Universitá degli Studi "Roma Tre"

Facoltá di Ingegneria

Metodi e Strumenti per l'Analisi delle Metriche nelle Wireless Mesh Network: uso di probe basati su pacchetti *unicast*

Tesi di Laurea Vecchio Ordinamento in

Ingegneria Informatica

Relatore

Prof. Giuseppe Di Battista

Candidato

Fabrizio Giordano

Anno Accademico 2006/2007

Indice

Prefazione

Il mondo delle reti é in continuo fermento ed evoluzione. Con questo lavoro si vorrebbe aggiungere un piccolo mattone nell'edificio senza confini che si va costruendo: l'interconnessione e la comunicazione senza frontiere tra gli utenti più disparati.
Sempre piú studi e ricerche si stanno orientando verso il mondo della comunicazione wireless, che essendo libera dalle connessioni via cavo, è più flessibile ed adattiva verso situazioni di impiego infrastrutturalmente leggere e mobili. Esistono diverse tecnologie con cui si puó costruire una rete wireless, in particolare, fra le tante, questo lavoro é focalizzato sullo studio di reti mesh costruite secondo lo standard 802.11, detto anche Wifi.
Le reti mesh sono lo stato più avanzato delle reti wireless che oggi stanno trovando aree di applicazione pratica in vari campi di utilizzazione e sono, quindi, suscettibili di un rapido allargamento della loro diffusione. Tuttavia, le reti mesh non sono esenti da problemi che dovranno trovare una soluzione nel prossimo futuro. Questo studio si focalizza su uno dei suddetti problemi: l'affidabilità delle reti mesh nell'assicurare un corretto e completo trasferimento dei dati fra i suoi nodi in relazione alle capacità di autoriconfigurabilità delle reti di link punto-punto che le sottendono I risultati di questo studio si concretizzano nella ideazione di uno strumento, detto TOOL, di supporto allo studio e all'analisi dei protocolli di routing mesh wireless in grado di

rilevare informazioni che permettano di migliorare l'efficienza dei networks wi-fi, fino al livello fisico della pila ISO/OSI. Uno studio approfondito dei protocolli di routing mesh quali OSLR E B.A.T.M.A.N. ha evidenziato che tali protocolli, comunemente utilizzati nelle reti mesh,talvolta (soprattutto un condizioni di criticità) non sono in grado di compilare nel modo migliore le tabelle di ruoting dei nodi e, pertanto, non instradano in maniera ottimale i pacchetti dati. Ciò comporta, conseguentemente, la possibilità di perdite, difficilmente rilevabili in real time, di dati/informazioni anche importanti. Tali perdite possono essere di particolare rilevanza per gli utenti, anche in considerazione de fatto che l'impiego delle reti mesh si indirizza soprattutto verso aree in cui i tempi operativi sono real/near real time (sistemi di sorveglianza mobili, protezione civile, controllo del territorio, raccolta dati sul campo,ecc.). Nel corso dello studio ci si é resi conto che il comportamento non corretto degli attuali protocolli di routing mesh era dovuto alle carenze del meccanismo con cui effettuano su i link il probing finalizzato ad individuare la configurazione di rete per le esigenze di istradamento, a ciò consegue anche una insufficienza delle metriche utilizzate a tale scopo. Nello studio si è individuato all'interno del TOOL un meccanismo di probing innovativo che permette un miglioramento quantitativo e qualitativo dei dati raccolti dallo sniffing nelle connessioni link. Con il supporto dei dati restituiti dal TOOL , si sono descritte delle proposte di metriche alternative a quelle attualmente utilizzate, che portano a valutazioni piú fedeli del reale stato di un link, permettendo, quindi, valutazioni operative della bontà delle sue prestazioni. Lo studio è stato svolto partendo dalla comprensione delle caratteristiche della tecnologia wireless e delle sue differenziazioni da quella delle comunicazioni su cavo. Nell'approfondire il mondo wireless si sono considerate le reti in modalità *infrastruttura* ed in modalità *ad-hoc* evidenziandone le differenze

pertinenenti agli scopi dello studio stesso. Di seguito si tratta delle reti mesh, dello standard802.11 in generale e dei protocolli di routing mesh piú comuni. Vengono poi analizzate alcune metriche utilizzate dai protocolli mesh al fine di avanzare delle osservazioni sulle loro criticitá ed indicare delle soluzioni alternative. Successivamente si è descritto il TOOL realizzato per testare e analizzare i link di una rete mesh, supportando il tutto con i risultati di alcuni test effettuati sul testbed della community wireless ninux.org.. Infine si illustrano alcune possibili integrazioni al TOOL.

Il TOOL é stato pensato per analizzare l'andamento nel tempo di un link wireless e memorizzare dati relativi al traffico unicast e brodacast ai livelli 1,2 e 3 della pila ISO/OSI. Osservando i dati restituiti, é possibile dedurre l'efettiva bontá del link. Il TOOL di per sé non propone una metrica alternativa, anche se la modalitá con cui effettua i probe potrebbe esserlo; sta a chi legge i dati trarre le conclusioni sull'andamento di ogni singolo link per poi confrontarle con le decisioni prese dal protocollo di routing e descrivere una metrica adatta allo scopo. Il TOOL permette inoltre di effettuare test eterogenei, impostando a piacimento dell'utente i suoi paramentri. Oltre ad essere un motore di test, dai dati che vengono restituiti, si puó procedere alla progettazione di una metrica piú precisa e completa.

Il codice generato é stato rilasciato sotto licenza GNU/GPL.

Capitolo 1

Introduzione alle Reti Mesh

1.1 Tipi di reti

Il termine wireless é sempre piú sinonimo di ampliamento rapido delle frontiere di comunicazione, abbattimento del digital divide, abbattimento dei costi, rispetto dell'ambiente. Vediamo perché.

1.1.1 Reti wireless e reti cablate

Le reti realizzate su cavo differiscono dalle reti wireless non solo per la tecnologia utilizzata, ma anche per l'impatto ambientale e l'importanza sociale. La grande flessibilitá delle reti wireless genera sempre piú numerosi studi e ricerche nel campo. Parlando in termini tecnologici, ció che le differenzia dalle reti su cavo é l'indipendenza dall'infrastruttura fisica legata alla stesura dei cavi stessi. Basta infatti che due o piú apparati wireless siano a portata radio per instaurare una comunicazione tra loro. La tecnologia wireless negli utlimi anni ha registrato un'enorme crescita in termini di soluzioni e servizi offerti, non piú solo alle imprese, ma anche ai singoli utenti. L'indipendenza dal cavo infatti permette una maggior libertá nel posizionamento degli appa-

rati e ha introdotto la mobilitá, cosa che su cavo non era pensabile. Inoltre, tramite link wireless, si rende possibile il collegamento tra zone in cui altrimenti sarebbe molto difficile fornire connettivitá. L'abbattimento del digital divide é un motore che spinge fortemente in questa direzione. La connettivitá alla rete mondiale é ormai parte integrante della realtá moderna e futura, e lasciare isolate alcune zone del mondo, equivale quasi a condannarle ad una stasi evolutiva. É chiaro quindi che la wireless favorisce effettivamente l'interconnessione tra sistemi e quindi tra uomini ed ha un importante impatto sociale nel contribuire a istaurare l'utilizzo di comunicazione e servizi anche dove sarebbe fisicamente molto difficile.

In termini di costi siamo ben al di sotto di quelli che servono per la realizzazione di reti wired. Mentre per realizzare un link cablato é necessario far passare un cavo da un'estremitá all'altra, per realizzare lo stesso link ma con tecnologia wireless basta posizionare due apparati entro gli orizzonti radio (che dipendono anche dall'hardware impiegato). Negli utlimi anni i costi degli apparati wireless si sono letteralmente abbattuti e si é registrato un incremento delle vendite tanto che é possibile andare in un qualsiasi centro commerciale per avere l'imbarazzo della scelta sull'access point wireless da comprare.

Se ci incentriamo sull'impatto ambientale, é evidente che stendere un cavo implica un impatto ambientale piú gravoso (scavi, palificazioni, etc..) che sistemare apparati wireless in punti elettromagneticamente adeguati (tralicci, montagne, tetti, etc...).

In termini di consumo energetico, in diverse situazioni oggi é giá possibile alimentare un apparato wireless Linksys WRT e un'antenna da 18dbl con un pannello a cellule fotovoltaiche da 45W.

Peró, se da un parte la tecnologia wireless diminuisce il digital divide, ri-

spetta di piú l'ambiente e ha dei costi nettamente inferiori, dall'altra parte é molto sensibile alle interferenze tanto da rendere in alcuni casi impossibile il suoutilizzo. Si possono presentare anche situazioni in cui un link wireless é attivabile in un solo senso o attivabile asimmetricamente in entrambi i versi. D'altra parte non sono da trascurare le norme legislative che regolano la trasmissione tra apparati wireless sul suolo pubblico.

1.1.2 Reti wireless Mesh e Reti wireless Managed

Ci sono due modalitá possibili di livello Data Link per costruire una rete wifi: la modalitá infrastruttura o managed, e la modalitá ad-hoc. Di seguito le illustreremo brevemente entrambe anche se nella trattazione di questa tesi ci si é incentrati sulle reti mesh realizzate in ad-hoc.
In una rete managed é adottata un'architettura del tipo server-client in cui sono previsti dei nodi in modalitá *master* detti Access Point (server), che hanno il compito di instradare il traffico deli altri nodi, detti Client, che non possono comunicare direttamente tra loro (figura 1.1).

Un limite imposto da tale modalitá é dovuto alla centralizzazione dei nodi addetti al routing; il venir meno degli Access Point, comporterebbe la mancanza di connettivitá fra i nodi Client. Se uno dei Client esce fuori dal raggio dall'orizzonte dell'Access Point, seppur rimanendo in quello di altri Client, non potrebbe comunicare con questi ultimi.
É possibile interconnettere piú reti managed tra loro creando un backbone di livello superiore tra gli Access Point di varie reti (figura 1.2).

Nella modalitá ad-hoc é stata eliminata la figura dell'Access Point ed é possibile far comunicare direttamente due o piú nodi. In tal caso non si parla

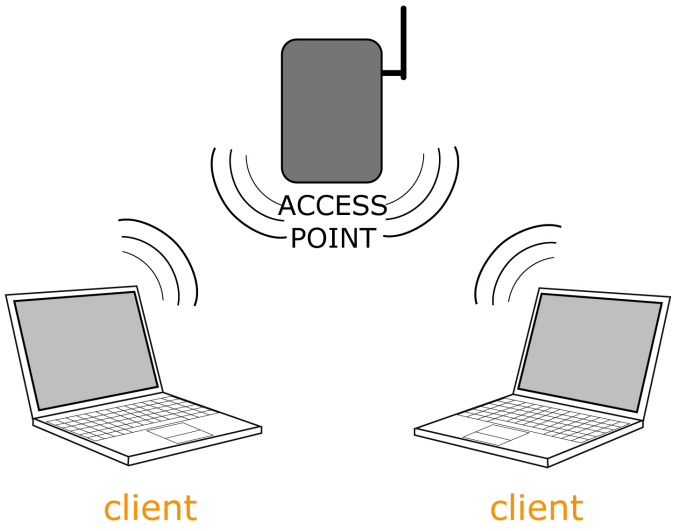


Figura 1.1: Modalità infrastruttura.

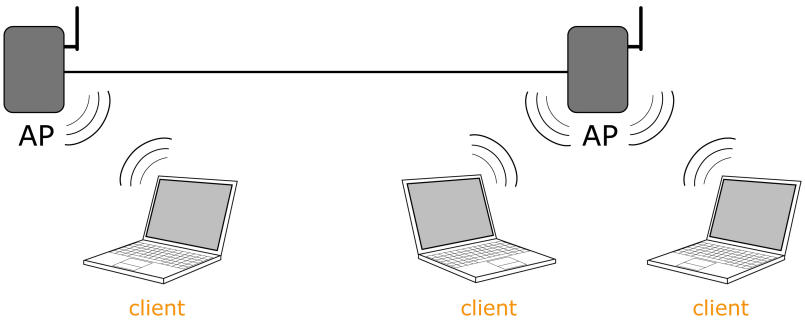


Figura 1.2: Più reti in modalità infrastruttura.

piú di Client e Access Point, ma solo di nodo. Si adottano pertanto i concetti di rete distribuita. L'unico limite che si presenta é dato dagli orizzonti radio dei nodi. Va evidenzioato che nella modalitá ad-hoc nessun nodo é in grado di fare da tramite nella comunicazione tra due nodi a lui adiacenti (cioé entro un hop di distanza figura 1.3).

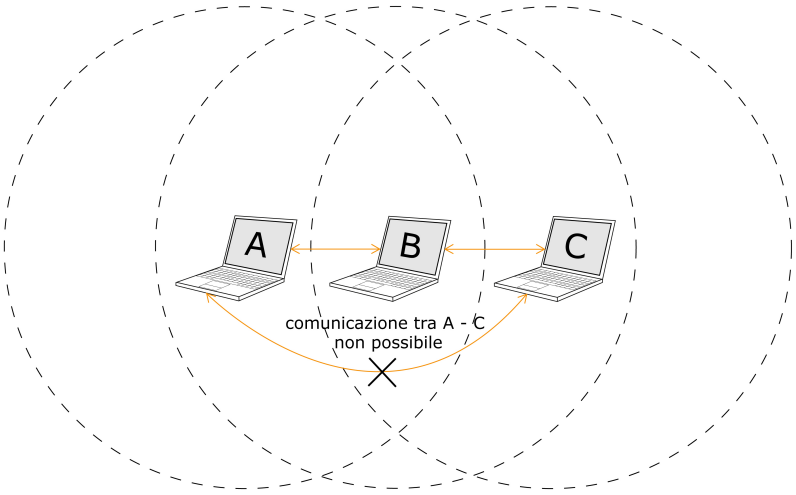


Figura 1.3: Modalità ad-hoc.

Per superare questo limite e rendere possibile la comunicazione tra nodi posti a piú di un hop di distanza, si deve prevedere un protocollo di routing che, in una rete in cui i nodi sono tutti in modalitá ad-hoc, prende il nome di protocollo di routing mesh. In una mesh si puó dire che ogni nodo é un router, poiché partecipa attivamente al routing occupandosi di instradare anche i dati che non sono diretti a lui (figura 1.4).

Una rete mesh amplifica di molto i vantaggi introdotti dalla tecnologia wireless. Basti pensare ad una realtá cittadina in cui é possibile mettere un nodo mesh su alcuni tetti, studiando una copertura radio, per interconnettere tutta la cittadinanza e offrire servizi di comune interesse come potrebbero eseere la realizzazione di una intranet cittadina, un servizio di telefonia VOIP, l'accesso a internet, ..., a costi bassissimi e indipendenti dalla stesura di cavi da parte degli ISP. Uno dei grandi pregi delle reti mesh é che riadattano dinamicamente il routing ai cambiamenti nella topologia della rete. La caduta di alcuni nodi non inibisce il traffico su tutta la rete, ma, se disponibili, vengono calcolate automaticamente delle nuove strade per raggiungere quei nodi che altrimenti rimarrebbero isolati.
Le reti mesh non sono peró prive di difetti. La gestione della sicurezza e la progettazione di un protocollo di routing mesh rappresentano tuttora un grosso ostacolo. Inoltre i nodi condividono il mezzo fisico, riducendo cosí la banda. Hanno inoltre trovato grandi consensi nel mondo delle '*sensors*

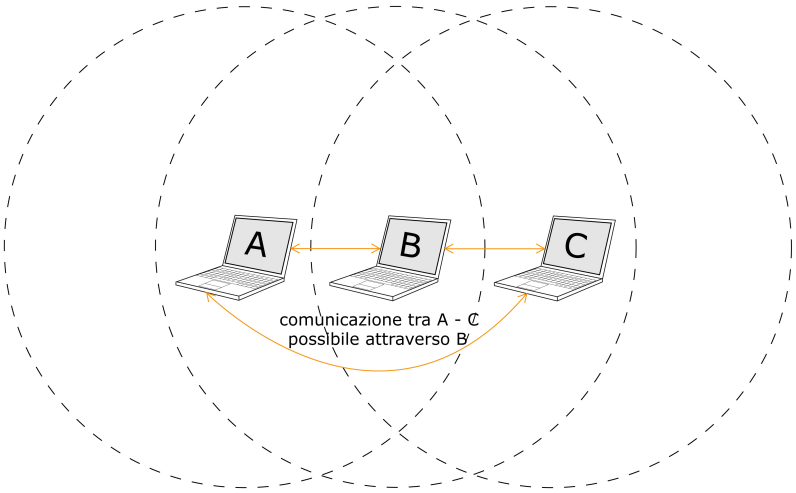


Figura 1.4: Rete wireless mesh.

network, ma questo argomento esula dalla trattazione di questa tesi.

1.1.3 802.11

Detto anche WiFi, é uno standard per le reti WLAN realizzato ad opera del gruppo 11 dell'IEEE 802. All'interno dello standard vengono definiti vari protocolli trasmissivi di livello fisico e Data Link della pila ISO/OSI, come mostrato in figura 1.5. Di seguito mostriamo quelli che ci interessano piú da vicino.

L'802.11b utilizza lo spettro di frequenze libero da licenze (banda ISM) nella banda dei 2.4 GHz. 802.11b ha la capacità di trasmettere al massimo 11Mbit/s e utilizza il Carrier Sense Multiple Access con Collision Avoidance (CSMA/CA) come metodo di trasmissione delle informazioni. Una buona parte della banda disponibile viene utilizzata dal CSMA/CA. In pratica il massimo trasferimento ottenibile è di 5.9 Mbit/s in TCP e di 7.1 Mbit/s in UDP. Metallo, acqua e in generale ostacoli solidi riducono drasticamente la portata del segnale. Il protocollo utilizza le frequenze nell'intorno dei 2.4GHz.

802.11g L'802.11g come l'802.11b utilizza la banda ISM. Utilizza le stesse frequenze del b cioè la banda di 2.4 GHz e fornisce una banda teorica di 54 Mbit/s che nella realtà si traduce in una banda netta di 24.7 Mbit/s, simile a quella dello standard 802.11a. È totalmente compatibile con lo standard b ma quando si trova a operare con periferiche b deve ovviamente ridurre la sua velocità a quella dello standard b.
Alcuni produttori introdussero delle ulteriori varianti chiamate g+ o Super G nei loro prodotti. Queste varianti utilizzavano l'accoppiata di due canali

per raddoppiare la banda disponibile anche se questo induceva interferenze con le altre reti e non era supportato da tutte le schede.

802.11b e 802.11g dividono lo spettro in 14 sottocanali da 22MHz l'uno. I canali sono parzialmente sovrapposti tra loro in frequenza, quindi tra due canali consecutivi esiste una forte interferenza. I 2 gruppi di canali 1, 6, 11 e 2, 7 e 12 non si sovrappongono fra loro e vengono utilizzati negli ambienti con altre reti wireless.

Utilizzando antenne direzionali esterne dotate di alto guadagno si è in grado di stabilire delle connessioni punto a punto del raggio di molti chilometri. Utilizzando ricevitori con guadagno di 80 decibel si può arrivare a 8 chilometri o se le condizioni del tempo sono favorevoli anche a distanze maggiori ma sono situazioni temporanee che non consentono una copertura affidabile. Quando il segnale è troppo disturbato o debole lo standard prevede di ridurre la velocità massima a 5.5, 2 o 1 Mbit/s per consentire al segnale di essere decodificato correttamente.

L'802.11a utilizza la banda ISM dei 5.8 GHz. Operando in bande di frequenze libere i dispositivi b e g possono essere influenzati da telefoni cordless, trasmettitori mobili e in genere da tutti gli apparecchi che utilizzano quella banda di frequenze. Questo standard utilizza lo spazio di frequenze nell'intorno dei 5 GHz e opera con una velocità massima di 54 Mbit/s sebbene nella realtà la velocità reale disponibile all'utente sia di circa 20 Mbit/s. La velocità massima può essere ridotta a 48, 36,34,18,9 o 6 se le interferenze elettromagnetiche lo impongono. Lo standard definisce 12 canali non sovrap-

posti, 8 dedicati alle comunicazioni interne e 4 per le comunicazioni punto a punto. Quasi ogni stato ha emanato una direttiva diversa per regolare le frequenze ma dopo la conferenza mondiale per la radiocomunicazione del 2003 l'autorità federale americana ha deciso di rendere libere le frequenze utilizzate dallo standard 802.11a.

Standard	Modulazione	Frequenza	Velocità di trasferimento (Mbit/s)
802.11 legacy	FHSS, DSSS, Infrarossi	2.4 GHz, IR	1, 2
802.11b	DSSS, HR-DSSS	2.4 GHz	1, 2, 5.5, 11
"802.11b+" non-standard	DSSS, HR-DSSS (PBCC)	2.4 GHz	1, 2, 5.5, 11, 22, 33, 44
802.11a	OFDM	5.2, 5.8 GHz	6, 9, 12, 18, 24, 36, 48, 54
802.11g	DSSS, HR-DSSS, OFDM	2.4 GHz	1, 2, 5.5, 11; 6, 9, 12, 18, 24, 36, 48, 54, 125

Figura 1.5: Tabella 802.11 (*Wikipedia*).

1.2 Tipi di protocollo

I protocolli di routing su mesh si possono classificare secondo il loro comportamento. Comunemente si usa dividerli in base al momento di elaborazione dei cammini distinguendoli in protocolli reattivi, protocolli proattivi, protocolli ibridi e protocolli multipath.

Per protocolli reattivi, detti anche *on demand*, si intende quella classe di protocolli che costruiscono le rotte "on demand". Le informazioni relative al routing per instradare il traffico vengono ricercate solo quando necessario. Come conseguenza si ha una potenziale riduzione del traffico sui link e un poteziale risparmio del consumo energetico, ma viene introdotto un maggiore ritardo nel determinare un nuovo cammino. Sono adatti per le reti in cui

si ha una banda limitata e si cerca di generare il minor traffico possibile. Un esempio di applicazione sono le *sensor networks*. I principali meccanismi su cui si basano sono *Route discovery*, per imparare i nuovi cammini, *Route maintenance*, per testare se i cammini che giá si conoscono sono ancora validi e *Route delection*, per eliminare i cammini non piú validi.

I protocolli proattivi, detti *table driven*, costruiscono le informazioni relative al routing in anticipo. Come vantaggio si ha una riduzione del ritardo nella scoperta dei nuovi cammini rispetto ai protocolli reattivi, ma si genera potenzialmente piú overhead sulla rete e si consuma piú energia sui nodi. Ogni nodo sa come raggiungere ogni altro nodo della rete, anche se non deve instradargli del traffico. Ogni volta che viene modificata una entry nella tabella di routing di un nodo, deve essere propagata agli altri nodi della rete. Il meccanismo con cui vengono calcolate le nuove entry nella tabella di routing, distingue ulteriormente i protocolli proattivi.

I protocolli ibridi hanno un comportamento proattivo con i nodi vicini e un comportamento reattivo con i nodi lontani. In tal modo si limita l'overhead introdotto dal carattere proattivo e si limita il ritardo nella scoperta di nuovi cammini proprio del carattere reattivo.

I protocolli multipath sono caratterizzati dal mantenimento su ogni nodo di piú *path* alternativi per raggiungere una destinazione. Se il primo path dá un *route failure*, il nodo proverá i percorsi alternativi. Solo nel caso in cui tutti i percorsi alternativi restituiscono *route failure*, il nodo cercherá un nuovo path secondo un meccanismo di *route discovery*. Tali protocolli sono pensati per reti con nodi in continuo movimento e la loro efficienza aumenta

al crescere del grado di magliatura della rete.

1.3 Approcci al routing su Mesh

1.3.1 Link State Routing

Il Link State routing si basa sulla percezione dello stato dei link adiacenti ad ogni nodo. Ogni nodo acquisisce periodicamente informazioni sullo stato dei link adiacenti e invia tramite un meccanismo di *link state broadcast* un pacchetto di link state a tutti i nodi della rete. I nodi che ricevono tale pacchetto, verificano che si tratta di un'informazione piú aggiornata di quella di cui sono giá in possesso e provvedono, se necessario, ad aggiornare il proprio Link state database. Viene poi applicata una variante dell'algoritmo di Dijkstra per determinare il cammino minimo per raggiungere ogni nodo della rete sulla base del Link State Database. Ogni nodo ha l'intera visione della rete e calcola i cammini minimi assumendo di essere il nodo radice. OLSR é un esempio di protocollo di tipo Link State routing su mesh. Al crescere del numero di nodi nella rete, crescono le tabelle di routing e di conseguenza cresce anche la richiesta di maggiori risorse di calcolo sui nodi.

1.3.2 Distance Vector Routing

Tale approccio al routing é basato sull'algoritmo di Bellman-Ford. Contrariamente al Link State Routing, nel Distance Vector routing, un nodo non conosce l'intera topologia, ma solo i next hop per raggiungere gli altri nodi della rete. I nodi inviano periodicamente ai propri vicini (1 hop-neighbor) un vettore delle distanze da tutti i nodi della rete. Combinando i dati ricevuti, ogni nodo tiene aggiornata una tabella di vettori delle distanze, sulla

quale baserá il calcolo del cammino minimo associato ad ogni destinazione, e per ogni destinazione ne memorizza il solo next hop.
Il Distance Vector Routing é affetto dal problema del Count to Infinity. AODV é un esempio di protocollo Distance Vector su mesh.

1.3.3 Hierarchical Routing- scaling verticale

Si tratta di un approccio al routing che tende a minimizzare l'overhead generato nelle reti mesh di grandi dimensioni e a diminuire la dimensione delle tabelle di routing sui nodi. Si organizza la rete mesh in piú livelli, ma generalmente ne bastano due. La rete mesh viene divisa in zone che comunicano tra loro tramite dei gateway. Su ogni zona puó girare un protocollo di routing mesh differente. I gateway si occuperanno di inoltrare il traffico tra una zona ed un'altra. Si viene a creare cosí una sorta di backbone. Ci sono peró delle problematiche che sono ancora oggetto di studio [Rif. tesi hitachi]:

- va gestito il caso di mobilitá di un nodo da una zona all'altra;
- va ricercato un metodo che imponga dinamicamente ed automaticamente la divisione in zone della mesh (*clustering dynamic*);
- va gestita, se prevista, l'elezione dei nodi che faranno da gateway.

1.3.4 Fish Eye- scaling trasversale

Si tratta di un meccanismo alternativo al routing gerarchico e meno invasivo, atto a diminuire l'overhead indotto sulla rete da OLSR e la dimensione

delle tabelle di routing sui nodi. É stato ideato da Gerla[Rif]. Il Fish Eye diminuisce la frequenza di invio di pacchetti di Topology Control verso le destinazioni piú lontane (intese come numero di hop) nella rete. É una soluzione obbligatoria per reti OLSR di grande dimensione (ex. Freifunk Network di Berlino). Non introduce nessun meccanismo di gerarchia tra i nodi e per questo si parla di scaling verticale.

1.4 Panoramica sui piú noti protocolli di Routing su Mesh

1.4.1 OLSR

OLSR, Optimized Link State Routing [Rif. RFC], é un protocollo di routing frutto della tesi di laurea di Andreas Tonnensen presso l'universitá di Oslo. Si tratta di un protocollo di tipo Link State routing e si basa sull'invio di messaggi broadcast di HELLO ad 1 hop, per far conoscere ai nodi della rete i propri vicini (*1-hop-neighbor*). I pacchetti di Link State broadcast prendono il nome di messaggi di Topology Control (TC) e vengono inviati periodicamente da ogni nodo su tutta la rete attraverso dei nodi particolari chiamati Multi Point Relay (MPR). Ogni nodo elegge i propri MPR tra tutti i suoi vicini con cui ha un link simmetrico. Gli MPR vengono eletti in modo da garantire la copertura di tutti i vicini a due hop (2-hop-neighbor). Con questo meccanismo si limita fortemente l'overhead sulla rete proprio dei protocolli Link State routing, anche se a volte a discapito della reattivitá ai cambiamenti della topologia. Nativamente nasce con una metrica chiamata ISTERESI, che ha il compito di considerare non funzionanti quei link che fluttuano eccessivamente nel tempo. La comunitá wireless Freifunk di Berli-

no ha proposto l'implementazione di una metrica alternativa, basata su ETX, che aggiunge un grado di bontá e simmetria sui link. Di questo tratteremo dettagliatamente nel capitolo relativo alle metriche.
Sono state proposte diverse variazioni nel corso del tempo di OLSR. Tra queste menzioniamo FastOLSR[Rif. RR-4510.pdf], che si propone di aumentare la reattivitá del protocollo, in caso di mobilitá dei nodi, generando dei pacchetti FastHELLO di piccole dimensioni e ad intervalli piú stretti. NoaOLSR é invece propone un meccanismo per implementare un servizio di DHCP su rete OLSR.

1.4.2 B.A.T.M.A.N.

B.A.T.M.A.N., acronimo di Better Approach To Mesh Ad-hoc Network [Rif. RFC], é un protocollo che vuole essere piú snello e piú leggero di OLSR. É stato pensato da Elektra, membro attivo interno a Freifunk, e realizzato in concomitanza con Marek Liner. Tale protocollo si basa sull'invio a intervalli regolari di pacchetti Originator Message (OM) molto piccoli di broadcast su tutta la rete in perfetto stile *flooding*. Viene effettuato un flooding selettivo solo nel caso di link asimmetrici, sui quali non viene inoltrato il traffico di controllo. Su ogni nodo é presente una tabella *ranking table* sulla quale vengono prese le decisioni relative al routing. Ogni entry nella ranking table corrisponde ad un vicino e serve a memorizzare il numero di OM relativi ad altri nodi della rete ricevuti da quel vicino. La entry relativa al vicino da cui si é ricevuto il maggior numero di OM relativi alla destinazione che si vuole raggiungere, sará scelto come next hop.

1.4.3 AODV

Ad-hoc On-demand Distance Vector é un protocollo reattivo di tipo Distance Vector. Le rotte vengono costruite "on demand" quando c'é bisogno di instradare del traffico. Si basa sull'invio di tre tipi di pacchetti quali:

- *Route request* (RREQ) é una richiesta inviata dal nodo sorgente in broadcast sulla rete per richiedere chi conosce il nodo cui deve inviare un pacchetto;
- *Route reply* (RREP) é una risposta alla richiesta RREQ;
- *Route error* (RERR) é un messaggio che serve ad indicare che un nodo non é piú raggiungibile.

La scelta del percorso migliore viene effettuata in base al minor numero di hop. Il principale vantaggio di AODV é quello di non generare ulteriore traffico di controllo per quei link che sono stabili nel tempo, ma presenta tutti gli svantaggi propri dei protocolli reattivi.

Capitolo 2

Metriche

2.1 Introduzione alle metriche

In questo capitolo verranno trattate le metriche utilizzate dai principali protocolli di routing mesh per instradare il traffico attraverso i nodi della rete al fine di avanzare poi delle proposte alternative di utilizzo, frutto dei mesi di studio e ricerca. Prima di tutto é bene capire che cos'é una metrica e a che cosa serve. La metrica é un algoritmo parametrabile finalizzato a misurare l'efficienza di una connessione analizzando quantitativamente il traffico scambiato correttamente.

2.1.1 Concetto di metrica e di protocollo

Una metrica é parte integrante del protocollo di routing ma é allo stesso tempo un concetto a sé stante. Un protocollo di routing prende le sue decisioni basandosi sui risultati calcolati dalla metrica su cui poggia.
Il compito del protocollo di routing é quello di calcolare, tra i vari percorsi che uniscono due nodi il migliore instradamento possibile dei pacchetti. D'altro canto una metrica deve dare al protocollo una valutazione sulla bontá di ogni

link della rete per permettergli di prendere le decisioni ottimali. Si potrebbe perció dire che il protocollo di routing si basa su un'informazione distribuita, calcolata localmente (su ogni singolo link) dalla metrica. É evidente che il concetto di metrica é di fondamentale importanza in quanto influenza direttamente il protocollo di routing.

2.1.2 Approcci alle metriche su mesh

Le metriche pensate per le reti cablate se applicate ad una rete mesh potrebbero portare il protocollo alla scelta di path non ottimali tra tutti quelli possibili tra due differenti nodi. Esiste una differenzza sostanziale tra una metrica pensata per una rete cablata ed una pensata per un rete mesh. Tale differenza sta nella percezione dello stato di un link.
Sul cavo non si verificano normalmente casi in cui la trasmissione funzioni solo in un verso tanto che si puó dire che lo stato di un link risponde ad una logica binaria (c'é o non c'é). Tale approccio per le reti mesh wireless non é corretto poiché un link wireless puó funzionare, puó funzionare male, puó essere soggetto a fluttazioni o puó non funzionare affatto; é piú correttopertanto parlare di logica 'fuzzy'. Inoltre un link wireless puó risultare asimmetrico, ossia percorribile prevalentemente in un solo senso.
Altra differenza sostanziale é che in una rete cablata non é previsto che i nodi si muovano mentre una rete mesh nasce anche per risolvere l'esigenza di mobilitá.

I problemi principali nella costruzione di un protocollo mesh wireless é proprio l'implementazione di un meccanismo per la percezione del grado di simmetria di un link e la gestione della mobilitá dei nodi all'interno della rete stessa. Quest'ultimo problema dipende dalla reattivitá del protocollo ai cambiamen-

ti della topologia. Piú un protocollo é reattivo migliore sará la percezione che ha dello stato della rete, anche se talvolta a discapito della banda effettiva a disposizione sui singoli link. Infatti la maggior reattivitá del protocollo é di solito direttamente proporzionale al flooding sulla rete. Ci sono alcune implementazioni orientate a risolvere il problema della mobilitá basate su OLSR tra cui menzioniamo (rif FastHELLO). OLSR stesso permette, agendo sul file configurazione, di rendere piú reattiva la percezione della topologia della rete. C'é da dire che in una mesh, costituita da nodi sono relativamete statici, non é necessaria un'alta reattivitá del protocolo e si preferisce generare meno traffico di controllo a discapito della reattivitá ed a favore della banda lasciata a disposizione degli utenti della rete.
Le metriche, che verranno illustrate di seguito, sono pensate essenzialmente per reti mesh wireless e utilizzano metodi e approcci differenti, ma sono tutte basate sull'invio e la ricezione di pacchetti di broadcast.

L'invio di un pacchetto di broadcast é fondamentale in una rete mesh poiché é l'unico modo per far conoscere ai propri vicini la propria esistenza. Per vicini si intendono quei nodi che si trovano entro la portata radio e contemporaneamente alla distanza di un solo HOP. L'utilizzo di tale tipo di pacchetto peró implica una serie di problemi. I potocolli di routing mesh effettuano i probe basandosi sull'invio e la ricezione di pacchetti di broadcast per poi utilizzare la rete per traffico generalmente unicast. Basarsi unicamente sui pacchetti di broadcast é peró un limite poiché vengono forgiati ai basic rate (1,2,6 Mb/sec) e con modulazioni perlopiú DSSS (*Direct-Dequence Spread Spectrum*) meno soggette all' interferenza intersimbolica, mentre i pacchetti di unicast sono forgiati a rate variabili e generalmente piú alti con modulazioni piú sensibili all'interferenza intersimbolica.

Sono stati fatti degli studi per conto dell'INRIA i cui risultati mostrano come i probe fatti in broadcast in alcune situazioni non sono adeguati a permettere al protocollo di routing di instradare quelli di unicast. Il fenomeno di cui stiamo parlando viene comunemente chiamato Gray Zone [Rif Gray Zone], mostrato in figura 2.1 che si presenta in maniera marcata quando i nodi sono in movimento. La zona grigia si verifica quando un nodo muovendosi percepisce solo il traffico broadcast ma non quello unicast poiché il driver della scheda di rete non ha avuto il tempo di riadattare il rate abbassandolo.

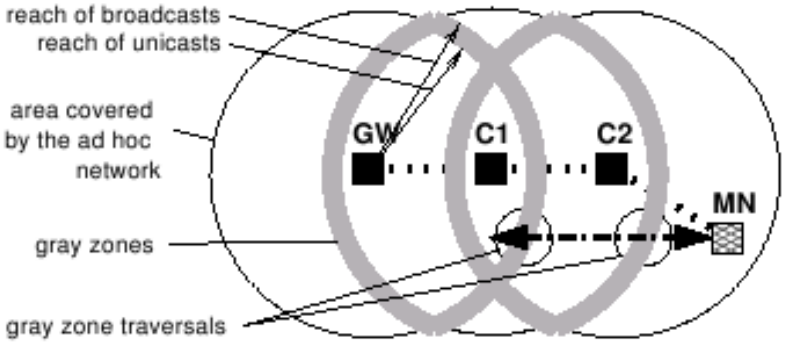


Figura 2.1: Gray Zone.

La Gray Zone é causata sostanzialmente da 5 fattori :

- i pacchetti di broadcast non prevedono meccanismi di acknoledgemnet e quindi di ritrasmissione dei pacchetti, cosa che invece avviene per i pacchetti di unicast;
- il movimento del nodo;
- il broadcast viene fatto al basic rate, quindi arriva piú lontano e la modulazione dei pacchetti é meno soggetta al rumore;

- i pacchetti di controllo dei protocoli di routing mesh sono relativamente piccoli;
- la fluttuazione del link nelle zone al limite della portata radio. Tale fluttuazione consente la percezione di messaggi di controllo broadcast nelle zone al limite della portata radio che portano i router a scegliere delle rotte piú brevi sulle quali non si riesce peró a far passare il traffico di unicast.

Se i nodi di una rete sono tutti fermi o si muovono relativamente poco, la Gray Zone puó essere rilevata osservando che il driver abbassa il rate con cui forgia i pacchetti di unicast secondo un meccanismo ben preciso finché non riceve gli acknoledgement del detinatario. C'é da dire che a pacchetti di unicast, forgiati con rate differente, corrispondono spesso modulazioni differenti. Man mano che si alza il rate, aumentano i livelli di informazione che vengono inviati con un unico pacchetto e quindi aumenta la sensibilitá all'interferenza intersimbolica. I protocolli di routing mesh, basando la percezione dei link sulla ricezione di pacchetti di broadcast, potrebbero incorrere in spiacevoli situazioni di Gray Zone. Una soluzione potrebbe essere data dall'integrazione di probe unicast tra i nodi ai capi di uno stesso link. Dai dati restituiti dal tool frutto di questa tesi si é in grado di rilevare se si é verificata la Gray Zone o meno.

La rilevazione della simmetria del canale é forse il piú grosso dei problemi che devono essere affrontati nella fase di progettazione di un protocollo di routing mesh. Prende il nome di *Link Sensing* ed ogni protocollo adotta una tecnica differente basata unicamente su pacchetti di broadcast. Nel caso di mobilitá dei nodi si potrebbe presentare il problema della Gray Zone risolvibile peró effettuando dei probe anche con pacchetti di unicast. Nel caso

di una rete relativamente statica, come quella cittadina, si potrebbe pensare ad un meccanismo che testa i link con pacchetti di unicast a differenti rate memorizzandone i risultati (effetuando delle statistiche basandosi sugli ack, sulle ritrasmissioni e sul rate nominale con cui sono stati forgiati), al fine di utilizzare il canale al meglio possibile diminuendo cosí il numero di ritrasmissioni e quindi rumore inutile sui canali stessi.

Per potersi avvalere di informazioni quali il numero di retry, il rate con cui sono forgiati i pacchetti e il livello del segnale (RSSI), bisogna poter comunicare con i livelli piú bassi della pila ISO OSI, quindi con i driver della scheda di rete. L'ostacolo piú grande nella comunicazione con i driver é che non ne esiste uno universale ma ogni chipset ne ha uno generalmente diverso dagli altri. Si potrebbe ovviare a questo problema implementando un'interfaccia standard per i driver delle diverse schede di rete, ma questo esula dallo scopo di questo lavoro. In questa tesi e nell'implementazione del tool é stato preso in considerazione il driver delle schede con chipset ATHEROS, BROADCOM e PRISM poiché si tratta di schede molto diffuse nelle reti mesh e montate su apparati quali il Linksys WRT.

2.2 Metriche su Mesh

In una rete mesh tutti o quasi tutti i nodi partecipano attivamente al routing. Il problema della costruzione di una metrica ottimale per una rete wireless Mesh é tuttora aperto. Sono stati fatti molti studi a tale scopo, alcuni dei quali solo a livello teorico, altri ancora in itinere e sono state individuate diverse soluzioni, ma nessuna di queste finora sembra risolvere appieno i problemi esposti sinora. Una metrica pensata per una rete mesh é strettamente

legata al progetto su cui si basa il protocollo di routing del quale sará parte integrante. Un esempio é B.A.T.M.A.N. in cui é veramente difficile trovare un confine netto tra i meccanismi di Neighbor Discovery, Link Sensing e i probe sui link. OLSR d'altro canto é un protocollo modulare che permette la scelta tra piú metriche. Nativamente implementa una semplice metrica basata su HOP COUNT e ISTERESI, ma é comunque possibile utilizzare in alternativa la metrica ETX. Di questo parleremo piú avanti.
Coma giá detto, normalmente i probe vengono effettuati al livello di rete, memorizzando l'avvenuta ricezione o meno dei pacchetti di probe broadcast. Di seguito vengono esposte alcune tra le metriche piú diffuse in ambiente mesh.

2.2.1 Hop Count

Hop Count é una metrica che calcola il cammino minimo tra tutti i cammini possibili tra due nodi secondo l'algoritmo di Dijkstra con peso pari al numero di hop ed é forse il primo tentativo di trasferire una metrica nata per reti cablate sulle reti mesh. Vediamo con un esempio come un semplice meccanismo di hop count possa ingannare un protocollo di routing mesh wireless. Immaginiamo di avere una rete mesh completamente magliata costituita da tre nodi come in figura 2.2 : A, B e C. Il link tra A e B é simmetrico ed ha un grado di bontá pari al 40% dei pacchetti inviati (si perde il 60% dei pacchetti inviati). I link tra B e C e tra A e C sono invece perfetti. Il nodo A per raggiungere il nodo B, secondo il meccanismo Hop Count, sceglie il link diretto invece di preferire il path piú lungo, passante per C, ma migliore. Come conseguenza il traffico tra A e B sará fortemente penalizzato dalla perdita di pacchetti.

Come mostrato nell'esempio non é la migliore metrica per le reti mesh poiché puó portare il protocollo ad instradare i pacchetti sul path non ottimo. Sarebbe una buona metrica per una rete mesh se i link wireless fossero tutti simmetrici e funzionanti alla stesso modo e della stessa tecnologia. Nella realtá i link wireless subiscono diverse oscillazioni e andrebbero penalizzati i link che oscillano di piú e preferiti i link cotruiti con tecnologie piú veloci (ex. 802.11 5Ghz) e piú stabili.

2.2.2 Hop Count + Isteresi

É la metrica nativa di OLSR secondo l'RFC implementata da Andreas Tonnensen. Fondamentalmente é un meccanismo per rimuovere dalle MIB dei nodi i link che oscillano continuamente. É un adattamento dell'HOP COUNT alla realtá delle reti wireless. Fornisce un dato binario sulla bontá del link del tipo funzionante/non funzionante. Quindi se secondo l'isteresi un link

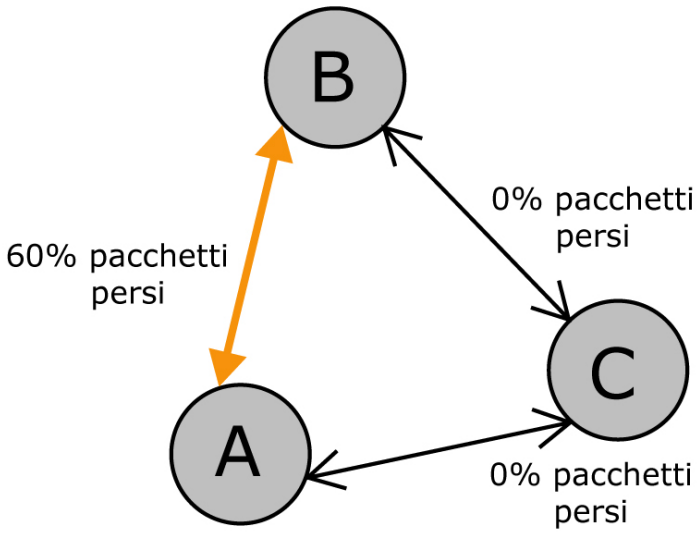


Figura 2.2: Esempio di instradamento secondo Hop Count.

c'é, OLSR applicherá l'algoritmo di Hop Count per popolare le tabelle di routing su tutti i nodi. Si basa sulla ricezione di pacchetti di broadcast e sull'utilizzo di due soglie e di un valore di scala. Ogni nodo mantiene in una tabella una entry per ogni link con i vicini che ne indica la bontá. Quando tale valore scende al di sotto della soglia minima, il link viene considerato non funzionante, mentre quando sale al di sopra della soglia massima viene considerato funzionante. Piú le due soglie sono vicine e piú il routing é instabile se nella rete ci sono molte oscillazioni, ma piú adattabile a rapidi cambiamenti della topologia. Si utilizzano due formule per calcolare il valore di isteresi di un link, una applicata in salita, quando cioé un messaggio di HELLO viene ricevuto con successo:

LinkQuality=(1-scaling)*LinkQuality+scaling

ed una in discesa, quando un messaggio di HELLO viene perso:

LinkQuality=(1-scaling)*LinkQuality

I messaggi di HELLO sono pacchetti di broadcast utilizzati da OLSR per effettuare il probing sui link e, allo stesso tempo, per implementare il meccanismo di *Neighbor Discovery*. Vengono generati ad intervalli regolari consentendo ai nodi di determinare se sono stati persi durante la trasmissione. Nell'ipotesi della figura ... ISTERESI avrebbe potuto considerare il link diretto tra A e B come non funzionante, instradando il traffico attraverso il path migliore passante per C. Non viene ancora risolta la rilevazione dei link costituiti da tecnologie piú veloci. Inoltre se ci sono due strade per collegare due nodi della rete che secondo ISTERESI sono funzionanti, verrá ancora

preferita quella con il minor numenro di hop e non la piú veloce. Con questo meccanismo di valutazione dei link wireless non ci si avvale di informazioni a livello fisico del canale.
Ipotizziamo che i link A-B e B-C siano realizzati con tecnologia 802.11a a 5Ghz, mentre il link A-C con tecnologia a 2.4Ghz (figura 2.3). Viene sempre scelto il link diretto.

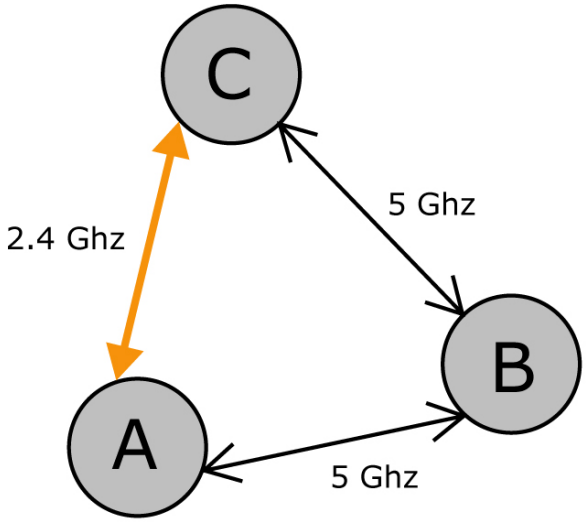


Figura 2.3: Esempio di instradamento secondo Hop Count e ISTERESI.

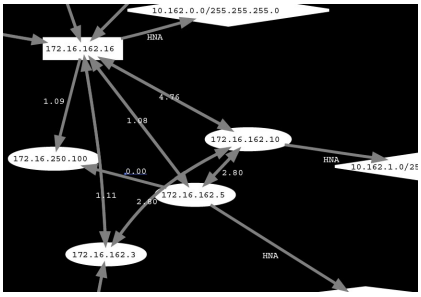
Il traffico che va da A a C attraversa il link diretto. Si puó osservare peró che se il traffico che genera A diretto verso C passasse per B, si avrebbe un incremento di prestazioni e velocitá.

2.2.3 ETX

ETX é una metrica in grado di dare effettivamente un valore “fuzzy” alla bontá del canale. Si basa sulla percezione del grado di simmetria dei due

nodi ai capi di un link e calcola un numero che indica la bontá del link percepita in entrambi i sensi. ETX é stato implementato per OLSR ad opera della comunitá wireless Freifunk di Berlino. Si basa su una semplice formula :

$$ETX = \frac{1}{LQ \times NQL}$$



dove LQ (Link Quality) é la percezione che ha il nodo A del link da A a B. Tale misura viene calcolata in base alla percentuale di pacchetti di controllo HELLO che A ha ricevuto da B; NLQ (Neighbor Link Quality) é la percentuale dei pacchetti HELLO che B ha ricevuto da A (questo dato viene inviato ad A da B stesso tramite i messaggi di controllo TC topology control). Tali valori vengono normalizzati ad 1; a 0 corrisponde il 100% dei paccheti pesri e ad 1 il 100% dei pacchetti ricevuti. Se per esempio vogliamo calcolare l'ETX su A con LQ=0.7 e NLQ=0.3 avremo:

ETX= 1/(0.7*0.3) = 4,76

Per calcolare i cammini minimi su una rete mesh, in tal caso OLSR utilizza Dijkstra con peso sugli archi dato dall'ETX e non piú il numero di hop minimo. Il cammino minimo tra due nodi sulla rete sará quello con la somma degli ETX minore. A paritá di ETX verrá scelto il cammino con il minor numero di salti secondo HOP COUNT. É possibile modificare la sensibilitá con cui varia ETX agendo sulla variabile LINKQUALITYMULT che si mol-

tiplica al denominatore della formula [RIF].

Puó capitare di avere due differenti valori di ETX sullo stesso link a seconda del verso di percorrenza, dovuti al mancato allineamento della percezione della simmetria tra i due nodi ai capi del link stesso.

ETX é effettivamente una metrica che tiene conto della bontá di un link in termini di percentuale di perdita di pacchetti di controllo (HELLO). Basarsi solamente su ETX per calcolare l'instradamento dei pacchetti sulla rete risulta peró ancora riduttivo poiché si basa sui soli pacchetti di broadcast e non tiene conto della tecnologia con cui é realizzato il link (per ETX un link a 2.4 Ghz equivale ad un link a 5Ghz). Inoltre possono ancora verificarsi situazioni in cui viene preferito un path peggiore a discapito di un path migliore ma con numero di salti maggiore, come mostrato in figura 2.4.

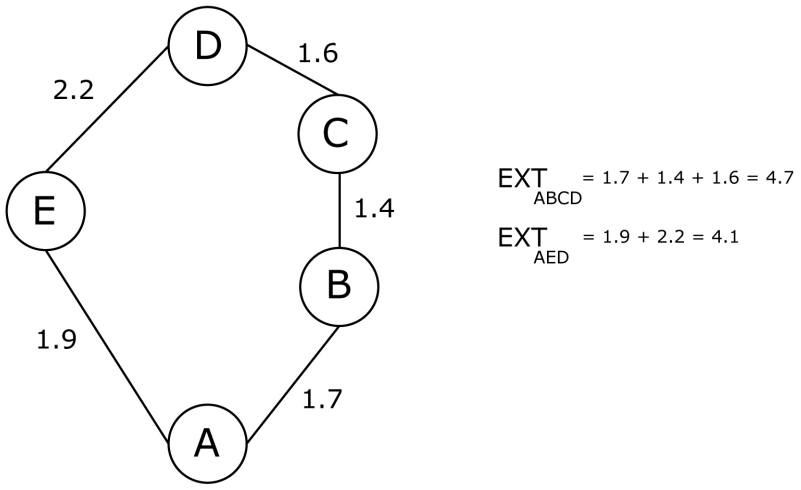
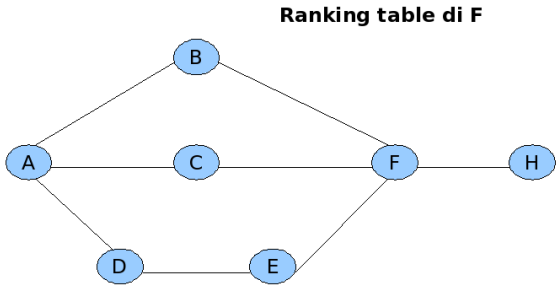


Figura 2.4: Qualità dei path secondo ETX.

Il cammino A-B-C-D é costituito da link con qualitá migliore rispetto al path A-E-D. Infatti L'ETX medio del path A-B-C-D é pari a (4.7/4) = 1.175, mentre l'ETX medio del path A-E-D é pari a (4.1/2) = 2.05. Se A utilizzasse il path piú lungo, si perderebbero meno pacchetti

2.2.4 Ranking

Questa metrica é propria del protocollo di routing mesh B.A.T.M.A.N.. É basata sul conteggio dei pacchetti di controllo broadcast ORIGINATOR MESSAGE ricevuti. Ogni nodo salva in una tabella di RANKING, per ogni vicino il numero di pacchetti di controllo Originator Messagge ricevuti (figura 2.5).



Ranking Table

	Neighbor B	Neighbor H	Neighbor C	Neighbor E
A	12	0	11	10
B	14	0	9	7
C	7	0	10	5
D	11	0	10	12
E	8	0	9	11
H	0	12	0	0

Routing Table di F

dest	gateway
A	B
B	d.c.
C	d.c.
D	E
E	d.c.
H	d.c.

Figura 2.5: Ranking table e tabella di routing sul nodo F.

Anche in questo caso non viene considerato il livello fisico del canale e i probe vengono fatti esclusivamente in broadcast. B.A.T.M.A.N. é tuttora

in fase di sviluppo ed esiste una branche che sta studiando un modo per prendere i dati dai livelli 1 e 2 della pila ISO/OSI.

2.2.5 Conclusioni

Le metriche viste utilizzano tutte pacchetti di broadcast molto piccoli per effettuare i probe e non tengono conto dei livelli fisici del canale. Nei test effettuati é capitato che i protocolli di routing consideravano buono un link su cui non era peró possibile effettuare trasmissioni unicast o che considerasse mediocre un link su cui la trasmissione unicast era possibile grazie al meccanismo delle ritrasmissioni unito all'adattamento del rate. Si é pensato allora a come potesse essere costituita una buona metrica per una rete mesh, che tenesse conto allo stesso tempo dell'oscillazione dei link, della Gray Zone, del traffico unicast sostanzialmente diverso da quello di broadcast e una memoria dell'andamento del link nel tempo. Si é arrivati a piú proposte, ma quella che sembra essere la migliore é la costruzione di una metrica del tutto modificabile a seconda delle esigenze, che permettesse di effettuare probe del canale in tutte le situazioni. Tale metrica non puó non tenere in considerazione il livello fisico del canale e le trasmissioni unicast, oltre ai probe effettuati in broadcast. Nella sezione seguente verranno esposte alcune proposte che rimangono comunque del tutto teoriche.

2.3 Metriche proposte

Il problema comune a tutti i protocolli di routing mesh mostrati é che il probing del canale viene effettuato in broadcast, unicamente a livello di rete e senza mantenere una memoria dell'andamento temporale del link.
Il probing effettuato esclusivamente con pacchetti di broadcast puó essere

riduttivo poiché, come giá detto, tali pacchetti vengono forgiati al rate piú basso possibile (meno soggetto all'interferenza intersimbolica) e non sono previste ritrasmissioni. Durante i test effettuati si sono verificati imoltre casi in cui su un link, considerato daii protocolli di routing mediocre o addirittura inesistente, il tool rilevasse che i pacchetti di unicast raggiungevano comunque la destinazione [Rif test fatto dal tool]. Perché allora effettuare i probe sulla rete con pacchetti fortemente differenti da quelli che poi verranno maggiormente utilizzati nella comunicazione vera e propria? E perché non usare anche i link dove é possibile il traffico unicast anche se quello broadcast non é eccellente? E su una rete i cui nodi sono statici, perché non mantenere una memoria dell'andamento dei link che influisca sul giudizio della bontá del link nel tempo? Con questo lavoro si é cercato di trovare delle risposte a tali domande e si sono avanzate delle proposte illustrate di seguito. Hanno tutte in comune l'esigenza di spingersi al livello fisico e ad utilizzare i pacchetti di unicast in aggiunta a quelli di broadcast. Si potrebbe pensare di agire in diversi modi sfruttando i pacchetti di unicast insieme a quelli di broadcast per i probe dei link o utilizzando eclusivamente l'unicast per i probe e il broadcast per il *Neighbour Discovery*.

Procederemo per affinamenti successivi per poi presentare una metrica piú completa basata su broadcast, unicast e memoria dell'andamento del link. Nelle proposte seguenti per il *Neighbour Discovery* vengono utilizati sempre pacchetti di broadcast.

2.3.1 Conoscenza indiretta del link - livello di rete

Ogni nodo, una volta individuato un nuovo vicino, inizia una serie di probe del canale inviando pacchetti di unicast. Si prevedono due variabili BROADINTERVAL e UNIINTERVAL che indicano la cadenza temporale in secondi con cui inviare pacchetti di broadcast e unicast ripettivamente.
Inizialmente un nodo invia solo pacchetti di broadcast, uno ogni BROADINTERVAL secondi; i vicini che li ricevono memorizzano in una variabile LASTAWARE_NODO_X il momento in cui hanno ricevuto l'ultimo pacchetto di broadcast dal nodo X e l'IP del nodo X stesso; iniziano quindi a mandargli una serie di pacchetti di unicast. La variabile LASTAWARE_NODO_X serve a mantenere memoria dell'ultimo istante in cui si é sentito qualcosa dal vicino X e nel caso, ad eliminare quei nodi di cui non si é sentito piú nulla da diverso tempo. Procediamo con un esempio.

Siano dati due nodi A e B. A invia un pacchetto di broadcast per primo e B, sentendolo, inizia ad inviare una serie di pacchetti di unicast. In questo modo A ha una percezione di come funziona il link in unicast da B ad A mentre B ha una percezione di come funziona il link in broadcast da A verso B. Contemporaneamente B ha generato pacchetti in broadcast che A potrebbe aver recepito; nel qual caso A risponde con una sequenza di unicast verso B.
Ora A sa come va il link da B ad A sia in broadcast che in unicast e B analogamente sa come va il link nel verso contrario. Quindi entrambi i nodi conoscono la bontá del link nel senso opposto a come lo dovrebbero utilizzare. Per ovviare questo problema si potrebbe implementare uno scambio tra i due vicini di ció che conoscono l'uno dell'altro per costruire una metrica stile ETX basata sul LQ e NLQ basata sia su unicast che su broadcast :

$$\alpha * \text{ETXb} + \beta * \text{ETXu} = \alpha * [1/(\text{LQb}*\text{NLQb})] + \beta * [1/(\text{LQu}*\text{NLQu})]$$

dove α e β sono due numeri tra 0 e 1 che hanno la funzione di regolare il peso che si vuole dare all'ETXb di broadcast e all'ETXu di unicast. Impostati entrambi a 1 darebbero lo stesso peso agli ETXb, ETXu.
Con questo tipo di metrica si tenderebbe a valorizzare anche i probe effettuati in unicast, ma i test effettuati dal TOOL hanno dimostrato essere pressoché inutili al solo livello 3, poiché non si ha un riscontro sul numero delle ritrasmissioni e nessuna informazione sui rate utilizzati. Un link non si perde nemmeno un pacchetto di unicast al livello 3, potrebbe essere caratterizzato da un alto numero di retry e un data rate medio basso.

2.3.2 Conoscenza indiretta del link - livello Data Link

In questo caso si puó procedere analogamente al caso precedente, scendendo peró sino al livello fisico e tenendo conto del numero di ritrasmissioni che si sono verificate nella ricezione dei pacchetti di unicast. A questa informazione si puó aggiungere il rate con cui sono stati ricevuti i pacchetti di unicast per calcolare l'andamento statistico del link in termini di velocitá. In tal modo, due link che a livello 3 perdono lo stesso numero di pacchetti, possono essere distinti in base al numero di retry e il data rate medio.

2.3.3 Conoscenza diretta del link

Un altro tipo di approccio é quello di utilizzare i pacchetti di unicast sfruttandone gli *acknoledgement* per ricavare un'informazione diretta sulla bontá del link. Si tratta di verificare quanti riscontri sono pervenuti ai pacchetti unicast inviati avendo cosí una percezione del link nel verso stesso in cui si

utilizzerá il link. Se un nodo riceve un riscontro ad un pacchetto di unicast significa che quel pacchetto é arrivato a destinazione. Si prevedono sempre due variabili BROADINTERVAL e UNIINTERVAL.
In questo caso é necessario scendere al livello fisico per verificare il numero di acknoledgement e fare delle statistiche sul numero di pacchetti inviati e il numero di riscontri ricevuti. Un link, in cui vengono riscontrati tutti i pacchetti di unicast, dovrebbe essere considerato buono. Peró non tenendo conto del Data Rate medio si potrebbe considerare migliore un link meno veloce. Aggiungendo quindi anche tale informazione si é effettivamente in grado di valutare l'effettiva bontá del link.

2.3.4 Invio di sequenza di pacchetti unicast di dimensione differente

Durante le fasi di test effettuate per simulare un traffico di unicast con pacchetti di dimensione variabile, si sono riscontrate delle anomalie; si é risontrata infatti una maggiore perdita di pacchetti rispetto ai test effettuati con sequenze di pacchetti della stessa dimensione. Effettuare dei test con pacchetti di dimensione diversa ha il significato di simulare l'effettivo utilizzo della rete a livello utente.
Nel caso delle metriche esposte sopra, si potrebbe quindi prevedere l'invio di una sequenza prestabilita di pacchetti di unicast di dimensione differente.
Si potrebbero prevedere inoltre due tabelle di routing su ogni nodo, differenziate dal tipo di traffico, una per esempio per il traffico VoIP e l'altra per il traffico dati. La MIB per il traffico VOIP sará popolata con i link che hanno dimostrato di perdere meno pacchetti di piccole dimensioni. Al contrario la MIB relativa al traffico dati sará popolata con i link che hanno dimostrato di perdere meno pacchetti di dimensione grande.

2.3.5 Una metrica poliforme

Adesso vediamo una proposta di metrica che tiene conto contemporaneamente dei probe di unicast, del livello fisico e che mantiene una memoria sull'andamento del link nel tempo, cosa che nei casi precedenti non é stata contemplata.

Mettiamoci nelle ipotesi eposte per la "Conoscenza indiretta del link - livello Data Link". La metrica ora proposta si basa sul fatto che se un nodo ha ricevuto un certo numero di pacchetti ritrasmessi allora sono stati persi dei pachetti di acknoledgement nel verso opposto. I probe sui link sono divisi in finestre della lunghezza FINESTRA, atte a mantenere una memoria dell'andamento dei probe su ogni singolo link. Durante ogni fase di probe ogni nodo memorizza per ogni suo vicino il numero di pacchetti ricevuti ed il numero di retry per ogni pacchetto.
Tale metrica deriva strettamente dal TOOL sviluppato per questa tesi. Su ogni nodo sono presenti tre buffer di dimensione FINESTRA per ogni vicino di cui si é sentito almeno un pacchetto di broadcast; sul primo buffer UBUFFER viene messo un 1 se viene ricevuto un pacchetto di unicast, 0 altrimenti; sul secondo URETRY si contano il numero di ritrasmissioni ricevute per un singolo pacchetto; sul terzo URATE viene memorizzato il rate di ogni pacchetto ricevuto. Si ricorda che viene inviato un pacchetto unicast ogni UNIINTERVAL secondi.
Avendo a disposizione dati quali retry e data rate si puó procedere alla valutazione della bontá del link nei due modi illustrati nella figura. Si tratta solo di due possibili modi di organizzare e valutare i dati di cui si é a conoscenza (figura 2.6).

Si puó notare che l'albero nella figura A si puó ricavare dall'albero in figura B impostando δ =0. Se si segue l'approccio di figura A, si tende a penalizzare i link che a paritá di pacchetti ricevuti hanno un numero di ritrasmissioni piú elevato e un rate medio inferiore. Tra due link che perdono lo stesso numero di pacchetti di unicast il migliore sará quello che ha collezionato il minor numero di retry con un rate medio piú alto.
Se si segue l'approccio di figura B, si tende a non penalizzare eccessivamente quei link sui quali si é verificata una rara perdita di pacchetti durante le fasi di probe. Puó infatti capitare che anche su un link buono si verifichi qualche perdita di pacchetti, ma ció non vuol dire che il link é peggiore di un altro che non ha perso nemmeno un pacchetto ma presenta un alto numero di retry e un data rate medio molto basso.
In entrambi i casi sono previsti dei pesi sugli archi regolabili a piacimento dell'utente. Dai test effettuati si é visto che di solito le trasmissioni con alto numero di retry erano caratterizzate da un rate medio basso (prossimo al Basic Rate). In altre situazioni, ben piú rare, si é registrato un rate medio prossimo a quello massimo. Per questo motivo forse la giusta misura

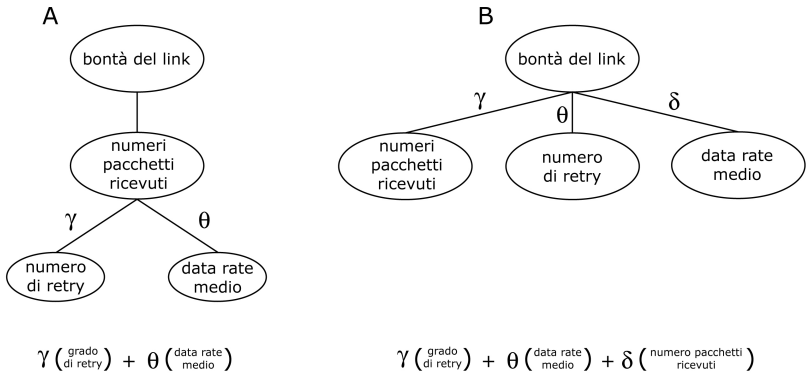


Figura 2.6: Alberi di riferimento per le metriche.

é impostare in entrambi i casi $\theta = \delta$, dando cosí lo stesso peso al numero di retry e al data rate medio.
Prendiamo ora in esame un link tra i nodi A e B con FINESTRA = 5 e UNIINTERVAL = 1 osservando il punto di vista di A rispetto a B.

1. B invia un pacchetto in broadcast.

2. A ha sentito il pacchetto in broadcast di B e inizia a mandare una serie di 5 (FINESTRA/UNIINTERVAL) paccheti di unicast verso B numerati con SQN incrementale da 0 a 4. SQN é un valore che viene inserito nel pacchetto che indica se il pacchetto inviato é il primo, il secondo, ...In ogni ciclo di probe verranno inviati FINESTRA/UNIINTERVAL pacchetti unicast quindi SQN andrá da 0 a (FINESTRA/UNIINTERVAL-1).

3. Non appena B sente per la prima volta un pacchetto di unicast inviato da A inizializza i tre buffer riempiendoli di 0; B legge l'SQN del pacchetto ricevuto al livello di rete e ne analizza i dati a livello fisico. Metterá un 1 sull'UBUFFER in posizione SQN (UBUFFER[SQN] = 1), aggiungerá un 1 sull' URETRY sempre in posizione SQN (URETRY[SQN]) se si tratta di un pacchetto ritrasmesso e memorizzerá il rate con cui ha ricevuto il pacchetto sul buffer URATE sempre in posizione SQN (URATE[SQN] = 11). Una possibili situazione dei buffer potrebbe essere la seguente :

last_aware : TIMESTAMP
UBUFFER : 0 1 1 0 1
URETRY : 0 3 0 0 2
URATE : 0 1 11 0 11

4. Un ciclo di probe finisce quando B riceve l'ultimo pacchetto della sequenza (SQN = FINESTRA/UNIINTERVAL), oppure quando B riceve un nuovo pacchetto con SQN minore rispetto all'ultimo memorizzato. In tal

caso B ha iniziato a ricevere una nuova sequenza di probe unicast da A.

5. A questo punto B puó procedere alla valutazione della bontá del link, LQ, da B ad A seguendo una delle due strade mostrate in figura. La qualitá del link da B ad A, NLQ, verrá spedita a B da A stesso. Indichiamo con UNIPROBE la metrica risultante e applichiamo l'approccio utilizzato da ETX sostituendo nella sua formula a LQ e NLQ quelli calcolati con il metodo di cui sopra. Aggiungiamo ora della memoria alla nostra metrica UNIPROBE e chiamiamola UNIMEM.
Se UNIPROBE é maggiore di UNIMEM, miglioriamo la metrica secondo un fattore di scala α :

UNIMEM = UNIPROBE + (1 - α) * UNIMEM

se invece UNIPROBE é minore di UNIMEM, peggioriamo la metrica :

UNIMEM = UNIPROBE - (1 - α) * UNIMEM

2.3.6 Conclusioni

Il problema delle metriche proposte in questa sezione é il potenziale crescente overhead che si verrebbe a generare all'aumentare del numero di nodi e della magliatura della rete. Le reti mesh, studio di questa tesi, sono peró mesh cittadine i cui nodi non sono in movimento, o se lo sono, si muovono relativamente poco. In questo contesto un'alta reattivitá del protocollo non

é necessaria. Si puó quindi impostare UNIINTERVAL ad un valore relativamente alto per limitare l'overhead sulla rete (ex. UNIINTERVAL = 5 secondi). D'altrocanto il BROADINTERVAL indica la cadenza con cui far conoscere ai propri vicini la propria esistenza. Data la bassa mobilitá dei nodi anche questo valore puó essere relativamente alto.

Capitolo 3

Caratterizzazioni dei link wireless

I collegamenti senza fili, come é stato più volte affermato, hanno comportamento diverso da quello dai link cablati. I protocolli attualmente più diffusi (noti con gli acronimi OLSR e B.A.T.M.A.N) utilizzano pacchetti destinati al sondaggio dei collegamenti (detti pacchetti di *probe*) in modalità broadcast: per via dei caratteri propri dei messaggi broadcast, questo tipo di test potrebbe comportare una valutazione poco precisa dell'effettiva *performance* dei collegamenti.

Lo scopo di questa indagine é sostanzialmente di scoprire se, modificando i probe dei link, adottati in un certo protocollo per reti mesh, sia possibile costruire una metrica più fedele alla realtà di quanto il protocollo stesso attualmente non faccia.

Nel presente lavoro si sono volute categorizzare, quindi, le varie componenti dei collegamenti wireless, per tenerne in considerazione tutti i possibili

aspetti, anche quelli di dettaglio, spesso non ritenuti di rilevanza primaria. Il tool, dei cui particolari ci si occuperà più avanti, é stato costruito per generare probe di diversa natura e tipo (es. broadcast, unicast, di dimensioni variabili) ed analizzare così l'andamento delle connessioni, sotto tutti gli aspetti tipici di una rete mesh wireless.

Per fare ciò il tool, oltre a generare probe diversi da quelli implementati dai protocolli di routing più diffusi, é stato configurato in modo da essere in grado di catturare i dati relativi ai livelli 2 e 1 della pila ISO OSI, di immagazzinare le informazioni e renderle disponibili per un'analisi evoluta da parte dell'utente.

Le caratteristiche che il tool si propone di rilevare possono essere classificate in due differenti macrocategorie: *Stabilità* e *Velocità*.

Resta comunque da tenere presente che, una buona metrica, per un protocollo di routing, é quella che sceglie il giusto compromesso tra il percorso (*path*) più stabile e quello più veloce; ma questo potrebbe non essere sempre vero. Alcuni utilizzi della rete, presuppongono per esempio la scelta di un percorso stabile piuttosto che veloce; per altri, invece, può risultare più efficace l'uso di un path più reattivo anzicché stabile nel tempo.
L'utente, perciò sarà in grado di analizzare le rilevazioni effettuate dal tool sulla base di parametri da lui scelti e di valutare le caratteristiche della connessione sotto gli aspetti considerati di interesse e che hanno per indici i parametri stessi. Inoltre l'utente pu anche decidere con quali dati rappresentare ogni aspetto dei collegamenti e in che misura essi incidano sulla valutazione complessiva dei link.
Qusti link, in ogni rete, sono impegnati in maniera differente, a seconda della

tipologia di traffico. Nel progettare il tool si é deciso quindi di *testare* i link della rete con pacchetti di diverso tipo e dimensione e di sovraccaricarli in modo da *stressarli* in maniera casuale: cos facendo si é sicuri di raccogliere dati caratteristici in un ampio spettro di possibili situazioni.

Veniamo ora a guardare più da vicino qualche aspetto delle connessioni. Un collegamento può essere impegnato in entrambi i versi contemporaneamente, come ad esempio avviene quando si usano protocolli con connessione di *livello applicazione* e di *livello 4* della pila ISO OSI. In questo caso un link é continuamente impegnato in un verso per la comunicazione, ma anche implicitamente in verso opposto per via dei messaggi di riscontro. Nella comunicazione *con connessione* l'efficenza del verso del canale dedicato ai riscontri influenza significativamente l'intera *performance* dello scambio dell'informazine.

Per emulare tale impiego, tipico del protocollo TCP in cui si usano pacchetti di ACK di riscontro (*ACKnowledgement*), nel tool si forgeranno appositi probe di tipo unicast.

Si ricorda che per lo standard 802.11b/g, usato in tutti i test per questo lavoro, i pacchetti di broadcast sono forgiati con il *rate* più basso, specificatamente 1, 2 o 6 Mb/s , mentre gli unicast possono essere forgiati a *rate* più veloci, come ad esempio 2, 5.5, 11, 24, 48, 54Mb/s. I rate più alti utilizzano modulazioni RF (*RadioFrequenza*) più complesse che, se per un verso sono più efficenti quanto a velocità, dall'altro presuppongono un canale di trasmissione non soggetto ad interferenze.

Inoltre, inserendo maggiori informazioni sullo stesso canale: la velocità dello scambio di dati aumenta, a ragione della quantità di informazione trasmessa, ma nello stesso tempo, in presenza di un canale poco pulito, potrebbero verificarsi fenomeni di interferenza intersimbolica. che invalidano la coerenza

dei dati ricevuti.

D'altro canto, la modalità unicast, poiché ritrasmette più volte i pacchetti non giunti a destinazione, consente la comunicazione anche in condizioni impossibili per i pacchetti di broadcast non riscontrati.
Tenendo conto di questi aspetti, l'indagine effettuata dal tool utilizzando probe di broadcast, valuterà in che misura il comportamento dei protocolli di routing si discosti da quello richiesto dalla reale condizione dei collegamenti. Come si é rilevato nella sezione relativa al problema della Gray Zone[**?**], questo comportamento costituisce un fattore fondamentale nella valutazione delle caratteristiche di un link.
Il confronto tra il comportamento dei probe usati dal tool -realizzati con pacchetti unicast di varie dimensioni, oltre che con pacchetti broadcast- e quello dei pacchetti utilizzati dai protocolli di routing, può quindi mettere in luce ulteriori caratteristiche dei link ancora più dettagliate.
Infatti, un collegamento che funziona al 100% in broadcast, con pacchetti di piccole dimensioni, potrebbe non reagire con la stessa efficienza se stressato da pacchetti di dimesioni maggiori o con pacchetti unicast a rate più elevati.
Nella procedura implementata nel tool, ogni nodo, rispettando un ciclo temporale di test, spedisce pacchetti in broadcast a tutti i nodi vicini , cioé quelli che fisicamente riescono a ricevere i suoi pacchetti. Il nodo in questione spedisce una finestra di pacchetti di tipo broadcast con dimensione diversa, a seconda di uno schema (o pattern) e una tempistica scelti dall'utente. Quando l'altro nodo riceve questo tipo di pacchetti, crea una struttura per immagazzinare le statistiche sulle trasmissioni broadcast ricevute relative al nodo mittente e attiva una procedura di risposta che prevede l'utilizzo di pacchetti di unicast, sempre con direttive di dimesione e tempistica im-

poste dal pattern configurato dall'utente. Ciascun nodo é, quindi, in grado di collezionare, in ogni ciclo temporale, determinato dall'utente e chiamato *EPOCA*, i risultati di test di broadcast e di unicast relativi a tutti i nodi vicini.
Questa metodologia permette la stesura su ogni nodo di statistiche relative ad ogni link, con finestre temporali che impegnano la rete mesh non invasivamente ma consentendo, ugualmente, una valutazione rappresentativa di ogni *epoca* di test.
In questo modo sarà possibile impiegare il tool anche su reti in produzione, cioé utilizzate per inoltrare traffico utile, senza comprometterne l'uso.
L'approccio statistico alle informazioni raccolte, anche nel caso di test non continui, garantirà la coerenza dei dati relativi ad ogni epoca di test. Per garantire la raccolta dei dati in ogni situazione di carico della rete, l'avvio dei test di broacast, che segna l'avvio dei test tra coppie di nodi, é delegata ad una componente casuale (randomica) della procedura, dimodocché ogni ciclo di test venga effettuato probabilisticamente in condizioni diverse.
I casi che possono verificarsi di fatto sono i seguenti:

- i nodi all'estremità di un link iniziano contemporaneamente la fase di test; il link é così attraversato da sei flussi di dati contemporanei per ogni coppia di nodi in comunicazione, ed ogni verso ospita un pacchetto di broadcast, un pacchetto di unicast di risposta e un pacchetto di riscontro (dovuto all'unicast);

- i nodi iniziano la fase di test in periodi disgiunti; il link é impegnato da tre flussi per ogni coppia in comunicazione, un pacchetto di broadcast e un riscontro in un verso e uno di unicast nell'altro.

Sulla base dei riferimenti temporali, risulta chiaro in fase di analisi dei dati quanti nodi impegnano il canale nello stesso momento e come si é comportata la rete in ogni caso.
Il confronto dei test avviene in una macchina appositamente incaricata di raccogliere il risultato dei test di tutti i nodi; i riferimenti temporali di ogni ciclo di test di ogni nodo sono registrati coerentemente grazie al protocolo NTP (*Network Time Protocol*).
Ad ogni modo, il tool é uno strumento completamente configurabile che consente di testare i link, anche, in modo continuo ed invasivo, abbreviando i tempi e con una visione precisa della rete in ogni istante.
La metodologia utilizzata in questo approccio rileva, non soltanto la componente statistica delle qualità dei link, ma anche quella comportamentale cioé il modo in cui la rete reagisce ai test effettuati in condizioni differenti.
Si può osservare che, la visione di un protocollo di routing attuale non é in grado di valutare se la percentuale di pacchetti persi é distribuita uniformemente nel tempo, o se c'é stata un'oscillazione con caratteristiche peculiari.Nello svilupo del tool si é scelto invece di conservare le informazioni relative alla sequenza dei pacchetti spediti e ricevuti, indicandone un numero di sequenza e un numero di ciclo di test in modo da poter così avere informazioni anche sull'effettivo andamento temporale. Il tool elaborato durante questo studio é così capace di indicare con precisione, per ogni nodo, quale sia stata la distribuzione nel tempo dei pacchetti ricevuti da ogni vicino. Questo modo di collezionare le informazioni di routing aggiungono pertanto all'usuale percezione dei protocolli di routing una componente cronologica o, se si preferisce storica all'andamento dei collegamenti nella rete.

Nelle prossime sezioni si illustreranno i significati dei dati raccolti relativi alla valutazione delle caratteristiche dei collegamenti.

3.0.7 Stabilità

Nella figura 3.1 si illustrerà la semantica di ogni singolo dato catturato durante la fase di test relativo alle caratteristiche di Stabilià.

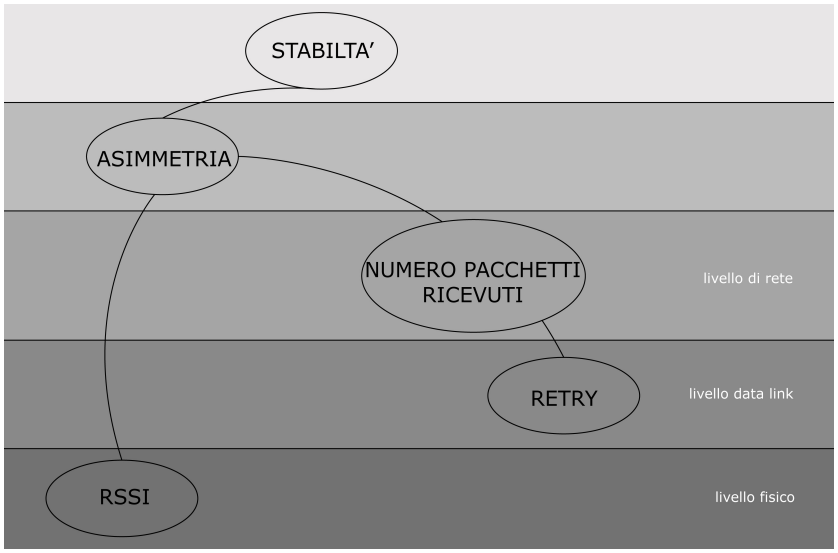


Figura 3.1: Albero della stabilitá.

Nella figura precedente, relativa alle caratterizzazioni degli aspetti di stabilità, si pone l'accento sul grado di specificità delle singole sottocategorie della stabilità. Nell'albero in questione si adotta la convenzione che ogni nodo figlio aggiunge un'informazione a più basso livello rispetto al padre. Come si é accennato, il grado di stabilità é caratteristica delle comunicazioni soggette a fluttuazioni come quelle basate su tecnologie senza fili. Una delle componenti maggiormente influenti sull'instabilità é la *simmetria* o la sua mancanza in un collegamento: in altri termini, un link che funziona con efficenza diversa nei due sensi della comunicazione é da considerarsi instabile.
Il criterio della simmetria può riguardare aspetti diversi di una connessione.

Si vedrà perciò nel seguito come sia possibile definire la simmetria in termini di vari parametri: nel tool é possibile utilizzare come rappresentazione di questo aspetto oltre che i dati di Livello Rete, anche dati di Livello Data Link e di Livello Fisico.

I protocolli di routing come OLSR e B.A.T.M.A.N per stimare il grado di simmetria utilizzano statistiche costruite, senza riferimenti temporali, sulla ricezione di pacchetti, tutti di broadcast e tutti di piccole dimensioni (circa 32Byte). La simmetria dei link, usando solamente questo tipo di pacchetti di probe, non potrà essere espressa più dettagliatamente di quanto non consenta l'uso della metrica ETX (*expected transmission count metrics*).
L'approccio adottato in questo lavoro per la valutazione della simmetria prende, invece, in esame più fattori di stima rispetto a quelli sopra accennati.

Asimmetria

L'asimmetria di un collegamento può essere rilevata, per esempio, dai dati di livello fisico tramite la potenza del segnale associato al pacchetto giunto a destinazione. Il significato del dato RSSI (*Received Signal Strength Indication*), messo a disposizione nel *PRISM HEADER* -come si vedrà nel capitolo successivo- é appunto quello di acquisire una stima dell'energia del segnale ricevuto. Se un link é caratterizzato da letture RSSI diverse nei due nodi che connette, il rapporto tra i segnali può fornire un'informazione quantitativa del grado di simmetria del collegamento.
L'idea, quindi, che sta alla base di questa indagine consiste nel confrontare la percezione delle qualità del collegamenti da parte dei protocolli di routing con quella rilevata dal tool, il quale però é in grado di stimare con misure

diverse le stesse caratteristiche, consentendo così misure più approfondite.
Tanto per fissare le idee, un indice catatteristico della simmetria di un canale -tipico della comunicazione con pacchetti di unicast, in cui si utilizza un meccanismo di riscontro anche a *livello di Data Link*- può essere costruito contando il numero di ritrasmissioni che sono avvenute in un verso a seguito di mancati riscontri nell'altro.
Se, per esempio, durante una sessione di test si rilevano ritrasmissioni in una comunicazione unicast in un certo verso di un link, in base a tali ritrasmissioni si potrà stimare l'efficienza del link stesso nel verso contrario.
Nel capitolo dedicato ai risultati si vedrà come é possibile affinare il grado di conoscenza dei link wireless sfruttando le citate informazioni.

3.0.8 Velocità

Nella figura 3.2 sono suddivise le componenti caratteristiche della velocità.

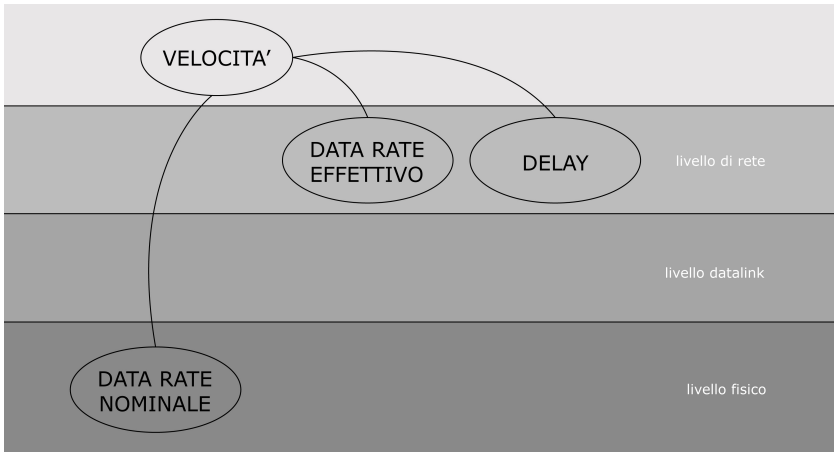


Figura 3.2: Albero della velocitá.

La stima della velocità di un collegamento senza fili, intesa come capacità di trasportare una certa quantità di informazioni nell'unità di tempo, sarà trattata con la stessa metodologia utilizzata per la caratteristica stabilità.
Questo aspetto verrà trattato in questo lavoro solo parzialmente, nella convinzione che esso, così come ritenuto per la stabilità, meriti approfondimenti ulteriori.
Nella tecnologia utilizzata in questo lavoro, i dati disponibili in base ai quali é possibile stimare la velocità di una connessione senza fili sono essenzialmente tre: il data rate nominale (che consiste nel rate al quale le schede di rete wireless si sincronizzano e che indica la velocità teorica di trasmissione senza gli overhead dovuti agli header di livello Rete e Trasporto), il data rate effettivo (*throughput* effettivo del canale in Mb/s) e il delay (ritardo in temini di tempo tra la spedizione di un pacchetto e la sua ricezione).
Come si evince dalla figura, solo il *data rate nominale* é una caratteristica riscontrabile tra le informazioni di livello fisico presenti nel *prism header*; é per questo motivo che -anche se l'argomento non verrà trattato in modo esaustivo nella nostra indagine- il tool é in grado di gestire e immagazzinare le informazioni relative al data rate nominale.
Le stime sul data rate effettivo e il delay sono ottenibili per mezzo di indagini attive, realizzabili indipendentemente dall'utilizzo di informazioni dei livelli *fisico* e *data link*.
Il data rate nominale, in questa trattazione, serve essenzialmente per indicare quali metodologie di trasmissione, tra quelle utilizzate nei protocolli 802.11b/g, sono state impiegate per spedire ogni singolo pacchetto. Nella rilettura dei dati forniti dal tool, quindi, 1,2,5.5 e 11Mb/s sono sinonimi di DSSS (Direct-Dequence Spread Spectrum), mentre 6,9,12,18,24,36,48,54 Mb/s lo sono invece per OFDM (Orthogonal Frequency-Division Multiple-

xing).
Nello specifico, ognuno dei rate sopra riportati corrisponde ad una specifica modulazione, come é possibile riscontrare nella seguente tabella d'esempio (802.11g).

Data Rate	Modulation	Encoding Rate	Bit per simbolo
6Mb/s	BPSK	1/2	48
9Mb/s	BPSK	3/4	48
12Mb/s	QPSK	1/2	96
18Mb/s	QPSK	3/4	96
24Mb/s	QAM-16	1/2	192
36Mb/s	QAM-16	3/4	192
48Mb/s	QAM-64	2/3	288
54Mb/s	QAM-64	3/4	288

Senza entrare nel merito dei singoli casi -la qual cosa esula dallo scopo di questa indagine- si può comunque notare come ogni rate sia caratteristico di una particolare modulazione, della frequenza di codifica e del numero di informazioni trasmesse su ogni portante.
Il data rate nominale costituisce, perciò, un'informazione preziosa per la rilettura dei dati di test, da momento che fornisce, non la misura diretta della velocità, ma piuttosto indicazioni sulle condizioni al *livello fisico* in cui é avvenuta ogni trasmissione.

Il rate effettivo (non trattato esaustivamente in questa trattazione, né implementato nel tool) consentirebbe di valutare l'effettiva capacità di un collegamento senza fili di trasportare informazioni. Esso però é legato (non al singolo link, ma) all'intero percorso attraversato da un flusso di pacchetti.

Per questo motivo non ci si é occupati di tale aspetto, dal momento che il tool vuole stimare le caratteristiche di singoli link, non di path completi.
Una metodologia utilizzata per la stima del rate effettivo di un intero percorso (*path*) prevede l'invio di due pacchetti di dimensione diversa: il primo dei due serve come riferimento temporale dell'inizio del secondo. Come é noto, si può sapere quando un pacchetto é stato recapitato per intero ma non quando é iniziata la ricezione, pertanto il secondo pacchetto -di rilevanti dimensioni- verrà utilizzato per la stima vera e propria della capacità (*throughput*) del canale di trasmissione. Conoscendo la tempistica sull'inizio e la fine della trasmissione e la quantità di informazioni spedite, sarà quindi possibile stimare l'entità del *throughput*.

Anche il *delay* potrebbe essere ricondotto ad una misura della velocità di trasmissione in una rete: esso infatti misura la latenza (ovvero il tempo necessario per trasmettere un messaggio minimale da un estremo del collegamento all'altro) che si verifica in un percorso di rete. Per gli stessi motivi espressi per il rate effettivo, esso non sarà trattato esaustivamente in questo lavoro né sarà implementata alcuna funzione nel tool per stimarne la misura.
Poiché il delay misura la latenza di un *path*, può essere assunto -con una similitudine- come l'indicazione di quanto sia impegnata la strada (link) seguita dai dati e quanto siano trafficati gli incroci (router), ossia quanto sia congestionato il traffico lungo un certo percorso.
Le stime di delay si potrebbero anche utilizzare per la valutazione di un protocollo per reti mesh nel caso specifico di particolari forme di utilizzazione, in cui si richiede che il traffico raggiunga senza alcun ritardo l'altro capo della comunicazione: questo per esempio il caso di utilizzo della rete per il servizio di comunicazioni in voce, ossia del cosiddetto VoIP (*Voice Over IP*), per cui

il quale il delay costituirebbe un ottimo indice di performance.

Capitolo 4

TOOL

Ogni protocollo mesh implementa un meccanismo di probing dei link per popolare le MIB dei vari nodi che compomgono la rete. Per i protocolli piú conosciuti il probing dei link viene effettuato eslusivamente in broadcast a livello 3, senza tenere conto del traffico unicast e delle informazioni relative al livello fisico del canale. Come conseguenza si possono verificare situazioni in cui la percezione della bontá di un link non rispecchia del tutto la realtá. Generalmente i protocolli di mesh routing utilizzati oggi non tengono conto della tecnologia con cui sono realizzati i link considerando allo stesso modo link a 5 Ghz e link 2.4Ghz.

Il tool, di cui al presente documento, si propone di offrire un livello di informazioni ben piú dettagliato sullo stato dei link facendo dei probe sia in broadcast che in unicast, memorizzando informazioni provenienti anche dai livelli 1 e 2 della pila ISO/OSI quali il livello del segnale (RSSI), il numero di retry dei pacchetti di unicast, il rate nominale con cui vengono forgiati e ricevuti i pacchetti. Da questa serie di informazioni é possibile trarre delle conclusioni accurate riguardo l'effettiva bontá del link per poi metterle a con-

fronto con le decisioni prese contemporaneamente dai protocolli che girano sulla stessa rete, e verificare la loro coerenza con il vero stato delle cose, e per pensare alla concezione di un nuovo tipo di metrica. Fondamentalmente il TOOL é uno strumento in grado di effettuare test su reti mesh wireless e supportare le scelte tecniche per un loro impiego piú efficace.

Il tool é stato interamente realizzato nel linguaggio di programmazione C e sviluppato su piattaforma Linux x86. La scelta di piattaforme Unix like é stata dettata in parte da affinitá concettuali, ma soprattutto dall'esigenza di avvalerci di strumenti quali il PRISM HEADER, l'RADIO TAP HEADER e le list head di linux, altrimenti non disponibili; inoltre ci ha permesso di avere un controllo totale sulle macchine su cui lo abbiamo sviluppato. Per i test ci si é appoggiati alla rete della community di "ninux.org" operativa su Roma. Si tratta di una rete OLSR costruita da appassionati aperta allo sviluppo e al testing di nuovi progetti. Tale rete é perlopiú costituita da apparati Linksys WRT o ASUS DW500 sui quali é stato necessario crosscompilare il tool per piattaforma MIPSEL.

In questo capitolo tratteremo approfonditamente come é stato concepito il tool partendo dalla metodologia, quindi la sua struttura e modularitá, esponendo le scelte e gli algoritmi principali adottati per memorizzare e gestire i dati ottenuti, sino ad arrivare alla tunabilitá dei parametri di test e alla visualizzazione dei dati.

4.1 metodologia

Il tool prevede 4 segmenti, mostrati anche in figura 4.1 : *i nodi della rete mesh, sui quali gira un protocollo di routing, che effettuano i probe ai livelli di rete, data link e fisico della pila ISO/OSI memorizzandone i risultati. Sono la parte attiva dei test; un server di memorizzazione che ha il compito di recuperare i dati salvati su ogni nodo; un motore di interrogazione sui dati del server che chiameremo server di analisi; un visualizzatore grafico dei risultati delle interrogazioni.* L'unica assunzione che é stata fatta é che i nodi siano temporalmente sincronuzzati tra loro almeno al secondo (per questo ci si é affidati al servizio NTP).

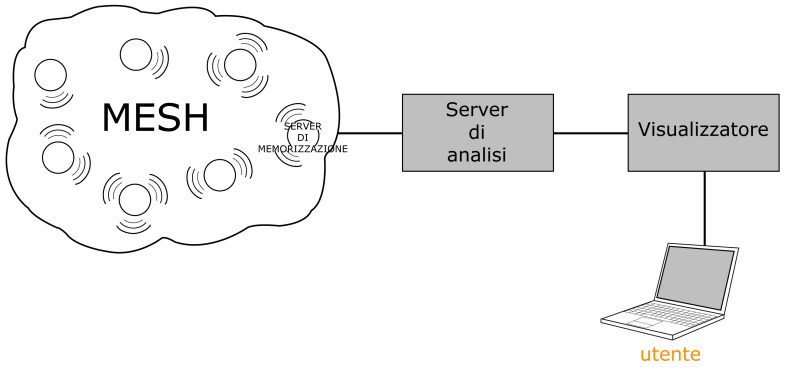


Figura 4.1: Modularitá del TOOL.

Sui nodi la fase di test é divisa temporalmente in epoche, dove un'epoca corrisponde ad una serie di invio di 30 pacchetti di broadcast ed ad una serie di invio di 30 pacchetti di unicast verso ogni vicino di cui si é sentito almeno un pacchetto di broadcast ed un eventuale tempo di inattivitá. Per ogni pacchetto ricevuto da un nodo vengono memorizzati il tipo (broadcast

o unicast), l'RSSI, il rate, la dimensione e nel solo caso dell'unicast anche il numero di retry. Ogni ciclo di test ha un riferimento temporale di inizio e di fine che servirá sul server per l'allineamento e il confronto dei dati provenienti da tutti i nodi della rete.
Alla fine di un ciclo di test con un vicino ogni nodo provvede a memorizzare i dati su una lista di tipo LIST HEAD di cui parleremo piú avanti. Oltre alla fase attiva di test, i nodi della rete mesh rimangono in ascolto su una determinata porta per le richieste inviate dal server di memorizzazione.

Il server di analisi provvede a comunicare i parametri di test ai nodi della rete e a collezionare i dati ricevuti per renderli disponibili al server di analisi. Puó comunicare con i nodi con tre tipi di pacchetti : una richiesta di start_test, piú richieste di get_data e una richiesta di end_test. Il nodo che riceve una richiesta di get_data spedisce tutti i dati memorizzati sino a quel momento al server e provvede a cancellarli per liberare la sua memoria. Una richiesta di end_test che temina l'esecuzione dei test sui nodi.
Per ora la memorizzazione dei dati avviene su un file di testo per semplice redirezione dello standard output.

Il server di analisi é del tutto personalizzabile rispetto alle esigenze che si hanno e puó essere piú o meno complesso. Si possono infatti implementare delle query che calcolino automaticamente se il path deciso dal protocollo di routing che gira sulla rete é effetivamente il piú rapido o il piú stabile nel tempo confrontando le MIB su tutti i nodi con le informazioni ottenute dal tool. Si puó chiedere di calcolare secondo i dati del tool in un dato momento qual è l'albero ricoprente migliore per raggiungere da un nodo ongni altro nodo della rete e confrontarlo con quello calcolato nello stesso momento dal

protocollo di routing. Oppure si puó semplicemente richiedere dei dati statistici sulla stabilitá o la velocitá nel tempo di alcuni link. Questi sono solo alcuni esempi di query che si potrebbero implementare.

Il visualizzatore infine ha il compito di fornire all'utente che utilizza il tool un front end grafico che riproponga le query previste dal processatore e renda i risultati leggibili e di facile consultazione.

Attualmente sono state implementate accuratamente la parte relativa ai nodi e al server e abbozzato un visualizzatore grafico realizzato in PHP su server web Apache. La figura del server di analisi non é stata implementata ma sostituita da noi stessi nella lettura del file di testo scritto dal server di memorizzazione.

4.1.1 nodi

I nodi della rete mesh svolgono la parte attiva di testing inviando e ricevendo pacchetti di broadcast e unicast e memorizzando informazioni dei livelli 1, 2 e 3 della pila ISO/OSI. Il flusso di esecuzione del tool sui nodi é come illustrato in figura 4.2.

I test su ogni nodo sono divisi in epoche a partire dall'epoca 0 per ogni vicino. L'epoca, di cui si illustra il concetto successivamente, l'ip del mittente del pacchetto e il tipo di pacchetto identificano univocamente l'esito di un probe su un link tra 2 nodi. Queste informazioni unite ai dati presi dal livello fisico e Data Link descrivono ampiamente l'andamento temporale del link in questione. Una volta terminato il dialogo tra 2 nodi, i dati vengono memorizzati e mantenuti in una lista neigh_node di tipo LIST HEAD e

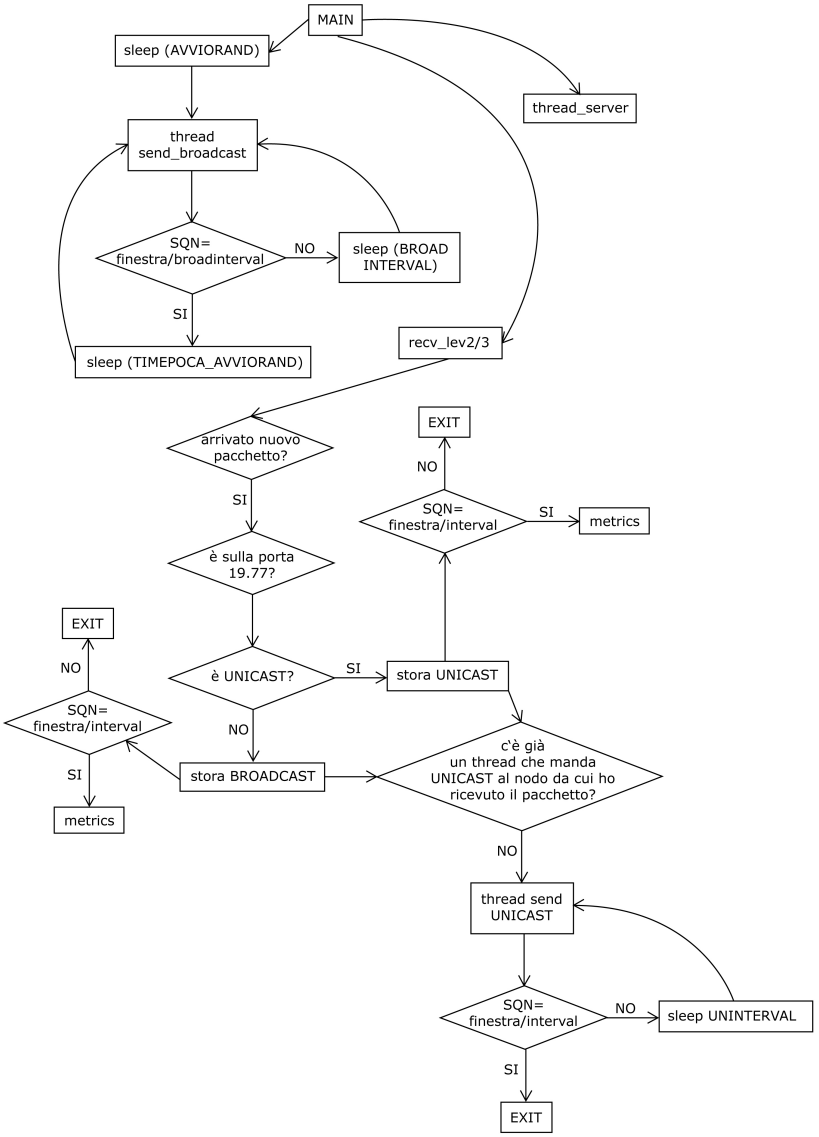


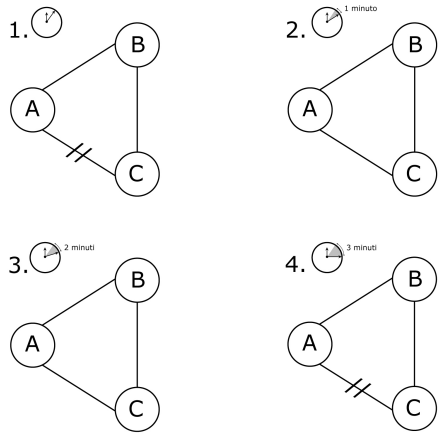
Figura 4.2: Flusso di esecuzione sul nodo.

contemporaneamente resi disponibili per la richiesta get_data del server. Ad ogni dialogo broadcast o unicast tra 2 nodi corrispondono due entry nella LIST HEAD neigh_node, una relativa al broadcast ed un relativa all'unicast. Il meccanismo secondo il avviene il dialogo tra due nodi é illustrato di seguito.

Per ogni nodo vicino di cui si é ricevuto almeno un pacchetto di broadcast o di unicast, si inizializza una struttura di tipo neigh_node individuata dall'ip del mittente e dal numero di epoca in cui si memorizzano i dati dei probe. Una sessione di ricezione di broadcast e unicast da uno stesso vicino é individuata sul nodo ricevente da un numero di EPOCA. Il concetto di epoca é quindi legato a chi riceve i pacchetti.

L'epoca é un segmento temporale in cui inizia e si conclude una sessione di probing di pacchetti unicast e broadcast scambiati tra due nodi. Ogni nodo, durante una sessione, trasmette in broadcast una ed una sola volta ma comincia a trasmettere in unicast solo con quei nodi dei quali ha ricevuto almeno un pacchetto di broadcast. Se ne deduce che un nodo isolato invierá solo pacchetti di broadcast.

Per illustrare meglio il concetto di epoca procediamo con un esempio. Sia A il nodo su cui stiamo osservando la lista neigh_node e B, C due nodi a portata radio con il nodo A. Si puó osservare dalla figura 4.3 che dopo il quarto ciclo di test nella lista neigh_node del nodo A sono memorizzate le epoche 0,1,2,3 del nodo B e le epoche 0,1 del nodo C.



Nodo A

	Invio Unicast Verso			Ricezione Broadcast Da	
	C	B	Entry Struttura Neigh_Node	C	B
1	—	X	vuota	—	epoca 0
2	X	X	salvataggio epoca 0 di B	epoca 0	epoca 1
3	X	X	salvataggio epoca 1 di B epoca 0 di C	epoca 1	epoca 2
4	—	X	salvataggio epoca 2 di B epoca 1 di C	epoca 1	epoca 3
5	—	—	salvataggio epoca 3 di B epoca 1 di C	epoca 1	epoca 3

X invio di una sessione di UNICAST
– nessun invio

Figura 4.3: Esempio di successione temporale di 4 sessioni di test sul nodo A.

La causa del disallineamento delle epoche sul nodo A relative ai probe con i nodi B e C potrebbe essere dovuta all'inoperabilitá del link tra C e A durante i cicli di test 1 e 4. Per semplicitá nell'esempio é stato omesso il traffico di unicast generato dai nodi in risposta al traffico di broadcast.
Analizziamo ora come sono strutturati i pacchetti scambiati tra i nodi della rete mesh (figura 4.4).

Figura 4.4: Struttura del pacchetto scambiato tra i nodi per i probe.

L'SQN é il numero di sequenza del pacchetto ed ha unimportanza fondamentale nella memorizzazione dei risultati dei probe. C'é una corrispondenza biunivoca tra SQN e i buffer di memorizzazione dei dati di probe. L'SQN serve al ricevente per individuare univocamente il pacchetto e per scrivere i risultati dei probe nella posizione (i-1)-sima dei buffer relativi al nodo mittente. I buffer d'altro canto hanno la funzione di mantenere l'andamento temporale della comunicazione tra 2 nodi e inizialmente vengono inizializzati tutti a 0. L'SQN varia da 0 sino FINESTRA/BROADINTERVAL per i pacchetti di broadcast e FINESTRA/UNIINTERVAL per i pacchetti di unicast.

Come esempio si ipotizzi che il nodo A stia inviando una sessione di pacchetti di broadcast al nodo B. I buffer sul nodo B seguiranno l'andamento mostrato nella tabella :

Pacchetti Inviati dal nodo A		Strutture memorizzate sul nodo B			
SQN pacchetto	ricevuto da B?	bbuffer	brate	brssi	bsize
1	no	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2	no	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
3	si	0 0 1 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0	0 0 70 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0
4	si	0 0 1 1 0 0 0 0 0 0	0 0 1 1 0 0 0 0 0 0	0 0 70 63 0 0 0 0 0 0	0 0 1 3 0 0 0 0 0 0
5	no	0 0 1 1 0 0 0 0 0 0	0 0 1 1 0 0 0 0 0 0	0 0 70 63 0 0 0 0 0 0	0 0 1 3 0 0 0 0 0 0
6	si	0 0 1 1 0 1 0 0 0 0	0 0 1 1 0 1 0 0 0 0	0 0 70 63 0 67 0 0 0 0	0 0 1 3 0 2 0 0 0 0
7	no	0 0 1 1 0 1 0 0 0 0	0 0 1 1 0 1 0 0 0 0	0 0 70 63 0 67 0 0 0 0	0 0 1 3 0 2 0 0 0 0
8	no	0 0 1 1 0 1 0 0 0 0	0 0 1 1 0 1 0 0 0 0	0 0 70 63 0 67 0 0 0 0	0 0 1 3 0 2 0 0 0 0
9	si	0 0 1 1 0 1 0 0 1 0	0 0 1 1 0 1 0 0 1 0	0 0 70 63 0 67 0 0 83 0	0 0 1 3 0 2 0 0 1 0
10	si	0 0 1 1 0 1 0 0 1 1	0 0 1 1 0 1 0 0 1 1	0 0 70 63 0 67 0 0 83 85	0 0 1 3 0 2 0 0 1 1

FINESTRA = 10 BROADINTERVAL = 1

- Il FLAG puó assumere i valori 'u' per unicast e 'b' per broadcast;
- SEC e USEC indicano in secondi e millisecondi quando é stato inviato il pacchetto; attualmente non viene utilizzato, ma ipotizzando una perfetta sincronizzazione della data di sistema tra i nodi della rete, si potrebbero utilizzare per calcolare il delay del link in esame;
- Il PADDING serve a generare pacchetti di dimensione arbitraria ed é riempito con 0. Si potrebbe utilizzare in futuro se si ha la necessitá di inviare altri dati.

Per ora é possibile lanciare il tool su un nodo per una sola interfaccia di rete. Non si é presentata la necessitá di implementarlo per nodi con piú di una

interfaccia poiché le reti mesh di solito sono costituite da nodi con una sola scheda di rete che parla con tutti.

4.1.2 server di memorizzazione

Come giá accennato, tale server si occupa di memorizzare i dati ricevuti dai nodi della rete mesh su cui é in esecuzione il tool. Il flusso di esecuzione é quello illustrato in figura 4.5.

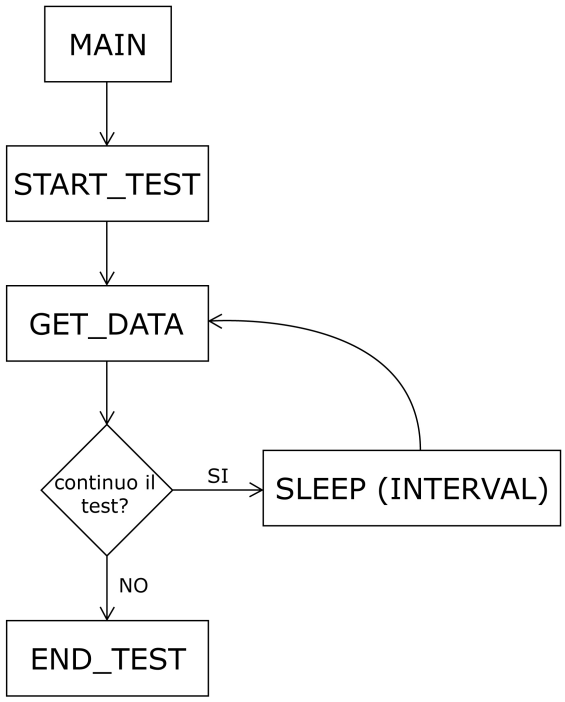


Figura 4.5: Flusso di esecuzione sul server di memorizzazione.

Il tool sul server viene lanciato da riga di comando digitando :

sh#./tool_server 172.16.162.99 172.16.0.100 172.16.0.2 172.16.0.1 172.16.0.3

dove gli IP che compaiono sono relativi ai nodi ai quali verranno inoltrate le richieste del server. É possibile prevedere in futuro un meccanismo di discovery basato su multicast per individuare automaticamente i nodi che hanno attivo come servizio il test. I dati ricevuti ad ora vengono memorizzati su un file tramite una semplice redirezione dell'output lanciando il tool sul server in tal modo :

sh#./tool_server 172.16.162.99 172.16.0.100 172.16.0.2 172.16.0.1 172.16.0.3 ¿ testX.txt

Il server deve poter raggiungere almeno un nodo della rete mesh per poter impartire le direttive. É consigliabile che possa comunicare con piú nodi per evitare di sovraccaricare un solo nodo con il traffico provemiente dal tutta la rete.

Il server invia 3 tipi di pacchetti distinti da un flag a tutti i nodi della rete mesh che partecipano al test : al flag 0 corrisponde un pacchetto di start_tool con cui comunica i parametri di test; al flag 1 corrisponde un pacchetto di get_data che richiede i risultati del test memorizzati sui nodi sino a quel momento; al flag 2 corrisponde un pacchetto di end_tool che fa terminare l'esecuzione del test sui nodi.

4.1.3 comunicazione tra nodi e server

Come detto sopra la comunicazione tra il server e i nodi si limita a soli 3 pacchetti : start_tool, get_data e end_tool. Vediamo ora nel dettaglio in cosa consistono tali pacchetti.
Il pacchetto di start é del tipo mostrato in FIGURA5a

Figura 4.6: Pacchetto di *start_test*.

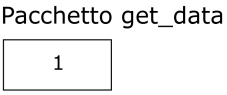


Figura 4.7: Pacchetto di *get_data*.

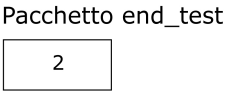


Figura 4.8: Pacchetto di *end_test*.

- il campo FLAG ha valore 0;

- il campo SEQ indica in che sequenza devono essere spediti dai nodi i pacchetti di dimensione differente per effettuare il probing dei link;

- il campo BROADINTERVAL indica in secondi quanto deve essere lungo l'intervallo di tempo che intercorre tra la spedizione di un pachhetto di broadcast e il seguente relativamente alla stessa epoca;

- il campo UNIINTERVAL indica in secondi quanto deve essere lungo l'intervallo di tempo che intercorre tra la spedizione di un pachhetto di unicast e il seguente relativamente alla stessa epoca;

- il campo BROADSLEEP indica il massimo numero in secondi su cui calcolare il tempo di attesa iniziale prima di cominciare a spedire pacchetti di broadcast;

- il campo TIMEEPOCHE indica quanti secondi dura una singola fase di test relativa ad una sola epoca;

- il campo EPOCHE indica quante epoche al massimo possono essere memorizzate sui nodi per ogni suo vicino. La gestione della memorizzazione dei datti relativi alle epoche é di ipo FIFO.

- il campo FINESTRA indica quanti secondi deve durare la fase attiva di invio pacchetti di broadcast. É strettamente legato a BROADINTERVAL e al SEQ e lo andiamo ad illustrare con un esempio. Ipotizziamo che FINESTRA sia di 30 secondi; se BROADINTERVAL fosse 2 invieremmo un pacchetto di broadcast ogni 2 secondi per un totale di 15 pacchetti in 30 secondi; se BROADINTERVAL fosse di 1 secondo ne manderemmo 30 in 30 secondi. Inoltre sarebbe auspicabile che la lunghezza della sequenza di pacchetti indicata in SEQ fosse un numero divisore di FINESTRA, in modo da poter essere ripetuta esattamente.

Ipotizziamo ora FINESTRA sia di 30 secondi, BROADINTERVAL pari a 2, SEQ 111223121311221 e che ad 1 corrisponda un pacchetto di 81 byte, a 2 uno di 749 byte e a 3 uno di 1449 byte. Con queste ipotesi ogni nodo invierebbe i propri pacchetti di broadcast secondo la tabella seguente :

sec	invio	tipo_pacchetto	dimensione_pacchetto
1	si	1	81 byte
2	no	-	-
3	si	1	18 byte
4	no	-	-
5	si	1	81 byte
6	no	-	-
7	si	2	749 byte
8	no	-	-
9	si	2	749 byte
10	no	-	-
11	si	3	1449 byte
12	no	-	-
13	si	1	81 byte
14	no	-	-
15	si	2	749 byte
16	no	-	-
17	si	1	81 byte
18	no	-	-
19	si	3	1449 byte
20	no	-	-
21	si	1	81 byte
22	no	-	-
23	si	1	81 byte
24	no	-	-
25	si	2	749 byte
26	no	-	-
27	si	2	749 byte
28	no	-	-
29	si	1	81 byte
30	no	-	-

Ora ipotizziamo che TIMEPOCHE sia 60 secondi e BROADSLEEP sia 15 secondi. Ogni nodo si calcola a partire da BROADSLEEP un numero random compreso tra 0 e 15 e lo memorizza nella variabile AVVIORAND. Questo sará il tempo che trascorrerá prima che il nodo inizi a mandare pacchetti di broadcast secondo lo schema descritto sopra. Alla fine di ogni epoca viene ricalcolato un nuovo AVVIORAND e iniziata una nuova epoca.
I pacchetti get_data e di end_tool non hanno informazioni ulteriori al proprio flag e sono rispettivamente una richiesta di invio dati e di termine applicazione. I pacchetti di start_tool e end_tool vengono inviati una volta soltanto durante una sessione di test, mentre il pacchetto di get_data puó essere inviato un numero arbitrario di volte.
La comunicazione tra client e server avviene per mezzo dell'instaurazione di una connessione TCP/IP classica. Nella figura 4.9 si puó osservare un esempio di scala temporale relativa a tale comunicazione.

4.1.4 comunicazione tra i nodi

Come giá accennato la comunicazione tra i nodi della rete consta nell'invio e nella relativa ricezione di pacchetti di broadcast e di unicast. I pacchetti di broadcast oltre ad essere oggetto di analisi per il tool, servono a far conoscere ai propri vicini la propria esistenza. Una volta ricevuto un pacchetto di broadcast da un vicino, immediatamente il nodo ricevente inizia a mandare in risposta una sequenza di pacchetti di unicast in numero pari a FINESTRA/UNIINTERVAL. Se la finestra é di 30 secondi e UNIINTERVAL é pari a 1 secondo, il nodo invierá 30 pacchetti di unicast con numero di sequenza incrementale da 1 a 30 ogni secondo (figura 4.10).
Per la comunicazione tra i vari nodi abbiamo implementato una SOCKET

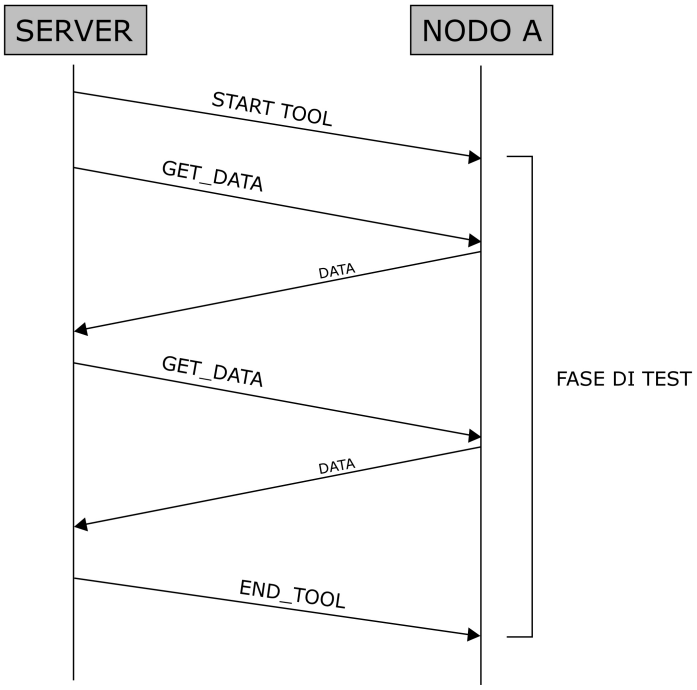


Figura 4.9: Comunicazione tra un nodo mesh e il server di memorizzazione.

RAW. Tale scelta é stata obbligata dalla necessitá di poter accedere ad informazioni sulle caratteristiche quali il rate e l'RSSI, sia del canale che del meccanismo delle ritrasmissioni dei pacchetti di unicast. Con le SOCKET RAW é stato possibile "sniffare" tutti i pacchetti ricevuti al livello fisico della scheda di rete. Vengono processati solo i pacchetti che arrivano sulla porta di comunicazione tra nodi definita nelle impostazioni del tool; i restanti pacchetti vengono scartati.

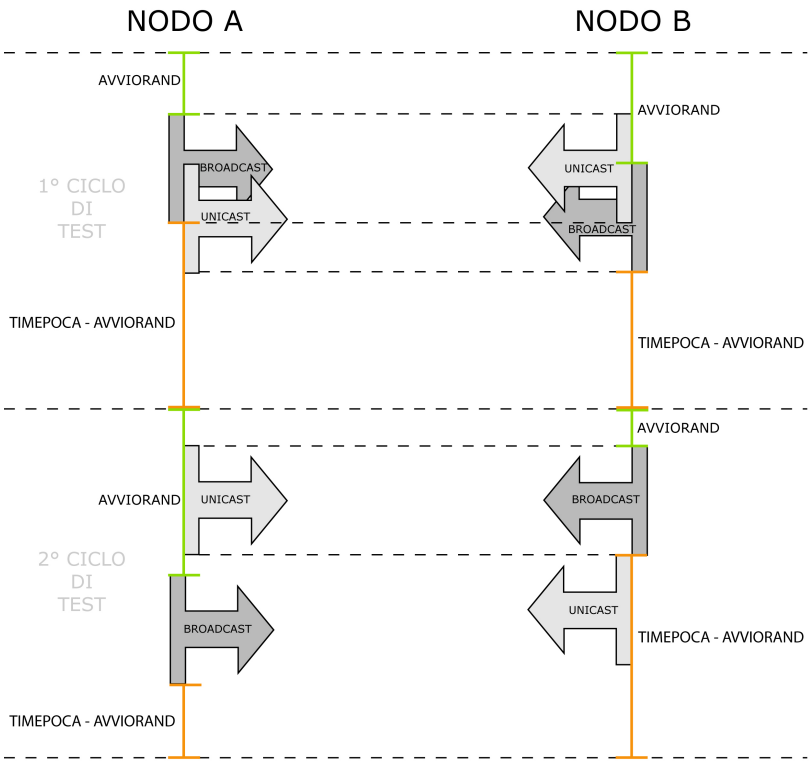


Figura 4.10: Comunicazione tra nodi mesh.

La figura illustra un esempio di applicazione del test non eseguita in continuo. É possibile modificare i parametri del tool affinché i test vengano

fatti in continuo senza intervalli di inattivitá tra un'epoca e l'altra. Basta impostare AVVIORAND pari a 0, e FINESTRA uguale a TIMEEPOCA.

Comunicazione ai livelli 1-2 e al livello 3

Per avere le informazioni dal livello fisico della scheda di rete abbiamo castato i pacchetti ricevuti al livello fisico tramite il PRISM HEADER . C'é da dire che il PRISM HEADER non é universale per tutte le schede di rete, poiché é stato concepito per schede con chipset ATHEROS, BROADCOM e PRISM. Del PRISM HEADER tratteremo piú specificatamente nei prossimi paragrafi. Per le schede con chipset INTEL quali ipw2200 e ipw3386 c'é un meccanismo analogo che tramite l'interfaccia di monitor rtap permette di accedere alle stesse informazioni ma al momento non l'abbiamo contemplato.
Ci sono inoltre schede di rete wireles per cui non é stato implementato nessun meccanismo di monitoring del canale. Per tali schede si é implementato una parte del tool che effettua il probing del canale solo al livello 3, senza quindi avvalersi di dati come RSSI, ritrasmissioni unicast, rate con cui sono stati inviati i pacchetti. Nodi con schede di questo tipo memorizzano la semplice ricezione o meno di un pacchetto con un 1 e uno 0 rispettivamente. Ció non va a discapito della memorizzazione dell'andamento temporale del link.

4.2 strutture utilizzate

In questa sezione viene trattato il metodo di memorizzazione dei dati sia sui nodi che sul server e quali strutture sono state utilizzate.

4.2.1 strutture sul nodo

Come detto in precedenza viene usato il PRISM HEADER per castare i pacchetti letti tramite la SOCKET RAW al fine di avere informazioni relative al pacchetto ricevuto quali l'RSSI, il numero di retry e il rate. Per ogni pacchetto ricevuto, tali informazioni vengono memorizzate sulla lista neigh_node realizzata ad hoc. Di seguito vengono mostrate in dettaglio tali strutture.

prism header

Quella che segue é parte del file prism.h. Si possono notare le voci RSSI, Data_Rate e Frame_Control all'interno del quale é presente il bit di retry.

```
    struct prism {
...
struct prism_value RSSI;
...
struct prism_value Data_Rate;
...
//ieee 802.11 header
struct Frame_Control fc;
...
unsigned char MAC1[6];
unsigned char MAC2[6];
unsigned char MAC3[6];
...
};
```

```
    struct Frame_Control {
```

unsigned char header;

unsigned char flags;

};

Per ogni pacchetto ricevuto, sia broadcast che unicast, viene effettuato un casting con degli offset ben precisi per accedere a tali dati.

La list_head neigh_node

La lista neigh_node é stata pensata per memorizzare i risultati dei probe. Su ogni nodo ci saranno 2 strutture neigh_node in memoria relative ad ogni nodo di cui si é sentito almeno un pacchetto di broadcast e uno di unicast. La distinzione tra il traffico unicast e quello broadcast viene fatta tramite la union delle struct unicast e broadcast. Tali strutture sono composte da una serie di buffer di lunghezza pari al numero di pacchetti che vengono inviati (FINESTRA/BROADINTERVAL e FINESTRA/UNIINTERVAL) all'interno dei quali vengono memorizzati in posizione (i-1)-sima i dati ottenuti tramite il PRISM HEADER relativi al pacchetto ricevuto con numero di sequenza i. I campi in comune tra le due strutture sono :

- *ip* é una sockaddr_in contenente l'IP del nodo da cui si é ricevuto il pacchetto;
- *type* è 'u' se il pacchetto ricevuto é di unicast e 'b' se il pacchetto ricevuto é di broadcast;
- *check_t_unicast* indica se c'é giá un thread che sta inviando pacchetti di unicast verso il nodo da cui si é ricevuto il pacchetto di broadcast;

- *epoca* indica per quel nodo in quale ciclo di test siamo. Inizialmente é 0 per tutti i nodi. Con il susseguirsi dei cicli di test tale valore viene incrementato di una unitá;
- *start_aware* indica il tempo espresso in TIMESTAMP relativo alla ricezione del primo pacchetto della sequenza inviata dal nodo X;
- *last_aware* indica il tempo espresso in TIMESTAMP relativo alla ricezione dell'ultimo pacchetto della sequenza inviata dal nodo X;
- *ubuffer* e *bbuffer* possono assumere i valori 0 o 1 in posizione i a se il pacchetto inviato dal nodo X con numero di sequenza i é arrivato o meno. Di default i buffer sono tutti a 0;
- in *urate* e *brate* viene memorizzato in posizione i il rate con cui si é ricevuto il pacchetto inviato dal nodo X con numero di sequenza i;
- in *usize* e *bsize* viene memorizzata in posizione i la dimensione del pacchetto inviato dal nodo X con numero di sequenza i;
- in *urssi* e *brssi* viene memorizzata in posizione i il livello del segnale con cui é stato ricevuto il pacchetto inviato dal nodo X con numero di sequenza i;
- in *uretry* viene memorizzato in posizione i il numero di retry relativi al pacchetto i inviato dal nodo X e ricevuto dal nostro nodo. Tale tipo di buffer é presente soo nella struttura unicast poiché il broadcast non in implementa il meccanismo di ritrasmissione dei paccheti;
- L'ultimo campo é la *list_head* di cui tratteremo piú avanti.

struct unicast{

```
char ubuffer[FINESTRA];
short urate[FINESTRA];
short usize[FINESTRA];
short urssi[FINESTRA];
short uretry[FINESTRA];
};
struct broadcast{
char bbuffer[FINESTRA];
short brate[FINESTRA];
short bsize[FINESTRA];
short brssi[FINESTRA];
short padding[FINESTRA];
};
struct neigh_node{
char type;
int check_t_unicast;
int epoca;
time_t last_aware;
time_t start_aware;
int last_sqn;
struct sockaddr_in ip;
union unione_bu {
struct broadcast b;
struct unicast u;
} node_type;
struct list_head neigh_list;
};
```

Si é scelto di gestire la memorizzazione dei dati tramite le LIST HEAD di Linux per la loro efficienza e provata affidabilitá. Sono infatti implementate nel kernel di Linux e utilizzate dallo scheduler dei processi. Si tratta di liste circolari doppiamente collegate (figura 4.11).

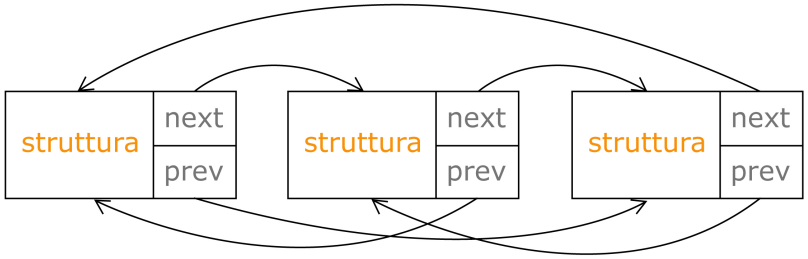


Figura 4.11: List head di linux.

Nel tool vengono utilizzate per memorizzare i risultati dei probe sui vari link adiacenti al nodo in esame. Ogni entry è relativa ad un'epoca,un ip e un tipo di traffico. Non é possibile avere due o piú entry con stessa epoca, ip e tipo (figura 4.12). Forse gli esperti di database storceranno il naso ma la terna epoca,ip e tipo costituiscono una sorta di chiave primaria.
Il server comunica ai nodi della rete mesh il numero di epoche che devono essere mantenute in memoria su ogni nodo secondo una politica di tipo FIFO. Quindi man mano che si va avanti nei cicli di test, i dati piú vecchi vengono cancellati per far spazio a quelli piú recenti. Se il server non ha provveduto a richiedere tali dati tempo, le informazioni da essi contenute saranno perse.

Nella dichiarazione di neigh_node si puó trovare la voce :

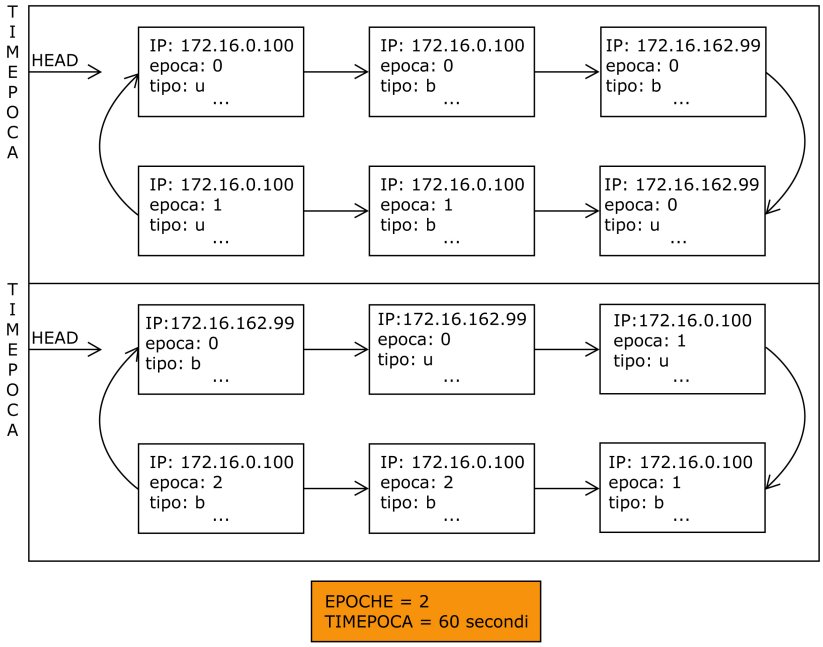


Figura 4.12: Esempio di memorizzazione dei risultati di probe sulla list_head neigh_node.

struct list_head neigh_list;

che non é altro che una struttura di 4 puntatori al prossimo e precedente elemento, alla testa e alla coda del lista.

livello 3

Per i nodi su cui non é possibile utilizzare il PRISM HEADER é stata implementata una parte del tool che lavora solo a livello 3 le cui strutture sono analoghe a quelle del livello 2 ma con meno informazioni :

```
struct unicast{
char ubuffer[FINESTRA];
short usize[FINESTRA];
};
struct broadcast{
char bbuffer[FINESTRA];
short bsize[FINESTRA];
};
struct neigh_node{
char type;
int check_t_unicast;
int epoca;
time_t last_aware;
time_t start_aware;
int last_sqn;
struct sockaddr_in ip;
```

union unione_bu

struct broadcast b;

struct unicast u;

node_type;

struct list_head neigh_list;

};

4.2.2 strutture sul server

Sul server si utilizzano due strutture fondamenrali. Una é la struttura neigh_node giá utilizzata sui nodi che serve per castare i dati ricevuti e poi stamparli a video.

La struttura from_server viene utilizzata per preparare le informazioni che verranno spedite ai nodi della rete.

struct from_server{

char flag; //0 per start_test, 1 per get_data, 2 per end_test

char seq[SEQ]; //sequenza delle dimensioni dei pacchetti

int uniinterval; //intervallo in secondi tra un unicast e l'altro

int broadinterval; //intervallo in secondi tra un broadcast e l'altro

int broadsleep; //tempo in secondi su cui calcolare randomicamente il tempo di start del tool sul nodo

int epoche; //quante epoche storare

int timeepoca; //quanti secondi dura un'epoca

int finestra; //quanti secondi è grande la finestra di attività

};

4.3 algoritmi

4.3.1 Ricezione dei pacchetti

Nella figura 4.13 viene illustrato come un nodo gestisce un pacchetto, unicast o broadcast, appena ricevuto.

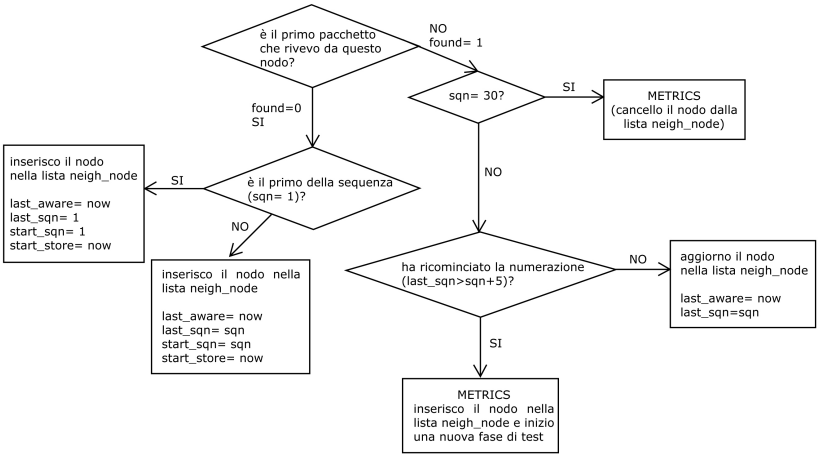


Figura 4.13: Algoritmo di ricezione pacchetto di probe.

4.3.2 epoche

Un'epoca é identificata temporalmente dalle variabili *start_aware* e *last_aware*. La numerazione delle epoche relative a sessioni di broadcast é svincolata dalla numerazione relativa a sessioni di unicast. Supponiamo che il nodo B inizi a spedire per la prima volta pacchetti broadcast e che venga sentito dal nodo A; immediatamente A risponderá con una serie di pacchetti di unicast e creerá una nuova entry nella lista neigh_node con ip di B, tipo b ed epoca 0. Nel frattempo A ha iniziato a generare paccheti broadcast che vengono sentiti da B, il quale a sua volta gli risponderá con una serie di pacchetti unicast; anche in tal caso A creerá una nuova entry per la lista neigh_node con ip di

B, epoca 0 ma tipo u. Quindi é nella lista neigh_node saranno presenti due entry con lo stesso ip, stessa epoca ma tipo differente.
Ogni epoca ha la durata massima di TIMEPOCA secondi e si possono memorizzare sino ad un massimo di 255*2(broadcast e unicast) epoche per ogni nodo a discapito della memoria. É quindi possibile memorizzare l'andamento di un link per 255*TIMEPOCA secondi. É ovvio che sta al buon senso dell'utente che utilizzerá il tool il compito di dimensionare a modo i parametri di test.

4.3.3 purge e metrics

La funzione sui nodi che rende disponibili i dati per le richieste del server é metrics. Tale funzione provvede a cancellare una entry dalla lista neigh_node e inserirli in un'altra lista di tipo LIST HEAD che verrá spedita al server. É in quest'ultima lista che viene effettivamente gestita la politica di memorizzazione di tipo FIFO poiché mantiene i dati per ogni nodo relativo alle utlime EPOCHE epoche.
Metrics viene lanciata ogni qualvolta viene ricevuto un pacchetto con numero di sequenza massimo (FINESTRA/BROADINTERVAL per i pacchetti di broadcast e FINESTRA/UNIINTERVAL per i pacchetti di unicast) oppure se il pacchetto ricevuto ha lo stesso ip, tipo di una entry nella lista neigh_node ma numero di sequenza ¡ almeno di 5. Questo meccanismo serve per capire quando il nodo mittente ha iniziato a spedire pacchetti relativi ad un'altra epoca senza che al ricevente sia pervenuto l'ultimo pacchetto dell'epoca precedente (figura 4.14).

Nel caso in cui il link tra due nodi funziona solo per parte di un'epoca e che venga perso l'ultimo pacchetto della sequenza, le entry della neigh_node

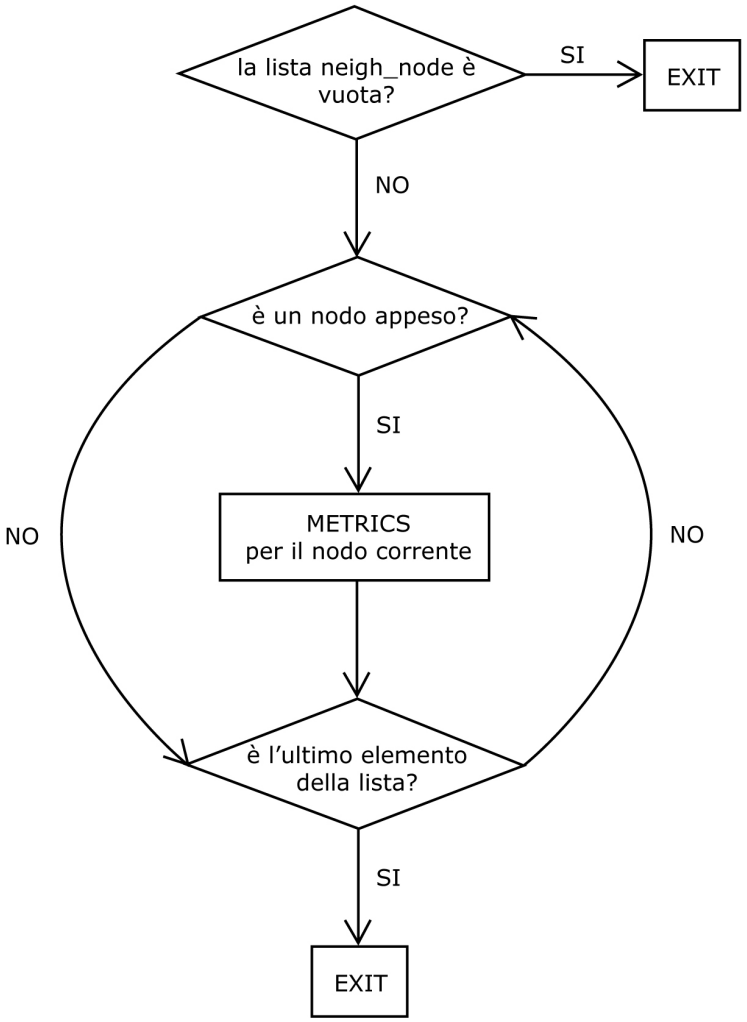


Figura 4.14: Algoritmo di esecuzione di metrics.

relative a tale probe rimarrebbero “appese” senza mai essere preparate e spedite al server. É stato quindi implementato un meccanismo di rintracciamento delle entry “appese” che viene lanciato ogni fine sessione di test prima di iniziare una nuova sessione, come mostrato in figura 4.15.

4.4 tunabilitá

Uno dei punti forti del tool é la sua alta personalizzabilitá a seconda delle esigenze di chi vuole testare la rete. Agendo sui paramentri di configurazione é possibile costruire dei test completmente differenti l’uno dall’altro ed in grado di effettuare dei probe ben definiti.L’utente che utizza il tool ha cosí la possibilitá di costruire dei test specifici.
Il tool puó essere configurato in modo da effettuare sia test statistici sulla rete mesh (ossia ad intervalli randomici) che test in continuo senza periodi di pausa tra una sessione e la successiva.

Nei test in continuo i nodi generano ininterrotamente pacchetti di broadcast e rispondono con altrettanti pacchetti di unicast. Questo modo di operare genera sicuramente dell’overhead sulla rete, ma rende le misure piú dettagliate.

esempio di test continuo
TIMEEPOCA = FINESTRA = 120
BROADINTERVAL = UNIINTERVAL = 1
BROADSLEEP = 0
SEQ = 1223333221

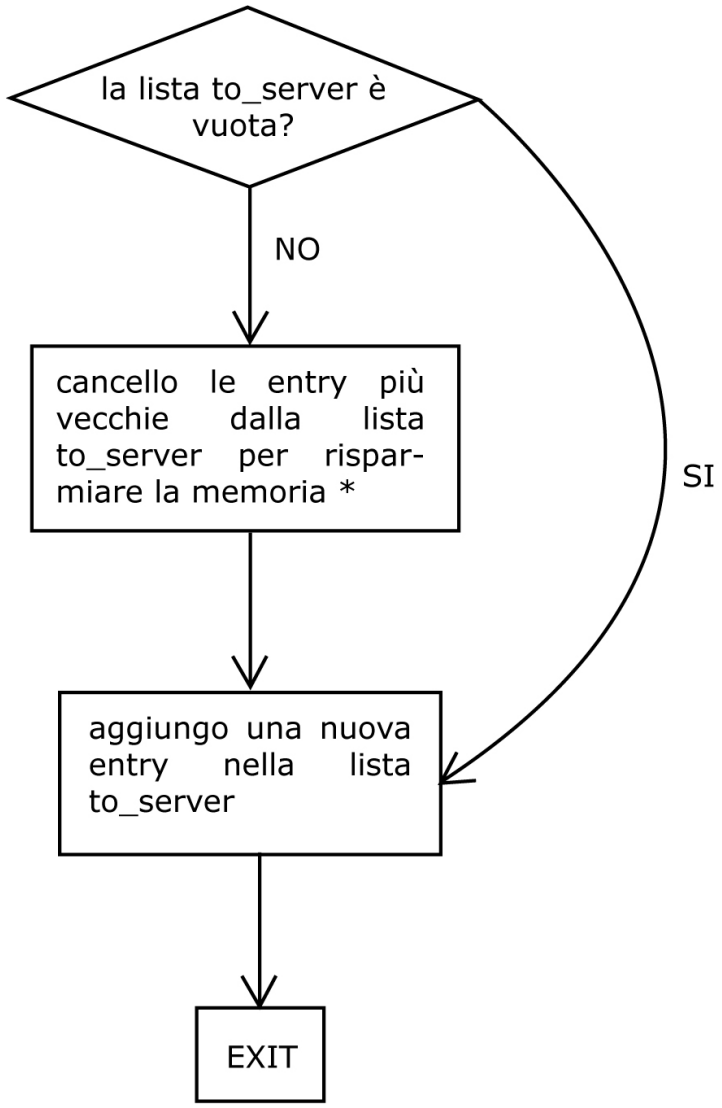


Figura 4.15: Algoritmo di esecuzione di purge.

Le epoche sono di 120 secondi; viene spedito un pacchetto di broadcast al secondo e uno di unicast al secondo verso ogni nodo da cui si é ricevuto almeno un pacchetto di broadcast. Notare che la lunghezza di SEQ é divisore di (FINESTRA/BROADINTERVAL) e di (FINESTRA/UNIINTERVAL). Tale tipo di test puó essere utile per stressare continuamente la rete e vedere come reagisce il protocollo di routing.
Un altro tipo di test continuo un pó meno invasivo si potrebbe costruire aumentando BROADINTERVAL e UNIINTERVAL a 4 in modo da mandare un pacchetto ogni 4 secondi per un totale di 40 pacchetti. Una possibile sequenza della dimensione dei pacchetti potrebbe essere SEQ = 11231.
I test statistici generano meno traffico sulla rete e contengono un livello di dettaglio minore. Si potrebbero classificare come traffico "burst". Sono consigliati per lunghe sessioni di test. Il tool ben configurato per fare test statistici puó essere lasciato girare anche per giorni o settimane sulla rete senza che se ne risenta.

esempio di test statistico
TIMEEPOCA = 120
FINESTRA = 60
BROADINTERVAL = 1
UNIINTERVAL = 1
BROADSLEEP = 10

Quanto detto sinora vale per le temporizzazioni delle sessioni di test sui

nodi. Passiamo ora a determinare alcune categorie di test.
É possibile agire sulla dimensione dei pacchetti e sulla sequenza SEQ per simulare un determinato tipo di traffico. Ad esempio é possibile simulare un traffico di tipo VOIP caratterizzato da pachetti di dimensione intorno ai KB nel modo seguente :

dimensione pacchetti di tipo 1 = KB
BROADINTERVAL = UNIINTERVAL = 0.5
SEQ = 11111111

Con tale approccio si puó simulare sulla rete mesh un qualunque tipo di traffico.

Se si vogliono invece testare le decisioni prese dal protocollo di routing che gira sulla rete e verificare se effettivamente vengono prese le decisioni migliori, l'approccio é un pó diverso e piú complesso. Oltre ai dati che vengono memorizzati dal tool, bisogna avere a disposizione anche il punto di vista del protocollo stesso per poter confrontare i dati. Il valore aggiunto del tool sulle metriche adottate dai protocolli di routing é che effettua test broadcast ma soprattutto unicast sino al livello fisico della pil ISO/OSI. Per eseguire dei test che intendono verificare le scelte prese dal protocollo di routing bisogna calibrare il tool in modo da non essere invasivo sulla rete. Una volta effettuati i test si possono trarre delle conclusioni sull'efficienza delle metriche utilizzate oggi e su cosa si potrebbe fare per renderle ancora piú efficienti.

4.5 rilettura dei dati

Sia sui nodi é possibile impostare 2 livello di debug. Il livello 1 (figura 4.16) stampa ad ogni ricezione di pacchetto inviato dal nodo X tutti i buffer relativi al nodo X dell'epoca corrente. Il livello 2 (figura 4.17) stampa l'output delle funzioni metrics e purge, mostrando quali strutture sono pronte per essere inviate alla prima richiesta get_data del server. Sui router Linksys é peró sconsigliato utilizzare il debug di livello 1, per questioni di memoria.

Sul server é previsto un solo livello di debug. Ad ogni ricezione di pacchetto stampa tutti i dati di tutti i nodi di cui é in possesso, come mostrato in figura 4.18.

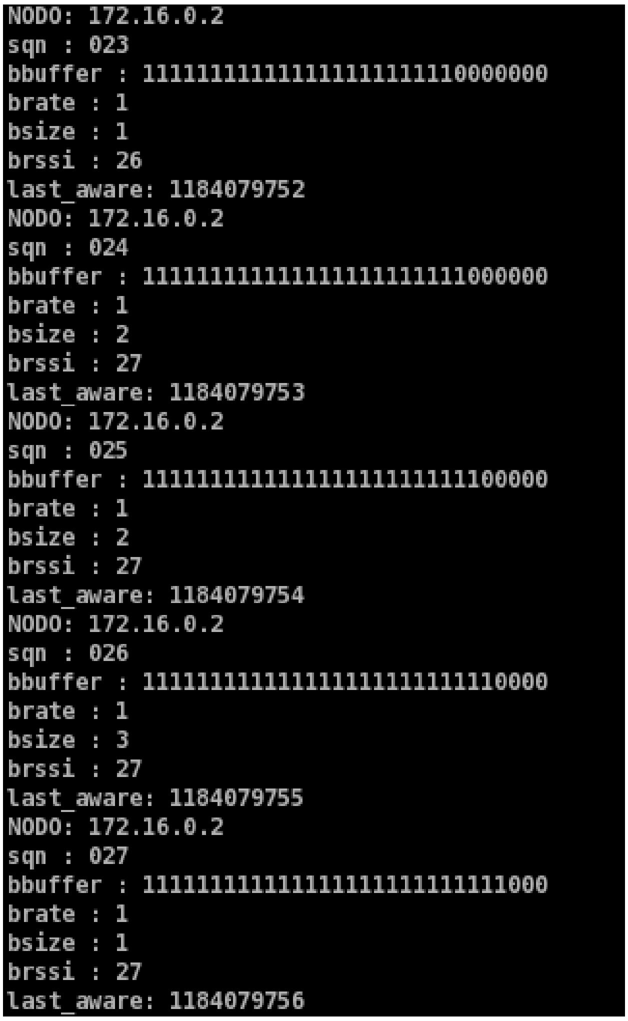


Figura 4.16: Output del livello 1 di debug sui nodi.

Figura 4.17: Output del livello 2 di debug sui nodi.

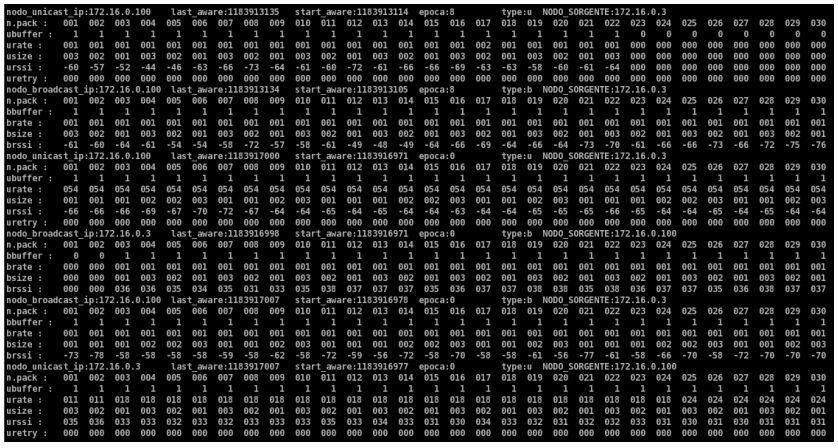


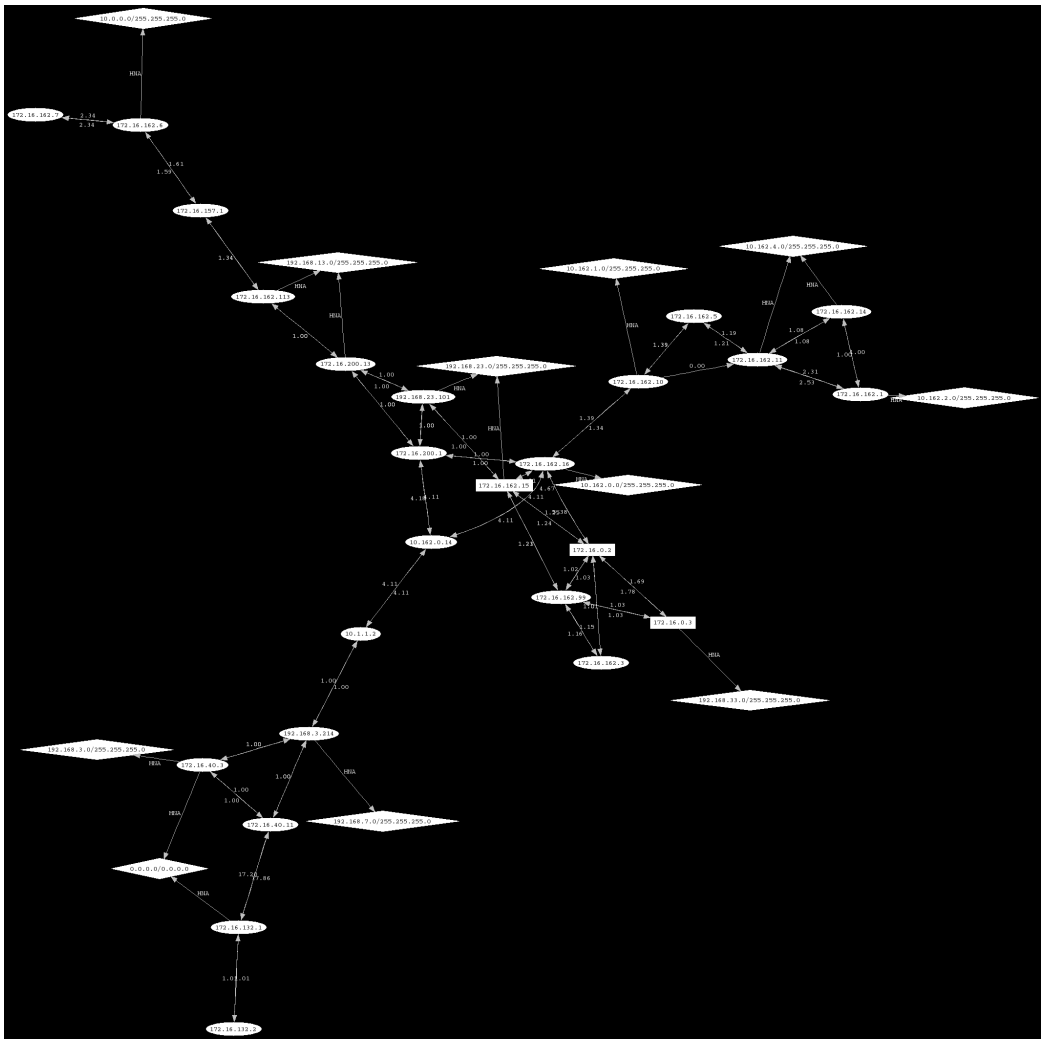
Figura 4.18: Output del debug sul server.

Capitolo 5

Test

5.1 Hardware utilizzato

I test che seguono sono stati effettuati sulla rete di ninux.org, costituita prevalentemente da router Linksys WRT, Asus DW-500, con architettura MIPSEL e portatili.



Sulla tale rete é in esecuzione il protocollo di routing OLSR e i nodi sono posizionati sui tetti delle abitazioni. Ci siamo limitati ad osservare l'andamento dei risultati dei test del TOOL sui singoli link, per poi espandere le valutazioni alle scelte che potevano essere prese da un protocollo di routing mesh. Nei test si é scelto di confrontare i risultati, derivati dai probe di unicast, con un ETX calcolato sui risultati dei probe in broadcast.

Vedremo essenzialmente due approcci differenti ai test: nel primo si distinguono i probe in unicast al solo livello 3 da quelli ai livelli 1-2; nel secondo si distinguono i probe unicast effettuati con pacchetti della stessa dimensione

da quelli effettuati con sequenza di differente dimensione. Per il primo caso proveremo che effettuare i probe al solo livello 3 non cambia di molto la percezione che si ha del link con i semplici probe effettuati in broadcast; al limite puó essere d'aiuto al rilevamento della Gray Zone. Nel secondo caso vedremo che nei test effettuati con invio di paccheti unicast di differente dimensione, si é verificata una maggiore perdita di pacchetti.
Lo scopo di questa serie di test é di mostrare che, integrando le metriche esistenti effettuate solo in broadcast, con probe in unicast di livello 1-2, si potrebbe condizionare il routing spingendolo a preferire i percorsi piú veloci e meno soggetti a ritrasmissioni.
Le valutazioni relative alla velocitá si riferiscono al Data Rate medio nominale, e non al Data Rate effettivo mostrato nel capitolo "Caratterizzazioni". Un Data Rate medio piú elevato, corrisponde ad un link piú veloce; un grado di retry elevato corrisponde ad un link su cui é presente molto rumore. I pacchetti utilizzati per i test sono di 3 dimensioni :

- pacchetti di tipo 1 di 81 byte simulano all'incirca la dimensione utilizzata dai protocolli mesh per effetuare i probe sui link in broadcast;
- pacchetti di tipo 2 di 749 byte servono a simulare un traffico intermedio tra quello di tipo 1 e quello massimo di tipo 3;
- pacchetti i tipo 3 di 1449 byte pari al paccheto massimo trasferibile sul link(MTU).

I test vengono effettutati secondo come illustrato nel capitolo [Rif TOOL], con una FINESTRA di 30 secondi,TIMEPOCA di 60 secondi e UNIINTERVAL = BROADINTERVAL pari ad 1 secondo.
Il TOOL testa il link in entrambi i sensi sia con pacchetti di unicast che con pacchetti di broadcast. Vediamo ora con un esempio come vanno interpretati

i risultati restituiti sul link tra i nodi 172.16.0.2 e 172.16.0.3.
L'output relativo al test del link in broadcast dal nodo 172.16.0.2 al nodo 172.16.0.3 é il seguente:

```
nodo_broadcast_ip:172.16.0.2 ,  last_aware:1183424611 , start_aware:1183424582 , epoca:0 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    0    1    1    0    1    1    1    1    0    0    1    0    1    1    1    1    0    1    0    0    1    1    1    1    0    1    1    1,
brate :    001  001  000  001  001  000  001  001  001  001  000  000  001  000  001  001  001  001  000  001  000  000  001  001  001  001  000  001  001  001,
bsize :    001  001  000  002  002  000  001  001  002  003  000  000  001  000  002  003  001  001  000  003  000  000  001  002  002  003  000  001  002  003,
brssi :    -86  -89  000  -92  -89  000  -93  -90  -91  -90  000  000  -86  000  -88  -86  -86  -88  000  -86  000  000  -90  -85  -88  -86  000  -88  -86  -90,
```

dove *nodo_broadcast_ip* indica il nodo che ha inviato i pacchetti di broadcast e NODO_SORGENTE indica il nodo che li ha ricevuti e inviati al server di memorizzazione. *n.pack* indica il numero di sequenza del pacchetto ricevuto e, allo stesso tempo, identifica univocamente una posizione nei buffer di memorizzazione del nodo ricevente. *bbuffer* é riempito con un 1 nella posizione *n.pack* se il nodo ricevente, 172.16.0.3, ha ricevuto un pacchetto di broadcast con numenro si sequenza pari a *n.pack*. *brate* indica il rate con cui é stato ricevuto il pacchetto con numero di sequenza *n.pack* (per i pacchetti di broadcast sará sempre pari al *basic rate*) . Allo stesso modo *bsize* e *brssi* indicano rispettivamente la dimensione e il livello del segnale del pacchetto ricevuto. *start_aware* e *last_aware* indicano quando il nodo ricevente 172.16.0.3 ha sentito rispettivamente il primo e l'ultimo pacchetto dal nodo 172.16.0.2. *epoca* indica a quale sessione di test si fa riferimento. Per le sessioni successive, verrá incrementato di 1. É un valore utile per la lettura dei dati ed essenziale nella gestione della memorizzazione del traffico di probe sui nodi.
Per quanto riguarda l'output dei probe effettuati in unicast dal nodo 172.16.0.2 al nodo 172.16.0.3, oltre alla semplice rinominazione dei buffer, si puó notare la presenza del buffer *uretry* che determina il numero di ritrasmissioni, percepite sui pacchetti, che il nodo ricevente, 172.16.0.3, ha ricevuto dal nodo mittente 172.16.0.2. Si noti come in questo caso, i valori con cui é riempito il buffer *urate* sono fortemente variabili a seconda della bontá del link.

```
nodo_unicast_ip:172.16.0.2 ,    last_aware:1183424674 , start_aware:1183424645 , epoca:0 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    001  001  001  002  002  002  002  002  002  001  001  001  005  002  002  002  002  002  001  002  002  001  005  005  001  005  001  002  002  001,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
urssi :    -93  -85  -86  -86  -91  -93  -91  -93  -95  -93  -93  -88  -88  -90  -88  -88  -85  -91  -90  -88  -95  -93  -88  -90  -88  -88  -93  -85  -88  -86,
uretry :   006  006  004  000  000  000  002  002  002  002  002  010  000  000  000  002  002  000  008  000  002  002  000  002  008  002  006  000  000  004,
```

Di seguito riportiamo per completezza anche i test effettuati sul link nel verso opposto, dal nodo 172.16.0.3 al nodo 172.16.0.2.

```
nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183424608 , start_aware:1183424578 , epoca:0 ,       type:b , NODO_SORGENTE:172.16.0.2,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    0    0    1    1    1    1    1    1    0    1    0    0    1    0    1    0    0    1    1    1    0    0    0    0    0    0    1    1    1,
brate :    001  000  000  001  001  001  001  001  001  000  001  000  000  001  000  001  000  000  001  001  001  000  000  000  000  000  000  001  001  001,
bsize :    001  000  000  002  002  003  001  001  002  000  001  000  000  002  000  003  000  000  002  003  001  000  000  000  000  000  000  001  002  003,
brssi :    -90  000  000  -86  -86  -86  -85  -86  -88  000  -85  000  000  -93  000  -86  000  000  -86  -90  -90  000  000  000  000  000  000  -90  -88  -91,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183424674 , start_aware:1183424644 , epoca:0 ,       type:u , NODO_SORGENTE:172.16.0.2,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    0    0    1,
urate :    002  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  000  000  001,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  000  000  003,
urssi :    -84  -83  -86  -88  -86  -88  -88  -94  -88  -86  -88  -86  -86  -91  -86  -84  -86  -84  -86  -86  -83  -84  -98  -84  -86  -89  -86  000  000  -93,
uretry :   000  002  000  002  002  000  004  004  004  002  002  002  000  002  000  000  000  004  002  002  002  000  002  000  000  000  002  000  000  002,
```

5.2 Rilevamento della Gray Zone

5.2.1 Condizioni di test

Sono stati presi in considerazione i risultati del TOOL sui nodi 172.16.162.10 e 172.16.162.15 posti su due tetti a circa 1 Km di distanza in linea d'aria. Si ricorda che la Gray Zone si rileva su un link quando il traffico di unicast, forgiato al rate piú alto supportato dalle schede di rete ai due capi del link (in questo caso pari a 54Mbit/sec), non viene percepito. Nel test che segue, si puó vedere che il traffico unicast dal nodo 172.16.162.10 al nodo 172.16.162.15 raggiunge la destinazione, ma con un rate pari al basic rate e un elevato numero di ritrasmissioni. Risulta quindi evidente che i pacchetti unicast forgiati a rate superiori a 1Mbit/sec vengono persi.

5.2.2 Il test - sequenza 11111111111111111111111111111111

```
nodo_broadcast_ip:172.16.162.10 ,       last_aware:1183387195 , start_aware:1183387166 ,epoca:9 ,       type:b ,NODO_SORGENTE:172.16.162.15,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    0    1    0    0    1    1    0    1    0    1    1    1    0    0    1    0    0    0    1    1    0    1    1    1    1    0    1    1    0    1,
brate :    000  001  000  000  001  001  000  001  000  001  001  001  000  000  001  000  000  000  001  001  000  001  001  001  001  000  001  001  000  001,
bsize :    000  001  000  000  001  001  000  001  000  001  001  001  000  000  001  000  000  000  001  001  000  001  001  001  001  000  001  001  000  001,
brssi :    000  -85  000  000  -85  -88  000  -88  000  -88  -88  -88  000  000  -87  000  000  000  -87  -85  000  -87  -86  -86  -85  000  -88  -86  000  -88,

nodo_unicast_ip:172.16.162.10 , last_aware:1183387209 , start_aware:1183387182 ,epoca:0 ,       type:u ,NODO_SORGENTE:172.16.162.15,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
usize :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
urssi :    -87  -85  -84  -85  -87  -84  -85  -87  -88  -88  -85  -85  -90  -86  -88  -85  -86  -86  -85  -85  -85  -88  -88  -88  -86  -83  -82  -86  -86  -86,
uretry :   005  006  004  002  004  004  004  006  008  008  008  004  004  006  004  004  006  008  002  008  002  008  006  006  004  008  010  006  004  000,
```

5.2.3 Risultati

Si puó notare che gli unici pacchetti di unicast che arrivano a destinazione sono quelli forgiati ad 1Mbit/sec e si riscontra un elevato numero di retry. Inoltre il numero di pacchetti di broadcast ricevuti é inferiore a quello dei pacchetti di unicast, poiché questi ultimi godono del meccanismo delle ritrasmissioni.

5.3 Probing unicast stessa dimensione livello 3

5.3.1 Condizioni di test

I nodi sono il 172.16.0.100 posto nella stanza di un'abitazione al primo piano e il 172.16.0.3 posto sul tetto. Effettueremo una serie di probe inviando tre sequenze di pacchetti della stessa dimensione, avvalendoci solamente dei dati di livello 3. Proveremo prima con una sequenza di pacchetti di tipo 1, poi con una di pacchetti di tipo 2, infine con una di pachetti di tipo 3. Ci aspettiamo una maggiore perdita, seppur leggera, di pacchetti di dimensione maggiore.

5.3.2 Il test

Dimensione 1 (sequenza : 1111111111111111111111111111111111111111):

```
nodo_broadcast_ip:172.16.0.100 , last_aware:1183911620 , start_aware:1183911591 , epoca:1 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
brssi :    -79  -71  -63  -65  -69  -63  -72  -71  -67  -69  -69  -71  -65  -63  -62  -65  -63  -62  -63  -65  -69  -65  -69  -67  -63  -74  -69  -74  -72  -74,

nodo_unicast_ip:172.16.0.100 ,  last_aware:1183911630 , start_aware:1183911600 , epoca:1 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    024  036  036  036  036  054  036  036  036  036  036  036  036  054  036  036  036  036  036  036  036  036  036  036  036  036  036  036  054  036,
usize :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
urssi :    -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -69  -76  -76  -76  -80  -76  -74  -70  -67  -69  -69  -69  -69,
uretry :   000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  002  000  000  000  000  000  000  000  000,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183911620 , start_aware:1183911591 , epoca:1 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    024  024  024  024  024  024  024  024  024  024  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036  036,
usize :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
urssi :    029  027  031  028  027  027  028  028  027  028  028  026  028  028  027  027  029  028  027  029  029  028  027  028  028  028  027  022  025  025,
uretry :   000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000,

nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183911630 , start_aware:1183911600 , epoca:1 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
brssi :    031  031  031  033  032  031  032  032  032  032  033  032  032  032  033  033  031  032  028  025  024  024  028  027  032  033  033  033  034  034,
```

Dimensione 2 (sequenza : 22222222222222222222222222222222222222):

```
nodo_broadcast_ip:172.16.0.100 , last_aware:1183910543 , start_aware:1183910513 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  000  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  000  002  002  002  002  002  002  002  002  002  002  002  002  002  002,
brssi :    -76  -74  -74  -74  -87  -84  -87  -84  -79  -90  -72  -82  -82  -76  -80  000  -77  -76  -77  -72  -80  -82  -80  -85  -82  -80  -87  -87  -85  -80,

nodo_unicast_ip:172.16.0.100 ,  last_aware:1183910543 , start_aware:1183910514 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    011  012  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  012  011  011  011  011  011  011  011  011  011  011,
usize :    002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002,
urssi :    -71  -79  -72  -74  -87  -77  -76  -80  -79  -84  -82  -82  -85  -80  -84  -82  -82  -82  -87  -79  -79  -77  -88  -82  -84  -82  -82  -84  -80  -80,
uretry :   000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  002  000  000  000  000  000  000  000  000  000  000  000  000  000,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183910543 , start_aware:1183910513 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    011  018  018  018  005  011  011  011  011  011  011  011  011  011  011  011  011  011  018  018  018  018  005  011  011  011  011  011  011  011,
usize :    002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002,
urssi :    024  019  020  019  013  019  022  021  021  013  022  013  018  019  016  023  022  018  017  017  009  009  010  008  010  009  010  010  012  011,
uretry :   000  000  000  000  002  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  002  000  000  000  000  000  000  000  000,

nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183910543 , start_aware:1183910513 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002  002,
brssi :    023  021  022  023  012  019  020  023  020  014  022  014  018  022  016  023  023  020  021  021  014  012  009  010  006  011  010  011  011  012,
```

Dimensione 3 (sequenza : 333333333333333333333333333333333333333):

```
nodo_broadcast_ip:172.16.0.100 , last_aware:1183909521 , start_aware:1183909495 , epoca:1 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    0    0    1    1    1    1    0    0    0    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    0    1    1    1    1    0,
brate :    000  000  001  001  001  001  000  000  000  001  001  001  001  001  001  001  001  001  000  001  001  001  001  001  000  001  001  001  001  000,
bsize :    000  000  003  003  003  003  000  000  000  003  003  003  003  003  003  003  003  003  000  003  003  003  003  003  000  003  003  003  003  000,
brssi :    000  000  -92  -92  -92  -94  000  000  000  -90  -89  -89  -87  -90  -92  -92  -89  -92  000  -92  -90  -90  -87  -86  000  -94  -94  -87  -89  000,

nodo_unicast_ip:172.16.0.100 ,  last_aware:1183909517 , start_aware:1183909488 , epoca:1 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    0    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    1,
urate :    001  000  001  005  001  001  001  005  001  001  001  000  005  005  005  005  005  005  005  005  005  005  001  000  005  005  005  005  005  005,
usize :    003  000  003  003  003  003  003  003  003  003  003  000  003  003  003  003  003  003  003  003  003  003  003  000  003  003  003  003  003  003,
urssi :    -92  000  -90  -90  -94  -90  -90  -92  -94  -90  -92  000  -90  -87  -86  -89  -92  -87  -90  -89  -89  -90  -94  000  -94  -87  -89  -89  -84  -90,
uretry :   002  000  002  002  002  004  002  000  004  002  002  000  000  000  000  002  000  000  000  000  002  000  002  000  000  000  002  000  000  002,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183909524 , start_aware:1183909495 , epoca:1 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    0    0    0    1,
urate :    001  001  001  001  000  001  001  002  002  002  002  002  002  002  005  001  005  005  005  005  000  005  005  001  002  001  000  000  000  001,
usize :    003  003  003  003  000  003  003  003  003  003  003  003  003  003  003  003  003  003  003  003  000  003  003  003  003  003  000  000  000  003,
urssi :    008  007  007  005  000  009  008  007  007  003  004  007  008  006  005  005  006  007  008  007  000  007  011  005  000  001  000  000  000  001,
uretry :   001  000  002  000  000  000  000  000  000  000  000  000  000  002  000  004  002  000  002  000  000  002  000  002  000  002  000  000  000  000,

nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183909517 , start_aware:1183909488 , epoca:1 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    0    0    0    0    0    0    0    0    1    1    0    1    1    1    1    1    1    1    1    1    1    1    0    1    1    1    1    1    1,
brate :    001  000  000  000  000  000  000  000  000  001  001  000  001  001  001  001  001  001  001  001  001  001  001  000  001  001  001  001  001  001,
bsize :    003  000  000  000  000  000  000  000  000  003  003  000  003  003  003  003  003  003  003  003  003  003  003  000  003  003  003  003  003  003,
brssi :    -01  000  000  000  000  000  000  000  000  005  006  000  009  007  008  008  007  003  007  007  004  006  006  000  005  007  007  007  008  008,
```

5.3.3 Risultati

Con questo semplice test non si fa altro che riscontrare che caricare un link con pacchetti pari al MTU porta ad un leggero deterioramente della qualitá del link. Si potrebbe pensare di integrare le metriche mesh con l'invio di pacchetti broadcast di dimensione differente per rilevare i link che soffrono di piú con pacchetti di grosse dimensioni. Non avrebbe senso infatti prevedere tale meccanismo con probe unicast, che stravolgerebbe completamente le metriche attuali, poiché, osservando i dati al solo livello 3, non si hanno dati relativi ai livelli inferiori della pila ISO/OSI.

5.4 Probing unicast dimensione diversa livello 3

5.4.1 Condizioni di test

Procediamo ancora con i test di livello 3 inviando una sequenza arbitraria di pacchetti di diversa dimensione al fine di simulare un traffico arbitrario sul

link tra 172.16.0.100 e 172.16.0.3. Ci si aspetta che i pacchetti piú piccoli arrivino a destinazione piú frequentemente di quelli grandi.

5.4.2 Il test - sequenza : 3213213213213213213213213213213

```
nodo_unicast_ip:172.16.0.100 ,  last_aware:1183912910 , start_aware:1183912881 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
usize :    003  002  001  003  000  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001,


nodo_broadcast_ip:172.16.0.100 ,last_aware:1183912887 , start_aware:1183912880 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    1    0    1    0    1    1,
bsize :    000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  002  000  003  000  001  000  002  001,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183912909 , start_aware:1183912880 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
usize :    003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001,

nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183912949 , start_aware:1183912921 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    0    1    1    1    1    1    0    0    1    0    0    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
bsize :    000  002  001  003  002  001  000  000  001  000  000  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001,
```

5.4.3 Risultati

Procedendo alla considerazione dei soli dati al livello3 si osserva che vengono persi moltissimi pacchetti dal nodo 172.16.0.100 al nodo 172.16.0.3, infatti il link é caratterizzato da un ETX elevato. Se peró si osserva l'andamento dei pacchetti di unicast, si nota che sono giunti quasi tutti a destinazione. Nel verso contrario la differenza tra pacchetti broadcast e unicast ricevuti é fortemente attenuata. I protocolli di routing mesh, tenendo conto dei soli pacchetti di broadcast, considerano il link come fortemente asimmetrico, quando invece é comunque possibile instaurare una trasmissione in entrambi i versi.

5.5 Probing unicast dimensione diversa livelli 1 e 2

5.5.1 Condizioni di test

Ripetiamo il test “Probing unicast stessa dimensione livello 3” ma osservando i dati anche ai livelli 1 e 2. Ci aspettiamo di avere una maggiore informazione sull’effettivo andamento del link nelle trasmissioni unicast. Per questo ci concentreremo sul numero di retry e il data rate medio.

5.5.2 Il test - sequenza : 32132132132132132132132132132132

nodo_unicast_ip:172.16.0.100 , last_aware:1183912910 , start_aware:1183912881 , epoca:4 , type:u , NODO_SORGENTE:172.16.0.3,

n.pack :	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030,
ubuffer :	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1,
urate :	002	001	001	001	000	001	001	001	001	001	001	001	001	001	001	001	001	005	001	001	001	001	001	001	001	001	001	001	001	001,
usize :	003	002	001	003	000	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001,
urssi :	-89	-95	-92	-92	000	-94	-92	-94	-92	-90	-92	-90	-92	-94	-90	-94	-89	-92	-92	-92	-94	-94	-92	-93	-90	-92	-93	-93	-90	-90,
uretry :	002	004	002	002	000	002	002	002	002	002	002	002	000	002	002	000	002	002	000	002	002	002	002	002	000	000	000	000	000	000,

nodo_broadcast_ip:172.16.0.100 , last_aware:1183912887 , start_aware:1183912858 , epoca:4 , type:b , NODO_SORGENTE:172.16.0.3,

n.pack :	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030,
bbuffer :	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1,
brate :	001	001	000	001	001	001	001	001	001	000	001	001	001	001	001	001	000	001	001	001	001	001	001	001	000	000	001	001	001	001,
bsize :	003	002	000	003	002	001	003	002	001	000	002	001	003	002	001	000	000	001	003	002	001	003	002	001	000	000	001	003	002	001,
brssi :	003	004	000	003	002	004	006	005	003	000	004	005	006	004	003	000	000	005	003	005	005	006	005	004	000	000	003	005	006	004,

nodo_unicast_ip:172.16.0.3 , last_aware:1183912909 , start_aware:1183912880 , epoca:4 , type:u , NODO_SORGENTE:172.16.0.100,

n.pack :	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030,
ubuffer :	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1,
urate :	001	001	001	001	001	001	001	001	001	001	001	001	001	002	002	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001,
usize :	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001,
urssi :	004	-01	006	005	004	002	001	002	004	-01	005	005	002	003	003	003	003	005	003	004	003	003	003	005	004	004	006	004	002	005,
uretry :	000	000	000	002	002	000	000	002	000	000	000	004	002	002	002	002	004	000	000	000	002	000	000	000	000	000	000	000	000	000,

nodo_broadcast_ip:172.16.0.3 , last_aware:1183912949 , start_aware:1183912921 , epoca:4 , type:b , NODO_SORGENTE:172.16.0.100,

n.pack :	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030,
bbuffer :	0	1	1	1	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1,
brate :	000	001	001	001	001	001	000	000	001	000	000	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001	001,
bsize :	000	002	001	003	002	001	000	000	001	000	000	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001	003	002	001,
brssi :	000	004	003	006	002	005	000	000	003	000	000	005	005	005	004	003	006	005	004	004	004	005	006	003	004	007	006	004	006	006,

5.5.3 Risultati

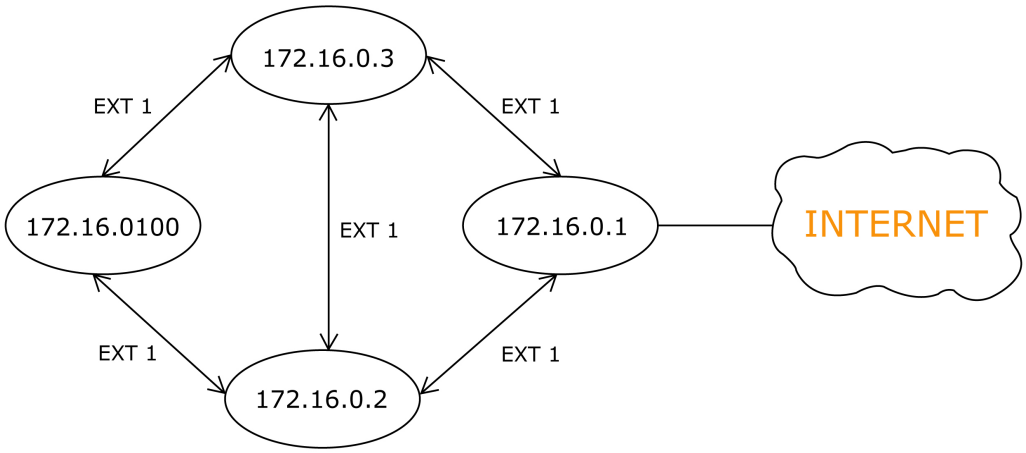
Si puó notare che c’é una grande differenza osservando i dati al solo livello 3 o ai livelli 1 e 2. Osservando solo il livello 3 sembra che il link sia discreto poiché si perde un solo pachetto di unicast dal nodo 172.16.0.100 al nodo 172.16.0.3. Guardando peró anche il numero di retry e il data rare medio, si evince che il link in questione non é effetivamente buono come sembra. Infatti si sono verificati 42 retry e un data rate medio pari a 1.13 Mb/sec. Se

una metrica tenesse conto di tali informazioni, potrebbe indurre il protocollo a preferire i link con minor numero di retry e data rate prossimo a quello massimo.

5.6 Probing unicast dimensione diversa livelli 1 e 2 su 2 link uguali a livello 3 broadcast e unicast

5.6.1 Condizioni di test

Con questo ultimo test vogliamo mostrare come due link, apparentemente uguali a livello 3 (stesso numero di pacchetti broadcast), sono differenti se osservati ai livelli 1 e 2. Si é ricreato un testbed cosituito da soli 4 nodi equipaggiati con schede di rete in grado di trasmettere fino a 54Mbit/sec. Vedremo come i link tra il nodo 172.16.0.100 e i nodi 172.16.0.2 e 172.16.0.3, a paritá di pachetti ricevuti in broadcast e unicast, differiscono per data rate medio e numero di retry. La tolpologia della rete su cui si é effettuato il test é la seguente :



Sulla rete gira OLSR e si puó notare che l'ETX é uguale su tutti i link, indicando che i link sono perfetti (ETX = 1.0). Il nodo 172.16.0.1 é il gateway verso internet per tutta la rete mesh. Le osservazioni che faremo di seguito sono relative al path che il nodo 172.16.0.100 sceglierá per raggiungere internet. Inizialmente si é tenuto spento il nodo 172.16.0.2 per fare in modo che il nodo 172.16.0.100 scegliesse il nodo 172.16.0.3 come gateway, poiché é il primo ad aver raggiunto l'ETX massimo. Una volta acceso il nodo 172.16.0.2, anche questo link ha raggiunto il valore di ETX pari ad 1.
Ci concentreremo sui link tra il nodo 172.16.0.100 e i nodi 172.16.0.2 e 172.16.0.3 e osserveremo i risultati del TOOL su tali link. Vogliamo vedere se, avendo a disposizione informazioni al livello 2, é possibile condizionare il routing a prendere decisioni migliori.

5.6.2 Il test - sequenza : 1111122233111112223311111222 33

```
nodo_broadcast_ip:172.16.0.100 , last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
brssi :    -79  -79  -79  -79  -80  -74  -79  -76  -77  -76  -76  -77  -79  -80  -77  -76  -79  -77  -77  -76  -82  -82  -82  -74  -76  -74  -76  -77  -80  -74,

nodo_unicast_ip:172.16.0.100 ,  last_aware:1183900758 , start_aware:1183900729 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.3,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011  011,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
urssi :    -74  -77  -76  -77  -76  -76  -74  -77  -77  -77  -79  -79  -79  -77  -77  -79  -76  -76  -79  -77  -77  -77  -82  -80  -72  -71  -69  -72  -77  -76,
uretry :   000  004  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  000  004  000  000  000  000  000  000,
rate medio =  11 Mbit/sec
retry = 8


nodo_broadcast_ip:172.16.0.3 ,  last_aware:1183900758 , start_aware:1183900729 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
brssi :    019  018  019  017  017  018  019  018  019  018  020  018  018  018  019  019  017  018  019  019  019  019  015  020  020  020  019  020  019  018,

nodo_unicast_ip:172.16.0.3 ,    last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    011  011  011  011  011  011  011  011  011  018  005  005  005  011  011  011  011  011  011  011  011  011  011  011  018  005  005  005  011  011,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
urssi :    018  018  018  019  018  019  019  019  020  014  017  018  018  017  018  019  019  018  018  019  015  018  020  021  014  019  018  018  017  022,
uretry :   000  000  000  000  000  000  000  000  000  000  002  002  002  000  000  000  000  000  000  000  000  000  000  000  000  002  002  002  000  000,
rate medio = 10.2 Mbit/sec
retry = 12
```

```
nodo_broadcast_ip:172.16.0.2 ,  last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  002  002  003  001  001  002  003  001  001  001  002  002  003  001  001  002  003  001  001  001  002  002  003  001  001  002  003,
brssi :    031  032  030  028  031  035  030  034  035  026  029  031  023  025  023  020  025  022  023  023  023  023  022  023  025  024  027  023  024  024,

nodo_unicast_ip:172.16.0.2 ,    last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.100,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    0    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1,
urate :    054  054  054  054  054  054  054  054  054  054  054  054  054  054  000  024  036  036  036  000  024  024  024  024  024  024  024  024  024  024,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  000  002  002  002  003  000  001  001  001  001  001  002  002  002  003  003,
urssi :    030  027  028  027  030  035  026  029  035  022  026  027  022  022  000  017  021  018  021  000  019  021  022  020  021  022  022  022  022  020,
uretry :   000  000  000  000  000  000  000  000  000  002  000  000  002  002  000  002  002  000  002  000  002  000  002  000  000  000  000  000  000  000,
rate medio = 37.6 Mbit/sec
retry = 16


nodo_broadcast_ip:172.16.0.100 , last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:b , NODO_SORGENTE:172.16.0.2,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
bbuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
brate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  001,
bsize :    001  001  001  002  002  003  001  001  002  003  001  001  001  002  002  003  001  001  002  003  001  001  001  002  002  003  001  001  002  003,
brssi :    -66  -63  -66  -66  -69  -74  -78  -71  -69  -66  -76  -65  -69  -84  -79  -81  -80  -75  -80  -73  -72  -73  -72  -70  -73  -82  -80  -77  -82  -78,

nodo_unicast_ip:172.16.0.100 ,  last_aware:1183900757 , start_aware:1183900728 , epoca:4 ,       type:u , NODO_SORGENTE:172.16.0.2,
n.pack :   001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030,
ubuffer :    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1,
urate :    054  054  054  054  054  054  048  048  054  054  048  048  048  036  036  001  024  006  018  036  006  048  048  048  048  048  048  048  001  054,
usize :    001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003  001  001  001  001  001  002  002  002  003  003,
urssi :    -70  -70  -70  -71  -69  -61  -72  -73  -74  -79  -75  -70  -82  -81  -80  -78  -79  -82  -78  -83  -80  -76  -79  -79  -79  -78  -71  -76  -67  -79,
uretry :   000  000  000  000  000  000  000  000  000  002  000  000  000  000  000  010  000  000  000  002  000  000  000  000  002  000  004  002  004  002,
rate medio = 30 Mbit/sec
retry = 28
```

5.6.3 Risultati

Osservando solamente i risultati del test effetuati in broadcast, si nota che non si é verificata nemmeno una perdita di pachetti broadcast, il che conferma la percezione che ha OLSR tramite ETX della bontá dei link. Guardando peró i risultati dei probe di unicast, é evidente che il link tra 172.16.0.3 e 172.16.0.100 é piú lento anche se presenta un minor numero di retry. Se si condizionasse il routing di OLSR basandolo anche sul traffico unicast ai livelli 1 e 2, il nodo 172.16.0.100 sceglerebbe il nodo 172.16.0.2, piú veloce, come next hop verso il gateway.

I protocolli di routing mesh non tengono conto delle informazioni derivate da probe unicast e dai livelli piú bassi della pila ISO/OSI, limitandosi al solo traffico broadcast. Il test fatto dimostra che puó essere utile integrare un meccanismo di probe unicast in grado di fornire maggiori informazioni nella costruzione delle metriche sui link.

Capitolo 6

Conlusioni

Con questo lavoro si é voluto porre le basi per la costruzione di uno strumento di supporto allo studio e allo sviluppo delle reti mesh. Il nome stesso TOOL, attrezzo, vuole rendere l'idea di un oggetto in grado di operare sulla rete, per "aggiustare" la percezione dello stato dei link. Si é pervenuti alla costruzione di questo strumento in seguito allo studio e all'osservazione del comportamento dei protocolli di routing mesh quali OLSR e B.A.T.M.A.N. . Tali protocolli costruisono le rotte sulla rete basandosi unicamente su pacchetti di broadcast della stessa dimensione, per natura molto differenti da quelli di unicast. Basarsi semplicemente sui pacchetti di broadcast, puó portare il protocollo alla scelta di strade sulle quali il traffico unicast non é possibile (Gray Zone); d'altro canto tale metodo puó portare a non considerare quelle strade dove il traffico broadcast non é ottimo, contrariamente a quello unicast che si avvale del meccanismo delle ritrasmissioni. Per contro osservare i pacchetti di unicast ai livelli Data Link e Fisico della pila ISO/OSI, apporta un grado di informazione ulteriore finora non considerato/valutato; in particolare conoscere il data rate medio con cui é possibile trasmettere su di un link pacchetti di unicast, puó portare il protocollo a distinguere due link che

al solo livello di Rete broadcast sembrerebbero uguali. Sempre dai livelli piú bassi della pila ISO/OSI si puó osservare il numero di ritrasmissioni che un nodo ha effettuato per inviare uno stesso pacchetto: piú alto é il suo valore, minore è la qualità del link.
Il contenuto innovativo del TOOL consiste nel testare l'andamento dei link fra i nodi su cui é in esecuzione, effettuando una serie di probe non solo broadcast ma anche unicast; inoltre il TOOL é caratterizzato da un'alto dinamismo, permettendo di essere modificato in tutti i suoi parametri, tra cui elenchiamo i piú significativi :

- l'intervallo di invio dei pacchetti di unicast;
- l'intervallo di teinvio dei pacchetti di broadcast;
- la sequenza di pacchetti di differente dimensione con cui effettuare i probe;
- la durata di una singola sessione di probe tra due nodi;
- il livello della pila ISO/OSI di riferimento per i probe.

Al fine di non sovraccaricare le CPU e saturare le memorie dei router su cui é in esecuzione, è possibile selezionare il numero massimo di sessioni di probe, (EPOCHE), che possono essere memorizzate su un nodo per ogni suo nodo vicino,
Oltre alla parte attiva che agisce sui nodi, é stato previsto un server di memorizzazione, che puó essere un nodo stesso della rete mesh o meno, con il compito di richiedere e memorizzare i dati in possesso dei nodi che partecipano ai test. E' compito di detto server di inoltrare ai nodi della rete mesh i parametri secondo cui effettuare i probe.

E' previsto anche un server di analisi. Quest'ultimo agisce sui dati memorizzati dal server di memorizzazione, incrociandoli, osservandone l'andamento temporale, eseguendo query definite in fase di progettazione. Il server di analisi può eseguire in modo automatico l'analisi dei dati secondo parametri presettati ed é possibile popolarlo con query definite a tempo reale dall'utente.
Per agevolare l'interazione tra utente e server di analisi é previsto un server di visualizzazione, non ancora implementato.

In conclusione il TOOL si propone non solo come strumento di monitoring on-line del comportamento dei link mesh ma anche come supporto nella costruzione di una metrica, adattabile, alle diverse realtá delle reti mesh.Il metodo con cui il TOOL stesso esegue i probe sui nodi, potrebbe essere applicato come metrica per i protocolli di routing mesh.
Effettuare probe in unicast, come mostrato nel capitolo 5 relativo ai test, in aggiunta a quelli di broadcast, puó apportare un ulteriore grado di informazione per la costruzione di una nuova metrica. É possibile inoltre ipotizzare di aggiungere un grado di memoria sull'andamento di un singolo link nel tempo, penalizzando i link che fluttuano maggiormente. Una metrica siffatta avrebbe il problema di generare eccessivo traffico di controllo sulla rete. Per evitare di generare tale overhead, si puó allargare l'intervallo di invio dei pacchetti di unicast. Uno dei problemi strutturali su cui ci si é scontrati nella realizzazione del TOOL, é dovuto alla eterogeneitá dei driver delle schede di rete. Allo stato attuale di svilppo, si puó godere dei livelli 1 e 2 della pila ISO/OSI solo per le schede che implemetano il PRISM HEADER. Per tutte le altre schede é previsto un adattamento che lavora al solo livello di Rete. In futuro si provvederá alla portabilitá anche su schede con RADIO TAP

HEADER. C'é da dire che per la modularitá con cui é stato sviluppato, aggiungere la compatibilitá per schede con driver differenti, non comporterebbe lo stravolgimento della progettazione del TOOL.

6.1 TODO

Di seguito vengono mostrate alcune parti del TOOL che, per motivi di tempo, non sono state implementate o che sono rimaste incomplete :

- Il "porting" su schede che implementano il RADIO TAP HEADER. Si tratta di "aggiustare" gli *offset* con cui vengono prese le informazioni dai livelli Data Link e Fisico dei pacchetti sniffati;
- L'implementazione di un server di analisi che preveda delle query prestabilite (come rilevamento di cicli, andamento di un link, raggiungibilitá di un nodo, cammini minimi, albero ricoprente, ...), e allo stesso tempo consenta all'utente di stabilirne delle altre;
- Il server di visualizzazione per ora é rappresentato da un server web Apache, con pagine scritte in PHP. Permette di visualizzare i risultati dei probe relativi ad un link per volta in formato testuale. Si prevede peró di renderlo molto simile allo stile di visualizzazione proprio di BGPlay;
- Si prevede inoltre di equipaggiare i nodi della rete mesh con dispositivi GPS, al fine di ricostruire una mappa della rete realistica;
- É stata pensata anche una possibile integrazione con Mapserver, motore open source di georeferenziazione, che si avvale del set di strumenti propri del PostGIS.

6.2 Ringraziamenti

Elenco delle figure

Elenco delle tabelle