FACOLTÀ DI INGEGNERIA
Corso di Laurea Specialistica in Ingegneria Informatica

Reti Radiomobili II

Reti Mesh

Analisi dei protocolli e delle metriche energetiche proposte, con focus su BATMAN-adv e proposta di una variante energeticamente efficiente.

Docente:
Ing. Floriano De Rango

Studente:
Natale Vinto
137523

ANNO ACCADEMICO 2011/2012

Indice generale

Introduzione

L'oggetto del seguente lavoro è stato lo studio delle Reti Wireless Mesh, definite nello standard IEEE 802.11s, e dei principali algoritmi di routing applicati in tale contesto, con particolare attenzione sulle metriche energy-aware e le ottimizzazioni in tal senso.

Sebbene 802.11s sia stato ormai ampiamente recepito dalla comunità scientifica ed implementato largamente, rimane tuttavia aperta la questione sulla ricerca di un algoritmo ottimale che possa ottimizzare i vari indici prestazionali dei nodi coinvolti nella rete, con recente attenzione anche all'aspetto energetico degli stessi. Poiché si tratta di nodi mobili eterogenei in una rete senza nodo coordinatore, risulta sempre più strategica la scelta di politiche energy-aware per la sopravvivenza dei nodi stessi durante i cicli di trasmissione all'interno della network che possano rispettare i criteri di affidabilità, velocità e tolleranza ai guasti richiesti per una buona comunicazione.

E' stata fatta un'ampia panoramica su 802.11s, partendo dalla più recente definizione dello IEEE (2012), analizzandone il funzionamento ed il meccanismo di risparmio energetico. A partire dalla metrica *airtime* definita dello standard, sono state analizzate diverse metriche energy aware, con *survey* approfondito sul protocollo EAPSM/EAPM, il quale adotta una metrica energetica in 802.11s raggiungendo buoni risultati in termini di prestazioni e tempo di vita del sistema.

Successivamente viene analizzato il protocollo OLSR, con i vari adattamenti per il mesh (olsrd) a partire dal RFC di partenza, fino alla considerazioni in merito alla disseminazione energetica, con le strategie sugli MPR e sul *willingness* attualmente in uso, e l'analisi della soluzione proposta con il protocollo EE-OLSR.

Viene presentato infine il protocollo BATMAN, sorto per sopperire alle limitazione di olsrd, e la recente variante di livello 2 BATMAN-adv implementato solo su kernel Linux, presente stabilmente nella mainline del kernel insieme a 802.11s. Vengono fornite le caratteristiche salienti ed analizzate le funzionalità, con i risultati di un test reale eseguito nel laboratorio di Telecomunicazioni del dipartimento, e proponendo infine delle ottimizzazioni per la considerazione dell'aspetto energetico nel protocollo, che fino ad ora non va oltre il Power Saving di 802.11.

1. Reti Mesh

In una rete mesh ciascun nodo partecipa alle trasmissioni all'interno della rete sia disseminando le proprie, sia fungendo da relay per gli altri nodi. Il risultato di tale comportamento è una topologia ibrida, molto simile a quella delle MANET, le quali però differiscono per il problema della mobilità dei nodi, non direttamente previsto in una rete mesh.

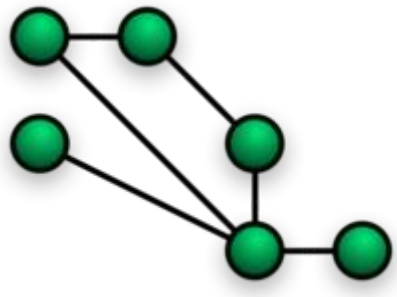


Fig 1: Esempio topologia rete mesh

I nodi scelgono automaticamente il path migliore (solitamente in termini di velocità e affidabilità), interpretando pienamente il concetto di routing dinamico. Le WMN (Wireless Mesh Network) rappresentano dunque una valida alternativa per la copertura wireless a lungo raggio o la propagazione e la condivisione di Internet poiché il segnale può essere replicato da nodi stessi nella rete, i quali oltre a fungere da nodi per lo smistamento del traffico locale, possono diventare mezzi di accesso alla rete esterna senza la necessità di AP per ogni sottoinsieme della rete.

1.1 Protocolli di routing

Poiché il livello fisico (PHY) e il livello datalink (MAC) sono largamente implementati nei vari chipset “Wi-Fi” in commercio secondo le specifiche dello standard IEEE 802.11, ciò che rende un nodo partecipante ad una rete Mesh è l'utilizzazione di alcuni algoritmi di routing al Layer 3 o al Layer 2 dello stack ISO/OSI, insieme a delle accortezze su MAC.

Quelli attualmente più usati sono 802.11s e BATMAN-adv per il Layer 2, e OLSR, BATMAN e Babel per il Layer 3. Vi sono altre implementazioni proprietarie non trattate in questo documento che pone il focus sulle implementazioni open source, in particolare quelle su kernel Linux.

Sebbene i protocolli di routing siano per definizione appartenenti al Layer 3, essi hanno lo svantaggio di produrre molto overhead, il quale può ridurre il throughput e richiedere a priori una potenza di calcolo maggiore sul nodo, come è suggerito per l'utilizzo di OLSR. I protocolli che operano a livello 2 invece forniscono uno switching che permette di ridurre il traffico di segnalazione, pertanto potendo aumentare il throughput senza richiedere necessariamente potenza di calcolo troppo elevate (e quindi disseminando verosimilmente meno energia).

1.2 802.11s

IEEE 802.11s è lo standard per il mesh networking delle reti IEEE 802.11; consente alle station 802.11 di formare reti auto-configurate multi-hop che supportano trasmissioni dati sia broadcast/multicast che unicast [1]. Esso rappresenta una soluzione basata su MAC per le WMN, fornendo le funzionalità di routing a livello MAC per la selezione del path migliore.

802.11s permette comunicazioni dirette fra due nodi senza il bisogno di un AP (Access Point) intermediario della comunicazione. Con tale standard, i nodi possono formare una rete multi-hop dove ogni link della rete è wireless. Pertanto non è richiesta un'infrastruttura

cablata per effettuare il setup della rete e ciò garantisce la massima libertà e adattabilità. Lo standard IEEE 802.11 definisce un WDS (Wireless Distribution System) dove solitamente più AP sono interconnessi utilizzando una rete cablata, mentre 802.11s rimpiazza la necessità del collegamento cablato con i link wireless, garantendo flessibilità e minor costo.

Lo standard definisce tre tipologie di nodi:

- Mesh Point (MP)
- Mesh Portal (MPP)
- Mesh Access Point (MAP)

Un esempio di una topologia derivante da queste descrizioni è rappresentato in Fig. 2

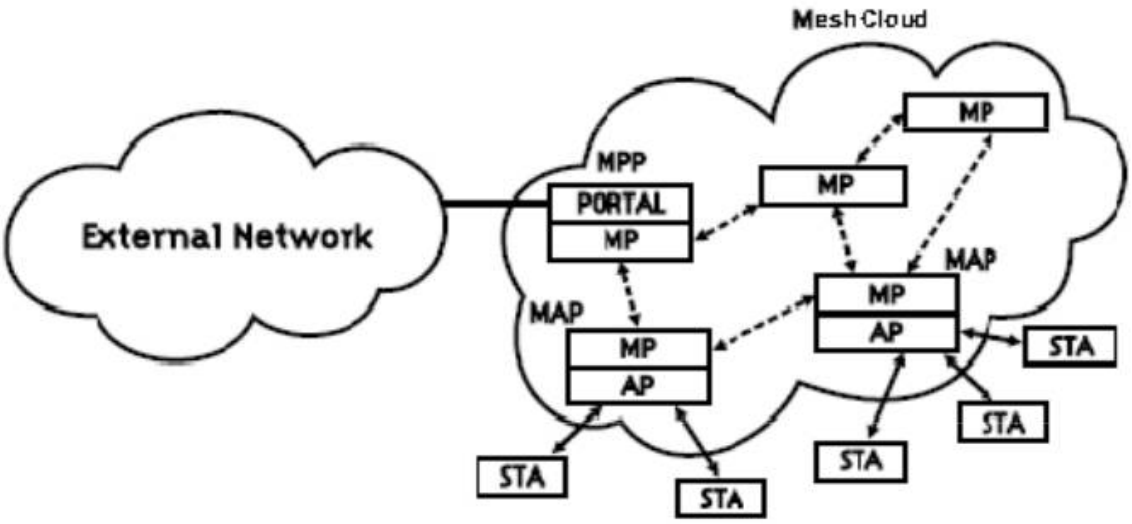


Fig. 2: Esempio mesh network

Ciascun nodo ha la capacità di inoltrare le trame ricevute destinate ad altri nodi e può quindi svolgere il lavoro di *relay* come precedentemente discusso.

I nodi Station (STA) sono i client e sono coloro i quali richiedono i servizi, ma non inoltrano

i frame, né partecipano nei maccanismi di path discovery.

Un Mesh Point (MP) supporta il protocollo Peer Link Management, utilizzato per la scoperta dei nodi vicini e per tenere traccia di questi. Tale discovery è pertanto limitato ai nodi che sono nel raggio di azione del Mesh Point. Per la comunicazione con i nodi che sono distanti più di un hop, un MP fa uso del protocollo HWMP (Hybrid Wireless Mesh Protocol), che verrà discusso in seguito.

I Mesh Portal (MPP) sono i nodi che fungono da gateway per l'accesso alla rete esterna. Essi sono connessi sia alla rete mesh, sia a quella esterna e pertanto hanno necessità di effettuare il bridge di almeno due interfacce di rete per fornire tale funzionalità di gateway.

Un Mesh Access Point (MAP) è un AP tradizionale con l'aggiuntiva funzionalità di mesh integrata, così da poter servire sia da AP che da nodo di una rete mesh.

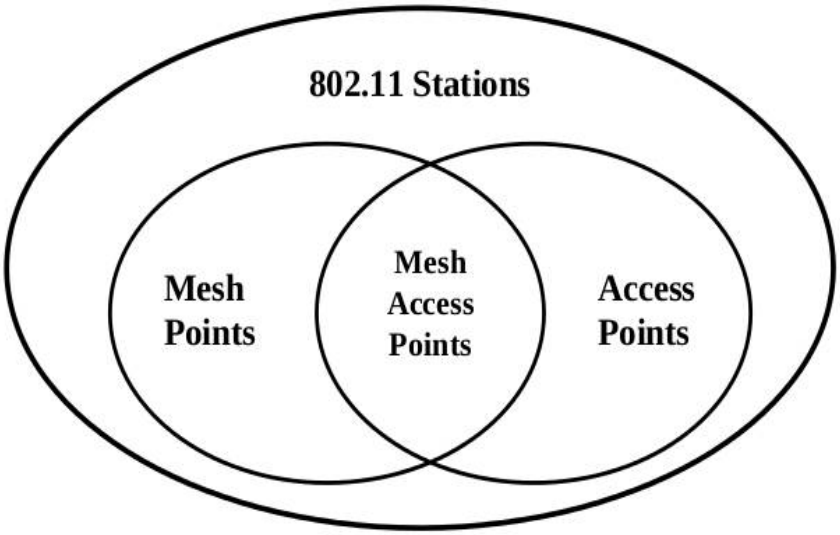


Fig.3: Diagramma degli insiemi di un mesh BSS

Nello scenario descritto tale mesh BSS è formato e operato da un insieme di servizi chiamati appunto servizi mesh. Nella descrizione del mesh in 802.11s [3] vi sono una serie di funzionalità che contengono tali servizi:

- Mesh Peer Management
- Mesh Security
- Mesh beaconing and syncronization

- Mesh Coordinator Function
- Mesh Power Management
- Mesh channel switching
- Extended address frame format
- Mesh path selection and forwarding
- Interworking con reti esterne
- Controllo di congestione intramesh

1.2.1 Struttura del frame 802.11s

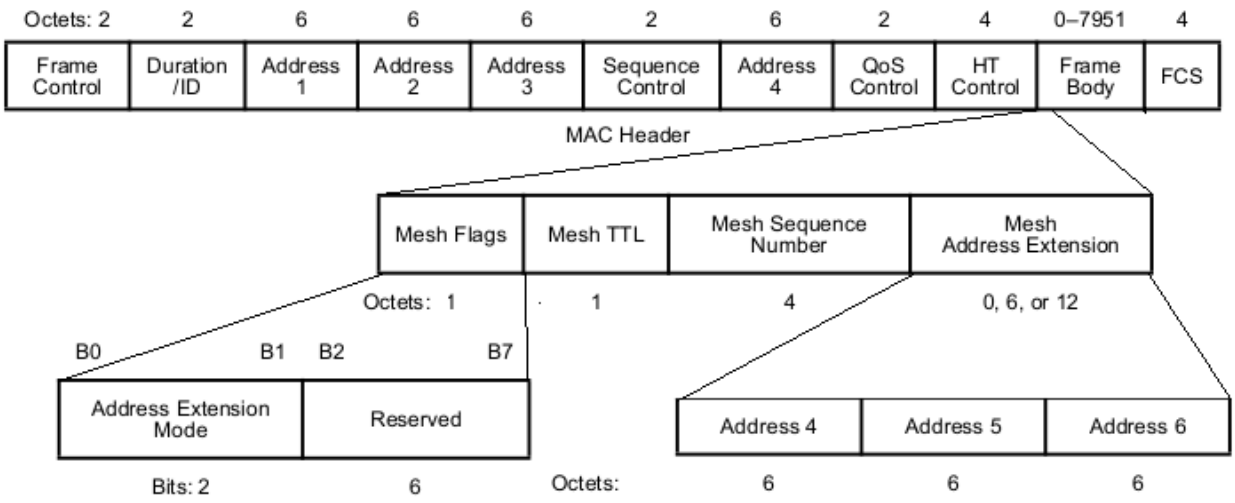


Fig. 4 : Mesh Header

1.2.2 MCF

Come si vede in Fig. 4, 802.11s estende il frame MAC tradizionale introducendo il MCF (Mesh Control Field) all'interno del body fino a 18 ottetti.

MCF è presente in un frame Mesh Data non frammentato, nel primo frammento del frame Mesh Data e nel frame di gestione dei subtype Action e Category Mutihop Action trasmessi

da una station mesh. Esso consiste in campo Mesh Flag, un campo per il mesh Time To Live (TTL), un sequence number mesh ed un campo per l'address extension. Il campo TTL ed il sequence number sono usati per prevenire cicli infiniti nella gestione dei frame mesh. I Mesh Point che comunicano su un singolo hop non hanno MCF all'interno delle loro trame.

I Mesh Flag indicano la presenza di ulteriori indirizzi MAC presenti nel MCF e gli address extension permettono un totale di sei campi address nel frame mesh, come esposto nell'esempio in Fig. 5.

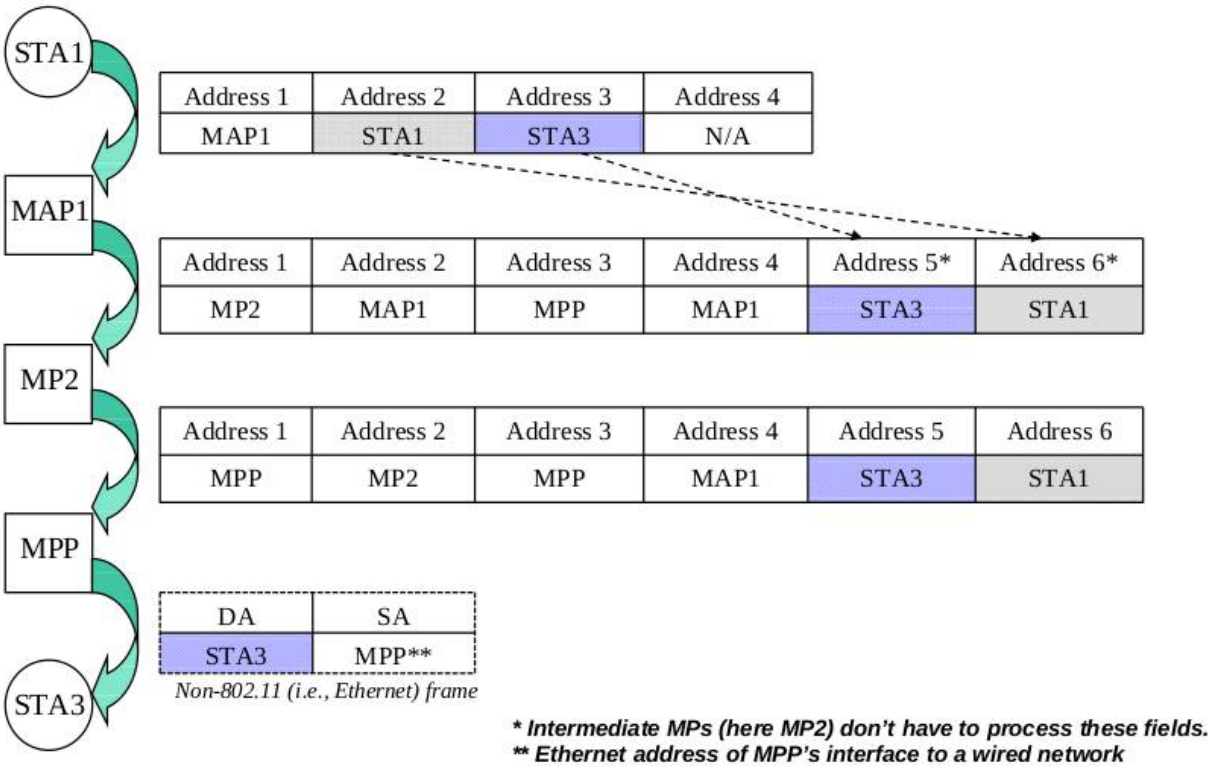


Fig 5: Comunicazione fra 802.11 station attraverso una rete mesh

Tale meccanismo è appunto utile quando il mittente ed il destinatario del frame non fanno parte della rete mesh, ma sono raggiungibili tramite il proxying dei Mesh Point.

1.2.3 Mesh beaconing

I beacon dei mesh point trasportano l'informazione riguardante il mesh e aiutano le altre

station mesh a rilevare la rete mesh e a partecipare alla stessa. I Mesh Point si rilevano fra di loro tramite uno *scanning* passivo o attivo. Sia il beacon specifico del mesh, sia i frame Probe di risposta contengono un Mesh ID, un elemento di configurazione che annuncia i servizi mesh e i parametri supportati dal Mesh Point trasmittente. Questa funzionalità abilita i MP a ricercare i peer adatti al tipo di trasmissione; una volta che è stato identificato un peer candidato, il MP usa il protocollo Mesh Peer Link Management per stabilire la connessione con un altro Mesh Point. Ed inoltre anche quando il link fisico non è più utilizzabile, i Mesh Point devono mantenere lo status del collegamento del peer per garantire una veloce riconnessione.

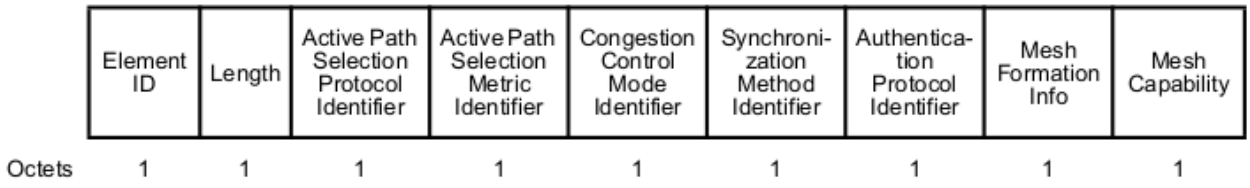


Fig. 6 : Mesh configuration element format

In Fig. 6 è visualizzato il Mesh Configuration element, usato per l'annuncio dei servizi mesh. Come già accennato, esso è contenuto nei frame Beacon e in quelli Probe Response trasmessi da una mesh station, ed è contenuto inoltre nelle trame del Mesh Peering Open e Mesh Peering Confirm [2].

1.2.4 Medium Access Control

I Mesh Point implementano una funzione di coordinamento (MCF) per l'accesso al mezzo. Tale funzione permette di trasmettere attraverso il protocollo di contesa EDCA, il quale è una variante migliorata del Distributed Coordination Function (DCF) del 802.11.

Attraverso la DCF, le stazioni trasmettono un singolo frame di lunghezza arbitraria. Con EDCA invece, le stazioni trasmettono frame multipli la cui durata totale della trasmissione non deve un certo limite, definito come TXOP (transmission opportunity), ed il ricevitore riconosce qualsiasi ricezione del frame avvenuta correttamente. Tale TXOP può essere

assegnato sia in base alla contesa che da un Hybrid Coordinator (HC). Un'altra differenza sta nelle quattro categorie di traffico con diverse priorità nell'accesso al mezzo e pertanto consente un supporto limitato alla qualità del servizio.

Ci sono due tipi di TXOP in MCF:

- Enhanced Distribuited Channel Access (EDCA)
- MCF Controlled Channel Access (MCCA)

Il TXOP EDCA è ottenuto da una mesh station dopo aver vinto una contesa EDCA, mentre il TXOP relativo al MCCA è ottenuto da una stazione mesh che sta conseguendo il controllo del mezzo wireless durante una MCCAOP. Tale MCCAOP è un intervallo di tempo per la trasmissione delle trame che è stato riservato mediante lo scambio di frame MCCA.

Se è in uso il Frequency Hopping (FH) nel livello fisico PHY, entrambi i TXOP di EDCA e MCCA non devono eccedere una quantità di tempo denominata *dot11MaxDwellTime.*

1.2.5 EDCA

EDCA è un meccanismo basato su contesa di accesso al canale HCF, di cui l'unità base di allocazione del permesso di trasmettere sul mezzo wireless è il TXOP appunto. Integra la qualità del servizio (QoS) introdotto in 802.11e con back-off con priorità. Vi sono pertanto degli appositi timer di accesso al canale e ciascun di essi deve mantenere una funzione di back-off, che ha un valore misurato in ogni slot di back-off.

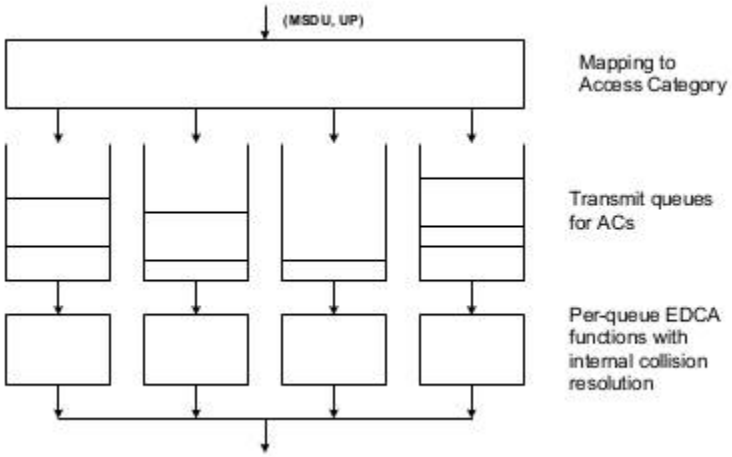
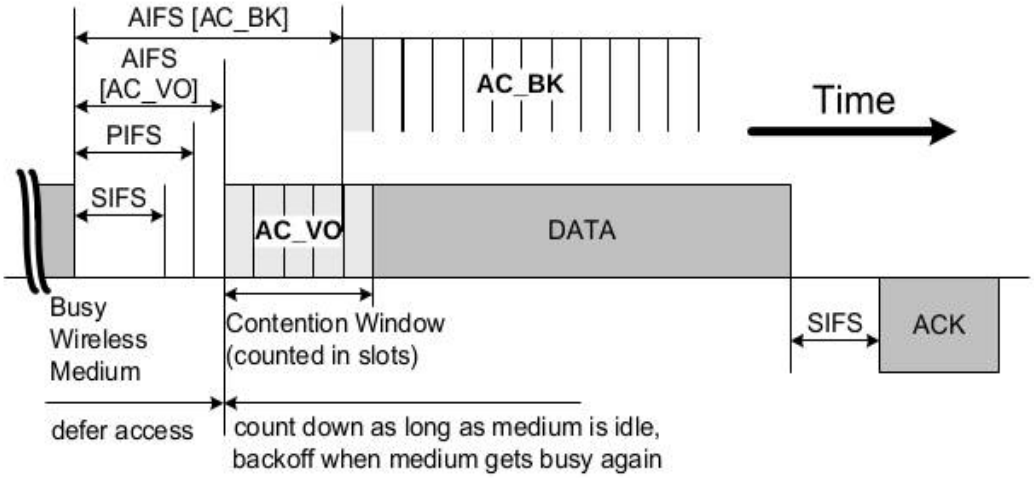


Fig. 7 Modello di implementazione di riferimento

Il modello mostrato in Fig. 7 illustra il mapping dal tipo di frame o di User Priority (UP)



verso le Access Category (AC); si notano le quattro code di trasmissione e i quattro EDCAF indipendenti.

Fig 7: Meccanismo EDCA

La durata di un AIFS[AC] è la durata derivata dal valore di AIFSN[AC] secondo la relazione:

$$AIFS[AC] = AIFSN[AC] \times aSlotTime + aSIFSTime$$

Il valore del AIFSN[AC] dovrebbe essere maggiore o uguale a 2. In un'infrastruttura BSS,

tale AIFSN[AC] è annunciato dall'Access Point nell'insieme di parametri EDCA nei frame di Beacon e Probe Response trasmessi dall'AP stesso. Un TXOP EDCA è abilitato ad ottenere un EDCAF quando quest'ultimo determina l'inizio di una trasmissione di uno scambio di sequenze di frame. Ciascun EDCAF dovrebbe fare una scelta per effettuare o l'inizializzazione della trasmissione di uno scambio di sequenze di frame per la funzione di accesso, o decrementare il timer di back-off per la funzione di accesso.

1.2.6 MCCA

MCCA è un metodo di accesso opzionale che abilita le station mesh all'accesso al mezzo wireless in tempi selezionati con la minima contesa possibile. Questo standard non impone l'utilizzato di MCCA a tutte le station mesh e dovrebbe essere usato da un sottoinsieme di stazioni mesh nel MBSS. Tuttavia le riserve MCCAOP dovrebbe essere effettuate solo dalle stazioni mesh che hanno il parametro *dot11MCCAActivated* abilitato e che operano sullo stesso canale. Le performance di MCCA sono degradate dal numero di partecipanti che non rispettano le riserve MCCAOP.

Una riserva MCCAOP specifica una schedulazione per le trasmissioni delle trame. I periodi temporali schedulati per le trasmissioni nella riserva sono chiamati appunto MCCAOP e lo scheduling è impostato fra chi possiede un MCCAOP e uno o più soggetti che rispondono al MCCAOP. Una volta effettuata la riserva, l'accesso al canale da parte delle station mesh con MCCA abilitato viene governato da alcune procedure di setup e la riserva MCCAOP è annunciata secondo altre procedure di advertising [4].

La riserva effettua lo scheduling di una serie di MCCAOP con una durata comune descritta in MCCAOP Duration nel campo della riserva MCCAOP. Questa serie è iniziata dopo un primo Beacon Delivery Traffic Indication Map (DTIM) seguito da una procedura di setup MCCAOP avvenuta con successo e terminata quando la riserva è terminata. Tale riserva definisce uno schema regolare per i MCCAOP, nell'intervallo DTIM di chi ha ottenuto il MCCAOP. La cardinalità degli MCCAOP nell'intervallo DTIM è data dal valore del subcampo MCCAOP Periodicity.

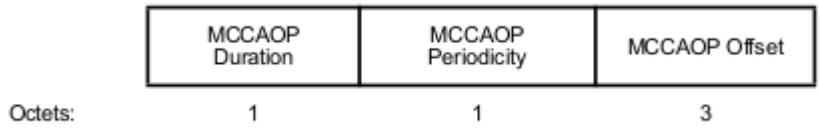


Fig. 8 : MCCAOP Reservation field

Il campo MCCAOP Offset specifica l'offset del primo MCCAOP schedulato dello schema di trasmissione relativo all'inizio dell'intervallo DTIM di chi possiede il MCCAOP. I MCCAOP seguenti sono separati da un intervallo di tempo con una durata uguale alla lunghezza del periodi di DTIM diviso il valore nel campo MCCAOP Periodicity. Un esempio di scheduling MCCAOP è mostrato in Fig. 9 dove il campo Periodicity ha valore 2.

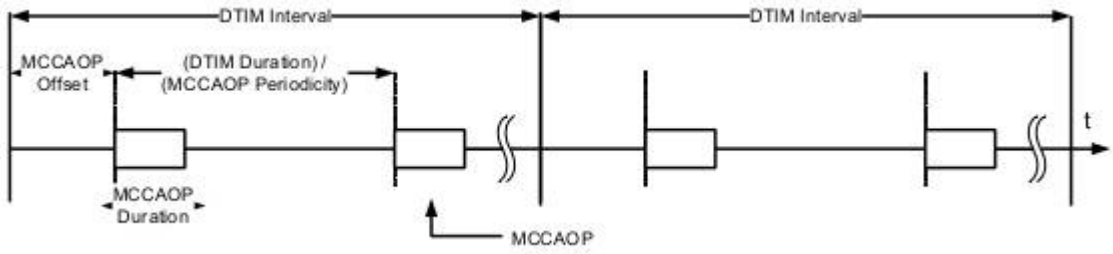


Fig 9: Esempio scheduling MCCAOP

1.2.7 Routing e HWMP

All'interno di una rete mesh, tutti i nodi usano la stessa metrica di path e lo stesso protocollo di selezione del path. La funzione di default per il calcolo del path è chiamata metrica airtime, ed è definita dalla seguente equazione calcolata da ciascun link come segue:

$$c_a = \left[O_{ca} + O_p + \frac{B_t}{r} \right] \frac{1}{1 - e_{fr}}$$

O_{ca} = overhead accesso al canale

O_{p} = overhead MAC

B_{t} = numero di bit nel frame di test (costante)

r = rate al quale il nodo vorrebbe trasmettere un frame di dimensione Bt con un error rate e_{fr} , basato sulle condizioni correnti dell'ambiente radio

Il routing in questo contesto corrisponde al calcolo del percorso (path) per l'inoltro dei frame (forwarding). Tale funzione in 802.11s è svolta dal Hybrid wireless mesh protocol (HWMP).

HWMP è un protocollo di selezione del path che combina la flessibilità della selezione del path on-demand con le estensioni della topologia ad albero dei protocolli proattivi. Sicché una siffatta combinazione di elementi reattivi e proattivi (in termini di protocolli) nell'HWMP abilita una selezione efficiente del path in una vasta varietà di reti mesh (con o senza l'accesso all'infrastruttura). Esso è basato su tre MMPDU:

- Path Request (PREQ)
- Path Replay (PREP)
- Path Error (PERR)

più un altro elemento chiamato Root Announcement (RANN) per l'annuncio del nodo radice.

HWMP usa un insieme comune di elementi protocollari, generazione e processing delle regole ispirato da Ad Hoc On-Demand Distance Vector (AODV) delle MANET, adattato per la selezione del path basato su indirizzi MAC e la conoscenza della metrica del link. Esso opera in tre diverse modalità:

- Schema di selezione del percorso opera con l'utilizzo del protocollo Radio

  Metric Ad-Hoc On-demand Distance Vector (RM-AODV)

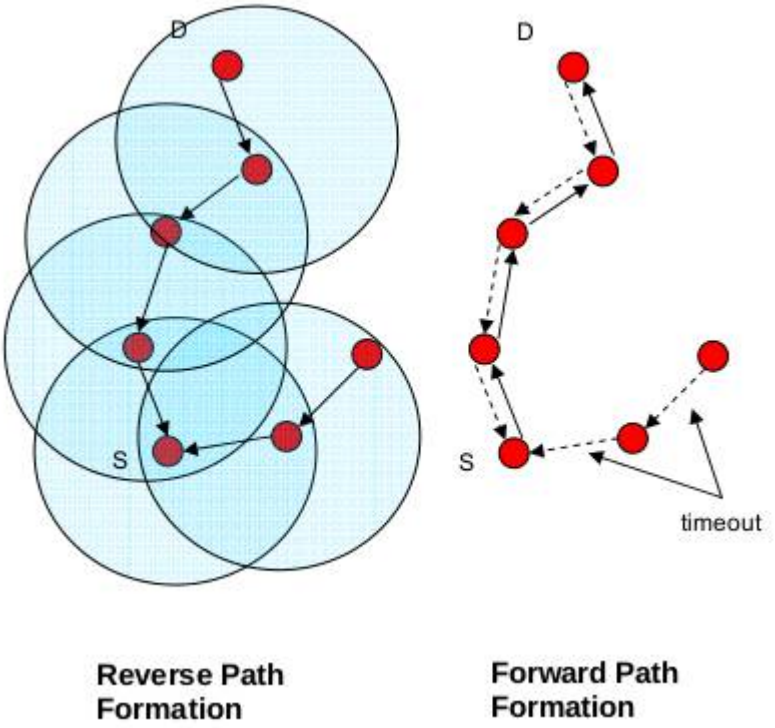


Fig. 10 : Routing on-demand in HWMP

- Approccio proattivo orientato all'uso di un albero dove uno specifico Mesh Point nella rete mesh diventa il MP radice (root). Invia in modo proattivo i messaggi PREQ per mantenere i path fra tutti i Mesh Point e la radice oppure invia messaggi di tipo RANN che abilitano i MP a costruire i percorsi verso il nodo root on-demand.

  E' importante ricordare a questo proposito che il RANN è usato per disseminare le metriche del path dalla radice della station mesh, ma la ricezione del RANN non stabilisce un percorso.

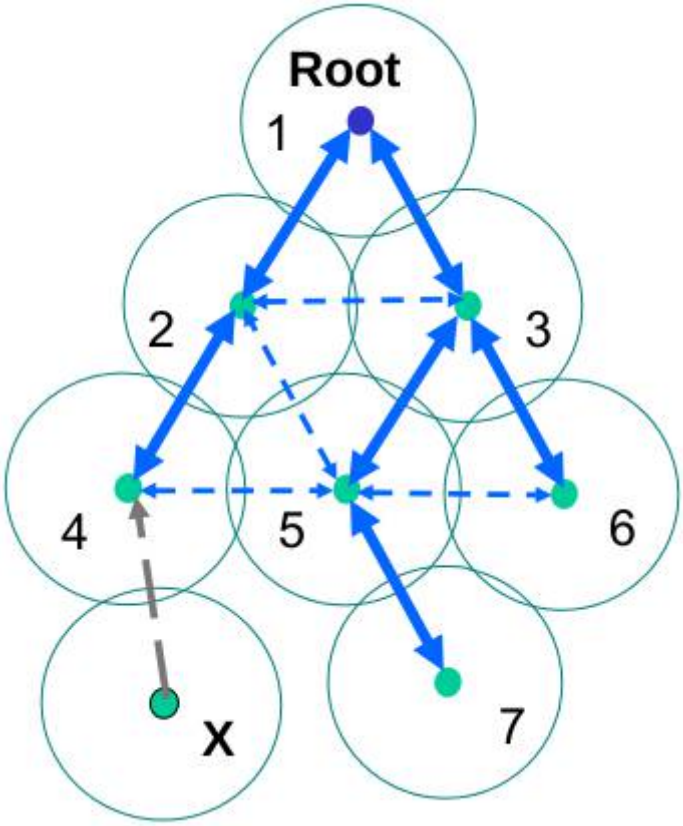


Fig. 11 : Routing ad albero in HWMP

- La selezione di path nulli indica che il Mesh Point non effettua l'inoltro delle trame.

Tali modalità non sono esclusive. Sia modalità on-demand, sia quella proattiva sono usate in modo concorrente poiché le modalità proattive sono estensioni di quella on-demand. Nella selezione del path in HWMP viene usato un meccanismo di Sequence Number il quale assicura che le stazioni mesh possano, in ogni istante di tempo, distinguere l'informazione riguardante il path corrente da quella relativa ad un percorso scaduto, al fine di mantenere una connettività priva di cicli (*loop-free*) . Ciascuna stazione mesh mantiene il suo proprio Sequence Number (HWMP SN), il quale viene propagato alle altre stazioni mesh negli elementi dell'HWMP.

1.2.8 Power Management

La gestione dell'energia in 802.11s riprende i concetti già utilizzati nel meccanismo BSS/IBSS con alcune modifiche. Il concetto base è banalmente quello che una stazione

mesh dovrebbe usare delle politiche energy-aware nel suo contesto di riferimento al fine di ridurre la disseminazione dell'energia. Una stazione mesh controlla ciascuno dei suoi *mesh peering* con una specifica modalità di gestione dell'energia ed imposta il *power mode* per l'accoppiamento mesh indipendentemente dagli altri accoppiati (*peer*), ma vengono anche fornite delle politiche per la gestione dei *nonpeer.* Inoltre, le stazioni dovrebbe avere la capacità di bufferizzare i frame e di effettuare il tracking del power mode per ciascuna modalità specifica per i suoi peer. Le station trasmettono gruppi di frame indirizzati dopo il frame di Beacon contenenti il DTIM quando ciascuno dei suoi mesh peer è nella modalità *light sleep* o *deep sleep* per il peering.

Sono definiti due differenti stati in merito alla gestione energetica:

- Awake
- Doze

La transizioni fra diversi stati è determinata dai power mode dei peer e dei nonpeer per

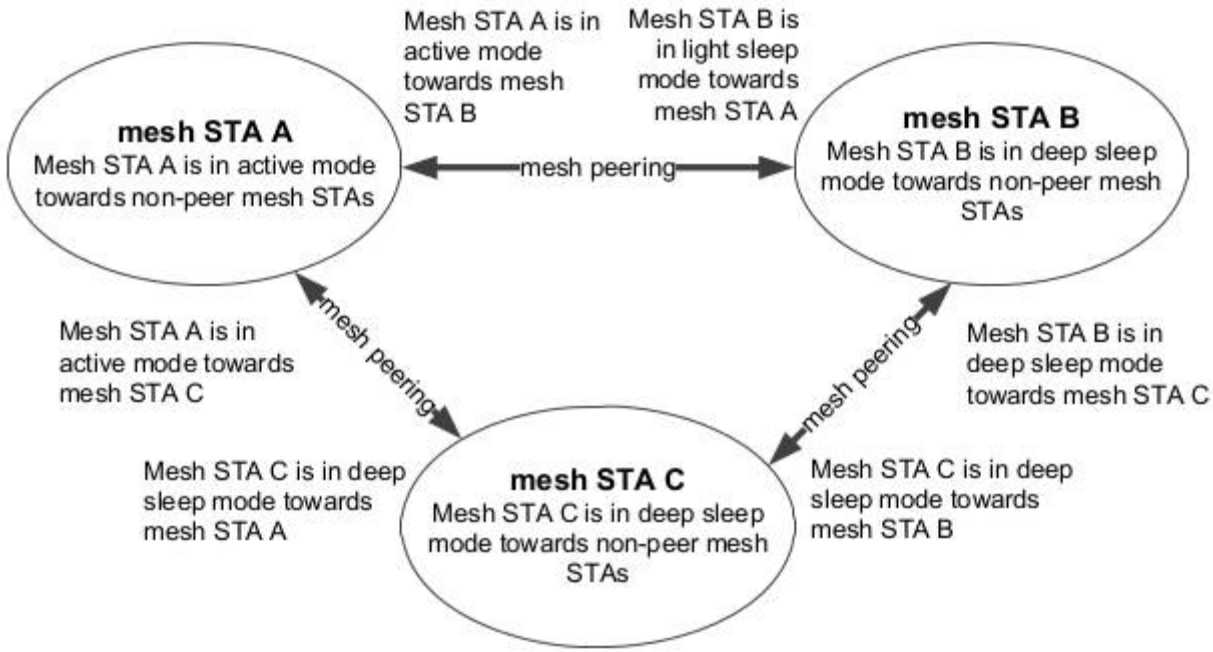


Fig. 11 : Status per modalità peer e nonpeer

ciascun nodo mesh. Il nodo è nello stato di Awake se il nodo non opera nello sleep mode o non ha terminato l'invio del gruppo di frame indirizzati dopo il suo Beacon DTIM, altrimenti è nello stato di Doze.

La differenza sostanziale nel power management in 802.11s sta nell'uso ridotto della frequenza di *beaconing*, limitata solo al tempo del DTIM, ed nella condivisione efficiente della *beaconing responsability*. Inoltre lo standard fornisce l'advertising della modalità del Power Save (PS) in maniera altrettanto efficiente, indicandola nei frame stessi di Beacon e dal bit PS nel campo del Frame Control. Definisce meccanismi per abilitare i Mesh Point ad essere posti nello stato Awake solo per porzioni di tempo richieste per la ricezione corrente, con un uso funzionale del *more data bit* e del *EOSP*.

La gestione della transizione del passaggio di stato fra Sleep e Awake è affidata ad un meccanismo basato su Announcemente Traffic Indication Message (ATIM) per il quale vi è una finestra garantita di tempo di awake dopo alcuni beacon periodi DTIM. Tale intervallo DTIM è definito come multiplo degli intervalli di beacon ed è unico globalmente nel mesh.

Il controllo della comunicazione è trasferito durante la finestra ATIM; può indicare traffico pendente, cambi nello stato del PS o flussi arrestati. I rimanenti tempi di awake dopo la finestra ATIM dipendono dal controllo stesso di comunicazione scambiato durante la finestra ATIM.

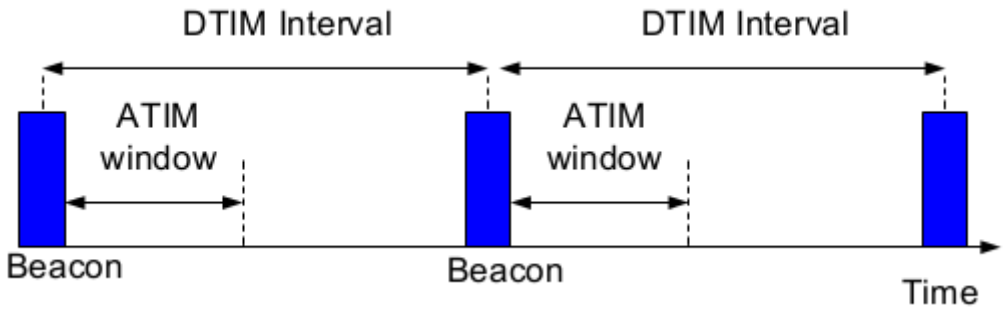


Fig. 12 : Operazioni awake-sleep basate su ATIM

1.2.9 Energy aware routing

La definizione della gestione delle politiche di risparmio energetico, estensione del meccanismo di Power Saving del 802.11 al caso di una rete multi-hop mesh, permette come si è visto un'amministrazione efficiente degli stati di attivo/inattivo delle varie stazioni, in riferimento alla topologia e alle comunicazioni in corso nella rete. Tuttavia la disseminazione dell'energia nella rete dipende da un fattore cruciale per le reti mesh, ovvero la scelta delle metriche per il protocollo di routing, pertanto è necessaria una ricerca di metriche di routing per HWMP energeticamente efficiente, aspetto cruciale in previsione di terminali alimentati da batteria per i quali definire le opportune politiche.

Il protocollo HWMP, come visto nel capitolo 1.2.8, combina l'approccio reattivo e quello proattivo risultato pertanto ibrido; in tale configurazione bisogna effettuare la ricerca di un routing energy-aware che non deteriori la bontà del protocollo in termini di prestazioni, possibilmente che almeno lo eguagli. Nella rete di WMN, l'energia della batteria è una risorsa molto limitata che necessita appunto un uso efficiente. Il fallimento delle operazioni di alcuni nodi può essere un aspetto negativo piuttosto rilevante in termini di performance sulla rete. L'esaurimento energetico dei nodi durante il routing inficia la disponibilità nel nodi e quindi percorsi scaduti o erronei. Il problema sostanziale inoltre nelle WMN è che una volta trovato il percorso più breve, questo viene usato per ogni comunicazione, consumando massivamente l'energia dei nodi coinvolti arrivando a creare un partizionamento della rete indesiderato. L'energia può essere dunque un fattore cruciale quando si viene designata una metrica per la selezione del path nelle WMN, e la ragione sta nel fatto che la durata funzionamento della rete è banalmente proporzionale in base alla

quantità di energia preservata. Conseguentemente potrebbe essere necessario “accontentarsi” di path subottimi in termini di pura metrica, ma non eventualmente in termini di disseminazione energetica. Questo assicura che l'energia dei nodi coinvolti nel percorso ottimale non venga esaurita e che la rete degradi in modo graduale nel complesso piuttosto che venire partizionata. Questo tipo di approccio viene denominato Energy Aware Routing [8]; suggerisce il calcolo di più percorsi fra il nodo sorgente e quello destinatario per ciascuno dei quali viene assegnata una probabilità di essere scelti, in base alla metrica energica.

Al fine di evitare il continuo calcolo delle diverse route per l'aggiornamento della probabilità, il protocollo definisce tre fasi:

- Setup
- Data Communication
- Route manteinance

La fase di Setup prevede una serie di *flooding* localizzati per cercare tutte le route e i relativi costi energetici dal nodo sorgente a quello destinatario, mentre nella fase successiva (Data) vengono inviati i dati in base alle informazioni ottenute dalla fase precedente. Infine, nell'ultima fase, viene utilizzato nuovamente il *flooding* selettivo per il mantenimento di tutte le route attive.

Energy Aware Routing è anche un protocollo di routing reattivo. E' un protocollo inizializzato dalla destinazione dove colui che usufruisce dei dati dà luogo alle route request e successivamente mantiene le route. Pertanto il meccanismo è simile a quello della diffusione in alcuni punti. Path multipli sono mantenuti dalla sorgente alla destinazione. La differenza sostanziale sta nel fatto che mentre con la diffusione i dati vengono inviati per mezzo di tutti i path ad intervalli regolari, l'energy aware routung invece usa un solo path in ogni istante. Di riflesso però, a causa della natura probabilistica delle scelta delle route, il protocollo potrebbe valutare costantemente le diverse route e scegliere la probabilità conseguentemente. Gli assegnamenti probabilistici e il calcolo dei costi di trasmissioni vengono effettuati nella fase di *Setup*.

Il nodo destinatario inizializza la trasmissione ed imposta il proprio costo definito come:

$$Cost(N_d)=0$$

Ciascun nodo intermedio inoltra la richiesta solo ai vicini che sono più vicini al nodo sorgente rispetto al nodo stesso, e più lontani dal nodo di destinazione. Pertanto al nodo N_i la richiesta è inviata solo ad un nodo vicino N_j che soddisfa tale equazione:

$$d(N_i,N_S)\geq d(N_j,N_S)$$
$$d(N_i,N_D)\leq d(N_j,N_D)$$

dove d sta per distanza.

Una volta ricevuta la richiesta, la metrica energetica per il nodo vicino che ha inviato la richiesta è calcolata ed aggiunta al costo totale del path. Dunque, data una richiesta inviata dal nodo N_i al nodo N_j , quest'ultimo calcola il costo del path come segue:

$$C_{NjNi}=Cost(N_i)+Metric(N_j,N_i)$$

I path con un alto costo energetico sono scartati, mentre quelli con basso costo energetico sono aggiunti alla Forwarding Table del nodo FT_j definita come:

$$FT_j=\{i \mid C_{N_j,N_i}\leq \alpha.(\min C_{N_j,N_K})\}$$

Il nodo N_j assegna una probabilità per ciascuno dei suoi vicini N_i presenti nella sua FT, e tale probabilità è inversamente proporzionale al costo:

$$P_{N_j,N_i}=\frac{\frac{1}{C_{N_j,N_i}}}{\sum\frac{1}{C_{N_j,N_k}}}$$

dove $k\in FT_j$

Ogni nodo N_j ha un numero di vicini per mezzo dei quali può instradare i pacchetti verso la destinazione. N_j successivamente calcola il costo medio del raggiungimento della destinazione in base alle Forwarding Table dei vicini:

$$Cost(N_j) = \sum P_{N_j,N_i} C_{N_j,N_i} i \in FT_j$$

Tale costo medio è immesso nel campo relativo del pacchetto di richiesta ed inoltrato verso il nodo sorgente secondo le politiche definite precedentemente. A questo punto, si entra nella fase Data Communication dove il nodo sorgente invia i pacchetti dati a tutti i suoi vicini presenti nella FT, con la probabilità che un vicino sia scelto uguale alla probabilità reperita nella FT. Ciascuno dei nodi intermedi inoltra i pacchetti dati verso un vicino presente nella sua FT scelto in modo casuale, con la probabilità che tale vicino sia scelto pari sempre alla probabilità presente nella sua FT. Il processo continua fintanto che il pacchetto non raggiunga il nodo di destinazione.

La metrica energetica che è usata per valutare le route è la parte fondamentale del protocollo (da cui appunto il nome). In base alla metrica, le caratteristiche del protocollo possono subire variazioni considerevoli. La metrica può includere informazioni in merito al costo associato all'utilizzo di un path, la stato energetico dei nodi coinvolti nel path, oppure la topologia della rete stessa.

Nella prima proposta del protocollo (Shan e Rabaey, 2002), veniva ripresa una semplice metrica per le reti ad hoc [10] definita come segue:

$$C_{ij} = e_{ij}^{\alpha} R_i^{\beta}$$

C_{ij} è il costo metrico fra i nodi i e j

e_{ij} è l'energia usata per trasmettere e ricevere sul link

R_i è l'energia residua al nodo i, normalizzata all'energia iniziale del nodo

Infine α e β sono fattori di peso che possono essere scelti per cercare il path a costo energetico minimo, il path con i nodi che hanno più energia, o una combinazione di entrambi gli scenari.

Questa prima versione di Energy Aware Routing è stata pensata e testata per una rete di sensori, in particolare sono stati considerati 76 nodi di cui 72 fissi e 4 mobili, di cui 47 trasmettevano ogni 10 secondi e 18 ogni 30.

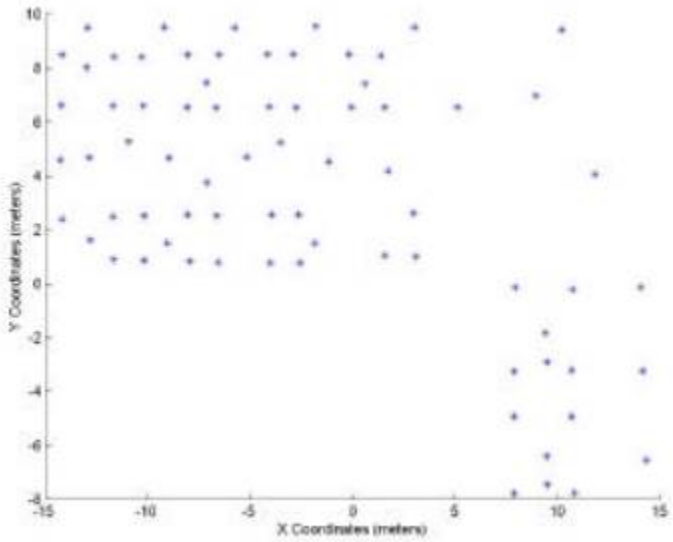


Fig. 13 : Layout dei nodi statici nella rete

E' stato fatto un confronto fra il direct diffusion e l'energy aware routing, astraendo il livello MAC senza contesa per l'accesso per l'invio dei dati nel mezzo assegnando un solo canale, scenario accettabile per mole di dati piccole (i pacchetti nella simulazione sono di 256 bit). Inizialmente i nodi sono stati dotati di un identico quantitativo di energia, rispettivamente in trasmissione 20 nJ/bit + 1pJ/bit/m^3 ed in ricezione 30nJ/bit.

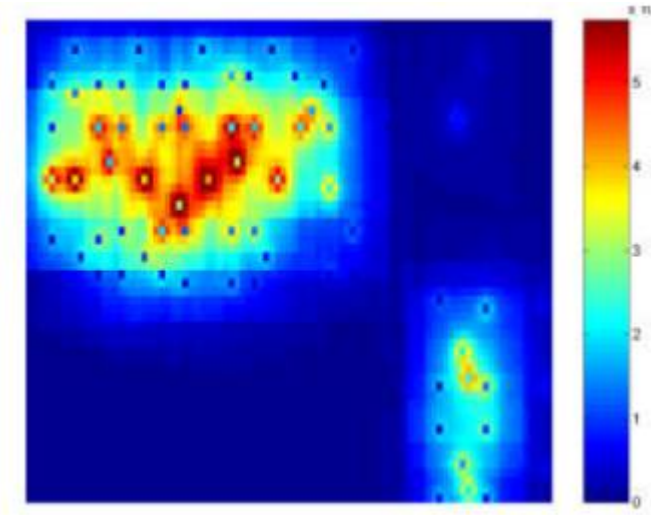


Fig. 14 : Consumo energetico in Energy Aware Routing

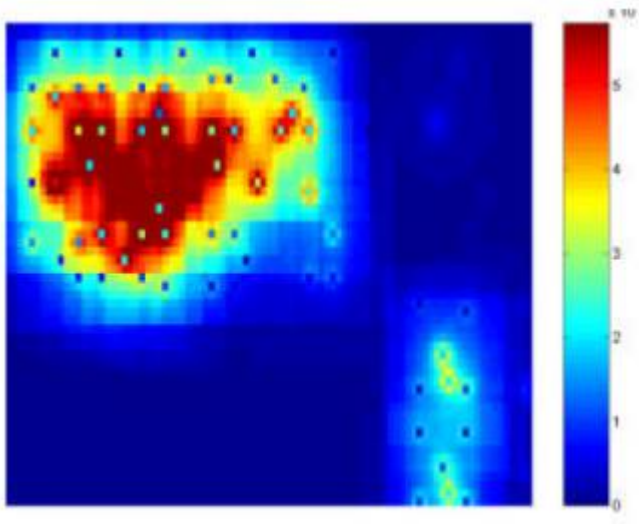


Fig. 15: Consumo energetico in Direct Diffusion

Dopo un'ora di simulazione, da come si evince anche analizzando le immagini in Fig. 14 e Fig. 15, l'energy aware routing diffonde il traffico sulla rete in maniera meno dispersiva in termini energetici. Conseguentemente, i nodi nel centro della rete conservano l'energia più a lungo. La media di energia consumata per nodo è un range che va da 14.99 mJ fino a 11.76 mJ, con un miglioramento del 21.5%. Nelle prove successive sono state fatte invece i test fino all'esaurimento energetico totale dei nodi, ottenendo come risultato che il direct diffusion consuma circa 1.4 volte in più dell'energy aware routing.

1.2.10 EAPM

Da questi risultati, una seconda versione del protocollo studiato appositamente per 802.11s è stata proposta (Mhlanga,Tolwal, Nntlatlpa 2009) con il nome di Energy Aware Path Selection Metric (EAPM)[11]. Si mantengono le considerazioni fatte per energy aware routing ma viene proposta una metrica modificata per le WMN alimentate da batterie, definita come segue:

$$C_{ij} = e_{ij}^{x_1} R_i^{-x_2} E_i^{x_3}$$

dove quindi, oltre alle grandezze già precedentemente definite, E_i è l'energia iniziale al

nodo i e x_1, x_2, x_3 sono fattori di peso non negativi per ciascun elemento. E' preferito il link che richiede meno energia di trasmissione $e_{ij}^{x_1}$, e allo stesso tempo è preferito un nodo trasmittente con alta energia residua $R_i^{-x_2}$ che apporta un migliore bilancio energetico. La combinazione dei fattori di peso determina le scelte effettuate in base alla metrica. Ad esempio la tripla { x_1, x_2, x_3 } nulla indica come path a costo minimo quello che un minore numeri di hop, mentre la combinazione {1,0,0} quello a minore costo energetico di trasmissione. Oppure se $x_2 = x_3$ è usata l'energia residua normalizzata, se invece $x_3 = 0$ allora è utilizzata l'energia assoluta residua.

Il diagramma di flusso in Fig. 16 riassume il funzionamento del path selection dell'energy aware routing per EAPM valido per 802.11s:

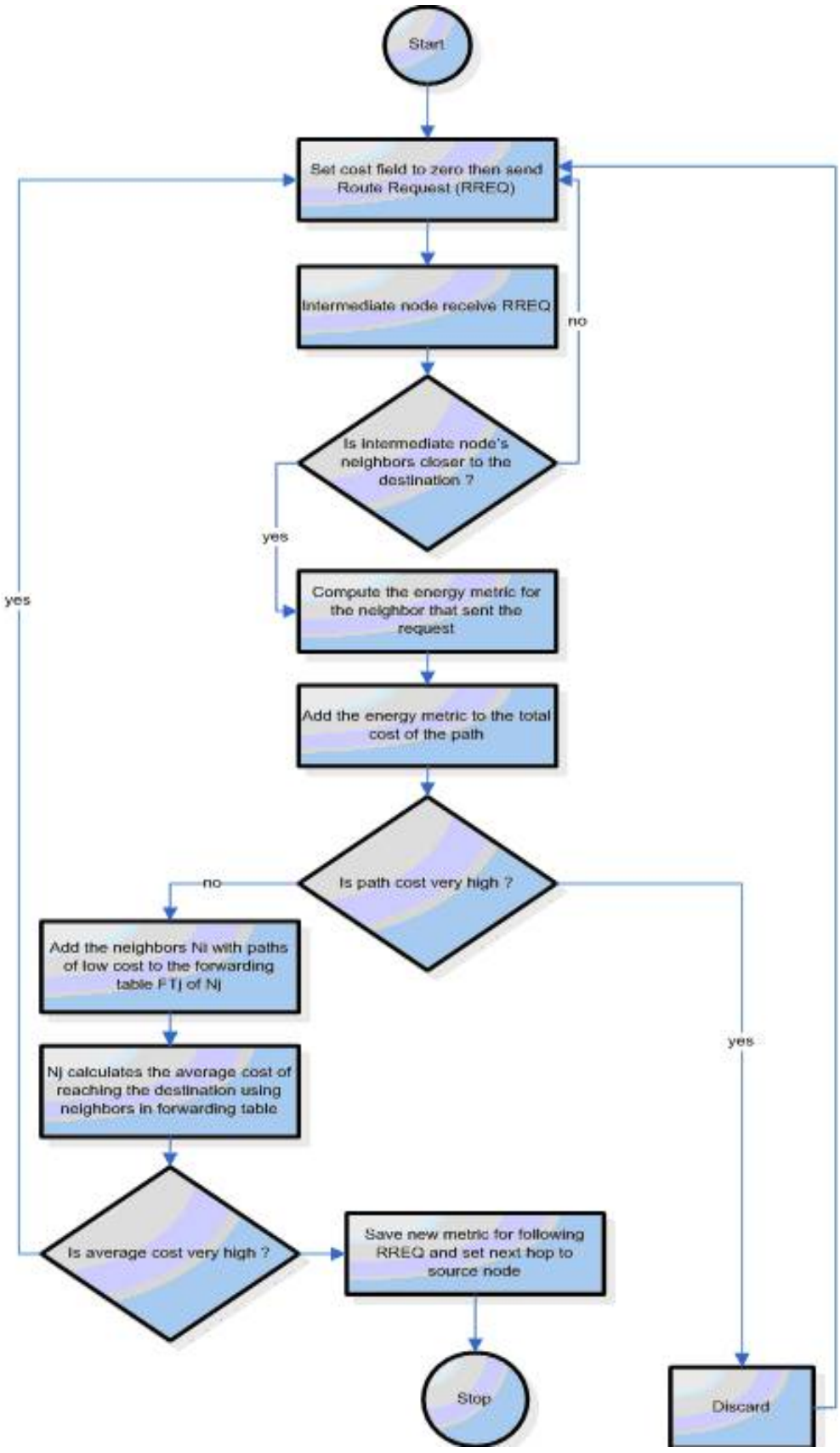


Fig. 16: Diagramma di flusso energy aware routing

EAPM introduce inoltre il seguente concetto: anziché minimizzare l'energia consumata in una rete mesh, viene considerato come scopo finale piuttosto l'idea di massimizzare il tempo operazionale della rete. L'approccio alla massimizzazione del tempo di vita della rete è lo stesso della massimizzazione del tempo per un porzione della rete. Il problema della massimizzazione del partizionamento della rete è considerato un problema NP-Complete, mentre quello sul massimo tempo di vita della rete è identificato come un problema di programmazione lineare. Poiché questi tipi di problema sono risolvibili in tempo polinomiale, allora in questo contesto si tratta di un problema PLI.

Il protocollo affronta pertanto il problema della selezione del path per massimizzare il tempo di vita del problema. Si considera il grafo di una WMN espresso come un grafo diretto G(N,A), dove N rappresenta tutti gli insiemi di nodi nella rete e A tutti gli insiemi di link diretti (i,j) con i, j ∈ N. Tutti gli insiemi dei nodi che possono essere raggiunti dal nodo i sono rappresentati da S_i e il suo range dinamico ha un certo livello di potenza energetica. Si assume che se j ∈ S_i allora esiste un link (i.j). Sia E_i come già precedentemente discusso l'energia della batteria assegnata a ciascun nodo i e sia $Q_i^{(c)}$ il rate al quale viene generata l'informazione al nodo i con c ∈ C *commodity*.

Assumendo che l'energia di trasmissione richiesta dal nodo i ai suoi vicini sia, come già visto, e_{ij} e che il rate di trasmissione di informazione per la commodity c dal nodo i al nodo j sia il flusso q_{ij} , siano i flussi aggregati Q_i e q_{ij} definiti come:

$$Q_i = \sum_{c \in C} Q_i^{(c)},$$

$$q_{ij} = \sum_{c \in C} q_{ij}^{(c)}.$$

Per un dato flusso q = { q_{ij} } il tempo di vita del nodo i è dato da:

$$T_i(q) = \frac{E_i}{\sum_{j \in S_i} e_{ij} \sum_{c \in C} q_{ij}^{(c)}}$$

Sotto il flusso q il tempo di vita del sistema può essere definito come il tempo che impiega la prima batteria a scaricarsi completamente rispetto a tutti i nodi in N. In base l'equivalenza, il tempo di vita del sistema può essere rappresentato come il tempo di vita minimo su tutti i nodi. Vi è la necessità di cercare il flusso che può aiutare nella massimizzazione del tempo di esistenza della rete e al difficoltà di massimizzare questo tempo (network lifespan)[12] è definita come:

$$\text{Maximize} \quad T_{sys}(q) = \min_{i \in N} \frac{E_i}{\sum_{j \in S_i} e_{ij} \sum_{c \in C} q_{ij}^{(c)}}$$

$$\text{s.t.} \qquad q_{ij}^{(c)} \geq 0, \qquad \forall i \in N, \forall j \in S_i, \forall c \in C,$$

$$\sum_{j:i \in S_j} q_{ji}^{(c)} + Q_i^{(c)} = \sum_{k \in S_i} q_{ik}^{(c)}, \forall i \in N - D^{(c)}, \forall c \in C$$

La condizione di conservazione del flusso della commodity al nodo i si applica a ciascuna commodity separatamente, il che ricade sotto la categoria dei problemi di programmazione lineare. Ciò che segue è il problema PLI che è equivalente al problema della massimizzazione del tempo di vita della rete, sotto la conduzione che il rate di generazione dell'informazione $Q_i^{(c)}$ è dato all'insieme dei nodi d'origine $O^{(c)}$ e l'insieme dei nodi destinazione $D^{(c)}$ per ogni commodity c. Per ciascun nodo, se la trasmissione dell'energia è rigida e senza alcuna considerazione del livello di energia trasmesso a ciascun nodo, rimane rigido è indipendentemente dal suo nodo hop successivo, quindi se non c'è controllo di energia $e_{ij} = e_i, \forall j \in S_i$ e risolvere il problema equivale a risolvere il problema del massimo flusso dove la capacità al nodo è data da:

$$\sum_{j \in S_i} \sum_{c \in C} \hat{q}_{ij}^{(c)} \leq E_i / e_i, \quad \forall i \in N.$$

$$\frac{e_i}{1} \sum_{j \in S_i} \sum_{c \in C} \hat{q}_{ij}^{(c)} \leq \frac{E_i}{e_i} \times \frac{e_i}{1}$$

$$e_i \sum_{j \in S_i} \sum_{c \in C} \hat{q}_{ij}^{(c)} \leq E_i$$

con

$$\hat{q}_{ij}^{(c)} = Tq_{ij}^{(c)}$$

che rappresenta la quantità di dati della commodity c trasmessa dal nodo i al nodo j, nell'unità di tempo T. La quantità di risorse energetiche che sono consumate da un flusso unitario è dipendente dall'energia di trasmissione consumata al prossimo nodo che ritrasmette. Pertanto, la ricerca del minimo numero di nodi da tagliare non è banale e dovrebbe essere nuovamente possibile identificare il traffico da tagliare se è stato assunto che tali nodi sono stati trovati.

Partendo da queste considerazione, per massimizzare la somma dei costi dei link in un path, viene considerato il Flow Augmentation Routing (FAR) in HWMP che presume che la rete sia statica e che identifica il miglior possibile path routing per un particolare nodo sorgente e i nodi destinatari. L'algoritmo proposto per EAPSM utilizza il Flow Augmentation come segue: il costo del path più corto da nodo sorgente a suoi nodi destinatari previsti in $D^{(c)}$ viene calcolato in ogni iterazione ed in ogni nodo d'origine o $\in$ $O^{(c)}$ della commodity c. Successivamente, la quantità $\lambda Q_i^{(c)}$ aumenta il flusso per il suo costo di path più corto, dove λ rappresenta la dimensione del passo di augmentation. Dopo l'aumento del flow, i path dal costo più basso sono ricalcolati, e ripetute le procedure fino a che non vi sia nessun nodo $i \in N$ che abbia esaurito la sua energia totale E_i . I flussi per suddividere il traffico in arrivo in modo appropriato e da usare in ogni nodo sono ottenuti come risultato dell'algoritmo.

Poiché il fine ultimo del protocollo è quello di aiutare tutti i nodi a massimizzare il loro tempo minimo di vita, un buon path candidato dovrebbe essere in grado di evitare i nodi con poca energia residua e di aumentare il path che consuma meno energia. L'ottimizzazione non può essere fatta per entrambe le tecniche allo stesso tempo, il che riflette che debba esistere un compromesso fra le due. Prima che ogni trasmissione per cui ogni nodo ha il massimo dell'energia abbia inizio, conviene scegliere il path che ha consumato il minimo di energia totale, mentre è significativo evitare l'uso dei nodi con poca energia residua verso la fine. Seguendo questa logica, il costo della metrica deve essere ridisegnato in una maniera

tale che vi sia un enfasi sul consumo di energia quando vi è in abbondanza nei nodi e che tali enfasi sia accentuata appena l'energia residua diventa minore. Si ottiene una modifica così alla metrica di EAPSM come segue:

$$C_{ij} = \frac{e_{ij}^{x_1}\left(e_i \sum_{j \in S_i} \sum_{c \in C} \hat{q}_{ij}^{(c)}\right)}{R_i^{x_2}}$$

Sebbene un link wireless (i,j) usi $e_{i,j}$ e

$$e_i \sum_{j \in S_i} \sum_{c \in C} \hat{q}_{ij}^{(c)}$$

come costanti in un link wireless (i,j), e fintanto che vi è traffico, R_i continua a scartare.

Una soluzione ottimale in un dato istante potrebbe non essere ottimale all'istante successivo poiché i costi dei link di R_i e i suoi corrispondenti sono cambiati. Per questa ragione, FAR risolve in maniera iterativa come segue:

- Per il primo passo la route ottimale può essere risolta;
- Un'altra route ottimale può essere risolta dopo l'aggiornamento dell'energia rimasta ai nodi e i costi dei link per il prossimo passo all'istante successivo

Si presume, durante ciascun passo, di avere il tasso di generazione dei dati per ogni nodo prima. Il calcolo del costo del path, rappresentato come C, è dato dalla somma dei costi dei link sul path che è stato scelto come path ottimale. Il costo del path è definito come:

$$C_{pi} = \sum_{\substack{i=m \\ j=i+1}}^{n} C_{ij} = C_{m,m+1} + C_{m+1,m+2} + C_{m+2,m+3} + \ldots + C_{n-1,n} + C_{n,n+1}$$

$$= \sum_{\substack{i=1 \\ j=i+1}}^{n=10} C_{ij} = C_{1,2} + C_{2,3} + C_{3,4} + \ldots + C_{10-1,10} + C_{10,11}$$

Nel processo di scoperta della route più ottimale fra un nodo sorgente ed un nodo destinatario previsto, non vi è la possibilità di avere path multipli possibili da usare come route per la destinazione. Tuttavia, un solo path ottimale può essere scelto per assicurarsu che nel path scelto i nodi abbiamo sufficiente energia per la trasmissione e la quantità favorita quando avviene la selezione. Tutti i path che hanno poca energia residua non possono essere mai scelti dai nodi sorgenti, anche quando l'energia di trasmissione che viene consumata è minimale. Scegliere path con poca energia nei loro nodi potrebbe avere come conseguenza che la rete venga partizionata velocemente e ciò deteriorerebbe decisamente il tempo di vita della rete.

La simulazione del protocollo è stata fatta utilizzando l'ambiente di simulazione NS-2 con un bit rate nominale di 2 Mbps, un numero fissato di dimensione dei pacchetti pari 512 byte, nodi fissi ed un rate di trasmissione di 4 pacchetti al secondo per 15 minuti di tempo di simulazione totale. Sono state mostrate confronti di performance di EAPM con ETX, metrica Airtime link e Multi-metric.

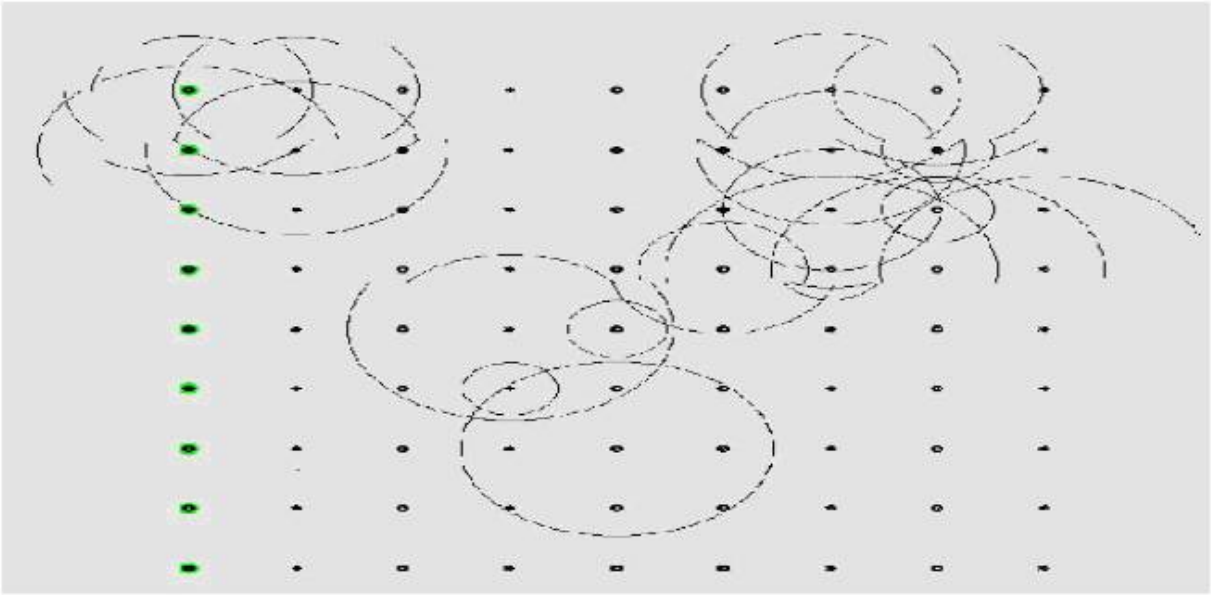


Fig. 16 : Topologia della rete con EAPM e route scelte

Trasmettendo dal nodo sorgente (in basso a sinistra) verso il nodo destinazione (in alto a destra), la valutazione del link è dal costo del link stesso come definito nel protocollo. Il costo della metrica è stato designato considerando la quantità di energia consumata durante la trasmissione sul link e_{ij} , l'energia iniziale E_i e l'energia rimanente R_i al nodo trasmittente i. Per il minimo tempo di vita di tutti i nodi da massimizzare, è scelto un buon path candidato che consuma meno energia evitando i nodi che hanno poca energia residua. E' preferito il link che richiede minore quantità di energia di trasmissione.

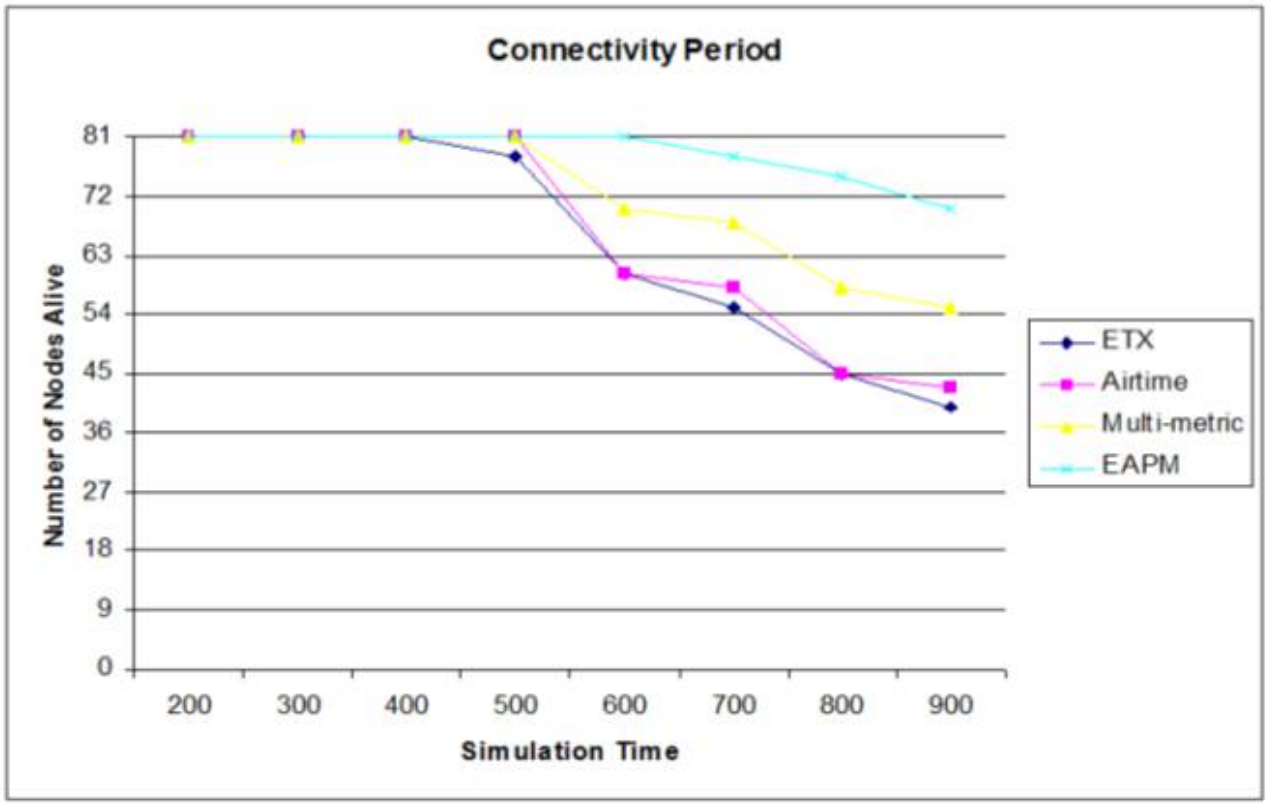


Fig. 17: Confronto di performance dei numeri di nodi attivi al tempo di simulazione

EAPM ha l'86% del numero di nodi ancora attivi mentre Multi-metric il 68% , Airtime 53%. Con un energia iniziale di 8J ai nodi, EAPM risulta migliore perchè ogni volta che i dati devono essere inviati verso i nodi destinazione previsti, i nodi sorgenti scelgono un path

in modo casuale considerando l'energia che ciascun path consuma. Pertanto è preferito il path che consuma il minor quantitativo di energia in base alle probabilità . Nessun path singolo è usato per ogni volta, soltanto un path dei tanti disponibili, in maniera del tutto probabilistica.

EAPM usa una route singola per volta ed è capace di switchare verso le altre route per migliorare la tolleranza ai guasti nella rete grazie al fatto che path diversi sono provati continuamente. Il path selezionato deve vaere i nodi con sufficiente quantità di energia per la trasmissione e la selezionabilità deve essere la più alta. Il nodo sorgente non può scegliere i path che hanno insufficiente quantità di energia, anche se i nodi coinvolti dovrebbero consumare solo un quantitativo molto piccolo di energia per la trasmissione nella rete.

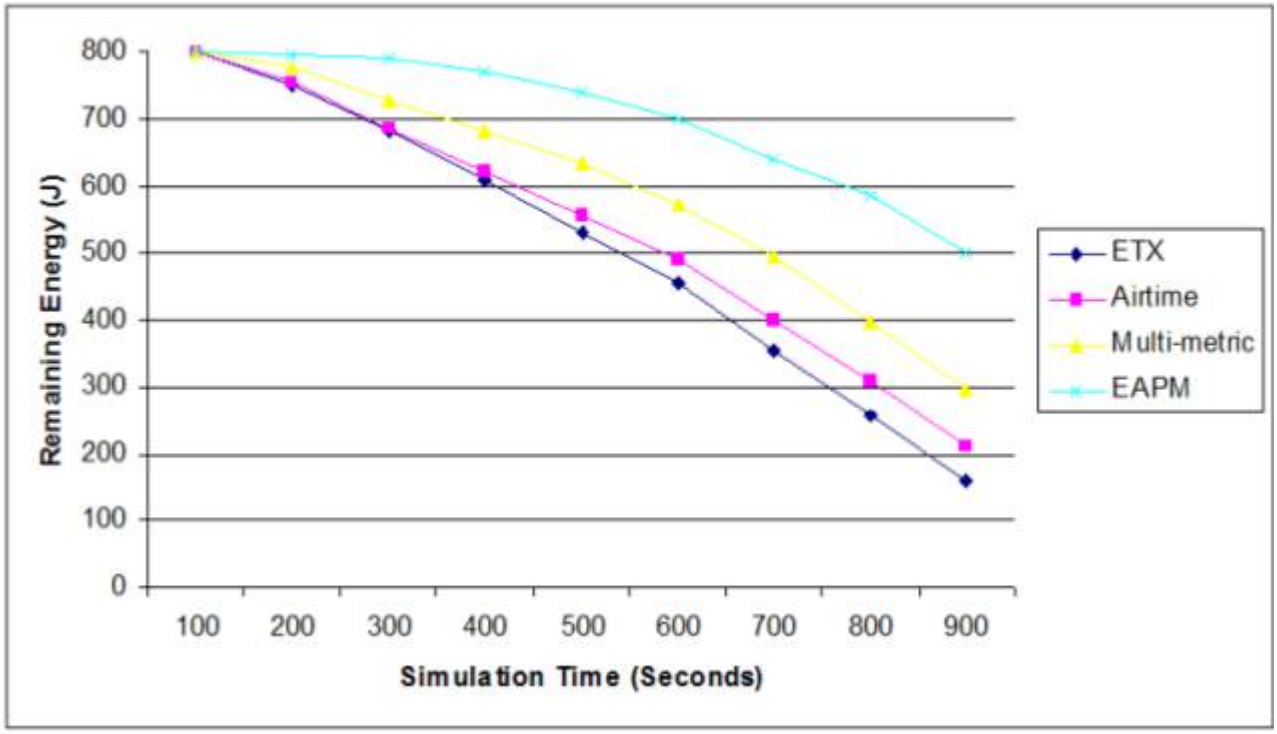


Fig. 18 : Performance energia rimanente/tempo di simulazione

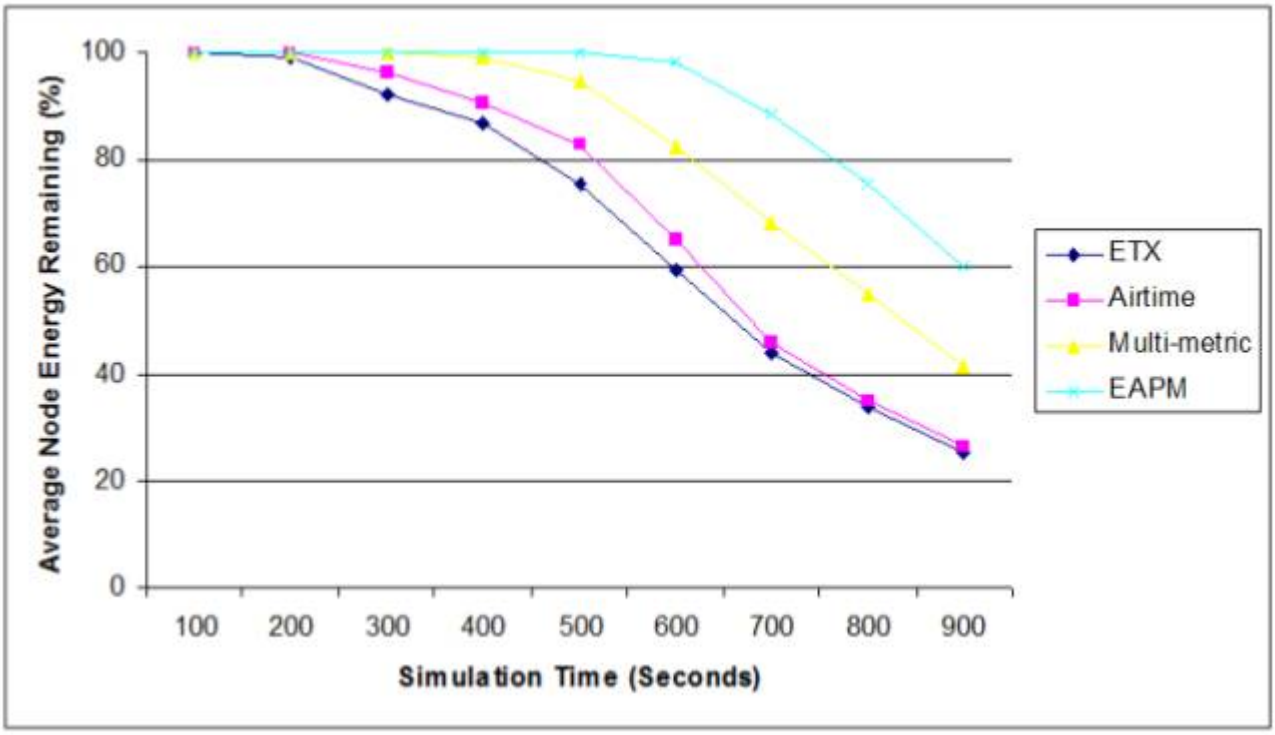


Fig. 19 : Performance energia media rimanente/tempo di simulazione

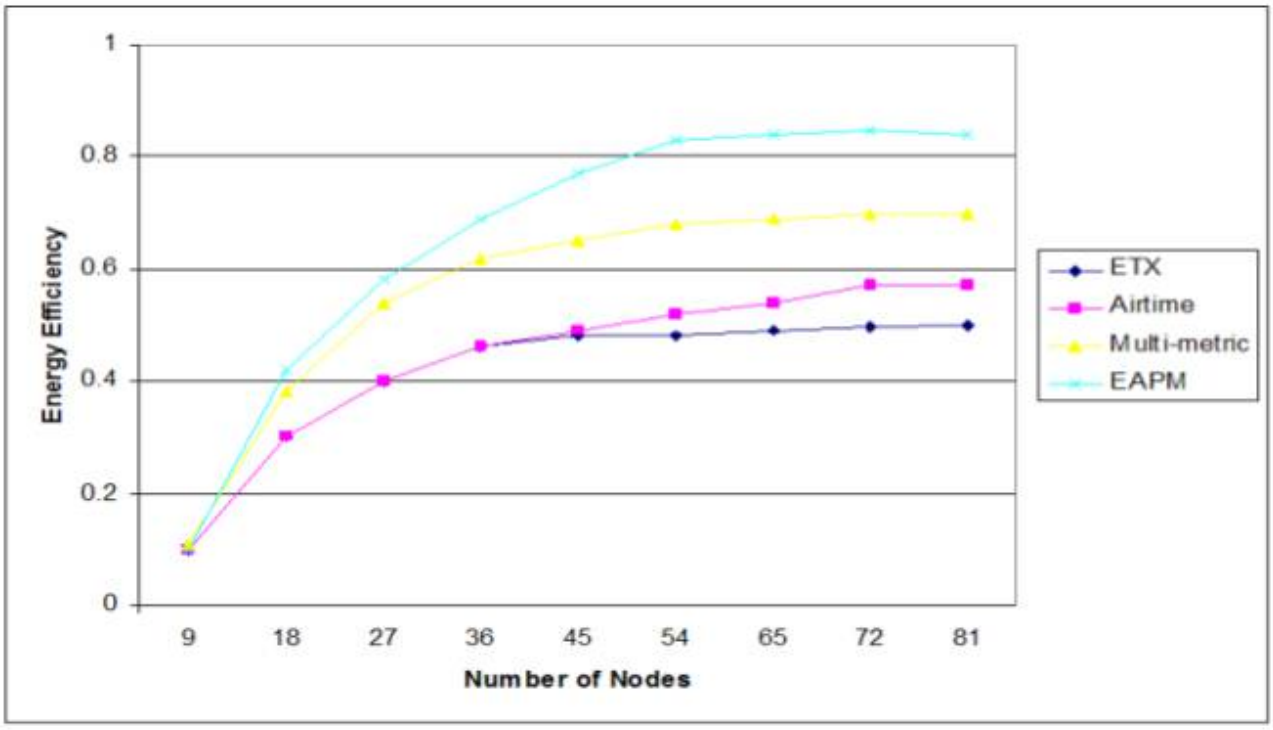


Fig.20 : Efficienza energetica in base al numero di nodi sulla rete

La ragione per cui i nodi in una rete densa sono più vicini agli altri sta nel fatto che essi riducono la loro energia di trasmissione. L'energia viene ridotto con una margine maggiore per assicurare la connettività e questo assicura che la connessione ai vicini meno lontani.

Le performance sono state comparate con le performance in reti dove EAPM non è stato impiegato. Il risultato è che EAPM ha apportato migliorie in termini di efficienza energetica, tempo di vita della rete, ritardo punto-punto e scalabilità nelle reti mesh.

1.3 OLSR

Optimized Link-State Routing (OLSR) è un protocollo per le Mobile Ad-hoc Network (MANET), che rappresenta un'ottimizzazione dell'algoritmo classico Link-State per le WLAN, definito nel RFC 3626 [7]. Il concetto chiave usato nel protocollo è quello dei MultiPoint Relay (MPR), ovvero nodi selezionati o “eletti” che inoltrano messaggi broadcast durante il topico processo di *flooding* di questo tipo di algoritmi. Nel caso delle reti mesh, esso è utilizzato in versione leggermente modificata e in coesistenza con 802.11s poiché è un protocollo IP e quindi non prevede tecniche di accesso al mezzo e contesa necessarie per la trasmissione in RF. Un esempio è il Radio Aware OLSR (RA-OLSR). Tale protocollo prevede il comportamento proattivo per il mantenimento dello stato dei link per il routing attraverso la comunicazione sui cambiamenti ai link stessi ai nodi vicini, ed è basato sullo standard OLSR più il Fisheye State Routing (FSR). Quest'ultimo, per valutare un controllo sulla sequenza di messaggi scambiati, suddivide logicamente i nodi in una composizione da anello simile per certi versi al modello atomico di Borh dove il nucleo è rappresentato dal nodo centrale e gli anelli corrispondono a degli *Scope*, in base alla distanza in termini di hop dal nodo centrale . Minore è le frequenza dei messaggi scambiati rispetto al nodo centrale, maggiore è il grado dello Scope. Il suddetto meccanismo serve per ridurre l'overhead dello scambio messaggio in termini temporali, mentre un miglioramento spaziale avviene con il meccanismo dei nodi MultiPoint Relay (MPR) mediante il quale avviene una riduzione sostanziale dell'overhead delle ritrasmissioni durante il processo di flooding. Il *radio awaring* della variante del protocollo avviene nella selezione degli MPR e dei path in modo da impedire le interferenze.

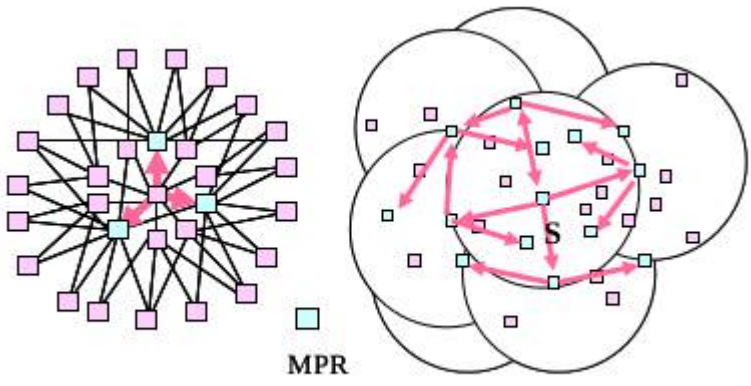


Fig. 21 : MPR in OLSR

Dunque il concetto chiave in OLSR è proprio l'utilizzo degli MPR. Sono nodi selezionati che inoltrano messaggi broadcast durante il processo di flooding riducendo così appunto l'overhead rispetto ai classici meccanismi di flooding per i quali ogni nodo ritrasmette ciascun messaggio ricevuto. L'informazione relativa allo stato dei link in OLSR è generata dai nodi eletti come MPR e ciascuno di questi nodi ha facoltà di scegliere di segnalare solo i link fra se stesso e i suoi selettore MPR. In virtù di ciò, a differenza del classico algoritmo di link-state, è distribuita nella rete un'informazione parziale sullo stato dei collegamenti, che viene usata per il calcolo delle route. Grazie a questo meccanismo OLSR fornisce delle route ottimali in termini di hop; come si vede in Fig. 11, il nodo MPR gestisce un insieme di vicini distanti un hop, che a loro volta coprono il vicinato nell'intorno di due hop di distanza.

I protocolli proattivi generano molto overhead a causa della loro necessità intrinseca del mantenimento dell'informazione relativa alla topologia e le tabelle di routing sincronizzate fra tutti o almeno fra i vicini adiacenti. Se il protocollo non gestisce il mantenimento delle tabelle di routing sincronizzate, ci potrebbero essere dei loop nella rete fino allo scadere del TTL. Un'implementazione funzionante del RFC 3626 per le reti mesh è stata fornita dal consorzio http://olsr.org il quale ha apportato alcune modifiche all'iniziale draft indicato adattarlo alle WMN con parametri prestazionali accettabili. In particolare sono stati introdotti il meccanismo LinkQuality LQ/ETX ed il Fish-Eye routing per aggiornare le informazioni sulla topologia.

OLSR utilizza un meccanismo comune per la popolazione dell'informazione di base sul link locale e sul vicinato, avviando periodicamente uno scambio di messaggi HELLO. Affinché avvenga il *link sensing*, il rilevamento dei nodi vicini e la segnalazione sull'elezione del nodo MPR, è stato costruito il messaggio HELLO con i campi adatti a contenere queste informazioni, come espresso in Fig. 22:

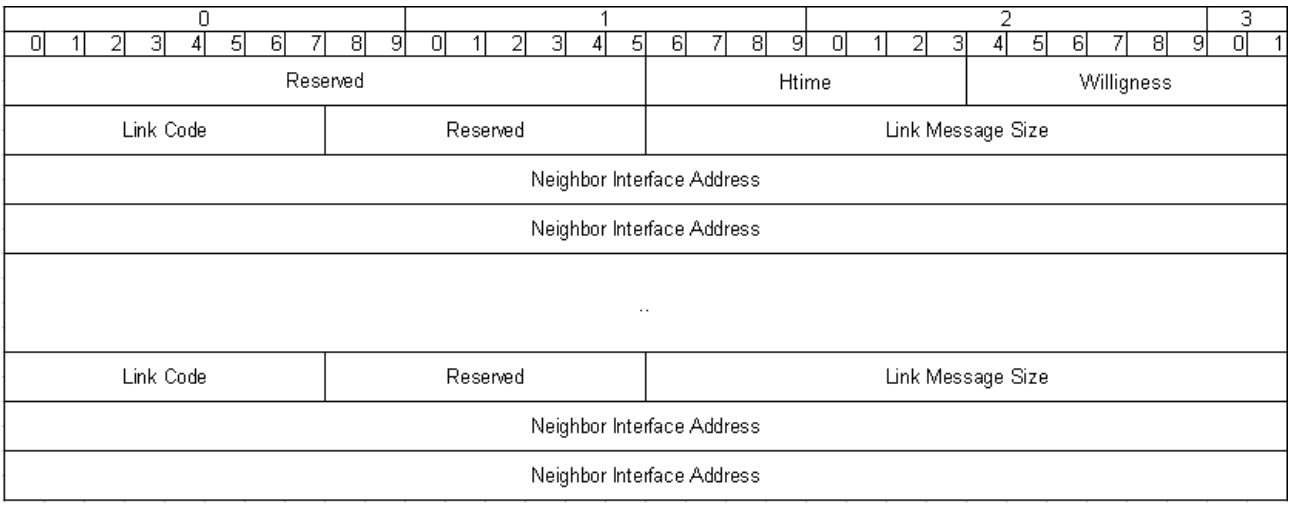


Fig. 22 : Formato messaggio HELLO

Il campo *Willingness* specifica la volontà di un nodo di trasportare ed inoltrare il traffico verso gli altri nodi. La willingness di un nodo deve essere settata ad un qualsiasi numero intero da 0 a 7 e specifica in termini quantitativi la volontà di un nodo di inoltrare il traffico di altri nodi. I possibili assegnamenti sono:

WILL_NEVER = 0
WILL_LOW = 1
WILL_DEFAULT = 3
WILL_HIGH = 6
WILL_ALWAYS = 7

Di default un nodo assume il valore WILL_DEFAULT, mentre WILL_NEVER è usato quando un nodo non intende trasportare il traffico per conto di altri nodi a causa delle limitazioni delle sue risorse, come ad esempio un livello di batteria basso. Al contrario, WILL_ALWAYS indica che un nodo dovrebbe essere sempre scelto, per esempio perché collegato ad una sorgente di corrente continua. I nodi possono cambiare dinamicamente la

loro willingness, se le condizioni lo richiedono. Uno scenario tipico potrebbe essere un nodo collegato ad un alimentazione esterna e con la batteria carica che annuncia il suo stato WILL_ALWAYS e successivamente, a causa di un'interruzione dell'alimentazione, aggiorna il suo stato e annuncia lo stato WILL_DEFAULT, continuando ad aggiornare e a disseminare l'informazione sul willingness, ogni qualvolta si creino le condizioni per farlo.

Nella più recente implementazione del protocollo, la willingness è impostata in base allo stato energetico del nodo e segue la seguente procedura:

```
/* If AC powered */
if(ainfo.ac_line_status)
  return 6;

/* If battery powered
 *
 * juice > 78% will: 3
 * 78% > juice > 26% will: 2
 * 26% > juice will: 1
 */
return (ainfo.battery_percentage / 26);
```

dividendo la percentuale del livello di batteria in ventiseiesimi ed assegnando la willingness in base al risultato ottenuto. Con la seguente euristica come si vede, il valore di WILL_DEFAULT è assegnato con un livello di batteria superiore al 76%, mentre LOW e NEVER sono assegnati rispettivamente con il valore di 2 e 1, differendo di un unità rispetto a quella definita nel RFC. L'assegnamento del willingness risulta strategico perché è anche usato per l'elezione del nodo MPR, in base all'euristica definita dal protocollo [13]. Da ciò si deduce che una politica energy-aware per OLSR debba considerare questo aspetto e cercare delle soluzioni che possano mantenere la funzionalità del protocollo, potendo essere anche energeticamente efficiente e quindi considerando anche il tempo di vita del sistema, come proposto in EE-OLSR (De Rango, Fotino, Marano 2009).

1.3.1 EE-OLSR

Energy Efficient-OLSR è una versione di OLSR energeticamente efficiente pensato per la MANET, considerando delle metriche delle metriche energy-aware note come MTPR, CMMBCR e MDR [14]. EE-OLSR utilizza per il calcolo della willingness al nodo due metriche: la capacità della batteria e il tempo di vita previsto, basato sul drain-rate. In Tab.1 è esplicato l'assegnamento del valore in base alla coppia (batteria, lifetime), mentre in Fig. 23 il listato dell'euristica che effettua tale assegnamento.

/ Battery Lifetime	**Low**	**Medium**	**High**
Short	W_LOW	W_LOW	W_LOW
Medium	W_LOW	W_DEFAULT	W_DEFAULT
Long	W_DEFAULT	W_HIGH	W_HIGH

Tab. 1 : Selezione willingness in base all'energia

ENERGY AWARE WILLINGNESS

```
double battery = ENERGY/INIT_ENERGY;
double lifetime = 65535;
if (drain_rate() != 0.0)
   lifetime = ENERGY/drain_rate();
willingness() = OLSR_WILL_DEFAULT;
if (lifetime < 10.0)
   willingness() = OLSR_WILL_LOW;
else {
   if (battery < 0.1 && lifetime < 100.0)
      willingness() = OLSR_WILL_LOW;
   else if (battery > 0.1 && lifetime > 100.0)
      willingness() = OLSR_WILL_HIGH;
}
```

Fig. 22 : Euristica EA-Willingness Setting

Come si vede, la capacità della batteria misurata al nodo, è il rapporto fra l'energia attuale e quella iniziale, con un range da 0.0 a 1.0. Il tempo di vita del nodo è calcolato in secondi, ponendolo inizialmente ad un valore massimo di 2^{16} , che sta a rappresentare infinito.

Uno stato energetico LOW viene imposto al 10% della batteria, mentre il lifetime è considerato SHORT se risulta minore di 10 secondi, LONG se maggiore di 100 secondi. Un'altro meccanismo che permette di risparmiare energia in OLSR, senza inficiarne comportamento e prestazioni, è l'Overhearing Exclusion, il quale consiste nello spegnere il dispositivo quando avviene una scambio di messaggi unicast nel vicinato, potendo risparmiare così una quantità considerevole di energia. Si può ottenere semplicemente usando i meccanismi di segnalazione del livello 2 come RTS/CTS, poiché non vi è alcun vantaggio nell'informazione nei messaggi unicast diretta agli altri nodi, cosa che invece risulta utile in altri protocolli come DSR.

Il modello di consumazione dell'energia adottato nelle simulazioni del protocollo proposto è il seguente:

$$E(p, n_a) = E_{tx}(p, n_a) + E_{rx}(p, n_b) + (N-1) \cdot E_O(p, n_i)$$

dove l'energia necessaria a trasmettere un pacchetto p da un nodo i è espressa come:

$$E_{tx}(p, n_i) = i \cdot v \cdot t_p$$

dove i è la corrente, v il voltaggio e t_p il tempo impiegato per trasmettere il pacchetto p, in secondi. La formula precedente descrive l'energia consumata per trasmettere un pacchetto da un nodo a ad un nodo b, considerando l'energia consumata per trasmettere il pacchetto dal nodo trasmittente E_{tx} , quella per ricevere lo stesso pacchetto al nodo ricevente E_{rx} , e quella per effettuare l'overhear del pacchetto E_O . Dalle simulazioni si è visto come la selezione energy-aware del willingness può aumentare il tempo di vita dei nodi (e quindi del sistema) ed inoltre come EE-OLSR superi OLSR classico in termini di throughput, vita media dei nodi, tempo di vita delle connessioni, nei nodi, preservando il controllo dell'overhead normalizzato.

1.4 BATMAN e BATMAN-adv

Dall'implementazione e dal testing di OLSR sulle WMN nel corso degli anni si sono raccolte varie proposte e vari feedback per un miglioramento del protocollo stesso, con particolare riferimento alle reti mesh, che si sono sintetizzate nella presentazione del protocollo Better Approach To Mobile Ad-Hoc Network (BATMAN). BATMAN dunque è un evoluzione pratica e sperimentale di OLSR per le reti mesh, che si è a sua volta evoluto verso un protocollo di livello 2 denominato BATMAN-adv, attraverso il quale è possibile raggiungere prestazioni al pari di 802.11s e talvolta anche superiori.

I punti deboli del protocollo OLSR, dal quale si è giunti prima a olsrd e poi a BATMAN, sono stati molteplici, in particolare le funzionalità di OLSR che si sono rese maggiormente colpevoli dei deficit rilevati nei test reali sono il meccanismo di Link Hysteresis e quello degli MPR [15]. Dalla definizione del RFC 3626, l'isteresi dei link (Link Hysteresis) è un meccanismo che determina, per un nodo, il criterio per il quale un link verso un nodo vicino sia accettato o rifiutato. Poiché il collegamento tra un nodo e qualche interfaccia dei vicini potrebbe non essere buono, viene attuato una strategia di isteresi che permette di avere una conferma sulla bontà del link. Tale meccanismo però, in virtù del fatto che tiene gli MPR al di fuori della routing table, comporta di riflesso dei malfunzionamenti nell'infrastruttura che si fa carico della disseminazione dell'informazione topologica, e di conseguenza porta a rinegoziare gli MPR. La selezione degli stessi MPR avviene scegliendo nodi lontani per mantenere il numero di MPR basso, ma poiché i collegamenti sono generalmente instabili, ed avviene maggiormente il caso in cui il meccanismo di isteresi tiene tali link fuori dalla routing table, rispetto alla riduzione dell'overhead grazie al meccanismo stesso degli MPR. Ciò ha portato all'eliminazione del meccanismo di isteresi e di MPR, e nella scelta delle metriche in base ad un meccanismo di quality/throughput chiamato Expected Transmission Count/Link Quality ETX/LQ. Quest'ultimo è un meccanismo di packet loss che prevede di tenere traccia di alcuni pacchetti chiamati dell'avvenuta trasmissione o meno di pacchetti di Hello, inviati in broadcast attraverso il meccanismo LSR. Tuttavia successivi aggiustamenti del protocollo, come il LinkQualityDjikstraLimit per evitare grossi calcoli su CPU limitate

in grandi reti mesh, non hanno risolto del tutto i problemi di loop frequenti nelle route e dei problemi di gateway-switching in caso di più nodi internet-gateway, pertanto si è pensato ad un protocollo non incentrato sulla sincronizzazione sullo stato dei link, che potesse superare i limiti intrinseci di OLSR, dando luogo a BATMAN I.

1.4.1 Algoritmo di routing

L'approccio dell'algoritmo su cui si basa BATMAN consiste nel dividere la conoscenza sui migliori path end-to-end tra i nodi nella rete mesh, a tutti i nodi partecipanti. Ciascun nodo percepisce e mantiene solo l'informazione sul next hop migliore verso tutti gli altri nodi e in aggiunta viene usato un meccanismo di flooding ad eventi e timeless per prevenire l'accumulo di informazioni topologiche contraddittorie. L'algoritmo è designato per lavorare con reti che sono basate anche su link inaffidabili. Vengono introdotti dei messaggi chiamati OriGinator Message OGM utilizzati per la trasmissione in broadcast da ciascun nodo per informare i nodi vicini della loro esistenza. I nodi vicini che ricevono gli OGM li inviano in broadcast nuovamente in base a delle regole specifiche, per informare i loro vicini dell'esistenza del nodo che ha iniziato lo scambio di messaggi e così via. Avviene così nella rete il flooding degli OGM. Tali messaggi sono piccoli, la dimensione tipica del pacchetto grezzo è di 52 byte incluso l'overhead IP e UDP. Essi contengono almeno l'indirizzo del nodo originator, l'indirizzo del nodo che trasmette il pacchetto, il TTL ed un sequence number. Gli OGM che seguono un path dove la qualità dei link wireless è bassa o saturata, sono affetti da packet loss e ritardo nel mesh, mentre quelli che viaggiano su buone route si propagano in modo più veloce e più affidabile. Il sequence number serve a determinare quante volte è stato ricevuto un OGM ed è dato dal nodo originator. Ciascun nodo ritrasmette in broadcast ciascun messaggio OGM ricevuto almeno una volta e solo quei messaggi ricevuti dal vicino identificato come il miglior next hop corrente dal nodo nodo che ha iniziato il messaggio OGM.

In questo modo gli OGM sono disseminati (flooding) in maniera selettiva nel mesh ed informano i nodi riceventi dell'esistenza di altri nodi. Un nodo A viene a sapere dell'esistenza di un altro nodo B attraverso la distanza per la quale ha ricevuto i suoi OGM, quando gli OGM del nodo B sono ritrasmessi in broadcast dai suoi vicini distanti un hop. Se

il nodo A ha più di un vicino, può dire quale di questi deve essere scelto per l'invio dei dati verso un nodo distante, attraverso il numero di OGM che riceve più velocemente e in maniera più affidabile tramite uno dei suoi vicini lontani un hop. L'algoritmo successivamente seleziona questo vicino come il miglior next hop corrente verso il nodo che ha generato il messaggio e configura la sua tabella di routing rispettivamente.

Con l'avvento del protocollo BATMAN, nasce una task-force di sviluppatori, studiosi e appassionati che si congregano in un consorzio chiamato OpenMesh per favorire uno sviluppo libero del meshing su kernel Linux. Nasce allo stesso tempo l'evoluzione del protocollo, che viene denominata BATMAN Advanced (BATMAN-adv), un'implementazione del protocollo di routiung BATMAN nel kernel Linux, operando però al livello 2, divenendo così un protocollo di Path Selection. BATMAN-adv è vincolato al kernel Linux, il che se da una parte il protocollo guadagna in robustezza con un ambiente collaudato, eterogeneo ed uno sviluppo distribuito, dall'altra rimane appunto vincolato al sistema operativo in questione perdendo la necessaria astrazione e la duttilità dei protocolli di rete. Tuttavia poiché nel meshing vengono considerati dispositivi perlopiù embedded con diverse architetture, la scelta di Linux permette di avere la totale copertura di questi dispositivi per la sua natura intrinseca, focalizzandosi piuttosto sulla pratica e sui test reali, che non sull'aspetto puramente teorico e relativo alla simulazione.

1.4.2 Caratteristiche

Giacché BATMAN-adv opera a livello 2, le informazioni di routing ed il traffico dati vengono incapsulati in uno special frame Ethernet (Ethertype 0x0842) prima di essere trasmessi come pacchetti 802.11 o 802.3. L'implementazione nel kernel di BATMAN-adv ha il vantaggio di introdurre un processamento dell'overhead trascurabile anche per grossi carichi di dati. Il protocollo incapsula ed inoltra tutto il traffico finché non raggiunge la destinazione, emulando così uno switching di rete virtuale per tutti i nodi partecipanti. Tutti i nodi sembrano essere collegamenti locali non sono a conoscenza della topologia della rete come non sono affetti dai cambiamenti nella rete stessa.

Le caratteristiche principali di BATMAN-adv sono:

- Network-layer agnostic: più protocolli possono essere eseguiti al di sopra di batman-adv (IPv4, IPv6, DHCP, etc). E' indipendente dai vari requisiti dei protocolli di rete;
- I nodi possono partecipare al mesh senza avere un indirizzo IP: vengono usati gli indirizzi MAC per inoltrare il traffico nella rete mesh. Per partecipare al mesh con batman-adv, è necessario solo disporre di un indirizzo MAC;
- Integrazione agevolata dei client non mesh: viene utilizzata una tabella chiamata MAC translation table per permettere i clienti non mesh a partecipare all'infrastruttura mesh. Quando un nodo batman-adv rileva che un client desidera comunicare sul mesh, registra il suo indirizzo MAC nella sua translation table e dissemina l'informazione nella rete che la coppia indirizzo/client è attaccata al nodo batman-adv che la ha rilevata. Se gli altri nodi desiderano inviare dati al client, ricercano il MAC del client nella translation table globale, per trovare il nodo batman-adv corrispondente. Dopodiché i dati sono trasmessi al nodo batman-adv che li inoltra successivamente al client non mesh;
- Roaming dei client non mesh: I client che non partecipano al mesh con batman-adv effettuano il roaming attraverso il nodo batman-adv più vicino, il quale successivamente invia in broadcast l'informazione sull'indirizzo MAC del client all'intera rete mesh;
- Ottimizzazione del flusso dei dati nel mesh: batman-adv applica due tecniche che permettono di guadagnare la massima performance in merito all'ottimizzazione del traffico, che sono *interface alternating* ed *bonding*. La prima dà la possibilità effettuare lo switch fra le diverse interfacce wireless del nodo (o anche ethernet), la seconda permette di suddividere i flussi.

1.4.3 Ottimizzazioni Multi-link

I nodi mesh con più link fisici verso altri nodi possono usare questa proprietà per ottenere vari benefici, attraverso tecniche come il *bonding* e l'*interface alternating*. Queste però possono essere utilizzate solo sotto la condizione che vi siano link multipli tra i due vicini, e che la decisione di routing sia indipendente dal link attraverso il quale è stato ricevuto un pacchetto. L'esistenza di più interfacce sul nodo è indipendente dall'algoritmo BATMAN, il quale cerca solo di ottimizzare i flussi sfruttando questa proprietà, se presente. Vi è solo la necessità di verificare che più peer appartengono allo stesso nodo vicino, non essendo neanche un problema di routing multi-path, dove sarebbero richiesti link multipli indipendenti. Il processo di ottimizzazione attraverso il multi-link può essere riassunto in questi passaggi:

- verifica che il vicino sia raggiungibile tramite più link;
- selezione dei link candidati migliori tra quelli disponibili;
- uso dei link multipli per varie manipolazioni.

La verifica o scoperta del multi-link dei nodi è compito di batman-adv. L'indirizzo MAC (Originator) dell'interfaccia primaria del nodo è disseminato nella rete tramite i messaggi OGM che lo inviano in broadcast su tutte le interfacce disponibili con un alto TTL, assicurando che i nodi mesh raggiungibili solo tramite interfacce secondarie vengano a conoscenza del nodo originator. Tutte le interfacce rimanenti sono dette secondarie, e sono conosciute solo a distanza di un hop nel vicinato. Gli OGM che contengono i loro indirizzi MAC sono inviati in broadcast con un TTL piccolo solo sulle rispettive interfacce. Questi OGM “secondari” servono unicamente allo scopo di rilevare i vicini e stimare la qualità dei loro link. Da questo meccanismo si capisce esiste un solo Originator nel mesh, il che evita l'overhead dei messaggi OGM. Per verificare se un nodo vicino abbia link multipli verso un nodo, è sufficiente analizzare il messaggio “primario” OGM in arrivo. Se riceve tale messaggio tramite interfacce multiple, è possibile assumere in modo abbastanza sicuro che lo stesso ha dei link multipli verso il suo Originator, come illustrato ad esempio in Fig.24.

Tuttavia, potrebbero esserci dei problemi in merito al fatto che il protocollo richiede l'inoltro di tutti gli OGM ricevuti per il rilevamento del path, e questo potrebbe generare ambiguità e confusione della scoperta del multi-link poiché sarebbe necessario distinguere i vicini lontani un hop rispetto a gli altri più distanti. Per risolvere il problema è stato introdotto un flag denominato PRIMARIES_FIRST_HOP che solo i messaggi OGM mantengono, mentre tutti gli altri nodi eliminano prima di inviare in broadcast nuovamente gli OGM.

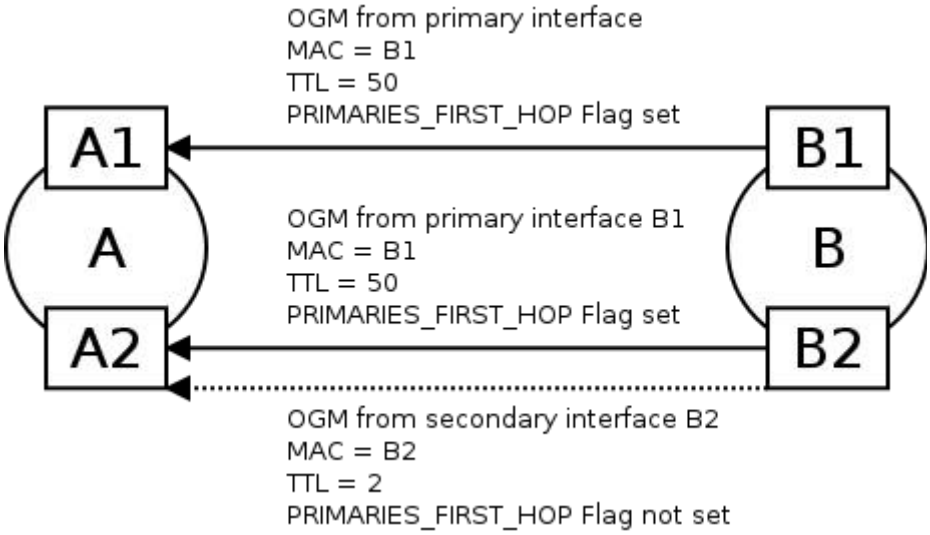


Fig. 24: Scoperta dei link candidati

Una volta ottenuta la lista di link candidati, avviene la selezione degli stessi. Essi devono avere qualità simili e dovrebbero operare su frequenze diverse. Viene così introdotta una soglia di qualità del link, denominata Thresold Quality TQ, per mezzo della quale viene stabilita la bontà del link in base al superamento o meno di un valore di soglia, definita dinamicamente. Le condizioni per la scelta dei candidati sono che la qualità del link candidato non deve essere al di sotto della soglia TQ, e che la destinazione del link sia diversa da quelle che già altri candidati scelti raggiungono, come avviene in Fig. 25.

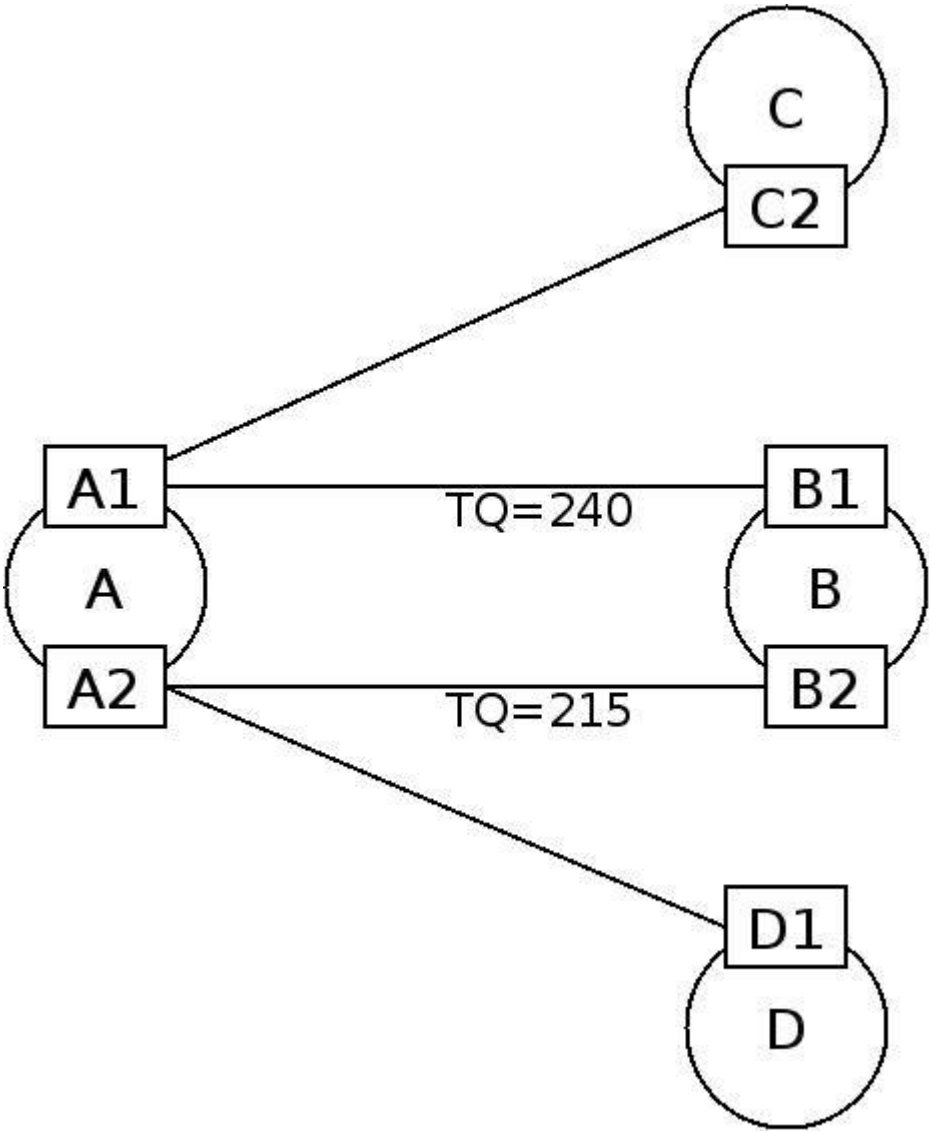


Fig. 25 Scelta dei link candidati

Dopo che un nodo mesh è stato selezionato dall'algoritmo di routing, i link individuali possono essere selezionati in modo egual modo per inviare i dati al prossimo nodo vicino senza interferire con il routing stesso.

1.4.4 Interface alternating

L'interface alternating è un meccanismo, abilitato di default nel modulo, che prevede lo smistamento dei frame su interfacce diverse da quella su cui sono stati ricevuti. Tale meccanismo, illustrato in Fig. 26, prevede l'alternanza delle interfacce disponibili sul nodo. Serve a ridurre le interferenze dovute alla trasmissione nel mezzo wireless, e a bilanciare il carico di rete sulle interfacce disponibili, al fine di massimizzare il throughput. La sola condizione affinché possa essere applicata questa tecnica è relativa alla qualità dei link candidati verso il next hop; per potere effettuare l'interface alternating è necessario che i link abbiano qualità simili, altrimenti verrebbero meno i vantaggi derivanti del suo uso.

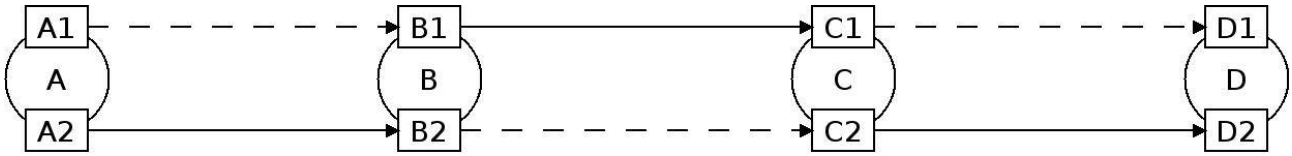


Fig. 26 : Catena di nodi con due interfacce

1.4.5 Bonding

Un'altra feature basata sulle qualità simili dei link candidati è il bonding, la quale permette la distribuzione dei frame su questi link disponibili. Viene utilizzata una modalità rond-robin per l'invio dei frame individuali e si ottiene un incremento del throughput in base al numero di link coinvolti nel processo, arrivando fino al 150% di aumento della bandwidth sul singolo link, laddove in teoria di aspettava di avere il 200%, probabilmente a causa dell'algoritmo di backoff durante i tentativi di riconnessione. [16]

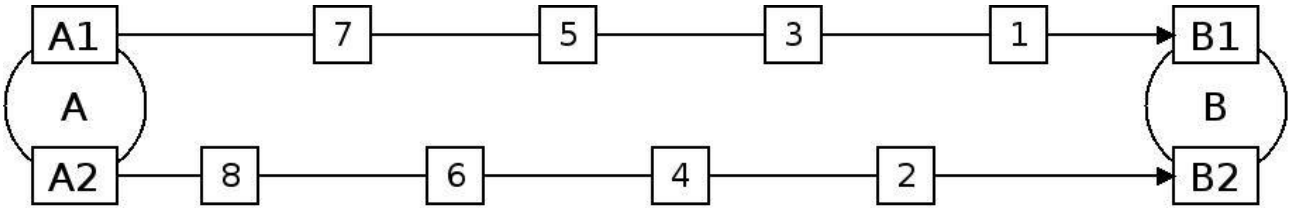
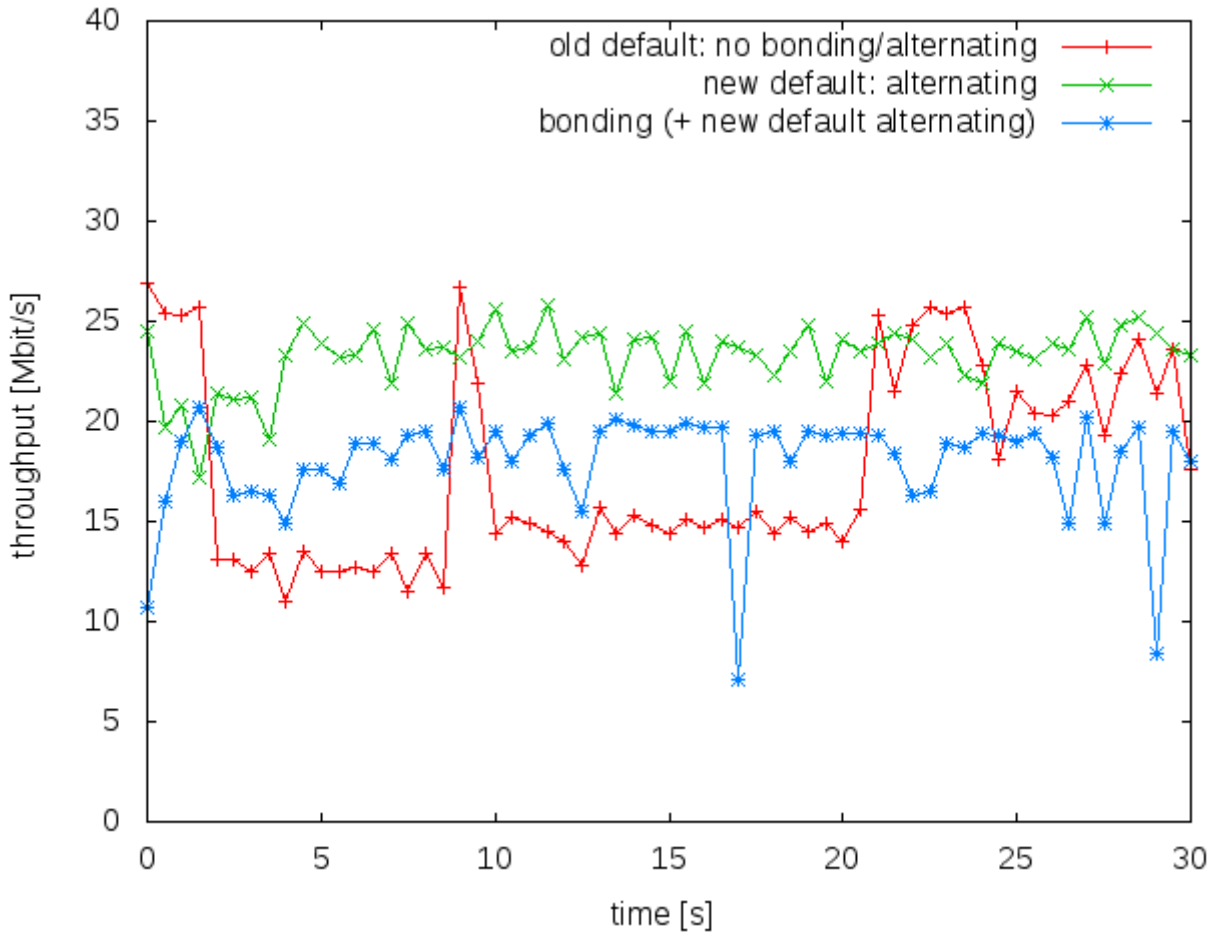


Fig. 27: Bonding

Tale incremento è vincolato alla velocità dei link. Poiché batman-adv non rileva la velocità dei link, se vi sono delle differenze sostanziali, il throughput potrebbe diminuire con la generazione di rallentamento generale dovuto alla presenza di link più lento. Pertanto queste feature non è abilitata di default nel modulo, e la di dovrebbe usare laddove si è abbastanza sicuri di variazioni trascurabili delle velocità dei link coinvolti.

Nella Fig. 28 è visualizzato il risultato di un test reale dove di evince il mantenimento delle performance avendo abilitato l'interface alternating.



1.4.5 Batman-adv gateway

Una delle proprietà peculiari di una rete mesh è quella di poter condividere la connessione alla rete internet attraverso tutti i nodi coinvolti nel mesh, fornendo così la possibilità di una copertura molto vasta. Vengono utilizzati, a tale proposito, dei nodi gateway per l'accesso esterno, ed è compito dell'algoritmo di routing aiutare i nodi nella scelta del path migliore da seguire (che corrisponde al path migliore verso il gateway più vicino). A causa delle controversie in merito al fatto che un protocollo di livello 2 come batman-adv debba gestire un problema di livello 3, come la manipolazione dell'IP di una default route, il meccanismo dei gateway è partanto disabilitato di default. Esso si basa inoltre sul DHCP, ovvero il protocollo assume che ciascun nodo avente funzionalità di gateway debba avere il proprio server DHCP e di conseguenza ciascun client un client DHCP. Quando il client DHCP inizia la sua richiesta DHCP la invia in broadcast in tutta la rete e ciascun server DHCP risponde con una comunicazione unicast al client, a cui spetta la decisione finale su quale server utilizzare e quindi quale nodo gateway, in base ad una politica di selezione del migliore in base alle informazioni ottenute come il ritardo o il packet loss. Il processo è illustrato in Fig. 29

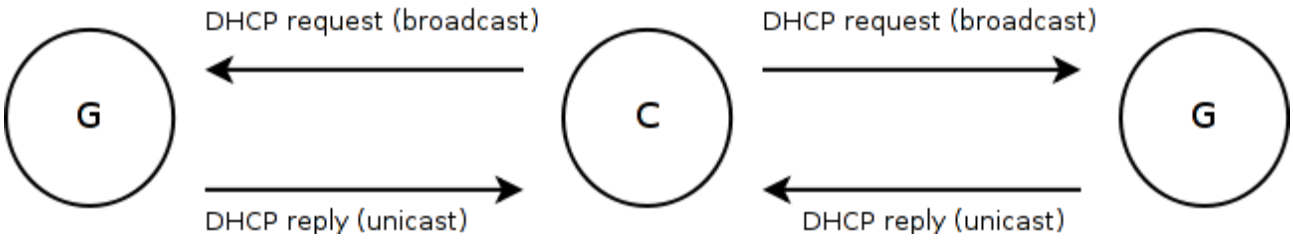


Fig. 29 : Richieste DHCP nella rete e risposte dei nodi gateway

Se la funzionalità è attiva, batman-adv effettua il flooding dell'informazione sui gateway nella rete, per renderlo noto ad ogni nodo batman-adv. Appena arriva una richiesta DHCP, il protocollo non invia in broadcast il pacchetto, ma lo invia in unicast verso il nodo gateway scelto. Il client successivamente seleziona il nodo gateway preferito. Al fine di garantire la

validità del DHCP release, anche a seguito di un cambiamento di gateway, batman-adv effettua l'ispezione dei pacchetti relativi alle richieste di rinnovo del DHCP. Se la destinazione non è il nodo gateway selezionato e se il link è sotto la soglia TQ, la richiesta di rinnovo non è inoltrata, esortando il client ad effettuare un ulteriore richiesta per un gateway migliore.

1.4.6 Modulo kernel linux

BATMAN-adv, come già accennato, è un protocollo di routing operante a livello 2, implementato su kernel Linux, con il nome di batman-adv. Poiché il processing dei pacchetti nello *user space* è molto dispendioso in termini di CPU, soprattutto alla luce del fatto che i dispositivi con poca limitata potenza di calcolo potrebbero risentirne in termini di prestazioni nel meshing, è stato implementato nel *kernel space*, come un driver del kernel, ottenendo così un processing dei pacchetti veloce ed affidabile, anche in caso di grossi carichi. Il modulo poi applica la gestione degli eventi da segnalare allo user space (uvents), per informare quest'ultimo dei cambiamenti avvenuti nel mesh. Le applicazioni nello user space rimangono in ascolto ed interagiscono in base al tipo di evento ricevuto, accedendo alle risorse per la cattura degli eventi generati al kernel (es. udev).

1.4.7 Console batctl

Al fine di configurare ed effettuare il debug del modulo del kernel batman-adv, è stata sviluppata una console di supporto, dal nome batctl. Essa è un'interfaccia un insieme di informazioni utili e un accesso facilitato a tutte le varie impostazioni del modulo. Contiene inoltre una versione a livello 2 di alcuni strumenti di rete di utilità come il ping, il traceroute e il tcpdump, giacché lo switch virtuale è completamente trasparente per tutti i protocolli sopra il livello 2.

1.4.8 Efficienza energetica in batman-adv

La questione energetica nel protocollo non è stata ancora considerata. Laddove i test reali premiano BATMAN-adv per risultati migliori in termini di throughput rispetto ad altri protocolli come 802.11s, non è possibile stabilire se una versione energy-aware dell'algoritmo di routing, o un'ottimizzazione in tal senso possa essere, possa continuare a garantire la bontà del protocollo. Il protocollo descritto nel capitolo 1.2.11, presenta una metrica energy-aware per 802.11s che ottiene buoni risultati in regime di alimentazione non continua a batteria. Per quanto riguarda BATMAN-adv, sarebbe necessario trovare delle funzionalità chiave, sulle quali provare ad impostare un'ottimizzazione in senso energetico, al di là del Power Saving di 802.11, implementato nel driver *mac80211* nel kernel Linux. Da un'analisi delle caratteristiche del protocollo e del modulo batman-adv (all'ultima revisione dal repository git), i punti sui quali lavorare potrebbero essere:

- Razionalizzazione dell'invio dei messaggi OGM: attualmente i messaggi OGM vengono inviati ogni secondo, anche quando necessario. Sebbene di carico leggero (52 byte “tutto compreso”), rimane uno spreco generare ed inviare messaggi quando se inutili.
- Determinazione della soglia TQ di qualità del link in base allo stato energetico del nodo: il link del path migliore potrebbe non essere più valido se l'energia dello stesso è in via di esaurimento, e il bilanciamento del carico con interface alternating e con bonding (se non ci sono grosse limitazioni alla potenza di calcolo dei nodi nel mesh) potrebbe essere fallace e le due feature degradanti per il sistema.

1.5 Test reale

Al fine di mostrare la bontà del protocollo, è stato effettuato un test reale in enviroment indoor con 5 nodi, di cui 4 fissi e uno mobile alimentato a batteria. In particolare è stato effettuato un test di rete su TCP e UDP sia con batman-adv che con 802.11s, e viene fornita nel dettaglio il test di esecuzione con il protocollo batman-adv, al fine di evidenziare le componenti e le fasi salienti della realizzazione di un infrastruttura di rete basata su BATMAN-adv. I dispositivi utilizzati sono eterogenei, e il test è stato effettuato in due stanze, come illustrato in Fig. 30.

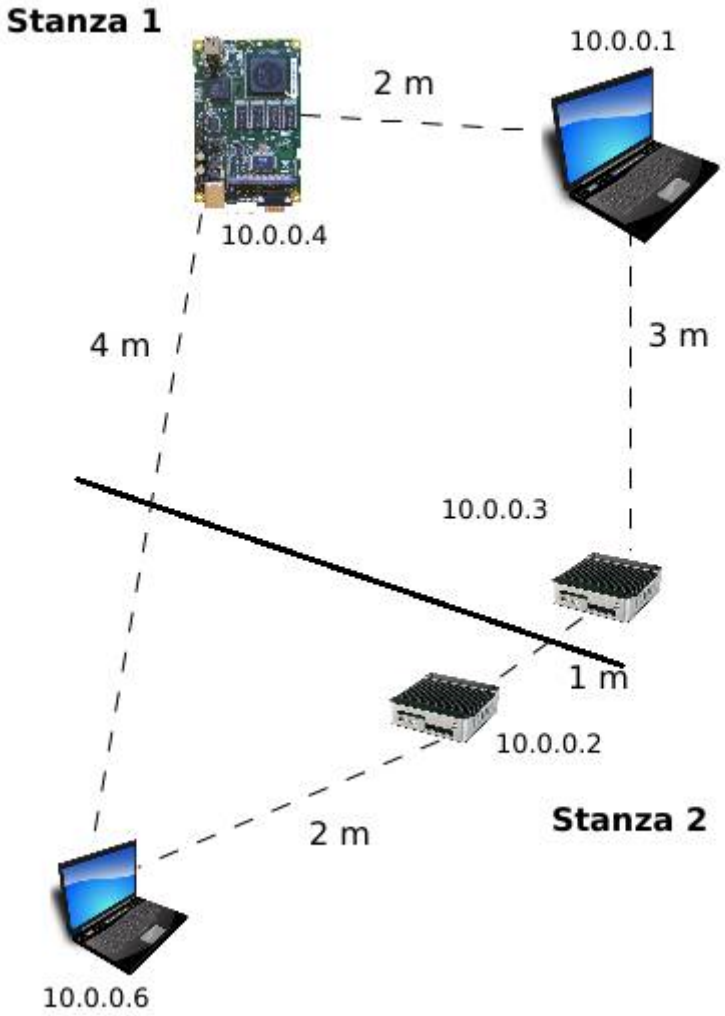


Fig. 30 : Disposizione dei nodi nell'ambiente di test

1.5.1 Specifiche hardware

Nodo 1

Tipo: Laptop HP Compaq 6720s

S.O. : Debian Linux 3.6.6

Processore: Intel Celeron 1,8 Ghz

RAM: 2 GB

Hard disk: 160 GB

batman-adv: 2012.3.0

Wireless: (2) USB Atheros / Ralink

IPv4: 10.0.0.1

IPv6: Auto su bat0

Note: VisServer, DHCP Server, Interface Alternating

Nodo 2

Tipo: Embedded board eBox 3300

S.O. : Debian Linux 3.2.0

Processore: DM&P Vortex86sx 330 Mhz

RAM: 128 MB

Hard disk: CF 4 GB

batman-adv: 2012.4.0

Wireless: (1) USB Prism54

IPv4: 10.0.0.2

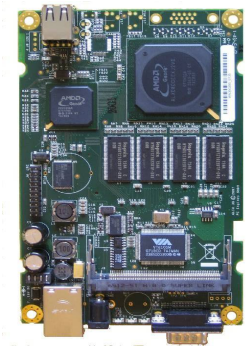
IPv6: Auto su bat0

Note: gw_mode client, tabella di routing non visualizzabile con batctl

Nodo 3

Tipo: Embedded board eBox 3300
S.O. : Debian Linux 3.2.0
Processore: DM&P Vortex86sx 330 Mhz
RAM: 128 MB
Hard disk: USB Flash 4 GB
batman-adv: 2012.4.0
Wireless: (1) USB Ralink
IPv4: 10.0.0.3
IPv6: Auto su bat0
Note: gw_mode client, antenna con guadagno 3dB

Nodo 4



Tipo: Embedded board Alix 3D
S.O. : Debian Linux 3.2.0
Processore: AMD Geode 500 Mhz
RAM: 256 MB
Hard disk: CF 1 GB
batman-adv: 2012.2.0
Wireless: (1) PCI Express Atheros
IPv4: 10.0.0.4
IPv6: Auto su bat0

Note: gw_mode client, iperf server

Nodo 5

Tipo: Netbook Asus 1001PX
S.O. : Ubuntu 12.04 Linux 3.5
Processore: Intel Atom N455 1,66 Ghz
RAM: 1 GB
Hard disk: 120 GB
batman-adv: 2012.2.0
Wireless: (1) PCI Atheros
IPv4: 10.0.0.6
IPv6: Auto su bat0
Note: gw_mode client, nodo mobile, alimentazione a batteria, iperf client

1.5.2 Specifiche software

Il sistema di riferimento, come già ampiamente discusso, è GNU/Linux su kernel >=3.2, con stack di rete basato su mac80211, cfg80211 e il nuovo userspace nl80211. [16]

L'hardware utilizzato nel test è presente nella tabella dei driver e delle funzionalità dei relativi chipset wireless [17], con il supporto sia della modalità *IBSS* (ad-hoc), sia quella *mesh* da usare con HWMP nell'implementazione 802.11s.

Per ciascun nodo sono state impostate le procedure di inizializzazione del mesh, come

illustrato nella Tab. 2 rispettivamente per batman-adv e per 802.11s

BATMAN-adv	802.11s
modprobe batman-adv ifconfig <iface> **mtu 1528** iwconfig <iface> mode ad-hoc essid BluesMesh ap <**02**:00:d3:ad:b3:3f> channel <reg_db> **batctl** if add <iface> ifconfig <iface> up ifconfig **bat0** up	ifconfig <iface> down iw dev <iface> set type **mp** iw dev <iface> set meshid BluesMesh iw dev <iface> set channel <reg_db> ifconfig <iface> up

Tab. 2: Fase di setup nodi mesh

I nodi partecipanti al mesh devono avere tutti lo stesso mesh-id (o ESSID), oltre ad essere sullo stesso canale, ed è cura del driver wifi mantenere la consistenza su questa informazione. Nel caso in cui non dovesse verificarsi tale associazione, in batman-adv è possibile configurare manualmente l'id della cella impostando il settimo bit del primo byte oppure impostando l'address con un valore iniziale di “02:”, come evidenziato in grassetto in Tab. 2. Inoltre per ciascun nodo batman-adv è consigliato un MTU di 1528 byte a causa dei 28 byte di header addizionali messi in ciascun pacchetto che deve essere inviato nel mesh.

1.5.3 Configurazione specifica per batman-adv

Poiché il focus implementativo è rivolto su batman-adv e 802.11s è mostrato solo per un confronto sulle performance dei due protocolli, viene illustrata di seguito in dettaglio la

configurazione dei nodi e della rete con batman-adv, con le relative tabelle di routing dei rispettivi nodi.

L'eterogeneità hardware e software del test è garantita dalla tabella di compatibilità del protocollo, la quale racchiude nella versione 14 tutte le versioni utilizzate nei vari nodi (v2012.2.0, v2012.2.0, v2012.3.0, v2012.4.0) [18] . Il modulo batman-adv è già presente nella mainline del kernel, ma nei due nodi eBox3300 è stata aggiunta l'ultima versione con un processo di cross-compilazione. La batctl utilizzata è la v2012.4.0, indipendente dalla versione del modulo a meno di differenze di versione macroscopiche, poiché è un tool che si limita ad accedere i file generati dinamicamente dal kernel nella cartella /sys/class/net/ $iface/batman-adv .

Per ciascun nodo partecipante al mesh vi possono essere più interfacce fisiche mappate su un interfaccia virtuale chiamata bat0 alla quale viene assegnato automaticamente un indirizzo MAC e un indirizzo IPv6, come illustrato:

Interfaccia virtuale bat0

bat0 Link encap:Ethernet HWaddr 56:df:3e:ad:5e:01
inet6 addr: **fe80**::54df:3eff:fead:5e01/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:1497 (1.4 KiB)

alla quale è associata una o più interfacce fisiche

Interfacce fisiche

wlan1 IEEE 802.11bgn ESSID:"BluesMesh"
Mode:Ad-Hoc Frequency:2.437 GHz Cell: 02:00:D3:AD:B3:3F
Tx-Power=27 dBm
Retry long limit:7 RTS thr:off Fragment thr:off
Encryption key:off
Power Management:off

```
wlan3     IEEE 802.11bgn  ESSID:"BluesMesh"
          Mode:Ad-Hoc  Frequency:2.437 GHz  Cell: 02:00:D3:AD:B3:3F
          Tx-Power=27 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
```

E' possibile assegnare ad ogni nodo un indirizzo IPv4 manualmente, oppure lavorare con quello IPv6, verificandone la funzionalità con un ping6:

```
ping6 fe80:<IP> -I bat0
```

L'assegnamento può essere inoltre fatto tramite DHCP Server o DHCP6 Server nel caso di voglia usare IPv6 con assegnamento dinamico. Attraverso il meccanismo DHCP è inoltre gestita la possibilità avere più gateway con accesso ad internet nella rete., ma poiché batman-adv è un protocollo layer 2, distingue concettualmente la manipolazione della default route IP dall'essenza stessa del protocollo di livello 2, pertanto disabilita di default tale opzione. Ai fini del test, è stata attivata sul nodo 1 con un server DHCP sull'interfaccia bat0. Sullo stesso nodo, è stato attivata anche la funzionalità di vis-server, per la visualizzazione grafica della topologia della rete.

Attivazione funzionalità gateway server e annuncio

```
batctl gw_mode server
```

Attivazione funzionalità vis server

```
batctl vm server
```

Vengono illustrati di seguito i risultati dei test di rete con iperf dal nodo 5 al nodo 4, e la topologia della rete in istanti di tempo diversi.

Nodo 5:

```
iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

---

[4] local 10.0.0.6 port 5001 connected with 10.0.0.4 port 52341

[ID] Interval Transfer Bandwidth

[4] 0.0-21.5 sec 3.00 MBytes 1.17 Mbits/sec

[5] local 10.0.0.6 port 5001 connected with 10.0.0.4 port 52342

[4] local 10.0.0.6 port 5001 connected with 10.0.0.4 port 52343

[4] 0.0-220.7 sec 8.00 MBytes 704 Kbits/sec

Tabella di routing

batctl o

[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/1c:4b:d6:85:bf:df (bat0)]

Originator last-seen (#/255) Nexthop [outgoingIF]: Potential nexthops ...

00:06:4f:7b:71:b3 0.252s (238) 00:06:4f:7b:71:b3 [wlan0]: 00:0c:42:61:d4:22 (0) 00:27:19:bd:2f:88 (0) 00:c0:49:55:3b:4e (0) 00:06:4f:7b:71:b3 (238)

00:0c:42:61:d4:22 0.652s (228) 00:0c:42:61:d4:22 [wlan0]: 00:06:4f:7b:71:b3 (0) 00:22:3f:05:74:29 (0) 00:c0:49:55:3b:4e (0) 00:27:19:bd:2f:88 (0) 00:0c:42:61:d4:22 (228)

00:27:19:bd:2f:88 0.880s (248) 00:27:19:bd:2f:88 [wlan0]: 00:0c:42:61:d4:22 (0) 00:22:3f:05:74:29 (0) 00:c0:49:55:3b:4e (0) 00:06:4f:7b:71:b3 (188) 00:27:19:bd:2f:88 (248)

00:c0:49:55:3b:4e 0.432s (255) 00:c0:49:55:3b:4e [wlan0]: 00:0c:42:61:d4:22 (0) 00:27:19:bd:2f:88 (0) 00:22:3f:05:74:29 (0) 00:06:4f:7b:71:b3 (0) 00:c0:49:55:3b:4e (255)

00:22:3f:05:74:29 0.144s (238) 00:06:4f:7b:71:b3 [wlan0]: 00:0c:42:61:d4:22 (0) 00:c0:49:55:3b:4e (0) 00:06:4f:7b:71:b3 (238) 00:22:3f:05:74:29 (237)

Nodo 4

iperf -c 10.0.0.6

TCP window size: 85.3 KByte (default)

---

[4] local 10.0.0.4 port 5001 connected with 10.0.0.6 port 52341

[ID] Interval Transfer Bandwidth

[4] 0.0-21.5 sec 3.00 MBytes 1.17 Mbits/sec

Tabella di routing

batctl o

[B.A.T.M.A.N. adv 2011.4.0, MainIF/MAC: wlan0/00:0c:42:61:d4:22 (bat0)]

Originator last-seen (#/255) Nexthop [outgoingIF]: Potential nexthops ...

00:06:4f:7b:71:b3 0.276s (246) 00:06:4f:7b:71:b3 [wlan0]: 1c:4b:d6:85:bf:df (220) 00:27:19:bd:2f:88 (218) 00:c0:49:55:3b:4e (241) 00:06:4f:7b:71:b3 (246)

00:27:19:bd:2f:88 0.764s (241) 00:c0:49:55:3b:4e [wlan0]: 1c:4b:d6:85:bf:df (220) 00:22:3f:05:74:29

(202) 00:06:4f:7b:71:b3 (192) 00:c0:49:55:3b:4e (241) 00:27:19:bd:2f:88 (235)

00:c0:49:55:3b:4e 0.236s (255) 00:c0:49:55:3b:4e [wlan0]: 1c:4b:d6:85:bf:df (219) 00:22:3f:05:74:29 (217) 00:06:4f:7b:71:b3 (209) 00:27:19:bd:2f:88 (216) 00:c0:49:55:3b:4e (255)

00:22:3f:05:74:29 0.148s (255) 00:22:3f:05:74:29 [wlan0]: 1c:4b:d6:85:bf:df (222) 00:27:19:bd:2f:88 (228) 00:c0:49:55:3b:4e (239) 00:06:4f:7b:71:b3 (246) 00:22:3f:05:74:29 (255)

1c:4b:d6:85:bf:df 0.068s (254) 1c:4b:d6:85:bf:df [wlan0]: 00:22:3f:05:74:29 (110) 00:27:19:bd:2f:88 (199) 00:c0:49:55:3b:4e (233) 00:06:4f:7b:71:b3 (106) 1c:4b:d6:85:bf:df (254)

Nodo 2

Tabella di routing

batctl o

[B.A.T.M.A.N. adv 2011.4.0, MainIF/MAC: wlan1/00:c0:49:55:3b:4e (bat0)]

Originator last-seen (#/255) Nexthop [outgoingIF]: Potential nexthops ...

00:06:4f:7b:71:b3 0.400s (194) 1c:4b:d6:85:bf:df [wlan1]: 00:0c:42:61:d4:22 (186) 1c:4b:d6:85:bf:df (194) 00:c0:49:55:3b:4e (192) 00:06:4f:7b:71:b3 (191)

00:0c:42:61:d4:22 3.920s (226) 00:0c:42:61:d4:22 [wlan1]: 1c:4b:d6:85:bf:df (190) 00:06:4f:7b:71:b3 (169) 00:22:3f:05:74:29 (213) 00:c0:49:55:3b:4e (217) 00:0c:42:61:d4:22 (226)

00:27:19:bd:2f:88 0.080s (250) 00:27:19:bd:2f:88 [wlan1]: 00:0c:42:61:d4:22 (0) 1c:4b:d6:85:bf:df (177) 00:22:3f:05:74:29 (213) 00:06:4f:7b:71:b3 (166) 00:27:19:bd:2f:88 (250)

00:22:3f:05:74:29 0.520s (238) 00:22:3f:05:74:29 [wlan1]: 00:0c:42:61:d4:22 (212) 1c:4b:d6:85:bf:df (205) 00:c0:49:55:3b:4e (228) 00:06:4f:7b:71:b3 (187) 00:22:3f:05:74:29 (238)

1c:4b:d6:85:bf:df 0.670s (233) 1c:4b:d6:85:bf:df [wlan1]: 00:0c:42:61:d4:22 (195) 00:06:4f:7b:71:b3 (105) 00:22:3f:05:74:29 (134) 00:c0:49:55:3b:4e (220) 1c:4b:d6:85:bf:df (233)

Nodo 3

Tabella di routing

batctl o

[B.A.T.M.A.N. adv 2011.4.0, MainIF/MAC: wlan1/00:27:19:bd:2f:88 (bat0)]

Originator last-seen (#/255) Nexthop [outgoingIF]: Potential nexthops ...

00:06:4f:7b:71:b3 0.400s (194) 1c:4b:d6:85:bf:df [wlan1]: 00:0c:42:61:d4:22 (186) 1c:4b:d6:85:bf:df (194) 00:c0:49:55:3b:4e (192) 00:06:4f:7b:71:b3 (191)

00:0c:42:61:d4:22 3.920s (226) 00:0c:42:61:d4:22 [wlan1]: 1c:4b:d6:85:bf:df (190) 00:06:4f:7b:71:b3 (169) 00:22:3f:05:74:29 (213) 00:c0:49:55:3b:4e (217) 00:0c:42:61:d4:22 (226)

00:c0:49:55:3b:4e 0.080s (250) 00:c0:49:55:3b:4e [wlan1]: 00:0c:42:61:d4:22 (0) 1c:4b:d6:85:bf:df (177) 00:22:3f:05:74:29 (213) 00:06:4f:7b:71:b3 (166) 00:c0:49:55:3b:4e (250)

00:22:3f:05:74:29 0.520s (238) 00:22:3f:05:74:29 [wlan1]: 00:0c:42:61:d4:22 (212) 1c:4b:d6:85:bf:df (205) 00:c0:49:55:3b:4e (228) 00:06:4f:7b:71:b3 (187) 00:22:3f:05:74:29 (238)

1c:4b:d6:85:bf:df 0.670s (233) 1c:4b:d6:85:bf:df [wlan1]: 00:0c:42:61:d4:22 (195) 00:06:4f:7b:71:b3

(105) 00:22:3f:05:74:29 (134) 00:c0:49:55:3b:4e (220) 1c:4b:d6:85:bf:df (233)

Nodo 1

Tabella di routing

batctl o

[B.A.T.M.A.N. adv 2012.3.0, MainIF/MAC: wlan1/00:22:3f:05:74:29 (bat0)]

Originator last-seen (#/255) Nexthop [outgoingIF]: Potential nexthops ...

00:0c:42:61:d4:22 0.340s (211) 00:0c:42:61:d4:22 [wlan1]: 1c:4b:d6:85:bf:df (153) 1c:4b:d6:85:bf:df (45) 00:27:19:bd:2f:88 (183) 00:27:19:bd:2f:88 (171) 00:c0:49:55:3b:4e (191) 00:c0:49:55:3b:4e (191) 00:0c:42:61:d4:22 (211) 00:0c:42:61:d4:22 (205)

00:27:19:bd:2f:88 0.340s (248) 00:27:19:bd:2f:88 [wlan1]: 00:0c:42:61:d4:22 (201) 00:0c:42:61:d4:22 (196) 1c:4b:d6:85:bf:df (61) 1c:4b:d6:85:bf:df (205) 00:c0:49:55:3b:4e (237) 00:c0:49:55:3b:4e (237) 00:27:19:bd:2f:88 (248) 00:27:19:bd:2f:88 (239)

00:c0:49:55:3b:4e 0.780s (255) 00:c0:49:55:3b:4e [wlan1]: 00:0c:42:61:d4:22 (198) 00:0c:42:61:d4:22 (191) 1c:4b:d6:85:bf:df (61) 1c:4b:d6:85:bf:df (202) 00:27:19:bd:2f:88 (203) 00:27:19:bd:2f:88 (220) 00:c0:49:55:3b:4e (253) 00:c0:49:55:3b:4e (255)

1c:4b:d6:85:bf:df 0.776s (231) 1c:4b:d6:85:bf:df [wlan3]: 00:0c:42:61:d4:22 (180) 00:0c:42:61:d4:22 (183) 00:c0:49:55:3b:4e (225) 00:27:19:bd:2f:88 (202) 00:27:19:bd:2f:88 (191) 00:c0:49:55:3b:4e (225) 1c:4b:d6:85:bf:df (68) 1c:4b:d6:85:bf:df (231)

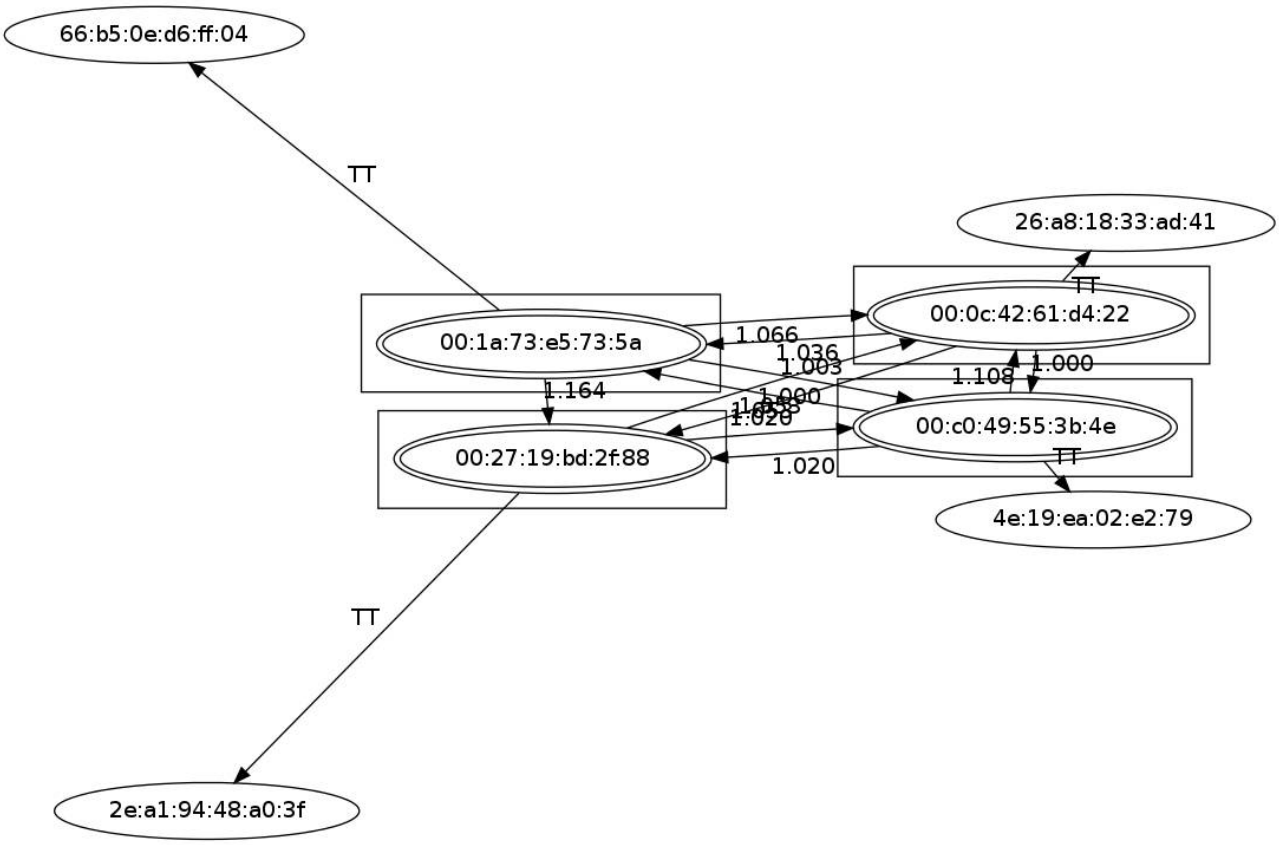


Fig. 31: Topologia setup 1

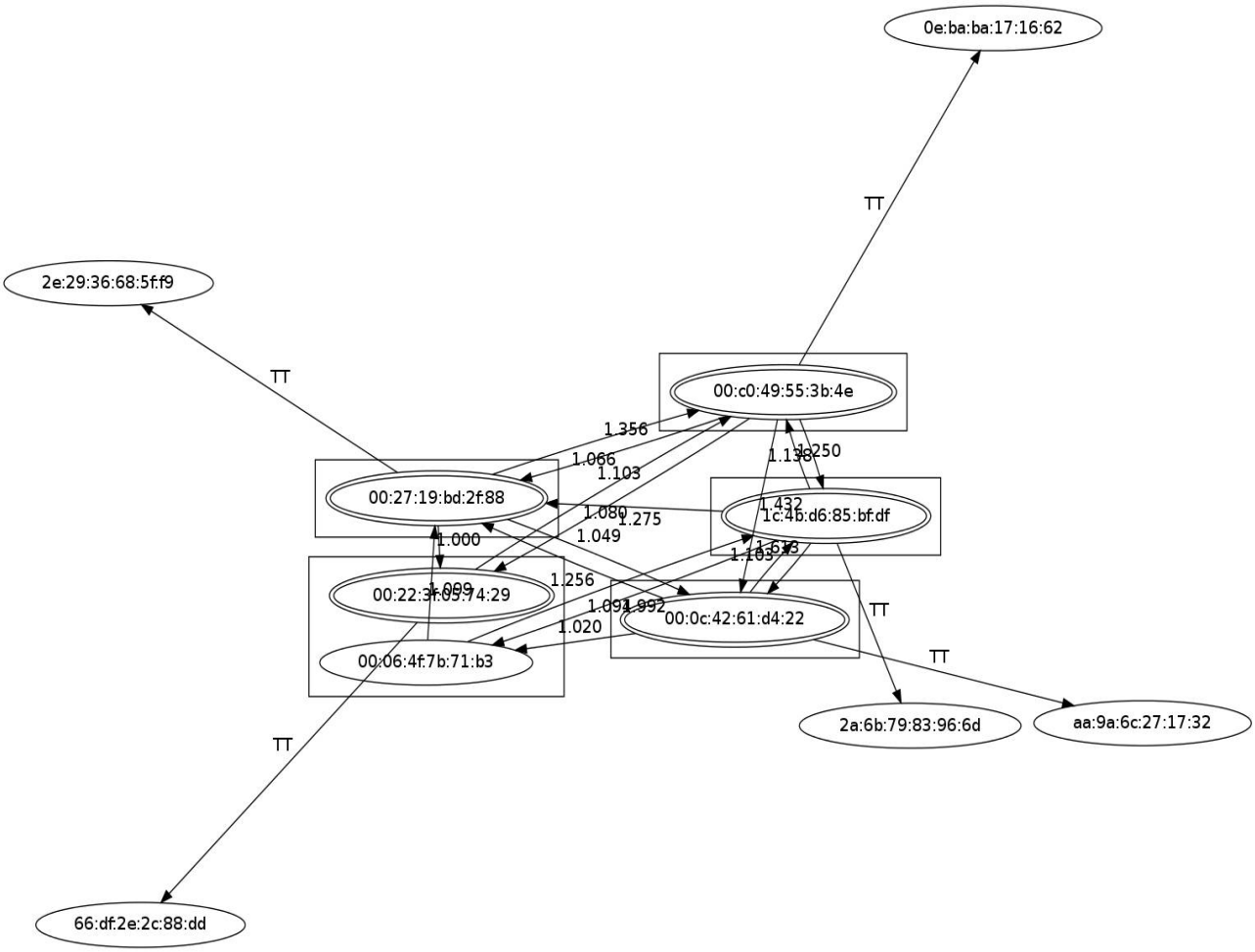


Fig. 32: Topologia a regime

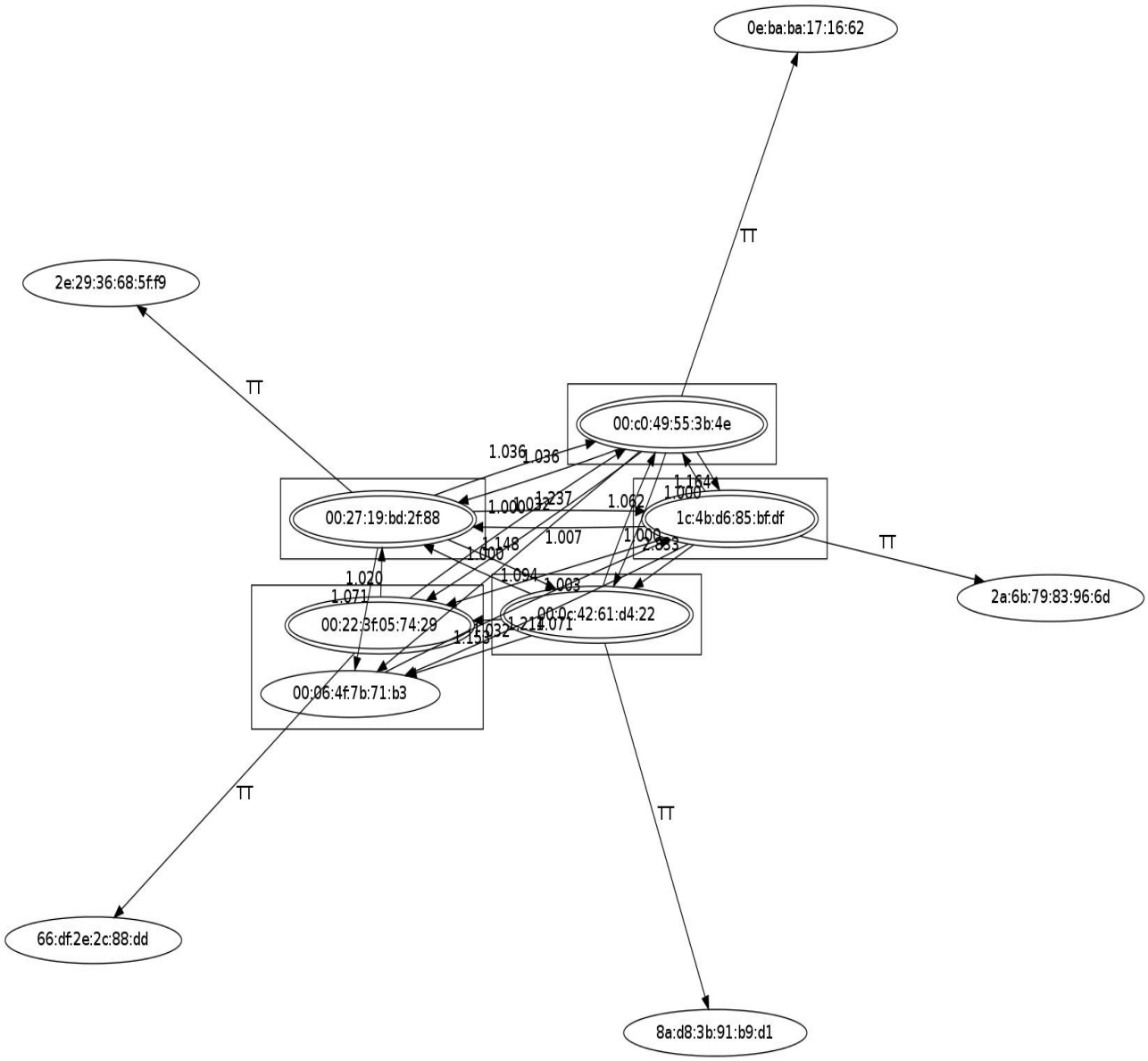


Fig. 32: Topologia test

La rappresentazione topologica può essere ottenuta anche tramite JSON, per una visualizzazione real-time via web, oppure tramite semplice testo.

Nella rappresentazione grafica:

- Ellissi: nodi e interfacce BATMAN-adv con il rispettivo indirizzo MAC
- Box: interfacce appartenenti ad un nodo BATMAN-adv
- Frecce con numeri: TQ (nella forma 1/TQ) da un interfaccia di un nodo BATMAN-adv ad un altra interfaccia di un altro nodo BATMAN-adv.

Per la modalità gateway inoltre è stato fornito l'accesso ad internet al Nodo 1, attraverso l'interfaccia ethernet eth0, dando l'accesso ad internet ad tutti i nodi del mesh, ai quali attraverso l'annuncio del server gateway inoltrano le loro richieste verso la rete esterna (internet) al Nodo 1, attraverso il path migliore individuato per il raggiungimento dello stesso.

Script condivisione internet

```
#!/bin/sh
echo 1 > /proc/sys/net/ipv4/ip_forward
/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
/sbin/iptables -A FORWARD -i eth0 -o bat0 -m state --state RELATED,ESTABLISHED -j ACCEPT
/sbin/iptables -A FORWARD -i bat0 -o eth0 -j ACCEPT
```

Tale script può essere sostituito con un bridge fra l'interfaccia che ha internet e la bat0.

1.5.4 Test combinato TCP/UDP

Attraverso iperf è stato effettuato il test anche su UDP. Dai test rilevati si sono ottenuti i risultati finali espressi in Tab. 4

	802.11s	batman-adv
Throughput TCP	1.18 Mbps	1.30 Mbps
Throughput UDP	2.16 Mbps	2.01 Mbps

Tab. 4 : Tabella confronto test di rete con iperf

Conclusioni

Dopo un'ampia panoramica sulle reti mesh e sui protocolli di routing (o path selection) in letteratura, con particolare attenzione sull'aspetto energetico, è stato presentato il protocollo layer 2 BATMAN-adv nelle sue funzionalità teoriche e pratiche. In particolare, attraverso un semplice test in ambiente indoor è stato possibile dimostrare la bontà del protocollo sia in termini di performance, sia in termini di eterogeneità e indipendenza dalla piattaforma, sotto determinate condizioni. Ciò che rimane da studiare e testare ulteriormente è l'aspetto energetico in merito all'adozione di questo protocollo per le reti mesh, sia in ambiente indoor, sia outdoor dove potrebbero essere utilizzati pannelli solari per l'auto-alimentazione dei dispositivi coinvolti, e batterie che eviterebbero la dipendenza del dispositivo dalla corrente continua e l'adattabilità in ambienti e contesti diversi, anche ostili. La diffusione inoltre del sistema operativo Android, basato su kernel Linux, permette di estendere il discorso a milioni di terminali, fino ad ora considerati solo nei classici meccanismi station-ap, che potrebbero invece partecipare attivamente al mesh ed essere essi stessi dei nodi intermedi per altre comunicazioni. Lo spostamento del focus sull'aspetto energetico richiede pertanto una ricerca esaustiva e comprovata su una metrica energeticamente efficiente da adottare all'interno del protocollo, oltre ai miglioramenti nel protocollo come ad esempio quelli elencati nel paragrafo 1.4.8 . La selezione di un link attraverso un interfaccia per mezzo della determinazione della soglia TQ potrebbe essere la chiave di volta per un meccanismo che possa mantenere le comprovate performance del protocollo, insieme a politiche energy-aware, sempre più necessarie in uno sviluppo tecnologico che tende ad essere più “embedded” e distribuito, piuttosto che standalone e fisso. Inoltre la natura open source del progetto legato all'implementazione del protocollo, inserito direttamente nello sviluppo del kernel Linux [19], favorisce un ambiente stimolante e creativo, che insieme alla forza della condivisione delle idee e del lavoro, diviene terreno fertile per la ricerca per il mondo accademico, spinto dalla stesso nobile e producente motus.

Bibliografia

[1] 802.11s-2011 - “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 10: Mesh Networking” - IEEE Standard for Information Technology--Telecommunications and information exchange between systems

[2] 802.11-2012 - “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications ” - IEEE Standard for Information Technology

[3] ivi p. 1353

[4] ivi

[5] ivi

[6] ivi

[7] Optimized Link State Routing Protocol (OLSR) – IETF RFC 3623

http://www.ietf.org/rfc/rfc3626.txt

[8] J. Chang and L. Tassiulas, “Energy conserving routing in wireless ad hoc networks”, IEEE Infocom, 2000 p.10

[9] ivi p.12

[10] ivi pp. 22-31.

[11] M.M. Mhlanga T.C. Nyandeni , T. Olwal , N. Ntlatlapa and M.O Adigun, “Energy-Aware Path Selection Metric for IEEE” , University of Zuzuland, 2009

[12] Chang, J-H. and Tassiulas, L. “Maximum lifetime routing in wireless sensor networks”, IEEE ACM Trans. Networking 12, 2004, no. 4, 609-619.

[13] Optimized Link State Routing Protocol (OLSR) – IETF RFC 3623 – Page 38, Cap. 8.3.1 http://www.ietf.org/rfc/rfc3626.txt

[14] F.De Rango,M.Fotino,S.Marano, “EE-OLSR: Energy efficient OLSR routing protocol for Mobile Ad-hoc networks”, D.E.I.S. Department, University of Calabria , 2009

[15] The OLSR.org story – OpenMesh Consortium

http://www.open-mesh.org/projects/open-mesh/wiki/The-olsr-story

[16] Multi-links optimization - OpenMesh Consortium

http://www.open-mesh.org/projects/batman-adv/wiki/Multi-link-optimizations-technical

[16] Linux Wireless IEEE 802.11 Subsystem

http://www.linuxwireless.org

[17] Compat wireless - tabella dei driver e dei chipset wireless supportati nel kernel Linux (torvalds/linville)

http://www.linuxwireless.org/en/users/Drivers

[18] Batman-adv COMPAT _VERSION

http://www.open-mesh.org/projects/batman-adv/wiki/Compatversion

[19] Kernel Archive

http://kernel.org