3.8 Calcolo della Routing Table

In RA-OLSR ogni MP ha una tabella di routing, ricavata dalla conoscenza del Link Set, del Neighbor Set, del 2-hop Neighbor Set, del Topology Set e dell'Interface Association Set. Ogni entry della tabella indica come raggiungere un qualunque MP della rete specificando la distanza in hop e in metric cost dalla destinazione, il next hop e l'interfaccia locale da utilizzare. Il calcolo della tabella di routing può essere svolto in RA-OLSR da un qualunque algoritmo di tipo link state. Tuttavia nell'architettura di questa tesi il compito del calcolo delle route è svolto dal PCE (come è stato descritto nel capitolo precedente). Per questo motivo non ci si soffermerà su come in RA-OLSR viene costruita la tabella di routing.

3.9 Association Station Discovery

Ci sono due diversi modi di procedure per l'association station discovery:

- Full Base Diffusion;
- Checksum.

Nel primo metodo ogni MAP annuncia l'intero contenuto del suo LAB, mentre nel secondo metodo annuncia solo le checksum dei block del suo LAB per rilevare possibili block inconsistenti nei GAB dei MP che ricevono questi annunci. In questa tesi verrà considerato solo il primo metodo: il Checksum mode apporta solo dei miglioramenti dal punto di vista del traffic overhead [2].

3.9.1 Associated Station Discovery in Full Base Diffusion Mode

Come spiegato precedentemente, nel metodo Full Base Diffusion ogni MAP inoltra in broadcast periodicamente dei messaggi, chiamati Local Association Base Advertised (LABA), che informano gli altri MP sull'intero contenuto del LAB dell'originator MP. Gli MP che ricevono questi messaggi popolano i loro GAB.

L'RA-OLSR LABA Information Element ha il seguente formato (figura 3.6):

Octets: 1	1	6	1	…	6	1	…
Block 1 Index	Block 1 Message Size	Block 1 STA Address 1	Block 1 STA Sequence Number 1	…	Block 1 STA Address x	Block 1 STA Sequence Number x	…

1	1	6	1	…	6	1
Block n Index	Block n Message Size	Block n STA Address 1	Block n STA Sequence Number 1	…	Block n STA Address y	Block n STA Sequence Number y

Figura 3.6 Formato del LABA Element

Ogni LABA annuncia tutti i block con le relative stazioni appartenenti al LAB dell'originator MAP.

Il campo Block Index indica il block index del block che si anuncia.

Il campo Block Message Size indica la lunghezza del block, conteggiato in byte e misurato dall'inizio del precedente Block Index fino al successivo Block Index.

Il campo STA Address indica il MAC Address della stazione 802.11 associata.

Il campo STA Sequence Number indica il sequence number del frame di management attraverso cui la stazione si è registrata al suo MAP.

I messaggi LABA sono inoltrati dagli MPR in broadcast in tutta la rete mesh; il forwarding di tali messaggi avviene secondo il Default Farwarding Algorithm. Essi sono generati prima che le entry dei precedenti LABA siano spirate.

In un'ottica di ottimizzazione, si può prevedere anche di generare LABA contenenti solo i block che hanno subito cambiamenti.

Alla ricezione di un messaggio LABA, ogni MP aggiorna il suo GAB come segue:

a) Tutti i block del GAB appartenenti al MAP con Originator Address riportato nell'header del messaggio RA-OLSR sono cancellati: poiché il LABA ricevuto contiene tutti i block del LAB dell'originator MP, le entry precedenti sono considerate obsolete.
   Vengono cioè cancellate le entry per cui:
   GAB_AP_address = Originator Address.

b) Per ogni STA Address contenuta nel LABA, l'aggiornamento delle entry avviene in questo modo:
   1) Se c'è un'entry in GAB in cui:
      GAB_AP_address = LABA Originator Address AND
      GAB_station_address = STA Address AND
      GAB_station _sequence_number <= STA Sequence Number
      allora l'aggiornamento è effettuato in questo modo:
      GAB_station_sequence_number = STA Sequence Number
      GAB_station_address = STA Address
      GAB_expiration_time = current time + tempo di validità.

2) Altrimenti, se esiste un'entry in GAB in cui:
   GAB_AP_address = LABA Originator Address AND
   GAB_station_address = STA Address AND
   GAB_station _sequence_number > STA Sequence Number
   il messaggio è scartato perchè considerato obsoleto.
3) Altrimenti viene creata una nuova entry in cui:
   GAB_AP_address = LABA Originator Address
   GAB_station_address = STA Address
   GAB_block_index = Block Index
   GAB_station _sequence_number = STA Sequence Number
   GAB_expiration_time = current time + tempo di validità.

3.9.2 Aggiornamento dei LAB

Poichè un MAP non sempre è informato direttamente della disassociazione di una STA, può capitare che la rilevi indirettamente dal LABA proveniente da un altro MAP, che annuncia la stessa STA con un nuovo sequence number.

La seguente procedura è adottata per trattare il verificarsi di questa situazione: se in LAB esiste un'entry per cui

- LAB_station_address = STA Address (del LABA)
- LAB:station_sequence_number <= STA Sequence Number (del LABA)

allora tale entry viene cancellata e la corrispondente STA è considerata disassociata. Questa informazione viene trasmessa agli altri protocollo 802.11.

In generale quando un MP ha bisogno di stabilire un path con una STA, esso cerca l'indirizzo della STA nelle entry del LAB e del GAB. Se dopo la ricerca sono state trovate più entry, il MP sceglierà l'associazione con il più alto sequence number, sia nel GAB che nel LAB.
La conoscenza dei MAP che fanno daProxy alle STA è ovviamente fondamentale per poter stabilire i path tra i MP e questi Proxy.
Le procedure, attraverso cui questi path sono stabiliti, sono svolte dal PCE (cha ha il compito di determinare i best path tra i nodi mesh in base alle informazioni fornite dal TED) e dal protocollo di segnalazione delle etichette (che ha il compito di instaurare questi path nella rete).

Cap. 4 PCE

In questo capitolo sono descritti le funzionalità generali del Path Computation Element e gli algoritmi classificati come Multi-Constrained Path (MCP) e Restricted Shortest Path (RSP); successivamente sono esaminati gli algoritmi implementati in questa tesi.

4.1 Introduzione al PCE

Come accennato precedentemente, il PCE è il componente architetturale che si occupa di calcolare un percorso, o più percorsi, tra due nodi di una rete, secondo determinati vincoli. Ad esempio nell'architettura MPLS i percorsi calcolati dal PCE saranno impiegati per la costruzione di LSP tra un nodo LSR d'ingresso e uno d'uscita.

Un PCE può essere realizzato secondo un modello centralizzato o distribuito [17]. Il primo si riferisce ad un modello per mezzo del quale tutti i path in una rete sono calcolati da un singolo PCE, localizzato in un server o in un router designato. Un PCE centralizzato utilizza una procedura ottima di calcolo che tiene conto sia delle informazioni sulle metriche dei link trasportate dal protocollo di routing, sia delle risorse riservate alle connessioni stabilite: infatti, poiché ogni connessione stabilita modifica le attuali risorse di rete, la procedura ottima deve tenere conto anche di questo

fattore. Gli algoritmi utilizzati nello scenario centralizzato sono chiamati off-line. Lo svantaggio nell'utilizzare un singolo PCE centralizzato consiste nel fatto che esso può diventare un collo di bottiglia.

Nello scenario distribuito il calcolo dei path avviene in diversi punti della rete, tipicamente in tutti i nodi. I path sono calcolati tenendo conto solamente delle metriche dei link e di eventuali vincoli amministrativi; queste informazioni sono trasportate attraverso il protocollo di routing sottostante. Gli algoritmi utilizzati in tale situazione sono chiamati on-line.

Coerentemente con le scelte progettuali adottate, si è deciso di effettuare lo studio di PCE distribuiti. Tale studio è decisamente complesso, visto che l'efficienza di un PCE deve essere valutata rispetto all'abilità nel soddisfare un insieme di richieste provenienti dai flussi. In particolare, tenendo conto delle metriche di QoS trasportate dal protocollo di routing adottato (che ripetiamo in questa tesi è una versione modificata di RA-OLSR), della topologia della rete e delle richieste di QoS dei flussi, il PCE deve effettuare:

1. *Path Selection*, ovvero calcolo del percorso per poter raggiungere il nodo mesh destinatario;
2. *Admission Control*, ovvero verificare che il percorso selezionato soddisfi le richieste di QoS dei flussi, e in caso positivo stabilire il path (processo svolto dal modulo di RSVP-TE).

Il problema principale consiste perciò nello studiare gli algoritmi di calcolo dei percorsi: tali algoritmi costituiscono il 'motore' dei PCE. E' ovvio che un algoritmo sarà migliore quando più alta è la possibilità di generare (almeno) un percorso che soddisfi contemporaneamente tutti i vincoli di QoS imposti; l'uso di un algoritmo migliore comporta perciò un PCE più

efficiente. I concetti appena esposti possono essere riassunti nello schema in figura 4.1.

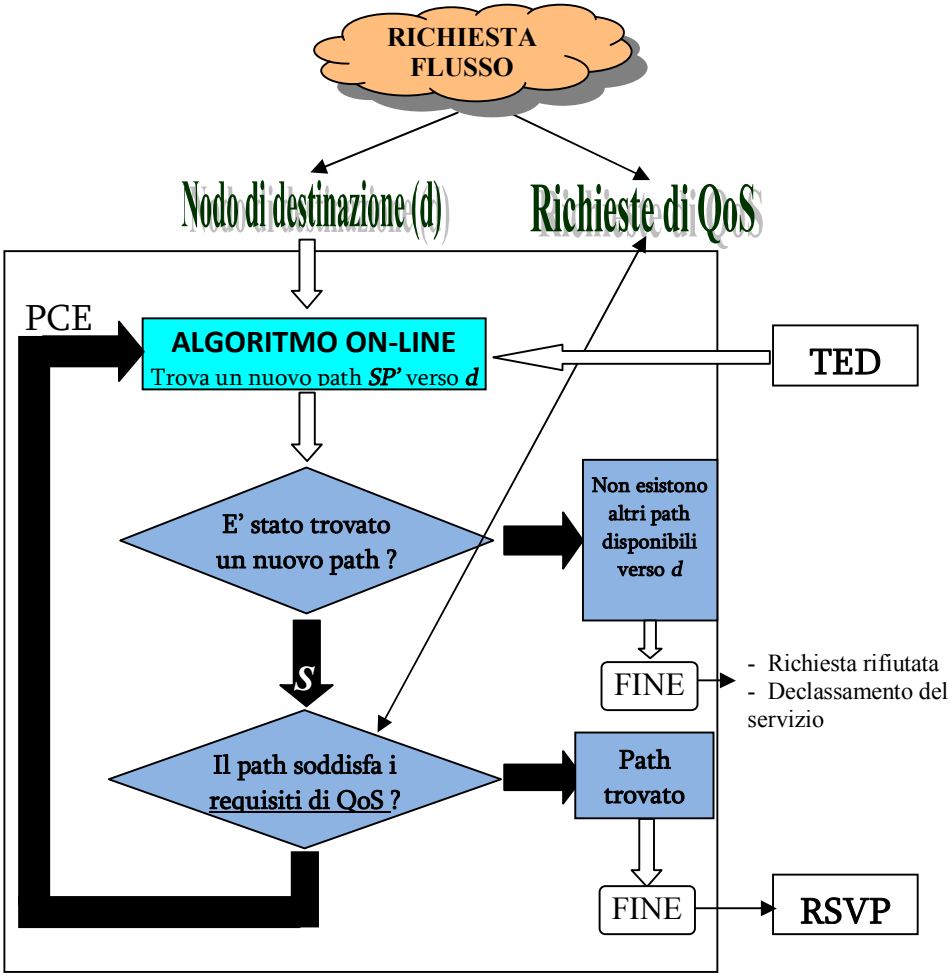


Figura 4.1 Struttura del PCE da implementare

In questo contesto la letteratura propone un insieme limitato di algoritmi on-line, che verranno a breve esaminati; tali algoritmi presentano inoltre dei limiti notevoli che saranno evidenziati.

4.1.1 Distribuited Path Computation Algorithm

Il primo algoritmo descritto è **CSPF** (Constrained Shortest Path First), che è l'algoritmo di path computation di default usato da OSPF-TE [18]. Ogni link *(i,j)* è caratterizzato da due metriche: b_{ij} (banda residua) e c_{ij} (costo). L'algoritmo punta a trovare un path con banda $B \geq BMIN$, minimo costo e tenendo conto eventualmente di vincoli amministrativi (colori dei link).
CSPF esegue i seguenti passi:

1. porre $c_{ij} = \infty$ se $b_{ij}<BMIN$ (o se il colore del link è diverso da quello prescelto);
2. calcolare lo shortest path *P* rispetto ai costi applicando Dijkstra.

Il secondo algoritmo è **WC** (Wang Crowcroft) [19]: questo, diversamente da CSPF, punta a soddisfare anche un vincolo sul costo massimo *DMAX*, oltre a quello della banda minima *BMIN*. Ogni link *(i,j)* è caratterizzato da due metriche: b_{ij} (banda residua) e d_{ij} (ritardo di propagazione). WC esegue i seguenti passi:

1. porre $d_{ij} = \infty$ se $b_{ij} < BMIN$;
2. calcolare lo shortest path *P* rispetto al ritardo di propagazione applicando Dijkstra, e ricavare il ritardo totale *D**;
3. confrontare *D** con *DMAX*. Se *D*<DMAX* selezionare il path, altrimenti la richiesta è rifiutata.

Il terzo algoritmo prende il nome di **DBCTE** (Delay and Bandwidth Constrained with TE) [20], ed estende l'algoritmo di Wang Crowcroft con il load balancing. Per supportare questa funzionalità viene introdotta una nuova metrica, BTE: essa è definita per ogni link come il rapporto tra la banda del link (B_{ij}) e quella residua:

$$BTE(i,j) = \frac{Bij}{bij}$$

BTE è una quantità che cresce quando la banda residua diminuisce ed è sempre maggiore o uguale ad uno.

L'algoritmo consiste dei seguenti step:

1. porre $d_{ij} = \infty$ se $b_{ij} < BMIN$;
2. calcolare lo shortest path P rispetto al BTE applicando Dijkstra, e ricavare il ritardo di propagazione totale D^*;
3. confrontare D^* con $DMAX$. Se $D^* < DMAX$ selezionare il path, altrimenti usare WC.

Questi tre algoritmi hanno tutti una pesante limitazione. Innanzitutto CSPF tiene conto solo di un vincolo: per questo motivo non può essere utilizzato nel PCE che si vuole implementare, dove è richiesto un algoritmo che tenga conto di più vincoli. WC e DBCTE considerano due vincoli, banda e ritardo, ma potrebbero comunque essere facilmente estesi considerando altri vincoli. Giusto per fare un esempio con quattro metriche (banda b_{ij}, ritardo d_{ij}, e in più jitter j_{ij} e percentuale di perdite dei pacchetti l_{ij}) una modifica a WC potrebbe essere la seguente:

1. porre $d_{ij} = \infty$ se $b_{ij} < BMIN$;

2. calcolare lo shortest path P rispetto ad una delle metriche, ad esempio quella più stringente come il ritardo di propagazione applicando Dijkstra, e ricavare il ritardo totale D^*, il jitter totale J^* e le perdite totali L^*;
3. confrontare D^* con $DMAX$, J^* con $JMAX$ e L^* con $LMAX$. Se $D^* < DMAX$, $J^* < JMAX$ e $L^* < LMAX$ selezionare il path, altrimenti la richiesta è rifiutata.

Un algoritmo di questo tipo risulta poco efficiente, perché quando il percorso trovato, a costo minimo rispetto ad una metrica, non soddisfa tutti i vincoli la richiesta viene direttamente rifiutata e non vengono trovati dei percorsi alternativi in grado di poter soddisfare i vincoli precedentemente non soddisfatti. È banale constatare infatti che se un algoritmo trova solamente un path, che risulta ottimo secondo una metrica e soddisfacente il vincolo sempre relativo a tale metrica, può non soddisfare gli altri vincoli, specie se questi sono numerosi. Perciò anche WC e DBCTE non hanno le caratteristiche del PCE di figura 4.1.

4.1.2 Multiconstrained path problem (MCP)

Cerchiamo perciò di formalizzare il problema in maniera più generale. Supponiamo di avere una rete con N nodi, e siano s e d rispettivamente i nodi sorgente e destinatario. Ad ogni link (u,v) siano associate inoltre **m** metriche $\mathbf{w_i}(u,v)$, con i=1,..m: supponiamo, per semplicità e senza perdita di generalità, che tali metriche siano additive e non negative. Dati **m** vincoli (constraints) $\mathbf{L_i}$, il nostro problema consiste nel trovare un path P da s a d tale che:

$$w_i(P) \underline{\text{def}} \sum_{(u,v)\in P} w_i(u,v) \le L_i$$
$$i = 1, \ldots, m.$$

Questo è il cosiddetto Multi-Constrained Path (MCP) problem [21]; un path che ubbidisce a questi vincoli è detto feasible ('fattibile'). Poiché è possibile trovare più path feasible tra *s* e *d*, una versione modificata (e più difficile) dell'MCP problem è trovare il path a 'lunghezza' minima tra quelli feasible, ovvero il path ottimo. Tale problema è conosciuto come Multiconstrained Optimal path (MCOP), e in pratica consiste nell'aggiungere una seconda condizione su *P*: $\mathbf{I}(P) \le \mathbf{I}(Q)$ per ogni path feasible *Q* tra *s* e *d*, dove $\mathbf{I}(\bullet)$ è una funzione che restituisce la 'lunghezza' del path dati in ingresso le metriche totali $\mathbf{w_i}(P)$, con i=1,..m.

E' stato dimostrato [22] che l'MCP problem è NP-completo: ciò equivale a dire che se il numero delle metriche QoS che si vorrebbero minimizzare è ≥2, l'MCP problem diventa intrattabile per reti di grande dimensione. L'MCOP problem presenta poi l'ulteriore problema della scelta della funzione $\mathbf{I}(\bullet)$: infatti siccome le metriche QoS sono eterogenee, non esiste una funzione ottima che 'combini' le metriche per restituirne una sola. Mentre il primo problema può essere affrontato realizzando alcune 'euristiche' che cercano di risolverlo in tempo polinomiale, il secondo può essere affrontato allargando l'MCOP problem alla categoria 'subottima' del Restricetd Shortest Path (RSP) problem, in cui l'obiettivo è trovare il percorso a costo minimo rispetto ad una sola metrica, per poi verificare il rispetto dei vincoli sulle altre metriche. Nel nostro caso, la metrica su cui ci si baserà per calcolare gli shortest path è quella del ritardo, mentre i vincoli

da verificare saranno quelli sul jitter massimo e sulle perdite massime (oltre quello sul ritardo stesso).

4.1.3 Obiettivo della tesi

Il lavoro di tesi si è perciò concentrato sulla realizzazione di quattro algoritmi RSP, i quali, ognuno in maniera diversa, cercano di ricavare più percorsi feasible tra un nodo sorgente ed un nodo destinatario, applicando l'algoritmo di Dijkstra. Questi algoritmi derivano da alcune intuizioni tratte dall'osservazione del funzionamento di algoritmi esistenti, oppure rappresentano implementazioni, opportunamente modificati, di pseudocodici di algoritmi utilizzati nella Teoria dei Grafi.
Il restante capitolo descrive questi algoritmi e le relative implementazioni.

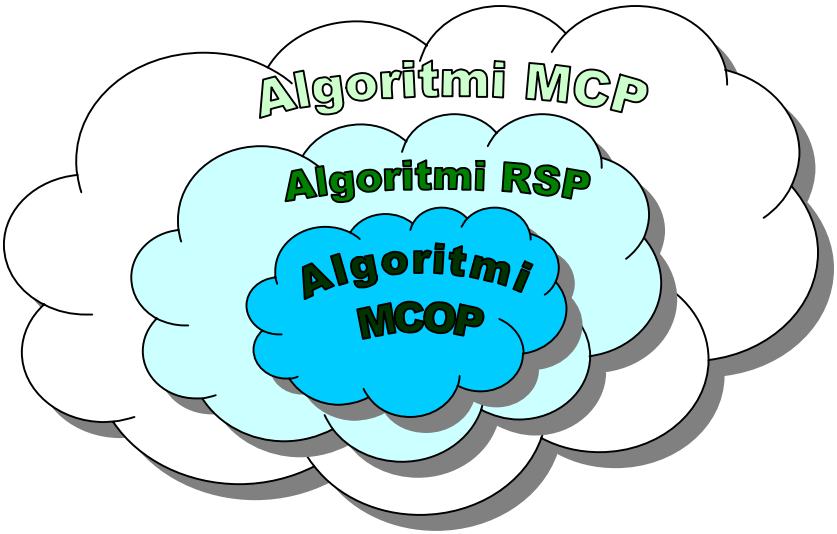


Figura 4.2 Classificazione dei QoS algorithm

4.1.4 Algoritmo di Dijkstra

Prima di passare in rassegna gli algoritmi implementati, è necessario richiamare brevemente il funzionamento dell'algoritmo di Dijkstra, visto che sarà usato da tutti e quattro gli algoritmi successivi.
Sia dato un grafo avente N nodi e si indichi con l_{ij} il costo non negativo associato al ramo che esiste fra i nodi i e j, ponendo $l_{ij} = \infty$ se non esiste un ramo che collega i e j. L'algoritmo di Dijkstra [23] permette di calcolare lo shortest path, rispetto al costo, da un nodo s ad un altro nodo d della rete.
L'algoritmo gestisce le due variabili seguenti: M indica l'insieme di nodi finora presi in considerazione dall'algoritmo e C(n) indica il costo del percorso da s a ciascun nodo n. Date queste definizioni, l'algoritmo è definito come segue:

M= {s}
per ogni n *in* N - {s}
  C(n)= l_{sn}
finchè (N ≠ M)
  M= M U {w} tale che C(w) sia il minimo per tutti i w in (N-M)
  per ogni n in (N-M)
    C(n)= MIN (C(n), C(w) + l_{wn})

In letteratura esistono diverse implementazioni dell'algoritmo di Dijkstra. Quella su cui ci si basa in questa tesi è detta forward search algorithm (algoritmo di ricerca in avanti). L'implementazione è riportata in appendice 4A.1: brevemente, vengono utilizzate due liste, `Confirmed` (che

comprende i nodi inseriti nell'albero), e `Tentative` (che comprende i nodi esaminati ma non ancora inseriti nell'albero).

Per fare un esempio, consideriamo la figura seguente (Figura 4.3):

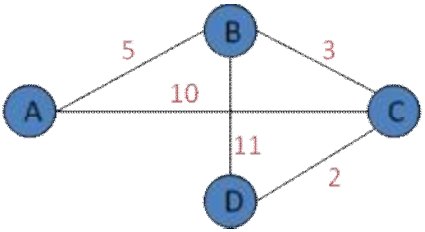


Figura 4.3 Esempio di topologia di rete

Supponiamo che il nodo sorgente sia *A*, che il nodo destinatario sia *C* e che i link siano contraddistinti da una sola metrica. I passi compiuti sono:

1. Si seleziona *A*;
2. Vengono esaminati i vicini di *A*, *B* e *C*, i quali vengono inseriti nella lista `Tentative` in questo modo: (B,5,A) e (C,10,A);
3. Il nodo raggiungibile con il costo minimo (quindi il dato (B,5,A)) viene estratto da `Tentative` e inserito in `Confirmed` (Figura 4.4);
4. Si seleziona *B* e si analizzano i suoi vicini.

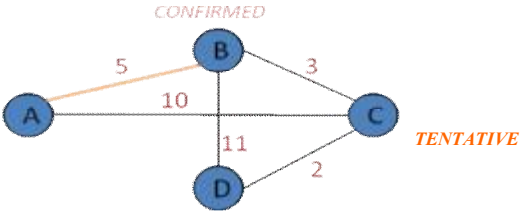


Figura 4.4. Applicazione di Dijkstra (1)

5. Il nodo *D* viene inserito nella lista `Tentative` (D,16,B); per quanto riguarda *C*, poiché il costo per raggiungerlo attraverso *B* è minore di quello già presente in `Tentative`, quest'ultimo dato viene cancellato e viene inserito (C,8,B). Poiché *C* è anche il nodo raggiungibile a costo minimo, viene estratto da `Tentative` e inserito in `Confirmed` (figura 4.5).

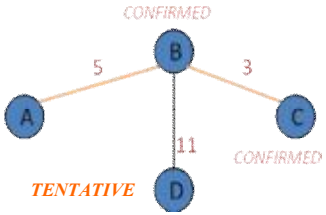


Figura 4.5. Applicazione di Dijkstra (2)

L'algoritmo è ripetuto N-1 volte: in questo modo possono essere ricavati gli shortest part da *s* verso tutti gli altri nodi della rete. Ovviamente l'algoritmo può anche essere utilizzato solo per trovare lo shortest path verso un determinato nodo destinatario: in tal caso esso viene fatto arrestare appena viene inserito nella lista `Confirmed` al passo 4 il dato relativo al nodo destinatario.

La complessità computazionale dell'algoritmo nel worst-case può essere calcolata considerando che l'algoritmo deve essere applicato N-1 volte per calcolare lo shortest path verso gli N-1 nodi della rete, e che al passo 3 devono essere esaminati tutti i vicini di Next, che nel caso peggiore possono essere sempre N-1 (rete completamente magliata). Perciò la complessità è dell'ordine di $O(n^2)$.

4.2 Descrizione degli algoritmi implementati

In questo paragrafo sono descritti i quattro algoritmi implementati:

- algoritmo di Dijkstra modificato;
- algoritmo di Iwata modificato;
- algoritmo di Mittal;
- algoritmo di Martin Santos.

Note generali sul codice sorgente relativo ai quattro algoritmi sono riportate in appendice 4A.2.

4.2.1 Algoritmo di Dijkstra modificato

Il primo algoritmo implementato consiste in una modifica dell'algoritmo di Dijkstra 'originale' per poter generale più percorsi tra due nodi mesh. Come detto precedentemente, nell'algoritmo di Dijkstra i percorsi verso un nodo a costo maggiore vengono sostituiti da quelli a costo minore venendo così eliminati. L'idea allora è quella di recuperare i percorsi scartati inserendoli in una nuova lista, `AlternativePath`: in questo modo è possibile ottenere alla fine più percorsi verso un nodo.

Riferendosi all'esempio nel sottoparagrafo 4.1.4, i primi quatto passi compiuti da quest'algoritmo coincidono con quello dell'algoritmo di Dijkstra originale. Al passo 5 accade che il nodo *D* viene inserito nella lista `Tentative` (D,16,B); per quanto riguarda *C*, poiché il costo per raggiungerlo attraverso *B* è minore di quello già presente in `Tentative`,

quest'ultimo dato viene spostato nella lista `AlternativePath` mentre quello nuovo (sempre (C,8,B)) viene inserito in `Tentative`.

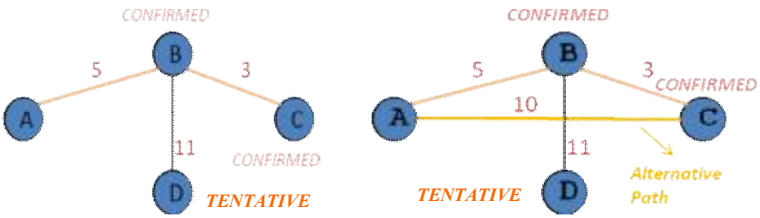


Figura 4.6. Differenze nell'esecuzione fra l'algoritmo di Dijkstra e quello modificato

Un dato relativo al raggiungimento di un nodo è inserito nella lista `AlternativePath` anche quando il costo per raggiungerlo è maggiore di quello già presente nella lista `Tentative` (nel Dijkstra originale in questo caso non si faceva nulla); viene inoltre aggiunto un dato relativo al raggiungimento di un nodo anche quando tale nodo è già presente nella lista `Confirmed` (cioè si scopre che un nodo presente già nell'albero può essere raggiunto dai nodi 'più lontani' che si stanno esaminando in quel momento).

Bisogna specificare che i dati inseriti nelle liste `Confirmed` e `AlternativePath` non sono i path trovati, ma indicano il nodo che è possibile raggiungere, le metriche cumulative per raggiungerlo e il previous hop attraverso cui si transita per raggiungerlo. Ad esempio, per la rete in figura 4.3 (ragionando per una sola metrica), alla fine dell'algoritmo la lista Confirmed sarà composta da questi dati: (A,0,A),(B,5,A),(C,8,B),(D,10,C). E' necessaria allora una funzione per ottenere da tali dati il path completo: senza entrare nei dettagli, questa funzione compie un percorso "a ritroso"

scorrendo la lista `Confirmed` fino a quando non viene trovato il nodo di partenza (che in questo caso è *A*). Questa funzione, indicata nel codice come "riordino_path", è presente anche nei successivi algoritmi visto che anche in essi i dati relativi ai path hanno la stessa identica forma.
Un'altra cosa che accomuna i quattro algoritmi è la modalità con cui viene effettuata la verifica del rispetto delle richieste di QoS. Poiché questi dati contengono anche le informazioni su ritardo, jitter e perdite totali necessari per raggiungere un determinato nodo, basta confrontare i membri delle strutture che costituiscono la lista `Confirmed` con le soglie. Se la verifica dà esito positivo, il path viene dichiarato feasible, altrimenti no.
Riportiamo di seguito ulteriori considerazioni.

- Per quanto riguarda la complessità computazionale, essa è chiaramente pari a quella dell'algoritmo di Dijkstra originaria, visto che il numero di passi eseguiti resta sempre lo stesso $O(n^2)$. Aumenta l'occupazione di memoria, visto che viene utilizzata un'altra lista: tuttavia questo svantaggio è del tutto trascurabile, poiché essa è legata al numero di nodi della rete che è limitato (come detto nel capitolo 2, massimo 50 nodi).
- A seconda della topologia della rete, viene trovato un numero di percorsi variabile. Come si vedrà dalle prove simulative, il numero di percorsi trovato per i nodi più lontani è generalmente maggiore di quello trovato per i nodi più vicini. Ciò si potrebbe spiegare in questo modo: man mano che ci si allontana dal nodo di partenza, l'albero sarà costituito da un numero sempre maggiore di 'rami'. I nodi più lontani saranno toccati perciò con più alta probabilità da un maggiore numero di 'rami' rispetto ai nodi più vicini.

- Sempre a seconda della topologia della rete, può capitare che venga originato un loop: in particolare ciò che può capitare è che venga ripetuto il nodo di partenza all'interno del path. A tal proposito è stato inserito un controllo opportuno per rilevare la presenza di questi loop.

Siamo così riusciti a trovare un algoritmo RSP, capace di generare più percorsi tra due nodi all'interno di una rete.

4.2.2 Algoritmo di Iwata modificato

Il secondo algoritmo [24] implementato nasce da questa intuizione: data una rete e definito un insieme di metriche, se si applica l'algoritmo di Dijkstra cambiando ogni volta la funzione vincolo è possibile trovare percorsi diversi, che sono rispettivamente gli shortest path rispetto alla metrica considerata. Per fare un esempio, consideriamo la rete in figura 4.7, dove ogni link è descritto da due metriche che per semplicità supponiamo additive.

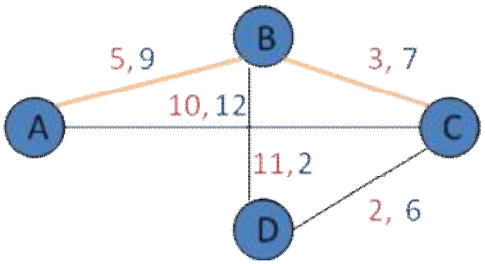


Figura 4.7. Algoritmo di Iwata

Lo shortest path per andare da *A* a *C* rispetto alla prima metrica è *C-B-A*; riapplicando Dijkstra rispetto alla seconda metrica si trova un path diverso, *C-A*. Calcolati i percorsi, si eseguirà poi l'admission control per verificare se le metriche totali dei percorsi soddisfano i vincoli richiesti dai flussi.

Il problema principale di quest'algoritmo è che comunque il massimo numero di percorsi che si riesce a trovare è pari al numero di metriche considerate (cioè ogni volta che si applica Dijkstra si trova un percorso diverso). Se perciò il numero di metriche è basso (nell'implementazione ne sono considerate tre) questo numero è anch'esso basso. L'idea allora è stata quella di applicare l'algoritmo di Dijkstra modificato, esposto nel paragrafo precedente, a quest'ultimo, che prende il nome del suo ideatore (Iwata), ottenendo così l'algoritmo di Iwata modificato.

Quindi ad ogni passo si applica l'algoritmo di Dijkstra rispetto ad una metrica: viene trovato il dato relativo allo shortest path più altri dati relativi ai percorsi scartati, che sono inseriti nella lista `AlternativePath`; dopodichè i path vengono calcolati. Quindi viene riapplicato Dijkstra rispetto ad un'altra metrica, ripetendo così il procedimento.

Dall'analisi di quest'algoritmo risulta che:

- fra gli algoritmi implementati, questo è l'unico che dipende esplicitamente dalle metriche: aumentando il numero delle metriche aumenta la possibilità di trovare più percorsi fra due nodi della rete;
- la complessità computazionale dell'Iwata modificato risulta pari a quella dell'algoritmo di Dijkstra moltiplicata per il numero di metriche M, quindi $O(M*n^2)$;

- poiché è possibile che alcuni path determinati rispetto ad una metrica siano gli stessi di quelli determinati precedentemente rispetto ad un'altra metrica, è stata implementata una funzione che analizza se il nuovo path coincide con i path già trovati. In caso affermativo, il path è scartato, altrimenti è aggiunto alla lista dei path trovati e quindi viene stampato. Questa funzione è utilizzata anche negli algoritmi di Mittal e di Martin-Santos
- Come nell'algoritmo precedente, anche ora c'è la possibilità che vengano trovati path con loop: ciò viene evitato tramite una funzione di controllo.

4.2.3 Algoritmo di Mittal

L'algoritmo di Mittal, e l'algoritmo di Martin Santos sfruttano la seguente idea. Supponiamo di avere a disposizione una certa rete, e di calcolare lo shortest path tra due nodi. A questo punto eliminiamo temporaneamente dalla topologia un elemento (link o nodo) appartenente allo shortest path e riapplichiamo l'algoritmo di Dijkstra: evidentemente viene trovato (se possibile) un nuovo percorso tra i due nodi, che è anche quello più breve relativamente al prune effettuato. Nell'algoritmo di Mittal (che è un'euristica) il prune viene "effettuato" sui link. Anche nell'algoritmo di Martin Santos (che invece è un algoritmo esatto), fra le varie operazioni compiute ad ogni passo, viene "effettuato" un prune, stavolta però sui nodi.
Descriviamo attraverso lo pseudocodice in figura Figura 4.8 come funziona l'algoritmo di Mittal [25]. Supponiamo di voler calcolare in una rete un certo numero K>1 di path tra un nodo *s* ed un nodo *d*. Indichiamo con

`Paths` la lista contenente gli n (con $n \leq K$) path trovati e con `PathsVector` la lista dei possibili path. Per ogni path *p* di `PathsVector` indichiamo con `L(p)` la lista dei link appartenenti a p ordinata secondo il loro costo crescente.

Paths = ∅;
PathsVector = ∅;
Trova lo shortest path *SP* tra *s* e *d* con l'algoritmo di Dijkstra;
PathsVector = *PathsVector* ∪ {*SP*};
Paths = *Paths* ∪ { *SP* };
n = 1;
While (n < K) and (*PathsVector* ≠ ∅)
    Prendi il primo path *p* of *PathsVector*;
    PathsVector = *PathsVector* – {*p*};
    Costruisci la lista *L(p)* dei link appartenenti a *p*;
    While (*L(p)* ≠ ∅) and (n < K)
        Prendi il link a costo minimo *l* of *L(p)* ;
        L(p) = *L(p)* – {*l*};
        Rimuovi il link *l* dalla rete e cerca il nuovo
        shortest path *SP'* tra *s* e *d* riapplicando Dijkstra;
        If *SP'* è trovato
            PathsVector = *PathsVector* ∪ {*SP'*};
            Paths = *Paths* ∪ {*SP'*};
            n = (n +1) ;
        End If
        Reinserisci *l* nella rete;
    End While
End While

Figura 4.8. Algoritmo di Mittal

L'algoritmo procede in questo modo. Inizialmente le liste `Paths` e `PathsVector` sono vuote; viene trovato lo shortest path *SP* con l'algoritmo di Dijkstra e inserito in `PathsVector` e in `Paths`, quindi si inizializza l'indice dei path trovati pari ad uno. Poiché `PathsVector` è non vuota e n<K, viene estratto da questa lista il primo path in testa (c'e' solo *SP*). I link di *SP* vengono inseriti nella lista `L(p)` in ordine di costo crescente.

A questo punto ad ogni ciclo viene estratto dalla lista `L(p)` il link a costo minimo, eliminato temporaneamente dalla rete e viene calcolato un nuovo shortest path, *SP'*, tra *s* e *d*, attraverso l'algoritmo di Dijkstra. Se viene trovato un nuovo path, diverso da quelli già memorizzati, questo viene aggiunto alle liste `Paths` e `PathsVector`, viene stampato e l'indice dei path trovati, n, viene incrementato. Quindi, alla fine del ciclo, il link viene reinserito nella rete. Quando tutti i link di *SP* sono estratti da `L(p)`, viene controllato se sono stati trovati almeno k path o se `PathsVector` è vuota (quest'ultima condizione indica che non sono stati trovati nuovi path). Se sì, l'algoritmo termina; altrimenti viene estratto dalla testa di `PathsVector` un nuovo path, *p*, e viene di nuovo eseguita l'operazione ciclica di prune sui link di *p*.

Lo pseudocodice si riferisce al caso in cui è definita una sola metrica: per questo motivo, esso è stato adattato in questa tesi per funzionare come RSP. In particolare, l'algoritmo calcola i percorsi utilizzando come funzione costo il ritardo; inoltre, per ognuno di questi percorsi vengono calcolati il jitter e le perdite totali. Infine viene effettuato l'admission control, che verifica se le metriche totali non superano le soglie.

Effettuiamo un analisi dell'algoritmo.

- A differenza dei due algoritmi precedenti, l'algoritmo di Dijkstra può essere arrestato non appena viene inserito nella lista `Confirmed` il dato relativo al nodo destinatario verso cui si sta calcolando il percorso (in altre parole, Dijkstra termina non appena è trovato lo shortest path verso il nodo destinatario).
- Il numero di percorsi trovato è variabile ed è dipendente dalla topologia della rete; inoltre, i percorsi trovati successivamente al primo non sono trovati in ordine di costo crescente (cioè l'n-esimo path può avere un costo maggiore dei successivi).
- La complessità in tempo è abbastanza complessa da valutare, visto che l'algoritmo consiste nell'applicazione ripetuta di Dijkstra. In generale si può affermare che tale complessità risulta sicuramente maggiore del Dijkstra modificato e nella maggior parte dei casi, anche dell'Iwata modificato. Studi effettuati in [26] dimostrano che essa è mediamente pari al prodotto fra il numero di nodi presenti nel grafo e la complessità media dell'algoritmo di Dijkstra, ovvero $O(2*n^2)$.

4.2.4 Algoritmo di Martin-Santos

L'algoritmo proposto da Martin e Santos appartiene alla categoria dei K-shortest loopless path algorithm [27], sviluppato nell'ambito della Teoria dei grafi. A differenza dei precedenti algoritmi, che sono euristici, esso è un

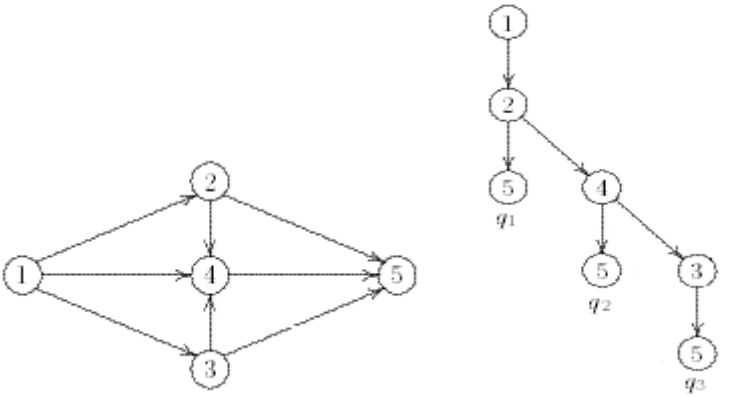
algoritmo esatto, in grado cioè di trovare tutti i percorsi possibili fra nodo sorgente e destinatario.

Siano *s* e *d* rispettivamente il nodo sorgente ed il nodo destinatario di un grafo, sia K un intero positivo e supponiamo che agli archi del grafo sia associato un solo costo. Determinare i K shortest loopless path significa determinare (se possibile) i path $\mathcal{P}_K = \{p_1,\ldots,p_K\}$ appartenenti all'insieme $\mathcal{P}$ tali che:

- p_K è loopless (cioè senza cicli), per ogni k=1,..K;
- il costo di p_K è minore o uguale a quello di p_{K+1}, per ogni k=1,.., K-1;
- il costo di p_K è minore o uguale al costo di qualunque path p appartenente a $\mathcal{P}$-$\mathcal{P}_K$
- p_K è determinato prima di p_{K+1}, per ogni k=1,..,K-1.

Lo studio di tali algoritmi è stato intrapreso fin dal 1951 grazie a Hoffman e Pavley [28]; da allora numerosi articoli su questo argomento sono stati pubblicati. Fra questi si citano gli studi di Eppstein [29] e l'algoritmo di Yen [30].

Gli algoritmi proposti da Martin e Santos sono nuove implementazioni dell'algoritmo proposto originariamente da Yen, e sono fra i più efficienti nell'ambito degli K-shortest loopless path algorithm. Essi fanno parte di un'altra sottoclasse di algoritmi, quelli di deviazione. Senza entrare troppo nei dettagli, tali algoritmi esplorano nuovi path a partire dai nodi appartenenti ai path precedentemente determinati: tali nodi sono chiamati nodi di deviazione. Per fare un esempio, consideriamo la rete in figura 4.9.a.



(a) (b)

Figura 4.9. Nodo di deviazione

Consideriamo i tre path in figura 4.9.b per andare dal nodo 1 al nodo 5. Il primo path ad essere determinato è q_1. Il secondo path determinato, q_2, ha un cammino in comune con q_1 fino al nodo 2 per poi deviare dal path precedente: il nodo 2 è il nodo di deviazione per q_2 rispetto a q_1. Segue quindi che per q_3 invece il nodo di deviazione rispetto a q_1 è sempre il nodo2; rispetto a q_2 è invece il nodo 4.

Gli algoritmi di Martin e Santos trovano ripetutamente nuovi percorsi *SP'* attraverso l'applicazione di Dijkstra tra i nodi v^k_{d-1}, v^k_{d-2},.., v^k_{s+1}, s appartenenti all'ultimo path determinato precedentemente, p_k, e d: in particolare con la notazione v^k_{d-1} intendiamo il previous hop di d relativo al path k-esimo, con v^k_{d-2} il previous hop di v^k_{d-1} sempre relativamente a p_k, e così via, per arrivare a v^k_{s+1} che è il next hop di s, e s stesso. Tali percorsi saranno poi concatenati rispettivamente ai sottopath $\text{sub}_{\text{pk}}(s,v^k_{d-1})$,

$sub_{pk}(s,v^k_{d-2})$, $sub_{pk}(s,v^k_{s+1})$ per formare i nuovi path.; ovviamente il percorso *SP'* ricavato tra *s* e *d* è considerato di per sé già un nuovo path e non ha bisogno di essere concatenato. In questo modo, per ognuno dei nuovi path trovati, i nodi di deviazione rispetto a p_k sono v^k_{d-1}, v^k_{d-2},.., v^k_{s+1}, *s*. Completata la ricerca, tra i nuovi path viene selezionato quello a costo minimo, p_{k+1}, e viene ripetuta una nuova ricerca esaminando i suoi nodi. Oltre ad usare il concetto di deviazione, l'algoritmo di Martin Santos utilizza anche quello di prune, come esposto all'inizio del paragrafo 4.2.3: in particolare, quando si determinano i nuovi path a partire da p_k, vengono eliminati dalla rete p_k sia i link di p_k, sia tutti i nodi appartenenti a p_k antecedenti quello dal quale si sta calcolando il nuovo path.

Gli algoritmi di Martin – Santos si differenziano tra loro a seconda delle ipotesi di partenza sulla rete (link simmetrici, link asimmetrici, ecc.): si riporta in figura 4.10 lo pseudocodice relativo all'algoritmo che si è deciso di implementare. *DP* è la lista dei nodi di deviazione che sono di volta in volta aggiunti; il simbolo ⌂ indica la concatenazione tra i due path.

Vediamo come quest'algoritmo è stato modificato per realizzare un MCP algorithm. Poiché il Martins-Santos agisce su una sola metrica, è stato necessario trasformarlo in un RSP algorithm, in cui il costo è rappresentato come sempre dal ritardo e i vincoli sono le perdite e il jitter. L'algoritmo modificato viene inserito nel PCE così come in figura 4.1.

$DP = \emptyset$;
$X = \emptyset$;
k=0;
Trova lo shortest loopless path SP tra s e d con l'algoritmo di Dijkstra;
$DP = DP \cup \{s\}$;
$X = X \cup \{SP\}$;
While $(X \neq \emptyset)$ and $(k \leq K)$
  k=k+1;
  Prendi il path a costo minimo p_k da X;
  Prendi il nodo di deviazione dp_k da DP;
  $X = X - \{p_k\}$;
  $DP = DP - \{dp_k\}$;
  v^k_{d-1}=previous hop di d di p_k;
  Rimuovi dalla rete i nodi appartenenti a $sub_{pk}(s, v^k_{d-1})$;
  For $(vik = \{ v^k_{d-1}, .., dp_k \})$
    Rimetti nella rete vik con tutti gli archi tranne:
    - quelli appartenenti a p_k
    - **If** vik = dp_k, quelli appartenenti a tutti i path precedenti, ovvero gli archi del tipo (dp_k, v^j_{dpk+1}) per tutti i j=1,..k; **End**

    Cerca uno shortest path SP' tra vik e d applicando Dijkstra;
    If SP' è trovato
      $p = sub_{pk}(s, vik) \triangle SP'$;
      $DP = DP \cup \{vik\}$;
      $X = X \cup \{p\}$;
    End
    Rimetti l'arco (vik, v^k_{vik+1});
  End
  Restaura la rete originaria;
End

Figura 4.10. Algoritmo di Martin Santos.

In appendice 4A.3 è riportata una nota sull'implementazione di questo algoritmo.

Descriviamo anche qui alcune considerazioni riscontrate durante l'implementazione del codice:

- L'algoritmo di Dijkstra può essere arrestato appena si ricava il dato relativo a *d* sia per trovare lo shortest path *SP*, sia durante l'esecuzione per trovare gli *SP'*.
- Analizziamo la complessità computazionale dell'algoritmo nel worst-case. Si può constatare che ogni volta che si analizza il path p_K , è necessario applicare l'algoritmo di Dijkstra per un numero di volte pari al numero di nodi che costituiscono p_K: nel caso peggiore questo path può essere composto da tutti gli n nodi della rete. Inoltre il Dijkstra dovrà essere applicato un numero di volte pari a K per trovare i K shortest path verso *d*: di questi K path poi, solamente alcuni saranno quelli che soddisferanno tutti i vincoli e saranno stampati dal PCE. Perciò la complessità è dell'ordine di $O(K*n*n^2)$.

Di seguito si riporta una tabella dove vengono riassunte le principali caratteristiche dei quattro algoritmi implementati.

	Dijkstra m.	**Iwata m.**	**Mittal**	**Martin Santos**
Tipo di algoritmo	*Euristica*	*Euristica*	*Euristica*	*Esatto*
Complessità	$O(n^2)$	$O(M*n^2)$	$O(2*n^2)$	$O(K*n*n^2)$

Appendice 4A

4A.1 Implementazione dell'algoritmo di Dijkstra

Viene descritta nel dettaglio l'implementazione dell'algoritmo di Dijkstra riportato nel paragrafo 4.1.4.

Ogni nodo utilizza due liste, note con il nome di `Tentative` e `Confirmed`. Ciascuna di queste liste contiene un insieme di dati aventi la forma (Destination, Cost, PrevHop). L'algoritmo funziona nel seguente modo:

1. Inizializza la lista `Confirmed` con (s,0,s) e `Tentative`=0.
2. Seleziona il nodo (chiamato Next) appena inserito nella lista `Confirmed` nel passo precedente, col costo necessario per raggiungerlo (CostNext) e il previous hop (PrevHop), ovvero l'insieme (Next, CostNext, PrevHop).
3. Per ciascun vicino (Neighbor) di Next, calcola il costo (Cost) necessario per raggiungere Neighbor come somma del costo da Next a Neighbor più CostNext:

a) se Neighbor non è nella lista `Confirmed` né nella lista `Tentative`, aggiungi (Neighbor, Cost, Next) nella lista `Tentative` (Next è ovviamente il previous hop di Neighbor);

b) se Neighbor non è nella lista `Confirmed` ma è nella lista `Tentative` e Cost è minore del costo attualmente associato a

Neighbor, sostituisci il dato attualmente memorizzato per Neighbor con (Neighbor, Cost, Next)

4. Scegli dalla lista Tentative, il dato avente il costo inferiore, spostalo nella lista Confirmed e torna al passo 2.

4A.2 Note sul codice sorgente

Il codice sorgente dei quattro algoritmi è realizzato in linguaggio C++. Essi hanno tutti alcune parti in comune:

- alcuni header, fra cui si segnalano "list.h", che comprende un insieme di funzioni sulle liste, e "timecalc.h", che comprende alcune funzioni utilizzate per il calcolo dei tempi di esecuzione;
- una parte di 'input', in cui sono aperti file in lettura e scrittura, inizializzate le matrici dei collegamenti, dei ritardi, del jitter e delle perdite;
- una fase di output, in cui vengono chiusi i file precedentemente aperti.

Rispetto all'algoritmo di Dijkstra, la struttura dei dati è modificata in modo da considerare anche le altre metriche. Più precisamente essa è definita in questo modo:

```
struct lista {
int nodo;
int cost;
int jitt;
double affid;
```

int prevhop;
lista pun;*
};

Come si può osservare, sono presenti più metriche: il ritardo di propagazione (cost), il jitter (jitt) e la probabilità di corretta ricezione (affid); i nodi sono visti come degli interi.

4A.3 Nota implementativa sull'algoritmo di Martin Santos

Rispetto agli altri algoritmi, oltre alla struttura Lista, è stata creata anche una nuova struttura Listadp:

struct listadp {
int nodo;
int cost;
listadp pun;*
};

In questa struttura sono registrati sia i nodi di deviazione relativi ai percorsi *SP'* calcolati, sia i costi (delay) totali dei path $sub_{pk}(s,vik) \triangle SP'$. Si è deciso di aggiungere anche i costi alla lista dei nodi di deviazioni in modo da velocizzare la ricerca del path a costo minimo.

Cap. 5 Test di simulazioni

In questo capitolo sono descritte le prove di simulazione effettuate sui quattro algoritmi che sono stati implementati per verificarne le prestazioni.

5.1 Generazione di topologie di rete

Per effettuare i test di simulazione, si è innanzitutto posto il problema sulla modalità di scelta della topologia della rete su cui effettuare queste prove. Una prima soluzione sarebbe stata quella di inventare alcune topologie e disegnarle: tuttavia questa opzione diventava inaccettabile se il numero di nodi della rete diventava alto (ad esempio 50) oppure se il numero di topologie da testare era numeroso.

Per questo motivo la scelta si è orientata sulla generazione casuale delle topologie tramite software. In letteratura esistono ad esempio diversi programmi per la generazione di reti random: fra questi si cita BRITE [31]. Tuttavia anche in questo contesto bisogna porre particolare attenzione: le topologie di rete che sono generate devono comunque essere simili alle reti mesh adottate, ovvero devono essere 'realistiche'.

Per questo motivo è stata effettuata un'accurata ricerca fra i numerosi modelli di rete esistenti ed in conclusione si è optato per una rete basata sul

modello della triangolazione di Delaunay. Per quanto spiegato in [32], i grafi di Delaunay sono infatti adatti a descrivere reti reali.

Brevemente, dato un set N di nodi in un piano, il grafo di Delaunay [33] è costituito da una serie di triangolazioni *DT(N)*. Tali "triangoli" hanno la proprietà che nessun nodo in N si trova all'interno del cerchio circoscritto ad ogni triangolo in *DT(N)*.

Il grafo è costituito quindi da tanti triangoli che uniscono i suoi nodi. Un esempio di triangolazione di Delaunay è in figura 5.1.

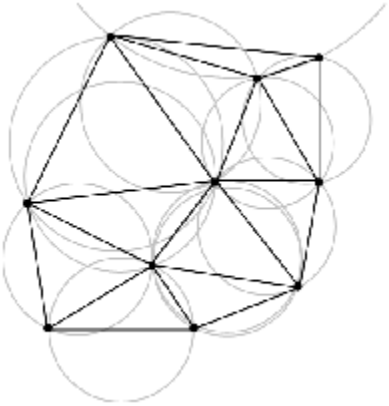


Figura 5.1 Grafo di Delaunay

Matlab fornisce la funzione TRI = delaunay (x,Coordinates), la quale, ricevute in input le coordinate cartesiane (generate a loro volta con il comando rand()) degli N nodi, fornisce la matrice TRI k x 3, dove k è il numero di triangoli ricavati: per ognuna delle k righe sono scritti i tre nodi che saranno uniti dalla triangolazione. Tale matrice viene poi opportunamente "manipolata" in modo da ottenere la matrice delle

adiacenze del grafo A, che sarà poi utilizzata dalla funzione gplot (A,Coordinates). Ecco lo script realizzato:

```
n=numeronodi;
for i=1:n
x(i)=rand(1);  y(i)=rand(1);
end
tri = delaunay(x,y);
xy=[x' y'];
A=manipolazione (tri);
gplot(A,xy)
```

Con questo script è possibile generare velocemente le topologie di rete. A questo punto è necessario generare le metriche per ogni link (i,j), che come detto sono il ritardo, il jitter e le perdite. Anche stavolta bisogna generare questi valori seguendo modelli il più possibile fedeli a quelli realistici. Seguendo questo principio, è stata fatta la seguente scelta.

- Ritardi $d(i,j)$: generati secondo una distribuzione uniforme tra a=1 e b=20 ms;
- Jitter $j(i,j)$: generati con distribuzione gaussiana, con valor medio pari a $d(i,j)$ e deviazione standard pari a 5;
- Perdite $l(i,j)$: generate secondo una distribuzione uniforme tra 0.01 e 1%.

Da ciò deriva che le variabili aleatorie $d(i,j)$ e $j(i,j)$ risultano correlate: infatti un link con alti ritardi avrà con elevata probabilità anche alti jitter e

viceversa. Le perdite di un link invece sono indipendenti dalle prime due metriche.
La generazione dei valori delle metriche viene effettuata nello stesso script che genera le reti di Delaunay. Questo script fornisce come output alcuni file di testo, che descrivono le matrici di adiacenza, dei ritardi, dei jitter e delle perdite; tali matrici saranno poi utilizzate dagli algoritmi RSP.

5.2 Modalità di svolgimento delle prove

Descriviamo ora le modalità con cui sono stati effettuati i test di simulazione per validare i path computation algorithm implementati e per confrontare le loro prestazioni. Questi test sono stati effettuati su diverse topologie di rete: in questa tesi riportiamo i risultati relativi alla topologia di figura 5.2. Essa è costituita da 20 nodi mesh; per ogni link, raffigurato attraverso una linea blu, sono indicate le metriche del ritardo (in verde), del jitter (in blu) e delle perdite (in rosso).
Relativamente ad ogni prova e ad ogni algoritmo, i test sono stati effettuati per tutte le possibili coppie di nodi, per un totale di 380 volte: questi test sono poi ripetuti in diverse condizioni che saranno descritte. Per il confronto dei diversi algoritmi sono stati considerati e misurati una serie di parametri.

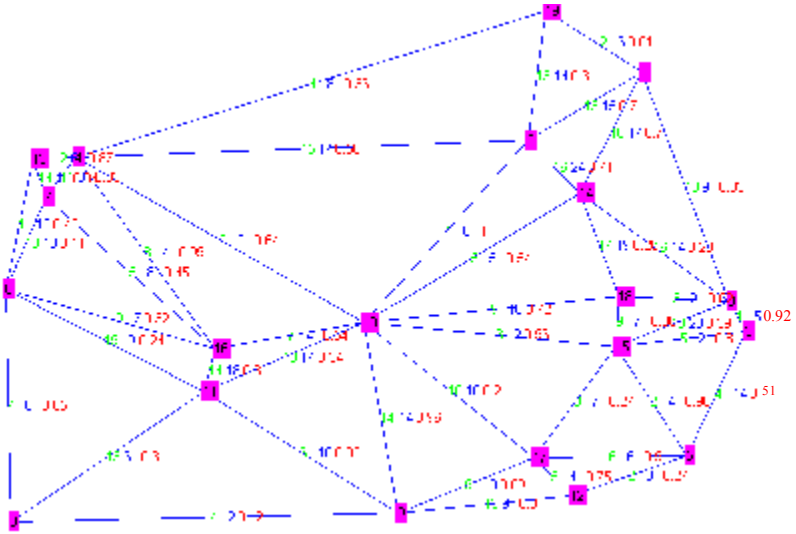


Figura 5.2 Topologia di test

I dati raccolti sono:

1) *Numero medio di percorsi totali:* è il numero di path che si sono riusciti a trovare da ogni nodo s a ogni nodo d, con $s \neq d$, indipendentemente dai vincoli QoS, mediato su tutte le 380 coppie.
2) *Numero medio di percorsi feasible*: è il numero medio di percorsi soddisfacenti le particolari richieste di QoS dei flussi.
3) *Percentuale di successo*: rappresenta la percentuale per cui l'algorimo riesce a generare almeno un percorso soddisfacente le richieste di QoS dei flussi.
4) *Tempo medio di Dijkstra*: rappresenta il tempo medio necessario all'algoritmo di Dijkstra 'originale' di generare un path (indipendentemente se questo poi risulta feasible o no); tale

percorso coincide in pratica con il primo percorso trovato da ognuno dei quattro algoritmi.

5) *Perdite, jitter e ritardi medi dell'i-esimo percorso feasible, i=1,2,3,4,5*: rappresentano i valori medi delle metriche accumulate dell'i-esimo percorso feasible (compreso anche quello eventuale generato da Dijkstra). Nel caso in cui non si è riusciti a determinare i path feasible non è riportato ovviamente nessun valore.
6) *Tempo di calcolo dell'i-esimo percorso feasible, i=1,2,3,4,5*: stesso discorso precedente, ma relativo ai tempi di calcolo.

Ricordiamo che i tempi, i ritardi e i jitter sono espressi in millisecondi.

Le performance degli algoritmi sono state valutate in base ai valori delle richieste di QoS. E' facile constatare che, a parte il numero medio di percorsi totali, tutti gli altri parametri dipendono dalle soglie sulle metriche. Un algoritmo sarà tanto più efficiente quanto più alto è il numero medio di percorsi feasible o più alta la percentuale di successo rispetto ad altri algoritmi, nonché quanto minore sarà il tempo impiegato per trovare questi percorsi.

Le simulazioni sono state condotte perciò variando i valori dei vincoli di QoS, così come illustrato in figura 5.3. In particolare ci si è mossi lungo il piano (perdite, ritardo + jitter). Il ritardo e il jitter sono stati considerati assieme visto che nella pratica incidono entrambi sul ritardo totale sperimentato dal nodo destinatario.

Nella prima e seconda fase di prove è stata fissata la soglia sulle perdite (rispettivamente a 1.3% e 2.5%), facendo variare le soglie su ritardo e jitter, con rapporto fra queste di 1:1 e 3:2. Nella terza e quarta fase le prove sono state effettuate fissando i vincoli sul ritardo e sul jitter, pari entrambi a 30

ms e poi a 20 ms, facendo variare la soglia sulle perdite progressivamente dall'1% al 2 e 2.3%.

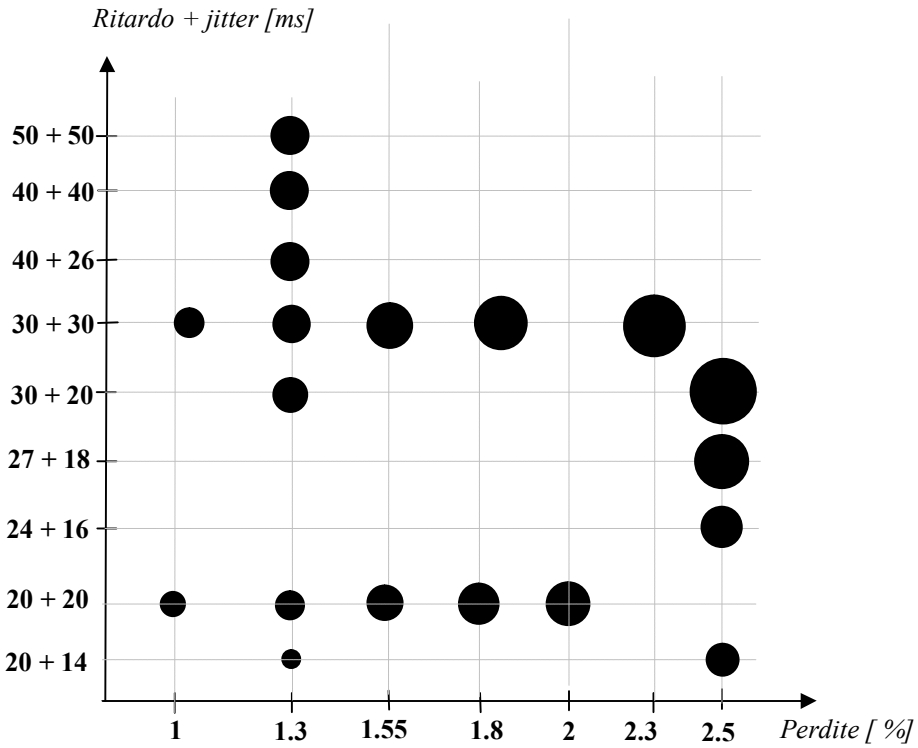


Figura 5.3 Percentuale di successo in funzione delle tre metriche

La figura 5.4 riporta tutte le prove effettuate. Ciascuna prova è individuata da una "bolla" il cui diametro è proporzionale alla percentuale di successo che ha il semplice algoritmo di Dijkstra. Questo serve per avere un'idea di quanto sono stringenti i vincoli sulla QoS che i flussi richiedono nei vari

scenari. Si passa ad esempio da una percentuale di successo del 24% per il punto (1.3, 20+20) ad una percentuale del 86% per il punto (2.5, 30+20).

5.2.1 Scenario 1

In questo scenario il vincolo sulle perdite è fissato a 1.3%.
Effettuiamo un confronto fra l'algoritmo di Dijkstra e i nostri quattro algoritmi implementati, in termini di percentuale di successo. I risultati sono riportati in figura 5.4 e 5.5 (abbiamo raffigurato separatamente il caso in cui le soglie su ritardo e jitter sono in rapporto 1:1 e quello in cui sono in rapporto 3:2).

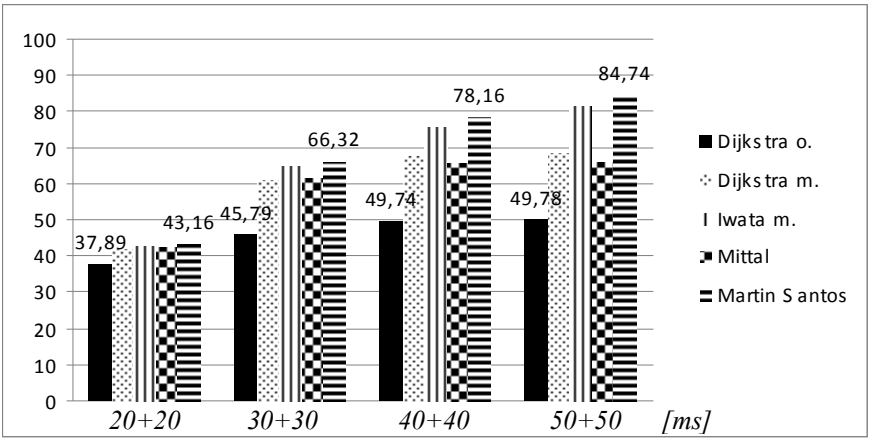


Figura 5.4 Percentuale di successo degli algortimi con soglia sulle perdite a 1.3% e rapporto ritardo-jitter 1:1

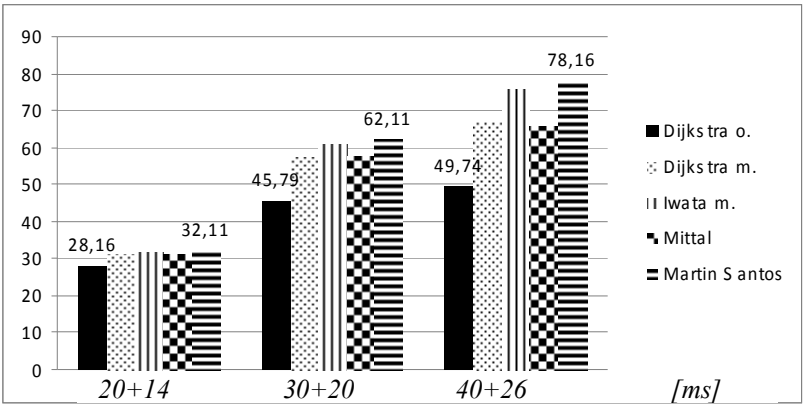


Figura 5.5 Percentuale di successo degli algortimi con soglia sulle perdite a 1.3% e rapporto ritardo-jitter 3:2

Da questi grafici già è possibile trarre alcune considerazioni. Innanzitutto vediamo che gli algoritmi implementati hanno una percentuale di successo sempre maggiore di quello di Dijkstra. Questo è ovvio, lo scopo degli algoritmi è trovare percorsi alternativi all'unico cammino fornito dal Dijkstra semplice. Fra questi algoritmi, quello con la percentuale di successo più alta è sempre quello di Martin Santos (da ora abbreviato MS). Anche questa cosa non deve stupire: l'algoritmo di MS a differenza degli altri tre è esatto (cioè si riescono a trovare tutti i percorsi tra un nodo ed un altro), mentre gli altri sono euristiche. Il fatto che sia esatto implica quindi che la percentuale del MS rappresenta l'upper bound per le percentuali di successo, ovvero non esiste nessun altro algoritmo che riesca ad avere una percentuale di successo maggiore. Questo ci potrebbe indurre a utilizzare

sempre MS: tuttavia, come vedremo fra poco, il notevole svantaggio è legato ai tempi di calcolo.

Fra gli altri tre algoritmi, osserviamo che l'unico che tende ad avvicinarsi alle percentuali di successo del MS è l'algoritmo di Iwata modificato (da qui abbreviato Im). Il Dijkstra modificato ha prestazioni inferiori, mentre quelle del Mittal (da ora in poi Mi) sono le più scadenti, nonostante abbia una complessità computazionale simile a quella di Im. Il motivo, che sarà visto ancor meglio nel sottoparagrafo 5.2.4, è legato al numero ridotto di percorsi che Mi riesce a trovare.

Osserviamo inoltre che per soglie di ritardo e jitter molto stringenti (parte inferiore di figura 5.3) le differenze di percentuali di successo tra i vari algoritmi sono molto piccole (massimo 5%). Quando queste soglie sono invece maggiori di 30 ms, si notano invece maggiori differenze fra l'algoritmo di Dijkstra e MS: la differenza di percentuale di successo è anche superiore al 30%. Perciò, riassumendo, per soglie su ritardo e jitter minori del 25-30% gli algoritmi sostanzialmente si equivalgono; viceversa per soglie maggiori le diverse prestazioni degli algoritmi cominciano a mettersi in evidenza.

Un confronto ulteriore viene effettuato sul numero medio di percorsi buoni (figure 5.6, 5.7). In questi due grafici si osserva ancora una volta che:

- l'algoritmo MS fornisce le prestazioni migliori;
- l'algoritmo Im è quello che si avvicina di più in prestazioni al MS;
- per soglie su ritardo e jitter maggiori di 30 ms le prestazioni cominciano a differenziarsi.

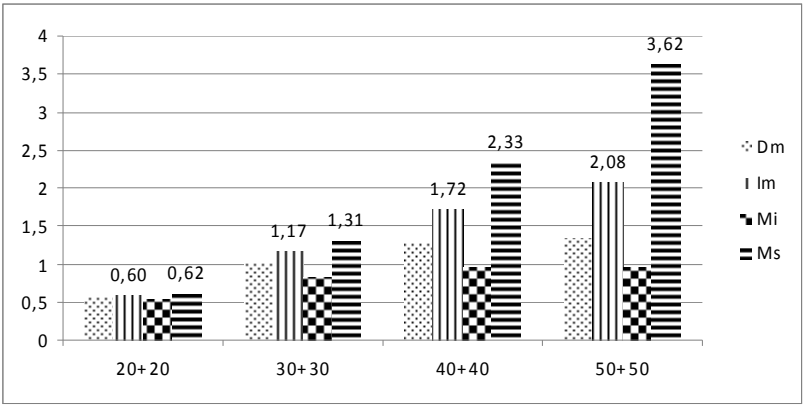


Figura 5.6 Numero medio di percorsi buoni con soglia sulle perdite a 1.3% e rapporto ritardo-jitter 1:1

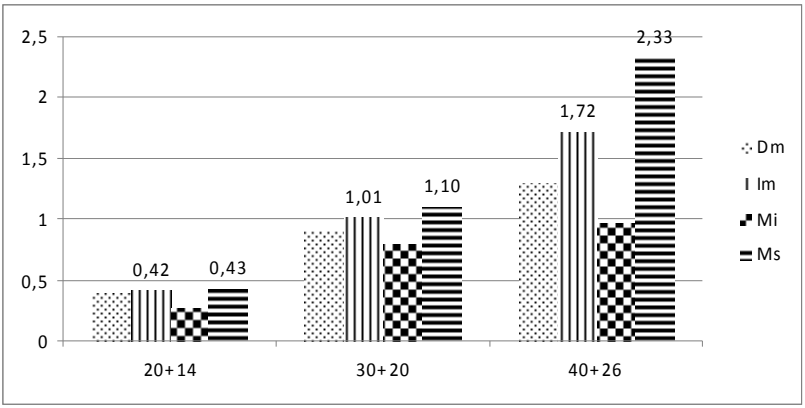


Figura 5.7 Numero medio di percorsi buoni con soglia sulle perdite a 1.3% e rapporto ritardo-jitter 3:2

Analizziamo ora più dettagliatamente i risultati ottenuti dalla prova in cui le soglie del ritardo e del jitter sono pari a 50 ms e quella sulle perdite al 1.3%:

infatti in tale condizioni è possibile osservare maggiormente le differenze di prestazioni fra i vari algoritmi.

I primi dati sono riportati in figura 5.8. Nella prima riga osserviamo come il numero medio di percorsi totali calcolati da MS sia nettamente superiore a quello degli altri algoritmi.

	Dijkstra Mod.	Iwata Mod.	Mittal	Martin Santos	Dijkstra Semplice
Media percorsi totali	3.7	7.4	2.7	> 60	1
Media percorsi buoni	1.4	2.1	1.0	3.6	0.5
Ritardo	27.1	32.7	24.9	30.6	n.d.
Jitter	17.6	19.7	16.3	18.3	n.d.
Perdite	0.84	0.84	0.85	0.87	n.d.
Tempo	0.085	0.160	0.136	0.720	n.d.

Valori relativi al primo path trovato escluso l'algoritmo di Dijkstra

Figura 5.8 Analisi dei risultati per soglia sulle perdite pari 1.3% e soglie su ritardo e jitter pari a 50 ms

In realtà quest'algoritmo riesce a trovare in questa particolare topologia di rete anche più di mille percorsi per ogni coppia di nodi: si è deciso di fissare un limite di percorsi totali per MS pari a 60 (poiché i percorsi calcolati da

MS sono ordinati secondo ritardo crescente, da un certo punto in poi questi cominceranno ad avere un ritardo maggiore di quello della soglia fissata sul ritardo, per cui non ha senso procedere con il calcolo). Notiamo che fra gli altri tre algoritmi, quello che dà prestazioni migliori è Im. Analizziamo ora le ultime quattro righe della tabella. Esse si riferiscono ai valori relativi al:

- primo percorso buono calcolato quando il primo path calcolato con l'algoritmo di Dijkstra non ha successo;
- secondo percorso buono ottenuto quando il primo path calcolato con Dijkstra è feasible.

Il dato più interessante è legato al tempo di calcolo: l'algoritmo MS a causa dell'elevata complessità computazionale, ha dei tempi di calcolo molto superiori rispetto agli altri. Non si nota invece un'eccessiva differenza riguardo ai valori medi legati alle metriche.

La figura 5.9 mostra invece le percentuali di successo nel trovare l'i-esimo percorso buono per i=1,..5 e i relativi tempi di calcolo. Si nota come anche in questo contesto MS sia l'algoritmo che fornisce decisamente i migliori risultati in termini di percentuale di successo. L'unico che riesce a fornire buone percentuali per i>1 è Im, che rispetto a MS ha comunque tempi di calcolo nettamente inferiori.

5.2.2 Scenario 2

In questo scenario i vincoli su ritardo e jitter sono fissati a 30 ms mentre viene fatto variare il limite sulle perdite.

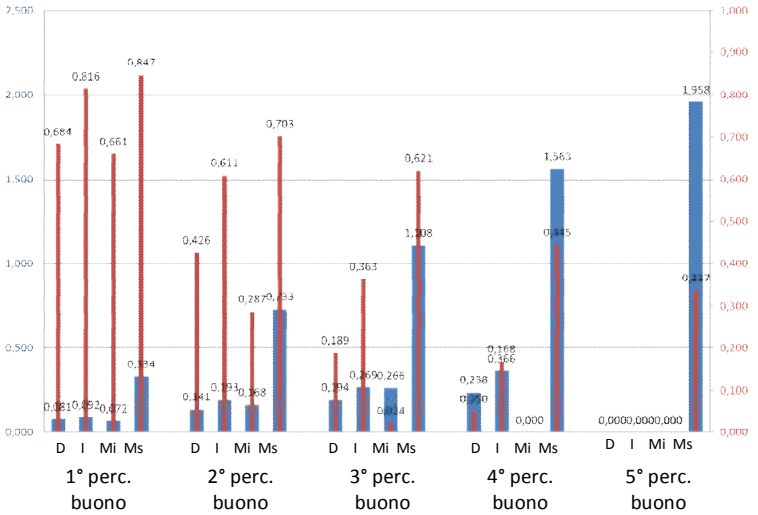


Figura 5.9 Percentuali di successo (in rosso) e tempi di calcolo (in blu) dell'i-esimo percorso per soglie sulle perdite, ritardo e jitter pari 1.3% e 50 ms

Anche stavolta il primo confronto riguarda le percentuali di successo dell'algoritmo di Dijkstra e dei nostri quattro algoritmi implementati, come è possibile vedere dalla figura 5.10. Notiamo che, rispetto al caso precedente, le prestazioni sono più "appiattite". Questo è dovuto ai vincoli su ritardo e jitter, che, come già visto, sono piuttosto stringeti. La differenza in percentuale di successo fra MS e Dijkstra originale è più alta per soglie sulle perdite basse. Questo grafico suggerisce di andare a vedere nel dettaglio i risultati relativi al punto (1,30+30).

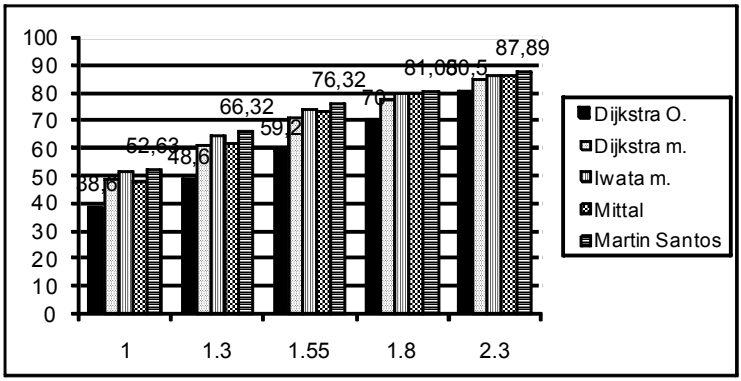


Figura 5.10 Percentuale di successo degli algortimi con soglie su ritardo e jitter pari a 30 ms

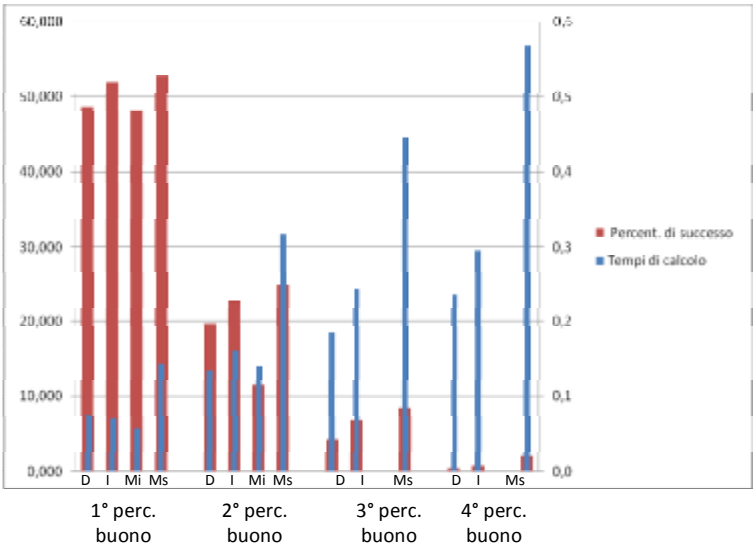


Figura 5.11. Percentuali di successo e tempi di calcolo dell'i-esimo percorso buono con soglie su ritardo e jitter pari a 30 ms e soglia sulle perdite pari a 1%

Dal grafico di figura 5.11 si notano ancora una volta le alti percentuali di successo del MS rispetto agli altri algoritmi. L'algoritmo IM è quello che si avvicina di più, mostrando però tempi di calcolo nettamente inferiori. Notiamo come in questo scenario nessun algoritmo riesca a calcolare cinque percorsi feasible.

5.2.3 Altri scenari

I risultati ottenuti con gli altri scenari non aggiungono esiti di rilievo rispetto a queli già commentati. Dalle figure 5.12 e 5.13 si notano che le percentuali di successo dei vari algoritmi sono abbastanza vicine: ciò è coerente con quanto visto prima, ovvero che per soglie su ritardo e jitter inferiori ai 30 ms le prestazioni degli algoritmi tendenzialmente son simili.

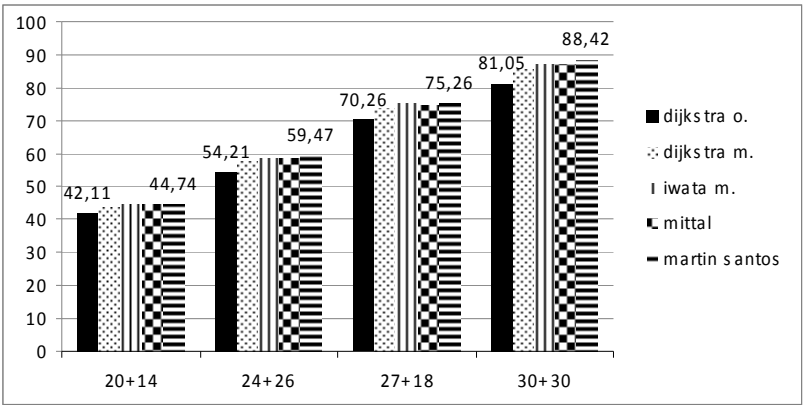


Figura 5.12. Percentuale di successo degli algortimi con soglia sulle perdite a 2.5%

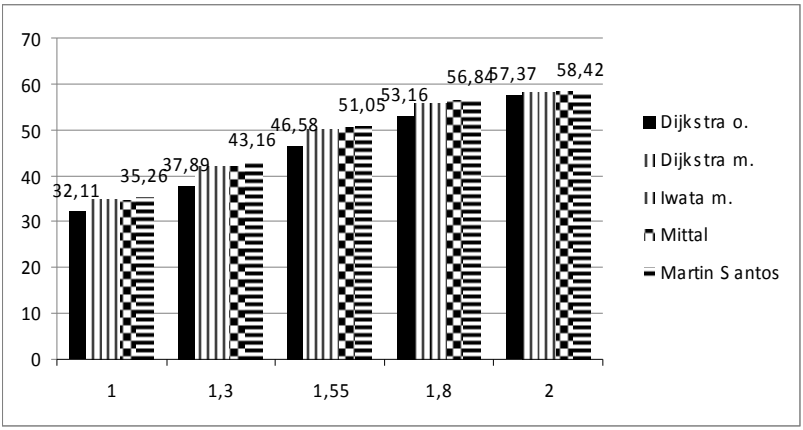


Figura 5.13 Perc. di successo degli algortimi con soglia su ritardo e jitter pari a 20ms

5.2.4 Anomalie del Mittal

Come accennato nei paragrafi precedenti, l'algoritmo di Mittal mostra prestazioni scadenti. Cerchiamo di spiegare con un esempio il motivo di questa anomalia.

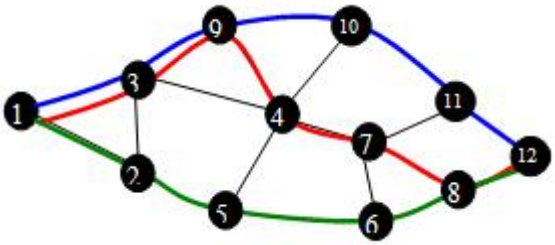


Figura 5.14 Anomalia del Mittal

Supponiamo di avere una topologia come quella in figura 5.14 e cerchiamo di calcolare più percorsi tra il nodo 1 e il nodo 12 con Mi. Inizialmente con il Dijkstra viene calcolato lo shortest path, che supponiamo sia quello in blu.

Quando viene effettuato il prune dei link, verranno trovati attraverso Dijkstra dei nuovi percorsi, in parte o del tutto diversi da quello precedente (supponiamo siano quelli in rosso e in verde). Ma quando il prune dei link viene effettuato sui nuovi percorsi, succede che il percorso migliore coinciderà con lo shortest path blu, per cui non sono trovati nuovi percorsi. Il risultato è che il numero di percorsi trovati alla fine è piuttosto ridotto.

Conclusioni

Gli obiettivi di questo lavoro di tesi sono stati il progetto di un'architettura con caratteristiche di Ingegneria del Traffico per reti wireless mesh e l'implementazione e l'analisi prestazionale di algoritmi per la selezione di percorsi basati su vincoli multipli di QoS.

Sono state descritte le caratteristiche principali delle WMN, che rappresentano un'evoluzione delle reti WLAN 802.11. A causa dell'interesse crescente verso quest'architettura, è necessario fornire supporto ad una nuova generazione di applicazioni richiedenti garanzie di QoS. La soluzione è ricorrere a tecniche di Traffic Engineering (TE) e MPLS, le cui funzionalità principali sono state riassunte.

E' stata quindi proposta un'architettura di rete WMN che supporti le tecniche di TE. Essa è costituita da diversi blocchi funzionali, appartenenti sia al piano di controllo (che si occupa dei meccanismi di segnalazione, instaurazione e gestione dei percorsi), sia al piano dati (che interviene direttamente sull'inoltro dei frame dati). Anche se quest'architettura ricalca quella utilizzata nelle reti cablate, è necessario, nel definire i blocchi funzionali, tenere conto sia delle problematiche del canale wireless sia del fatto che la rete WMN opererà presumibilmente a livello 2, specialmente se si fa riferimento al nascente standard IEEE 802.11s. Per questi motivi alcuni di questi elementi vanno riprogettati ad hoc, come ad esempio il protocollo

di routing e il Path Computation Element; altri invece possono essere ottenuti modificando entità già utilizzate in altre applicazioni, come il protocollo di segnalazione o il meccanismo di label switching.
E' stato quindi descritto il protocollo di routing che si è deciso di utilizzare, RA-OLSR. Tale protocollo, distribuito, link-state e preventivo, deve essere opportunamente modificato per poter trasportare più metriche di tipo radio-aware: sono state allora spiegate le modalità con cui tali modifiche possono essere realizzate. E' stato inoltre descritto il Traffic Engineering Database, che memorizzerà le informazioni trasportate dal protocollo di routing.
A questo punto ci si è concentrati sul PCE: lo scopo è stato lo sviluppo di algoritmi di QoS path selection alternativi a quelli già esistenti. Sono state sviluppate in particolare due euristiche, una modifica al comune algoritmo di Dijkstra, per aumentare il numero di percorsi trovati pur mantenendo la stessa complessità computazionale, una estensione dell'algoritmo di Iwata, che sfrutta sempre la prima euristica. Per confrontarne le prestazioni, sono stati implementati altri due algoritmi, quello di Mittal e quello di Martin Santos (MS). A differenza degli altri, quest'ultimo è un algoritmo esatto, che rappresenta perciò il limite superiore alle prestazioni in termini di percentuale di successo.
Abbiamo quindi generato alcune topologie di rete e ne abbiamo scelta una su cui testare i nostri algoritmi. I test sono stati effettuati variando le soglie sulle metriche introdotte (ritardo, jitter e perdite). I risultati ottenuti hanno dimostrato che l'algoritmo di Iwata modificato riesce ad ottenere una percentuale di successo molto vicina a quella di MS, mentre gli altri due forniscono prestazioni più scadenti. Anche per il numero medio di percorsi soddisfacenti le richieste di QoS l'algoritmo di Martin Santos è quello che

fornisce i migliori risultati in assoluto, e anche in questo caso l'algoritmo di Iwata modificato è l'unico che offre prestazioni paragonabili. E' stato quindi effettuato un ulteriore confronto tra questi due algoritmi in termini di tempo medio di calcolo di un percorso: il risultato è che i tempi di calcolo del Martin Santos sono tuttavia notevolmente superiori rispetto a quelli dell'algoritmo di Iwata modificato,.

Concludendo, l'algoritmo di Iwata modificato fornisce prestazioni in termini di percentuali di successo e numero medio di percorsi 'buoni' paragonabili a quelli del Martin Santos, mostrando però tempi medi di calcolo dei percorsi decisamente più bassi. Per questo motivo, esso rappresenta un utile strumento nel contesto di un'architettura WMN con caratteristiche di TE, poiché non richiede risorse di calcolo elevate e può quindi essere ospitato sui nodi mesh.

Bibliografia

[1] Y. Zhang, J. Luo, H.Hu, *Wireless Mesh Networking – Architectures, Protocols and Standards,* Auerbach Publications, Part III, Cap. 12, 2007

[2] Draft *P802.11s™/D1.01*, IEEE, marzo 2007

[3] IETF RFC 3031, *MPLS Architecture*, gennaio 2001, www.ietf.org/rfc/rfc3031.txt

[4] IETF RFC 3209, *RSVP-TE: Extensions to RSVP for LSP Tunnels*, dicembre 2001, www.ietf.org/rfc/rfc3209.txt

[5] IETF RFC 3212, *Constraint – Based LSP Setup using LDP,* gennaio 2002, www.ietf.org/rfc/rfc3212.txt

[6] IETF RFC 3468, *The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signalling protocols*, febbraio 2003, www.ietf.org/rfc/rfc3468.txt

[7] A.Acharya, S. Ganu, A. Misra, *DCMA: A Label Switching MAC for Efficient Packet Forwarding in Multihop Wireless Network*, IEEE Journal on Selected Areas In Communications, Vol.24, No.11, novembre 2006

[8] H. Liu, D. Raychaudhuri, *Label Switched Multi-path Forwarding in Wireless Ad-hoc Networks*, IEEE Proceedings of the 3rd Int'l Conf. On Pervasive Computing and Communications Workshops, 2005

[9] R. Wei, M. Wu, T. Yu, *LSMR: A Label Switching Multipath Routing Protocol for Ad Hoc Networks*, IEEE Computer Society, 2007

[10] M. Bahr, *Proposed Routing for IEEE 802.11s WLAN Mesh Networks*, WICON'06, agosto 2006

[11] M. Jabeen, S.Khan, *Label Switch Path with guaranteed Quality of Service in Mobile ad Hoc Network*, IEEE 2006

[12] J. Chung, *Wireless Multiprotocol Label Switching (WMPLS),* Internet Draft, agosto 2002

[13] K. Srinivasan, J.Chung, H. Soo, *Performance Analysis of Wireless Multiprotocol Label Switching (WMPLS) Networks*, Proceedings of the Fifteenth Annual International Conference on Wireless Communications, luglio 2003

[14] I. Muccini, *Progetto e sviluppo di un'architettura wireless mesh network basata su standard IEEE 802.11s*, Tesi di Laurea specialistica in Ingegneria delle Telecomunicazioni, settembre 2007, par. 1.5

[15] D. Adami, C.Callegari, S. Giordano, M. Pagano, *A New Path Computation Algorithm and its Implementation in NS2*, 2007

[16] IETF RFC 3626, *Optimized Link State Routing Protocol*, ottobre 2003, *www.ietf.org/rfc/rfc3626.txt*

[17] D. Adami, C. Callegari, S. Giordano, M. Pagano, *Path Computation Algorithms in NS2*, SIMUTools 2008, March 3-7, 2008, Marseille, France, pp.1-2

[18] T. Tofoni, *MPLS – Fondamenti e applicazioni alle reti IP*, Ed. Hoepli Informatica, Ottobre 2003, Par. 7.5.2, pp. 259 – 261

[19] Z. Wang and J. Crowcroft, *Quality of Service Routing for Supporting Multimedia Applications*, IEEE Journal on Selected Areas in Communication, settembre 1996

[20] D. Adami, C. Callegari,S. Giordano, M. Pagano, *Path Computation Algorithms in NS2*, SIMUTools 2008, March 3-7, 2008, Marseille, France, pag. 2

[21] F. Kuipers, P. Van Mieghem, T. Korkmaz, M. Krunz, *An Overview of Constraint-Based Path Selection Algorithms for QoS Routing*, IEEE Communications Magazine, dicembre 2002

[22] M. S. Garey, D. S. Johnson, *Computers and Intractability: Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979

[23] E. Dijkstra, *A note on two problems with connection in the graphs, Numerical Mathematics 1*, 1951, pp. 395-412

[24] A. Iwata et al., *ATM Routine Algorithms with Multiple QoS Requirements for Multimedia Internetworking*, IEICE Trans. And Commun., vol. E79-B, no. 8, 1996, pp. 999-1006

[25] S. Mittal, *Implementation of K-shortest path Dijkstra Algorithm used in All-Optical Data Communication Networks*, SIE 546 Project, 2004

[26] R. Mewanou, S. Pierre, *Dynamic Routing Algorithms In All-OpticalNetworks*, CCECE 2003

[27] E. Martins, M. Pascoal, *A New Implementation of Yen's Ranking Loopless Paths Algorithm*, ottobre 2000

[28] R. Hoffman, R. R. Pavley, *A Method for the solution of the n-th best path problem*, Journal of the Association for Computing Machinery, 6:506-514, 1959

[29] D. Epstein, *Finding the k shortest paths*, SIAM Journal on Computing, 28:652-673, 1998

[30] J. Y. Yen, *An Algorithm for Finding Shortest Routes from All Source Nodes to a Given Destination in General Network*, Quart. of Applied Math., Vol.27, No.4 (gennaio 1970), pp. 526-530

[31] BRITE, Boston University Representative Internet Topology gEnerator, *http://www.cs.bu.edu/BRITE/*

[32] F. Baccelli, M. Klein, M. Lebourges, S. Zuyev, *Stochastic Geometry and Arhitecture of Communication Networks*, settembre 1995

[33] Delaunay triangulation, *http://en.wikipedia.org/wiki/Delaunay_triangulation*