A Carlo

Università degli Studi "Roma Tre"

Facoltà di Ingegneria

Metodi e strumenti per l'analisi delle metriche nelle Wireless Mesh Network: probe basati su pacchetti *broadcast*

Tesi di Laurea Vecchio Ordinamento in

Ingegneria Informatica

Relatore

Prof. Giuseppe Di Battista

Candidato

Antonino Ciurleo

Anno Accademico 2006/2007

Indice

Sommario

La Topologia delle reti *Mesh*,sotto l'unico vincolo che i nodi possano riceversi reciprocamente a due a due, è l'unica in grado di sopravvivere di fronte alla caduta di alcuni suoi nodi. Essa consente di trovare percorsi alternativi che assicurano continuità di connessione anche in presenza di mobilità o di eventi inprovvisi.

Le reti *Wireless Ad-Hoc* consistono essenzialmente in un insieme di dispositivi instradatori (*router*) o di computer (genericamente denominati *nodi*), collegati uno all'altro in modalità paritaria (*ad-hoc*), capaci di comunicare tra loro e fornire i consueti servizi di rete. Quando i collegamenti assumono una configurazione a maglia, le reti vengono anche indicate con il l'espressione di reti *Mesh*, appunto *magliate*. Questo nuovo tipo di connettività senza cavi libera i nodi della rete dal vincolo di far capo ad una o più entità centrali predeterminate per poter comunicare tra loro. Si rendono quindi possibili topologie e configurazioni adattabili alle varie esigenze e situazioni comprese quelle che prevedono la mobilità dei nodi. Offrono condizioni di ridondanza delle risorse e possibilità di collegamenti in condizioni di criticità. Va da sé che un punto cruciale diviene il problema di elaborare schemi di collegamento ed instradamento (*routing)* dei dati efficienti e soprattutto adatti a questo tipo di reti. I software necessari per la gestione dei collegamenti (*protocolli di routing*) sono diversi dai tradizionali protocolli di routing validi per le

reti cablate: essi infatti devono essere in grado di stabilire le rotte in modo rapido -nel giro di pochi secondi- per poter far fronte tempestivamente ai cambiamenti della topolgia e operare scelte di itinerari ottimali tra il nodo sorgente del messaggio ed il nodo destinatario. A seconda delle destinazioni da raggiungere devono quindi essere in grado di effettuare un'immediata valutazione di elementi di varia natura che caratterizzano le connessioni *(link)* e di conseguenza decidere la scelta degli itinerari più opportuni. Tali valutazioni assumono perciò rilevante importanza ai fini dell'ottimizzazione dei percorsi e della riduzione del traffico di controllo, aspetti questi di rilievo fondamentale sotto il profilo della funzionalità e dell'efficienza di una rete wireless complessa. Allo stato attuale, nonostante la loro decisiva importanza, le valutazioni appaiono piuttosto approssimative. Anche se per certi tipi di comunicazione le approssimazioni possono considerarsi tollerabili, per altri servizi è invece auspicabile la rilevazione di dati più precisa e significativa. Comunque, indipendentemente dal tipo di servizio che si vuole implementare sulla rete, lo strumento *(tool)* di lavoro da noi elaborato, consente di effettuare una valutazione più fedele al reale stato della rete di quanto non facciano gli attuali protocolli; ciò significa che il tool può funzionare da supporto, sia all'adozione e alla valutazione di un protocollo di comunicazione tra quelli esistenti, sia allo sviluppo di protocolli futuri.

Capitolo 1

Introduzione

Una rete di computer, che per comodità chiamiamo tradizionale, ordinariamente realizzata o mediante connessioni cablate o mediante collegamenti radio *(wireless)* tra vari dispositivi è caratterizzata da una topologia predeterminata e rigida: anche nelle reti wireless in cui non si è vincolati dalla cablatura nella costruzione della topologia di rete, i collegamenti sono decisi nel momento della progettazione e nel dimensionamento della rete .

Nelle reti di cui ci si occuperà, le *Mesh Network*, invece, la topologia è decisa dinamicamente in fase operativa da un'entità software (*routing*). L'unico vincolo topologico nelle *Mesh Network*, è costituito dall'effettiva possibilità dei nodi di comunicare. Quindi è chiaro come una condizione essenziale perché l'insieme dei dispositivi costituisca una rete Mesh è che esista un processo software (*routing*) implementato sui nodi, capace di stabilire un percorso che colleghi ciascun nodo con qualsiasi altro nodo per la trasmissione dei messaggi.

Nelle reti cablate la qualità dei collegamenti tra i nodi è indipendente dal tempo, sia in relazione alla stabilità perché la bontà del segnale si mantiene costante, che riguardo alla velocità di trasmissione dei dati. Quest'ultima

infatti può bensì dipendere dal tipo di connessione fisica (cavo o fibra ottica, fili di rame, ecc.), ma non dipende né dal tempo, come si accennato, né dal verso del percorso origine-destinazione dei messaggi. D'altra parte, i link cablati tra i router sono governati, per così dire, dalla logica classica a due valori: il collegamento c'è o non c'è, e non si dà mai il caso di collegamenti parziali.

A differenza dalla prime, le reti wireless possono presentare due diversi tipi di configurazione: la configurazione *strutturata* e la configurazione *ad-hoc*. La prima impone una topologia in cui i vari calcolatori da collegare fanno capo ad un dispositivo rappresentato dalla stazione base o *access point* ed ognuna delle macchine è in certo senso dipendente da essa ed è solitamente denominata *client*. Non tratteremo oltre questa tipologia in quanto esula dall'ambito di questo lavoro.

La seconda configurazione è caratterizzata non più dalla presenza di un'unità centrale di riferimento, bensì da collegamenti *paritari*, senza alcuna implicazione gerarchica tra i vari nodi.

In realtà, anche se viene scelta per la costruzione di reti *Mesh*,la modalità usata non è stata concepita perché soddisfacesse questo tipo di esigenza: la modalità *ad-hoc* o *IBSS* del proocollo 802.11 è stata concepita per emulare una connessione tra computer in modo paritario proprio in sostituzione di un cavo ethernet *crossover*. Nella sua essenza, infatti, presenta notevoli limitazioni nelle performance e nell'ottimizzazione intesa come occupazione delle risorse radio. Nonostante le sue limitazioni prestazionali, meglio delle altre modalità, si presta alla flessibilità di una rete non gerarchica. Nelle reti ad-hoc i nodi possono comunicare direttamente solo con i nodi raggiungibili direttamente (con cui riescano ad estaurare una connessione fisica). Queste reti, in un certo senso, sono anch'esse caratterizzate da rigidità, nel senso

sopra specificato che i salti (*hop*) tra un nodo e l'altro sono predeterminati dalla *fattibilità* fisica (radio) del collegamento.

1.1 Le reti wireless

Si introduce qui il concetto di Stabilià. Nelle reti cablate, sopra accennate, il concetto di stabilità quasi non compare, un collegamento realizzato con una tecnologia basata su cavo non presenta fluttuazioni delle performance e della capacità del canale , a parte in rari casi di rottura. I collegamenti tra i nodi adiacenti nelle reti *wireless*, a differenza che nelle reti cablate, quindi, risultano instabili per costituzone, i collegamenti wireless (*link*) convivono con la fluttazione di molteplici parametri; l'instabilità deriva quindi da diversi fattori: la bontà di un link può, per esempio, dipendere dalle condizioni di propagazione del segnale radio, dalla lunghezza del salto, da ostacoli lungo il percorso, da interferenze, ecc. Inoltre i collegamenti, come si vedrà in seguito possono essere affetti da asimmetria , nel senso che uno stesso collegamento può rivelarsi diverso in qualità a seconda del verso di percorrenza del salto, ossia la connessione può essere soddisfacente quando il segnale va dal nodo A al nodo B e non esserlo quando il segnale va nel senso opposto da B ad A. In definitiva si potrebbe affermare, e questo vale in generale, che la qualità dei collegamenti wireless non obbedisce più ad una logica *vero-falso*, bensì ad una logica sfumata *(fuzzy)*, in cui sono possibili livelli di comportamento (*performance*) dipendenti in vario grado da fattori quali quelli sopra accennati.

Qui si inserisce, a nostro parere, l'utilità di uno strumento di lavoro finalizzato alla rilevazione, nella maniera più fedele possibile alle situazioni reali, di elementi oggettivi che determinano il comportamento dei processi di routing

e quindi la loro maggiore o minore rispondenza alle effettive esigenze della connettività. Sulla base della raccolta e dell'eleborazione di tali elementi sarà poi possibile tracciare un quadro della complessiva funzionalità di una rete gestita da un determinato software di routing. Come si vedrà in seguito, le indagini possibili in questo senso sono molteplici: quanto più precise saranno le metriche mantenendo le condizioni di probing invariate (pacchetti di probing di piccole dimensioni e broadcast) e tenendo in considerazione la lettura delle statistiche oltre che a *Livello Rete* anche a *Livello Data Link* e a *Livello Fisico*? quanto saranno precise utilizzando probe diversi (unicast e broadcast di dimensioni diverse) ma contiunando a leggere solamente il Livello 3? utilizzare entrambe le metodologie di indagine (broadcast e unicast di dimensioni variabili) e prendendo in esame tutti i dati disponibili a tutti i livelli potrebbe apportare raffinamenti ulteriori alla stima dei link?

1.2 Le Reti Mesh

Le reti con collegamenti ad-hoc, con i nodi disposti a maglia sono denominate *Reti Mesh o Wireless Mesh Network*. Esse costituiscono una categoria di reti senza fili formate da elementi che possono essere anche mobili anch'essi comunque dotati di funzionalità di routing e pur non esenti da instabilità, ma al contrario, caratteristiche per questo fattore, sono per in grado di offrire comunicazioni anche in aree che non si presterebbero alla cablatura grazie al ricorso a varie tecnologie di trasmissione senza fili (radio, ottiche o satellitari). Le apparecchiature utilizzate consistono in apparati disposti, come s' detto, secondo un'architettura paritaria, dotati di intefacce wireless punto-a-punto (*point-to-point wireless interfaces*) al posto delle tradizionali interfacce cablate. In tali reti ciascun nodo può raggiungere ogni altro nodo, direttamente o

indirettamente passando per nodi intermedi, con percorsi (*path*) costituiti da salti multipli tra i router detti anche percorsi *multihop*. Poiché ai nostri fini le reti mesh costituiscono la tipologia di maggiore interesse, riportiamo qui di seguito in sintesi alcune delle loro caratteristiche più salienti: - ogni nodo possiede una partecipazione attiva al processo di routing; - il funzionamento della rete indipendente dalla tecnologia adoperata; - il comportamento della rete è dinamico ed autoconfigurante: alcune reti di questo tipo possono essere accese lanciando i nodi in modo casuale da un elicottero; - possono offrire supporto alla mobilità; - larghezza di banda limitata e basse potenze dovute all'alimentazione con batterie; - rendimento delle connessioni non costante. Le reti mesh trovano impiego in vari ambiti applicativi come di seguito accennato: -collegamento ad Internet di aree topografiche o geografiche periferiche; - reti di sensori per rilevamenti di vario genere (es. reti di sensori per il rilevamento di incendi in aree difficilmente controllabili); - reti dinamiche costituite da nodi in movimento per l'organizzazione dei soccorsi in situazioni di catastrofi naturali; - realizzazione di reti urbane di comunicazione a basso impatto ambientale; - sorveglianza di punti sensibili mediante l'impiego di telecamere o altri dispositivi di sorveglianza; - messa a disposizione di risorse socialmente utili in situazioni di svantaggio ambientale (digital divide); - impieghi tattici in ambito militare; ecc. Si possono riassumere gli aspetti di criticità evidenziando che le reti mesh presentano i problemi tipici derivanti da configurazioni dinamiche e connessioni asimmetriche, dalla bassa potenza dei segnali che rende i collegamenti instabili anche a causa dell'assorbimento da parte degli ostacoli, delle interferenze e dell'attenuazione propria delle alte frequenze. Da non trascurare un aspetto interessante dal punto di vista energetico, della sostenibilità e dell'impatto ambientale. Le reti Mesh, per costituzione, sono caratterizzate da una moltitudine di link di lunghezza li-

mitata. Questo fattore permette l'interconnessione di una stessa zona, con tanti nodi a bassa potenza, anziché con poche *celle* dotate di enorme potenza radiante. In termini di impatto ambientale si traduce in istallazioni con tralicci più piccoli, e quindi che deturpano meno il paesaggio, e l'utilizzo di potenze emissime più basse. è insito nelle reti mesh una organizzazione della rete cooperativa, paritetica e democratica, in opposizione al organizzazione oligopolica caratteristica delle grandi imprese di comunicazione elettronica, che invece di cooperare, riducendo al minimo gli impatti ambientali, concorrono insallando grossi tralicci in zone limitrofe e cercando di prevaricare gli uni su gli altri con potenze sempre maggiori. Per questi motivi le wireless mesh network, nell'era democratica di Internet, avranno sempre più un ruolo fondamentale nella libertà di espressione e per questo motivo intorno ad esse si sono create attività comunitarie di piccoli gruppi interessati allo sviluppo di tecnologie alternative e sostenibili.

1.3 Protocolli di routing su Mesh Network

Al fine di contestualizzare questo lavoro, nel senso di illustrare l'ambito problematico in cui esso si inserisce, si ritiene opportuno fare un rapido accenno ai protocolli elaborati per queste reti particolari.

A differenza che nelle reti cablate caratterizzate come s'è detto dalla rigidità dei percorsi predeterminati i protocolli utilizzati nelle reti mesh organizzano le interconnessioni tra i nodi in maniera dinamica, in modo di assicurare al meglio il recapito del traffico alle destinazioni volute.

Questa organizzazione delle connessioni consiste essenzialmente in questo: quando due qualunque dei nodi della rete non hanno (o non hanno più) un collegamento diretto uno con l'altro, la funzionalità di routing deve innan-

zitutto stabilire una connessione; poi, deve ridurre al minimo il traffico di controllo (*overhead*) indotto, per garantire la scalabilità della rete e nello stesso tempo rilevare prontamente l'eventuale interruzione dei link, al fine di rendere minime le perdite nella trasmissione dei (*pacchetti*). È dunque un compito abbastanza arduo e complesso quello che grava sui protocolli, sia a causa dell'eventuale mobilità dei nodi, sia per la natura stessa dei link (non cablati). Si comprende da qui come le procedure di routing progettate per le reti tradizionali non possano risultare validi per le reti ad hoc wireless: ne discende che i relativi protocolli di routing comporteranno necessariamente soluzioni di compromesso e problemi di ottimizzazione preferenziale di certe caratteristiche. Non vogliamo qui far menzione di un'ulteriore possibile problema che potrebbe incombere sui protocolli di routing perch esula dai nostri scopi, e cio quello della sicurezza delle comunicazioni che, nel caso di emissioni radio non mai trascurabile.

1.3.1 Tipi di protocolli wireless

Possiamo categorizzare i protocolli in due tipologie diverse secondo la modalità adottata per la determinazione della rotta. Entrambi gli approcci si prefiggono di far fronte all'importante problema di ridurre l'entità parassitaria del traffico di controllo, ovviamente cercando di mantenere accettabili livelli di rendimento. Una prima categoria quella dei protocolli cosiddetti *reattivi*. Secondo questo tipo di protocollo i nodi non conoscono a priori la rotta per raggiungere la destinazione: questa infatti non è predeterminata, nel senso che il mittente non possiede una *tabella di ruoting* statica e predefinita (ossia una lista predeterminata di indirizzi -memorizzata in ciascun nodo della rete- a cui il nodo stesso può inoltrare i pacchetti), ma ogni router ne scopre una, solo al momento dell'invio di un pacchetto. Il protocollo

reattivo più diffuso nell'ambito delle reti mesh, utilizzato anche per reti di sensori,è conosciuto con l'acronimo *AODV (Ad-hoc On Demand Distant Vector)* . Il funzionamento di base è qui di seguito illustrato. (Fig. 1). Facendo riferimento alla Fig. 1, quando un router A vuole comunicare con un router B, invia il pacchetto di controllo a tutti gli altri nodi della rete, ma solo il nodo destinatario invia un messaggio di risposta seguendo lo stesso percorso seguito dalla richiesta, assunto automaticamente come percorso più adatto. Questo tipo di protocollo offre il vantaggio che non sono necessari messaggi di controllo per rotte inattive e inoltre risulta adeguato in caso di nodi mobili, che come tali non possono disporre di tabelle di routing predefinite (o statiche). Si presta meglio per le reti in cui solo alcune rotte convogliano il traffico dei dati (*traffico pagante*, come per esempio nelle reti di sensori con limitatissime potenze trasmissive e scarsa mobilità). Lo svantaggio risiede invece nel fatto che il mittente è costretto ad inviare a *tutti* i nodi *(flooding)* della rete la richiesta di connessione, determinando così un certo sovraccarico di traffico che si risolve in un impegno aggiuntivo parassitario dovuto alla trasmissione dei messaggi di controllo. Sono anche da tenere presenti i tempi morti necessari all'individuazione delle rotte.

L'altra categoria di protocolli, i cosiddetti protocolli *proattivi* fa in modo che, dopo lo scambio dei primi pacchetti, si venga a creare una rotta di connessione tra sorgente e destinatario, consistente in una tabella di routing per ogni nodo raggiungibile. In seno a questa categoria si possono distinguere due tipi di algoritmo di routing denominati *Distant Vector* e *Link State Packet*; essi differiscono principalmente nel tipo di informazione che trasmettono ad ogni router relativamente alla topologia della rete. Entrambi comunque richiedono che ogni nodo conosca i propri nodi vicini e l'importanza (il *peso* o *metrica*) di ogni link che lo collega a ciascuno di essi.

1.3.2 Algoritmi di Rounting

Distance Vector Routing

Il Distance Vector Routing si basa sull'algoritmo di Belman-Ford in cui ogni router comunica periodicamente ai suoi vicini (*neighbours*) la propria *distanza* da ogni altro router della rete La comunicazione consiste nell'invio periodico da parte di un router ai propri vicini della sua completa tabella di routing (ossia il *Distant Vector*). La tabella propria di un nodo contiene la lista dei suoi vicini attraverso cui passare per raggiungere ogni destinazione voluta. Inoltre, nella tabella sono contenuti anche i pesi corrispondenti ai link con ognuno dei vicini che solitamente sono rappresentati dal numero di *hop*. Per esempio, vediamo come si procede nel caso di una semplice topologia di tre nodi collegati in sequenza uno dopo laltro, A , B e C. Inizialmente A ha nella propria tabella il nodo B, questo ha nella propria tabella A e C, la tabella di C contiene solo B. Dopo un primo giro di diffusione dell'informazione, il nodo A apprende dell'esistenza del nodo C e che esso può essere raggiunto tramite B, A viene quindi incluso nella tabella di B. C, avendo appreso da B l'esistenza di A e la sua raggiungibilità tramite B stesso, include A nella propria tabella; B da parte sua apprende che C può raggiungere A tramite se stesso e registra a sua volta questa informazione nella propria tabella. Al secondo giro A può anche inviare a B le proprie informazioni e il ciclo si chiude. Ognuna delle acquisizioni di conoscenza viene dunque registrata nella tabella di routing di ciascun nodo insieme a valori dei link relativi ai vicini. Al ripetersi del giro, il meccanismo inserisce nelle tabelle di routing ogni nuovo contatto di un altro elemento con qualcuno dei tre nodi iniziali, e così via. Si nota subito che la dimensione del messaggio che ciascun router deve diffondere aumenta proporzionalmente al numero di nodi, il che comporta problemi

di *scalabilità*, ossia di possibilità di estensione della rete; inoltre si intravede la difficoltà per l'algoritmo di rilevare prontamente l'eventuale interruzione di un link, per cui si può incorrere in quell'inconveniente conosciuto come *count-to-infinity problem* o conteggio allinfinito. Per far fronte a tale problema, sono state necessarie alcune modifiche all'algoritmo. Una soluzione al problema delle grandi topologie (e del conteggio allinfinito) viene fornita da un altro protocollo di routing, basato sull'algoritmo di Dijkstra e detto *Link State Packet*.

Link State Packet

In questo protocollo, ogni router della rete comunica periodicamente a tutti i nodi della rete il proprio vicinato e per ogni neighbour il peso del *link*. L'informazione viene recapitata mediante l'invio di messaggi indicati con lacronimo LSP (*Link State Packet*) che reca l'identificativo del nodo mittente insieme alla lista dei suoi vicini e i pesi dei relativi link. Questi dati vengono periodicamente aggiornati. Ogni nodo destinatario (tra i vicini del nodo mittente) registra il pacchetto ricevuto e lo rispedisce a suo volta a tutti i propri vicini, in modo che il pacchetto aggiornato raggiunga tutti i nodi della rete (*flooding)*. Sulla base dell'insieme degli LSP relativi a tutti i router (immagazzinati in ogni router in un apposito database, il Link State Database) si può ricostruire l'intera topologia della rete. In questo modo ogni router dispone di una mappa dellintera topologia e può scegliere il percorso ottimale per ogni destinazione applicando al database l'algoritmo di Dijkstra. L'algoritmo cerca quale dei nodi vicini può raggiungere un nodo al di fuori del gruppo dei vicini con un percorso (*path*) dal costo più basso. Il protocollo basato su questo algoritmo di routing più utilizzato e sviluppato dalla comunità è senza dubbio *OLSR* (Optimized Link State Routing) .

1.3.3 OLSR *Optimized Link State Routing*

Il protocollo OLSR rappresenta attualmente lo standard de facto per le Mesh Network: gran parte delle implementazioni di reti Mesh esistenti utilizzano un'implementazione di questi protocollo. Il meccanismo di funzionamento è simile al classico *LSP* ma con l'aggiunta di una metodologia che consente di limitare il traffico di controllo (flooding), i MPR (*Multi Point Relay*). Esiste un OLSR standard descritto nell' RFC3626[1]. L'OLSR descritto nel RFC relativo, utilizza come metriche per l'algoritmo di *Dijkstra* il numero di hop, mentre come vedremo in seguito, alcune varianti del protocollo utilizzano metriche più specializzate all'uso in una mesh. Le funzionalità base di questo protocollo, che lo caratterizzano e fanno di esso un protocollo per reti mesh, possono essere riassunte come segue:

- Neighbour/Link sensing
- Optimized flooding (MPR)
- Link-state messaging.

La prima funzionalià, implementata mediante l'uso di pacchetti di HALLO, ha il compito di probare la rete, ogni nodo, così capirà quele è il suo vicinato (*neighborhood*). Questo meccanismo, in OLSR, prevede che il link con uno dei vuoi vicini possa essere asimmetrico, come è stato accennato, e potrebbe quindi funzionare solamente in un verso. Le informazioni sul vicinato sono mantenute in due tabelle chiamate rispettivamente *Link Set* e *Neighbor set* che, prevedono la possibilità per un link di essere asimmetrico. Come si vedrà più avanti nella trattazione, una delle varianti di OLSR prevede che questa informazione possa non essere solamente binaria, ma fuzzy: la bontà dei collegamenti è descritta in questa variante da una percentuale.

La seconda funzionalità, come accennato, è propria di OLSR, essa implementa il selective flooding. L'informazioni sulla topologia della rete, come in tutti i LSP, è distribuita in tutta la rete tramite pacchetti di controlo: qusto tipo di dato topologico in OLSR viaggia su pacchetti di *Topology Control*. Per evitare che la rete venga inondata (flooding) da pacchetti di questo tipo, in OLSR esiste un meccanismo con il quale ogni nodo elegge tra i suoi neighbour il minimo insieme che consente di raggiungere tutti i *2 Hop Neighbor* (Tutto il vicinato a distanza di due *Hop*). Questo processo è configurabile in due aspetti:

- Quanti MPR al minimo deve avere ogni nodo
- Quali informazioni topologiche far distribuire ai propri MPR

Il primo *tuning* consente da un lato, scegliendo un parametro basso e quindi inoltrando, per ogni nodo, i propri TC (Traffic Control) solamente a pochi MPR, di diminuire il traffico di controllo della rete, dall'altro, scegliendo un alto numero di MPR minimi, di aumentare l'efficenza e la reattivit della rete alle variazioni avendo a disposizione Relay (instradatori) ridondati.

Il secondo parametro invece indica quali informazioni distribuire nei pacchetti TC. Anche in questo caso si può scegliere una stategia che riduca il traffico di controllo, spedendo solamente le informazioni relative al rapporto nodo-MPR e a nodo-MPRs(nodi che hanno scelto il nodo in questione come proprio MPR), oppure sceglierne una che inoltra ai propri MPR le informazioni di vicinanza di tutti i neighbor.

La terza si incarica quindi di calcolare, sulla base delle informazioni sulla topologia della rete raccolte durante la fase di probing e contenuta nei Link State Database, tramite l'utilizzo di un algoritmo di ricerca di un cammino minimo come ad esempio Dijkstra la strada (o *path*) meno costosa.

Il protocollo OLSR, essendo un LSP, presenta aspetti problematici legati alla scalabilità della rete (es. cicli interminabili o *loop*) nel caso di topologie variabili molto rapidamente: ciò avviene a causa di possibili incongruenze tra i grossi database poco coerenti di nodi diversi in una grossa rete.

Si vedrà come nella vita del protocollo OLSR queste tre funzionalità sono state modificate per soddisfare le esigenze di stabilità e scalabilità delle implementazioni di reti mesh reali.

1.3.4 B.A.T.M.A.N *Better Approach To Mobile Ad-hoc Network*

è presentato adesso l'approccio di un protocollo molto giovane che ha meno di due anni e che, a differenza degli altri protocolli in cui si è partiti da considerazioni teoriche e per cui sono stati stilati prima gli RFC che le implementazioni, adotta un'approccio basato sulla sperimentazione diretta.
B.A.T.M.A.N è nato ed è tuttora sviluppato da una comunità eterogenea di appassionati di reti mesh che non ha alcuna componente accademica.
Lo scopo dei suoi sviluppatori è quello di creare un protocollo che possa sostituire OLSR, attuale dominatore assoluto, in tutte le implementazioni di rete mesh.
In contrasto con molti altri protocolli, B.A.T.M.A.N ha goduto di larghe fasi di test e notevoli riscontri del funzionamento reale fin dall'inizio dello sviluppo. La metodologia di sviluppo perciò è *Button-Up*. Partendo dal basso nuove feature dettate da nuove esigenze sono svilupate e i problemi vengono corretti man mano che sono riscontrati.
B.A.T.M.A.N di propone come un migliore approccio al routing per le reti mesh ad-hoc (*Better Approach To Mobile Ad-hoc Network*) garantendo scalabilità e stabilità non godute con OLSR.

La stessa comunità tedesca sviluppatrice di B.A.T.M.A.N ha sviluppato varianti del protocollo OLSR: ha introdotto l'uso della metrica ETX o del protocolo Fish Eye che vedremo più avanti in questa trattazione. L'approccio è stato sempre basato sulla sperimentazione diretta su reti mesh cittadine.
L'algoritmo di routing proposto da B.A.T.M.A.N si basa sulla suddivisione della conoscenza del miglior percorso da un punto all'altro della rete tra i diversi nodi attraversati. Non c'è bisogno che ogni nodo conosca l'intera topologia di rete affinche scelga il gateway migliore per ogni destinazione. Le informazione sulla rete, più si riferiscono a zone remote, maggiormente saranno imprecise.
Al contrario dei protocolli *Link State Packet* in cui ogni nodo conosce l'intera topologia della rete, B.A.T.M.A.N raccoglie solamente le informazioni sul gateway migliore (*best-ranking neighbour*)da utilizzare per ogni destinazione. B.A.T.M.A.N può essere categorizzato per questo motivo tra i Distanct Vector.
Ogni nodo genera dei pacchetti originator e li invia a tutti i suoi vicini a bordo di pacchetti broadcast. Ogni nodo che riceve gli *originator message* (OGM) li rinvia sulla rete ai propri vicini in un processo di flooding.
Ogni nodo intermedio inoltra gli OGM ai suoi vicini solamente se gli giungono dal miglior vicino corrente (*best-ranking neighbour*) per quella determinata destinazione. Il criterio con cui è scelto il *best-ranking neighbour* è che , per ogni destinazione di rete, gli OGM relativi siano giunti in misura maggiore e per primi che da tutti gli altri vicini.
La *ranking table* di ogni nodo conserva le migliori statistiche relative ad ogni nodo sulla rete e, in base ad essa, è costruita la tabella di routing.
La valutazione del grado di asimmetria di un link è fatta da ogni nodo sulle statistiche dei propri OGM ritrasmessi dai propri vicini: se un nodo riceve

gli originator message da un suo vicino ma non riceve da esso ritrasmissioni dei propri OGM ritiene il link con il nodo in questione asimmetrico. I link asimmetrici non sono utilizzati e le informazioni relative ad essi sono omesse. I pacchetti originator sono di piccole dimensioni essi contano di 52byte inclusi gli header IP ed UDP, per cui l'ovehead del protocollo è limitato.
Il riconoscimento dello stesso pacchetto ricevuto da path diversi presso i nodi è garantito da un numero si sequenza che lo identifica univocamente insieme all'indirizzo IP del nodo *originator*.
Una delle ultime funzionalità aggiunte nelle ultime versioni del protocollo è di decidere, per ogni interfaccia, quanto lontano in termini di hop count spedire le informazioni. Questa caratteristica consente di risparmiare traffico inutile quando per esempio si utilizzano interfacce diverse per costruire i backbone. Il traffico sulle intefacce secondarie di un nodo sarà così limitato in uno scope locale.

1.3.5 Overhead e Scalabilità

I protocolli di routing in quanto tali non possono funzionare senza lo scambio tra i router di informazioni per così dire di servizio che ne governano il funzionamento. Questo scambio costituisce quindi un aspetto essenziale dei protocolli ed una specie di traffico ambientale (già accennato con il termine *overhead*) inevitabile, in seno al quale i router operano per convogliare il flusso dei dati (traffico utile) ai destinatari voluti.
Il problema che sorge è quello di fare in modo che il traffico di servizio lasci il più possibile libere le risorse a beneficio del traffico utile, o quantomeno mantenga lontana la massa dei messaggi di controllo dalla saturazione parassitaria della rete; il traffico ambientale deve però, e prima di tutto, as-

sicurarne il corretto funzionamento.

Esistono diverse vie per la soluzione al problema, ossia per preservare lambiente di lavoro (in senso quasi ecologico) da inquinamenti eccessivi di traffico parassitario; si tratta sempre di soluzioni di compromesso (*tradeoff*) che tendono ad assicurare il funzionamento della rete anche in condizioni di svantaggio e nel contempo rendere massime le condizioni di utilizzazione della rete stessa da parte dell'utente.

Le tecniche di riduzione dell'overhead sono particolarmente impegnative per le reti ad-hoc, caratterizzate, come si sa, da variabilità delle topologie, movimento dei nodi, esigenze di aggiornare frequentemente le informazioni di routing.

Esistono varie tecniche di riduzione dell'overhead che riguardano i vari aspetti del problema sui quali si può intervenire: uno di questi è il flooding (inondazione).

Il flooding consiste nel meccanismo elementare per cui ogni nodo, quando riceve un messaggio, lo ridistribuisce ai propri vicini: questo processo a cascata ha come risultato voluto che *tutti* i nodi della rete ricevono l'informazione. Ma spesso il costo da pagare per questo risultato è un eccesso di traffico parassitario (overhead) sulla rete.

Nei protocolli di routing il flooding consiste dunque nel complesso di messaggi necessario a far pervenire a *tutti* i router della rete le informazioni indispensabili al loro funzionamento. L'algoritmo di flooding è quindi una parte essenziale del protocollo. Per esempio, i protocolli proattivi (tipo OLSR), per il fatto di dover disseminare periodicamente le informazioni sullo stato dei link, richiedono una grande quantità di flooding, il cui meccanismo occupa la gran parte del traffico indotto. Pertanto migliorando la procedura di flooding si possono conseguire drastiche riduzioni nel numero di messaggi

necessari a distribuire linformazione a tutti i nodi.
Per ridurre il carico di lavoro necessario a recapitare i messaggi a tutti i nodi della rete, si può affidare il compito di ripetere il messaggio, invece che a tutti i singoli nodi, solamente ad alcuni di essi, appartenenti ad un opportuno sottoinsieme.
Uno dei modi di determinare questo sottoinsieme è il sistema dei *Multi Point Relay* (MPR) come già visto a proposito del protocollo OLSR. Esso consiste essenzialmente nelle scelta da parte di un nodo di un insieme di nodi *vicini* cui affidare l'incarico di inoltrare il messaggio contenente la topologia del suo vicinato.
I protocolli di routing elaboratii nelle reti ad-hoc vanno incontro al problema della scalabilità ossia al problema relativo all'allargamento del numero dei nodi ad una scala più grande (*scaling*), fino ad un limite massimo che il protocollo può sostenere.
E evidente che all'aumentare di nodi in una rete, per esempio, sotto un protocollo proattivo, (diciamo ancora l'OLSR) il traffico di controllo cresce al crescere del numero di nodi, e poiché la larghezza di banda è necessariamente limitata ci sarà certamente un limite al numero di nodi sostenibili, nel senso che al di sopra di un certo numero, il traffico di controllo assorbirebbe tutta la larghezza di banda disponibile, non lasciando alcuno spazio al traffico dei dati.
Oltre tale limite, addirittura la banda non basterebbe neanche al solo traffico di controllo.
In pratica, tanto per rendere l'idea, con l'OLSR basato sulla tecnologia dell802.11b, si raggiunge al massimo il numero di 100 nodi complessivi.
Sorge quindi il problema di aumentare la scala del protocollo.
I metodi proposti sono sostanzialmente di due tipi, cosiddetti ortogonali tra

loro, e cioè uno verticale basato su uno schema ad albero e l'altro orizzontale, noto con l'espressione occhio di pesce o *Fish Eye*.

Scaling verticale o ad albero

Come lascia ad intendere l'aggettivo verticale, lo schema ad albero prevede una suddivisione della rete in gruppi di nodi e un routing gerarchico secondo una struttura ad albero. L'aggregazione di un nodo ad un certo gruppo avviene automaticamente in modo dinamico.
Alla base del meccanismo c'è l'individuazione dei nodi radice (*root* degli alberi logici (*tree*), i quali diventano i nodi di riferimento (i capi) dei vari gruppi (*cluster*) e sono quei nodi che hanno il maggior numero di nodi vicini.
Gli alberi si creano automaticamente in maniera dinamica mediante la scelta da parte di ogni nodo del proprio nodo-*padre* ossia il proprio vicino di grado maggiore ossia quello che ha il maggior numero di nodi vicini. Allora un nodo che ha localmente il grado massimo diviene il nodo radice del proprio albero che costituisce così un *cluster*.
L'intera rete si organizza come un insieme di alberi ed in seno ad essa il flooding viene effettuato lungo i rami degli alberi stessi. La formazione degli alberi logici avviene mediante l'invio periodico di messaggi detti messaggi di ramo (*Branch Messages*) che hanno la portata di un solo salto e contengono l'identificativo del nodo emittente, l'identificativo del suo nodo padre e la propria distanza dalla radice dell'albero (o profondità) espressa in numero di salto (*hop*).
Ai nodi radice è affidato il compito di comunicare con l'esterno del proprio albero e può stabilire la profondità (distanza) massima consentita ad un nodo per associarsi ad esso. Sicché se un nodo vuole associarsi ad un albero,

controlla la propria profondità rispetto al nodo padre e se essa si avvicina al limite fissato stabilisce se si può o no associare al nodo. In caso negativo a causa del superamento della profondità consentita, il nodo cerca di associarsi, tramite il nodo vicino, all'albero migliore più prossimo e disponibile; se non trova alcun albero disponibile cui associarsi, il nodo sceglie se stesso come radice.

Questa modalità di funzionamento del protocollo di routing deve instaurarsi e cessare automaticamente, quando si supera una certa soglia nel numero di nodi attivi (che può essere riferita a parametri del tipo l'unghezza del link state database memorizzato nei nodi o dalla frequenza dei messaggi di controllo) così come viene rilevata attraverso i messaggi di branca sopra citati.

Allinterno di un albero, il funzionamento del protocollo è quello normale, vale a dire che le cose vanno come se le strutture ad albero non esistessero, fatta salva la funzione del nodo radice di comunicare con lesterno del suo albero e l'esigenza che, se un nodo è in contatto con un altro albero, ne deve informare il proprio albero e in particolare la propria radice.

In conclusione si intuisce come il traffico di controllo destinato all'intera rete viene ridotto, con notevoli vantaggi, solamente allo scambio di informazioni fra i cluster costituiti nella topologia della rete.

Le decisioni riguardo il routing inter-cluster sono prese dai nodi *root* utilizzando anche un'istanza diversa dello stesso protocollo utilizzato all'interno.

Scaling orizzontale e *Fish Eye*

Come si è detto sopra, l'organizzazione gerarchica consente di ridurre l'overhead connesso con la numerosità dei nodi di una rete.

Un diverso approccio (non gerarchico) al problema dello scaling, quello del

routing a *occhio di pesce o Fish Eye*, proposto dallitaliano Gerla et al. si basa essenzialmente sul concetto che ciò di cui ha bisogno un nodo per inoltrare i dati verso una determinata destinazione lontana non è uninformazione dettagliata, ma piuttosto una direzione di massima verso cui convogliare i dati: le informazioni più accurate per il routing saranno disponibili man mano che i dati viaggiano verso la destinazione.

Il meccanismo consiste essenzialmente di rendere meno frequenti, proporzionalmente alla distanza, le informazioni riguardanti i nodi più lontani rispetto a quelle relative ai nodi vicini. In tal modo il traffico generato da un nodo diminuisce in forma inversamente proporzionale alla distanza da esso, in guisa che le informazioni di aggiornamento sulla topologia formano una specie di alone che svanisce con la distanza. In tal modo, scegliendo un'opportuna funzione della distanza, si può fare in modo che il traffico tenda ad un limite finito anche quando la rete cresce indefinitamente. L'applicazione del metodo Fish Eye al protocollo OLSR risulta particolarmente agevole se si agisce sui TTL (time to live) dei pacchetti di aggiornamento della topologia. Ma esso implica che la crescita di una rete mesh faccia aumentare proporzionalmente la sua estenzione, intesa come numero di hop.

Capitolo 2

Le metriche

Lo scopo di questo lavoro è di indagare sulle metriche usate per le reti mesh e sulle metodologie utilizzate per stimarle.
Prima di proporre nuove varianti più efficaci di calcolo delle metriche, è utile conoscere i funzionamenti attuali.

2.1 Metriche su reti cablate

Le metriche utilizzare nella pratica comune con i protocolli di routing per reti cablate, come potrebbe già essere chiaro, non sono adattabili in un contesto caratterizzato da collegamenti instabili per natura.
I link delle reti cablate o funzionano al 100% oppure non funzionano, non è menzionato il caso di funzionamenti parziali o variazioni dipendenti dal tempo.
Le metriche perciò utilizzate nei protocolli di routing concepiti per le reti su cavo, come ad esempio RIP o OSPF, utilizzano come metrica il numero di salti (*hop count*) oppure una metrica impostata manualmente dall'amministratore che ne indica la capacità. Nell'OSPF, per esempio, si usa indicare

manualmente la capacità dei collegamenti in base alla tecnologia utilizzata. Le prime versioni delle metriche utilizzate nei protocolli per reti mesh, come in OLSR, usavano il numero di salti associato all'uso di un meccanismo di isteresi per misurare la qualità di un *path*.

2.2 Metriche su Mesh Nework

Come si è accennato, le mesh network sono costituite,non solamente da link senza fili e quindi instabili per natura, ma anche da processi software (protocolli di routing) che ne accentuano la variabilità.
In questo contesto le metriche utilizzate per stimare un percorso, dovrebbero essere quanto più possibile indicative della capacità, intesa sia come portata di informazioni che come stabilità delle connessioni ma anche della variabilità nel tempo di queste componenti.
Le metriche implementate negli attuali protocolli per reti mesh, pur avvicinandosi più che quelle tipiche dei protocolli per reti cablate alla reale situazione della rete, valutano ancora troppo superficialmente le caratteristiche intrinseche delle connessioni utilizzate, vediamo perché.

2.2.1 Problemi di probing (Gray Zone)

In alcuni recenti lavori [2] è stato portato all'attenzione un particolare fenomeno che accade utilizzando i consueti protocolli di routing, basati sullo standard di trasmissione wireless IEEE 802.11b/g , sviluppati per le reti ad hoc.
Il fenomeno consiste nel fatto che mentre la rilevazione da parte di un nodo del proprio vicinato effettuata mediante la diffusione su tutta la rete dei cosiddetti messaggi di *neighbour's sensing* (basati su pacchetti *broadcast*) as-

sicuri la raggiungibilità dei vicini, lo scambio dei dati risulta impossibile .
Tutto ciò introduce una discrepanza tra lo stato delle rotte individuate (che altro non è che il database delle tabelle di routing memorizzato in ogni nodo) e l'effettiva possibilità di trasmissione dei dati.
Le zone (topografiche) in cui il fenomeno si verifica sono chiamate Zone Grigie o *Gray Zones*.
In altri termini, la perdita di pacchetti di dati non può essere attribuita all'elaborazione dedicata dal protocollo a ricostituire una rotta quando uno dei nodi perde il contatto con il nodo successivo, ma all' impossibilità del protocollo nel valutare correttamente lo stato delle connessioni. La questione è di grande rilievo per protocolli (come OLSR e B.A.T.M.A.N) che si fondano sul'utilizzo di pacchetti di broadcast per la determinazione dei nodi vicini.
L'algoritmo di scoperta del vicinato deve quindi essere in grado di stabilire se il link tra due vicini può o meno convogliare dati e in che misura.
Nello standard IEEE 802.11b avviene che i messaggi di *neighbour discovery* vengono diffusi a tutti i nodi (ossia in modalità *broadcast*) e come tali forgiati a bassa velocità (bit/s), a maggiore affidabilità e capaci di andare più lontano di quanto non facciano i pacchetti di dati.
Nelle reti mobili, in cui, come è stato detto, sono utilizzati protocolli reattivi, il fenomeno si manifesta più frequentemente; Durante lo spostamento di un nodo mobile dalla zona di copertura di un nodo ad un altro, le trasmissioni con riscontro (unicast), tipiche delle gran parte delle comunicazioni su rete IP, sono forgiate a rate più alti del minimo, mentre i messaggi di probe sono mandati al rate minimo. Mentre il nodo si sposta, riceve pacchetti di probe in broadcast dal nodo al quale si sta avvicinando e modifica le rotte perché venga intrapreso il nuovo percorso ma, trovandosi nella sua grayzone, non è possibile recapitare unicast senza abbassare il rate. Le comunicazioni unica-

st, come gran parte delle trasmissioni dati IP non potranno così giungere a destinazione.
Siamo quindi di fronte al caso tipico della gray zone: i messaggi di controllo darebbero via libera, ma il traffico utile trova, di fatto, impedimento.
Il caso della grayzone è maggiormente riscontrabile nei casi di reti in cui i nodi si muovono perché, anche se il protocollo 802.11b/g prevede un meccanismo di adattamento del rate di trasmissione, nella mobilità non si ha il tempo di adattamento del rate.
Vediamone ora le possibili cause di tali zone:

- La prima consiste nel uso rate di base, quindi modulazioni utilizzate più semplici e meno penalizzate dalla presenza di interferenze, per il probe dei link.

- La seconda consiste nel fatto che la diffusione in broadcast non attende conferma di ricezione (*aknowledgement ACK*) da parte dei destinatari, e quindi non garantisce l'utilizzo bidirezionale del link: ricevere un messaggio di HELLO (neighbour sensing), per esempio, non significa che sia possibile trasmettere dati in direzione inversa.

- La terza dovuta alla trasmissione di piccoli pacchetti che è meno soggetta a errori di trasmissione rispetto ai pacchetti più lunghi quali sono quelli che trasportano dati.

Per questi motivi tutti i messaggi di probe utilizzati dai protocolli per le reti mesh sono meno sensibili alla qualità dei collegamenti e quindi non rispecchiano la situazione reale cui va incontro la trasmissione di dati.
Tra le metodologie proposte in questo lavoro si prevede di utilizzare pacchetti di dimensione diversa per valutare lo stato delle connessioni senza fili ponendo rimedio alla terza causa delle Gray Zone.

Per godere di una percezione più precisa, e quindi non essere ingannato dalle limitazioni della gray zone , il tool è in grado di generare *probe* di diversa natura e raccogliere, quindi, statistiche anche sulle trassmissioni basate su unicast e su quelle forgiate a *rate* diversi da quelli base.
In questo modo sarà possibile confrontate le metriche costruite solamente con pacchetti broadcast ad 1Mb/s, di piccole dimensioni, con quelle costruite con pacchetti di dimensione variabile, unicast ed a rate elevati.
Nel corso di questa trattazione ci si limiterà, come nel caso dei protocolli di routing attualmente utilizzati, a considerare le metriche costruite con le statistiche sulla ricezione di soli pacchetti di broadcast.
Specificatamente si vedrà quanto è possibile affinare la percezione delle caratteristiche intrinseche dei collegamenti modificando, alla normale attività di indagine dei protocolli, la dimensione dei pacchetti di indagine della rete (probe). Analizzando i dati raccolti dal tool si valuterà, così, la percezione sul grado di conoscenza dei collegamenti del protocollo in questo caso e si potrà stabilire se qusta metodologia aumenti significativamente l'efficenza delle scelte di routing.
Un altro caso prevede l'aggiunta delle informazioni dovute alla lettura anche dei dati dei livelli più bassi per determinare se possano migliorare ulteriormente la percezione dei protocolli.
In seguito, onde consentire un confronto, si vedono le metriche utilizzate nei protocolli OLSR e B.A.T.M.A.M e su quali informazioni esse siano costruite.

2.2.2 ETX

La metrica in uso più diffusa nelle reti mesh è quella chiamata ETX (*expected transmission count metrics*).

Essa è un valore numerico che tiene conto delle statistiche di perdita di pacchetto nei due versi delle trasmissioni: $\frac{1}{LQ*NLQ}$ in cui LQ (*Link Quality*) esprime la percentuale (rispetto ad una *finestra* configurabile che tiene conto delle ultime ricezioni) di pacchetti giunta a destinazione in un verso e NLQ (*Neighbour Link Quality*) quella giunta nell'altro. Esso quindi può assumere valori positivi maggiori di uno oppure, quando il collegamento peggiora all'infinito, assume valore nullo.
Il significato di questa metrica è che un link è tanto più stabile quando più il valore dell' ETX è prossimo ad 1.
L'ETX è largamente utilizzato come metrica del protocollo OLSR e specificatamente nell'implementazione di Andreas T∅nnesen (http://www.olsr.org). Anche in questo caso LQ e NLQ sono valutati solamente tramite pacchetti di probe basati su pacchetti broadcast di piccole dimensioni. E questa è la pecca.
In questo uso dell'ETX si suppone che le tecnologie utilizzate per la realizzazione dei collegamenti siano omogenee e che, quindi, le statistiche sui pacchetti persi siano indicativi della valutazione dei *path* più efficenti.
Ciò non è sempre vero: utilizzando tecnologie eterogenee non è detto che il collegamento con una percentuale più alta di pacchetti persi sia da considerarsi il peggiore:per esempio un collegamento su linea telefonica cablata, realizzato con un modem analogico (con throughput massimi di pochi kB/s), non perderà quasi mai pacchetti e verrà valutato sempre in ottime condizioni. Esso per questo motivo verrà scelto dal protocollo di routing al posto di qualsiasi connessione senza fili, anche la più performante, perché in essa, per natura, qualche pacchetto andrà sempre perso.

2.2.3 Probing in B.A.T.M.A.N

L'approccio del giovane protocollo B.A.T.M.A.N è significativamente diverso dagli altri protocolli. Come si è visto, ogni nodo contemporaneamente spedisce a tutti i nodi vicini e riceve da questi pacchetti di controllo (chiamati in questo contesto Originator Messages o OGM). Ogni nodo, quando riceve un OGM, incrementa un indice relativo al nodo originator che ha spedito l'OGM e al nodo vicino attraverso cui l'OGM è transitato; tuttavia l'indice è incrementato soltanto con il pacchetto, relativo ad ogni nodo originator, giunto per prima tra tutti quelli inviati dai nodi vicini.

Tuttavia ,l'utilizzo di pacchetti di indagine di tipo broadcast e di dimensione fissa di 52Byte, non esenta B.A.T.M.A.N dalla percezione approssimativa a cui si cerca, con questo lavoro, rimedio.
Pacchetti di indagine di pochi byte non possono garantire una stima approfondita delle condizioni della rete che sia adatta a garantire una scelta di instradamento corretta. Nonostante la scelta di usare pacchetti di piccole dimensioni per il probe della rete sia dettata dal fatto che essi introducano minor overhead di controllo sulla rete rispetto a pacchetti di dimensioni più grandi, essendo granparte delle trasmissioni basate su pacchetti unicast e veicolando, per ogni pacchetto, ingenti quantità di dati, le statistiche sui pacchetti di probe di piccole dimezioni non rispecchiano la reale risposta della rete alle sollecitazioni delle comunicazioni ordinarie.
Si accenna qui alla metodologia proposta per una rilevazione più dettagliata delle caratteristiche che influenzeranno la costruzione della metrica. Come si è accennato nella sezione dedicata ai problemi di indagine della rete (probing), la metodologia che verrà utilizzata in questo lavoro per la costruzione della metrica prevede di forgiare pacchetti di probe nella modalità broadcast e di dimensione variabile. Le dimensioni dei pacchetti di probe del tool segui-

ranno un pattern scelto dall'utente. Nel pattern la posizione degli elementi indica la posizione dei pacchetti nella sequenza di invio mentre il valore indicherà, ripettivamente, la dimensione.
In particolare gli 1 indicano pacchetti piccoli di 32 byte (che simuleranno l'uso dei classici pacchetti di probe), i 2 indicano pacchetti di 800 byte e i 3 indicano quelli da 1400 byte (che simuleranno un uso massiccio della rete). Alternando nella sequenza le tre tipologie di pacchetto potranno farsi considerazioni in merito al fatto che, mentre su collegamenti poco performanti e poco stabili i pacchetti di dimensione maggiore avranno difficoltà a transitare, i pacchetti di piccole dimensioni, simili a quelli usati come probe dei protocolli comuni, passeranno senza alcun problema.
La rilevazione di pacchetti persi nelle statistiche raccolte durante i test saranno pesati in base alla loro grandezza : un pacchetto non giunto a destinazione di dimensione 3 influirà sulla valutazione della metrica in modo più rilevante che un pacchetto perso di dimensione 1.

Capitolo 3

Caratterizzazioni dei link wireless

I collegamenti senza fili, come è 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 è 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, è 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, è 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, è 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 è 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 è 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 è 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 è rilevato nella sezione relativa al problema della Gray Zone[2], 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 , cioè 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 è, 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, è 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 è 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);

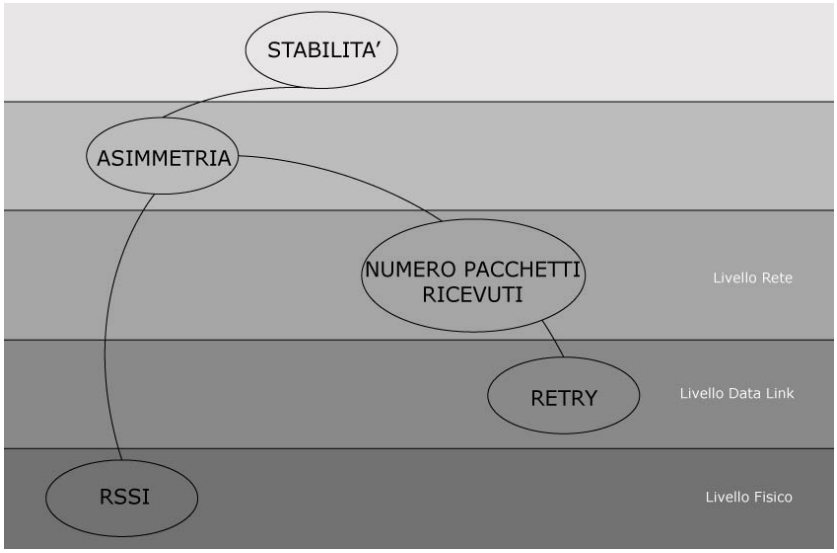
- i nodi iniziano la fase di test in periodi disgiunti; il link è impegnato da tre flussi per ogni coppia in comunicazione, un pacchetto di broadcast e un riscontro in un verso e uno di unicast nell'altro.

Sulla base dei riferimenti temporali, risulta chiaro in fase di analisi dei dati quanti nodi impegnano il canale nello stesso momento e come si è 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 è 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 cioè 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 è in grado di valutare se la percentuale di pacchetti persi è distribuita uniformemente nel tempo, o se c'è stata un'oscillazione con caratteristiche peculiari.Nello svilupo del tool si è 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 è 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.

3.0.4 Stabilità

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



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 è accennato, il grado di stabilità è caratteristica delle comunicazioni soggette a fluttuazioni come quelle basate su tecnologie senza fili. Una delle componenti maggiormente influenti sull'instabilità è la *simmetria* o la sua mancanza in un collegamento: in altri termini, un link che funziona con efficenza diversa nei due sensi della comunicazione è 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 è 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.

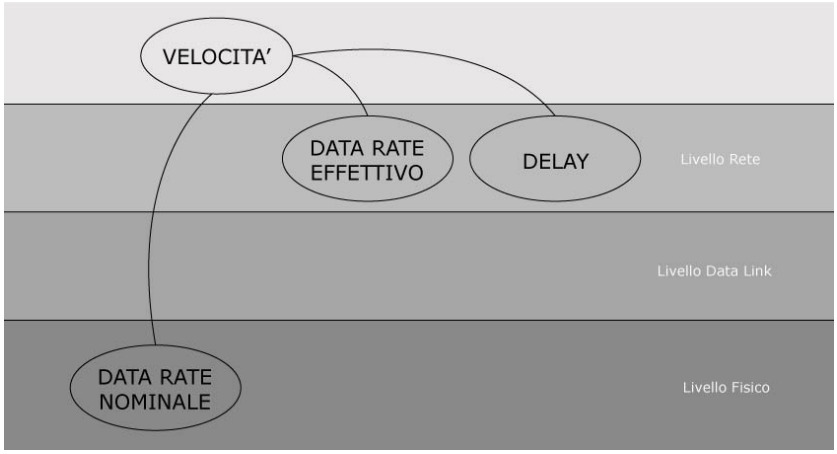
Asimmetria

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- è appunto quello di acquisire una stima dell'energia del segnale ricevuto. Se un link è 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 è 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 è possibile affinare il grado di conoscenza dei link wireless sfruttando le citate informazioni.

3.0.5 Velocità

Nella prossima figura sono suddivise le componenti caratteristiche della velocità.



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 è 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* è una caratteristica riscontrabile tra le informazioni di livello fisico presenti nel *prism header*; è per questo motivo che -anche se l'argomento non verrà trattato in modo esaustivo nella nostra indagine- il tool è 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 Multiplexing).

Nello specifico, ognuno dei rate sopra riportati corrisponde ad una specifica modulazione, come è 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 è 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 è legato (non al singolo link, ma) all'intero percorso attraversato da un flusso di pacchetti. Per questo motivo non ci si è 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 è noto, si

può sapere quando un pacchetto è stato recapitato per intero ma non quando è 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

Il 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 più 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 bontà di un link non rispecchia del tutto la realtà. 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 più 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 è possibile trarre delle

conclusioni accurate riguardo l'effettiva bontà del link per poi metterle a confronto 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 è uno strumento in grado di effettuare test su reti mesh wireless e supportare le scelte tecniche per un loro impiego più efficace.

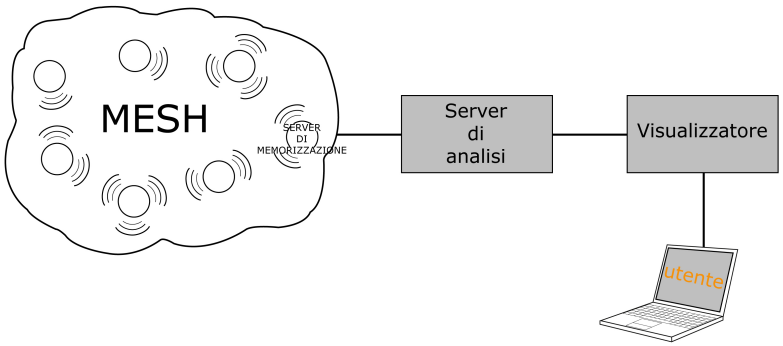
Il tool è stato interamente realizzato nel linguaggio di programmazione C e sviluppato su piattaforma Linux x86. La scelta di piattaforme Unix like è stata dettata in parte da affinità concettuali, ma soprattutto dall'esigenza di avvalerci di strumenti quali il PRISM HEADER, l'RADIO TAP HEADER e le list 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 è 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 è perlopiù costituita da apparati Linksys WRT o ASUS DW500 sui quali è stato necessario crosscompilare il tool per piattaforma MIPSEL.

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

4.1 metodologia

Il tool prevede 4 segmenti : *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 è stata fatta è che i nodi siano temporalmente sincronuzzati tra loro almeno al secondo (per questo ci si è affidati al servizio NTP).



Sui nodi la fase di test è 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 è sentito almeno un pacchetto di broadcast ed un eventuale tempo di inattività. 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 servirà 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 più 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. Può comunicare con i nodi con tre tipi di pacchetti : una richiesta di start_test, più 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 è del tutto personalizzabile rispetto alle esigenze che si hanno e può essere più o meno complesso. Si possono infatti implementare delle query che calcolino automaticamente se il path deciso dal protocollo di routing che gira sulla rete è effetivamente il più rapido o il più stabile nel tempo confrontando le MIB su tutti i nodi con le informazioni ottenute dal tool. Si può chiedere di calcolare secondo i dati del tool in un dato momento qual l'albero ricoprente migliore per raggiungere da un nodo ongni altro nodo

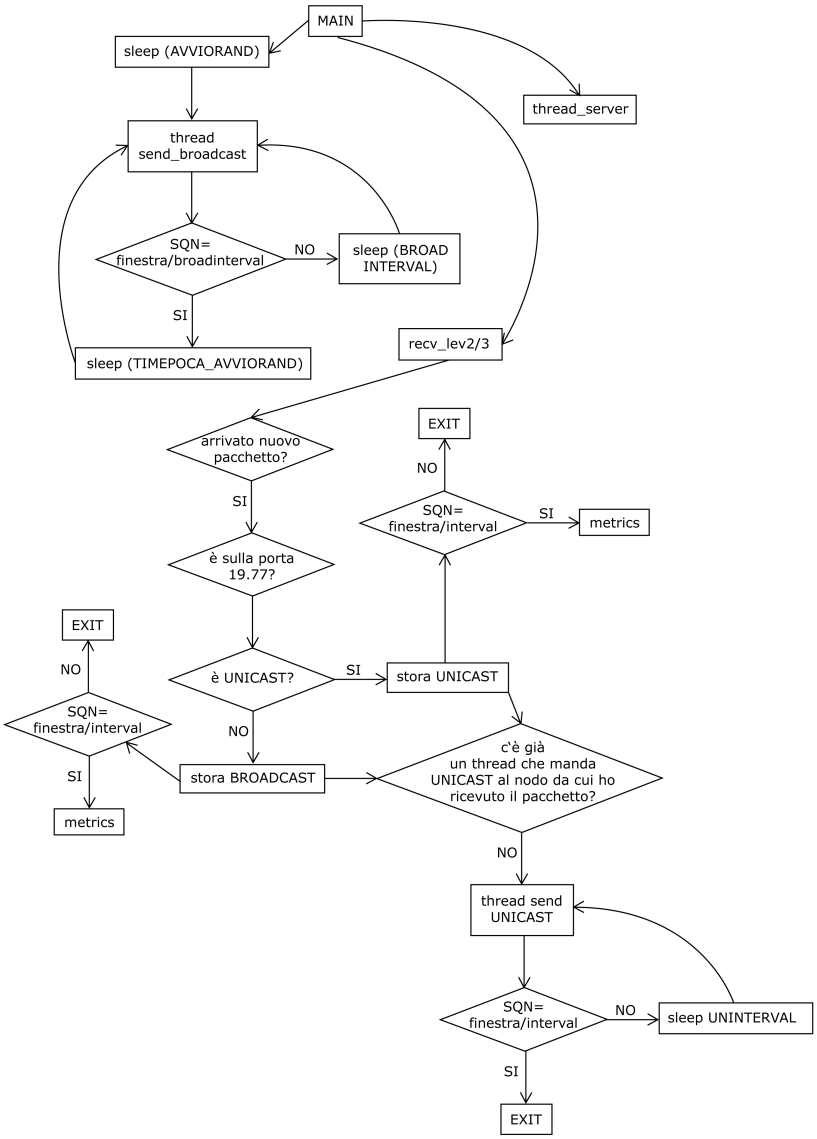
della rete e confrontarlo con quello calcolato nello stesso momento dal protocollo di routing. Oppure si può semplicemente richiedere dei dati statistici sulla stabilità o la velocità nel tempo di alcuni link. Questi sono solo alcuni esempi di query che si potrebbero implementare.

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 è 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 è come illustrato in figura.



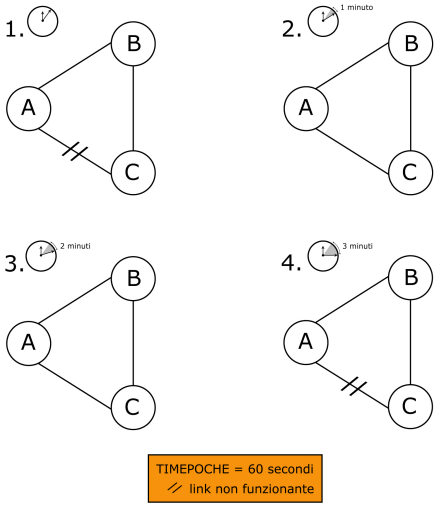
I test su ogni nodo sono divisi in epoche a partire dall'epoca 0 per ogni vicino. L'epoca, di cui si illustra il concetto successivamente, l'ip del mittente del pacchetto e il tipo di pacchetto identificano univocamente l'esito di un probe su un link tra 2 nodi. Queste informazioni unite ai dati presi 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

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 è illustrato di seguito.

Per ogni nodo vicino di cui si è 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 è individuata sul nodo ricevente da un numero di EPOCA. Il concetto di epoca è quindi legato a chi riceve i pacchetti.

L'epoca è 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 invierà 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 può osservare dalla FIGURA 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

La causa del disallineamento delle epoche sul nodo A relative ai probe con i nodi B e C potrebbe essere dovuta all'inoperabilità del link tra C e A durante i cicli di test 1 e 4. Per semplicità nell'esempio è 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.

L'SQN è il numero di sequenza del pacchetto ed ha unimportanza fondamentale nella memorizzazione dei risultati dei probe. C'è 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 può assumere i valori 'u' per unicast e 'b' per broadcast;

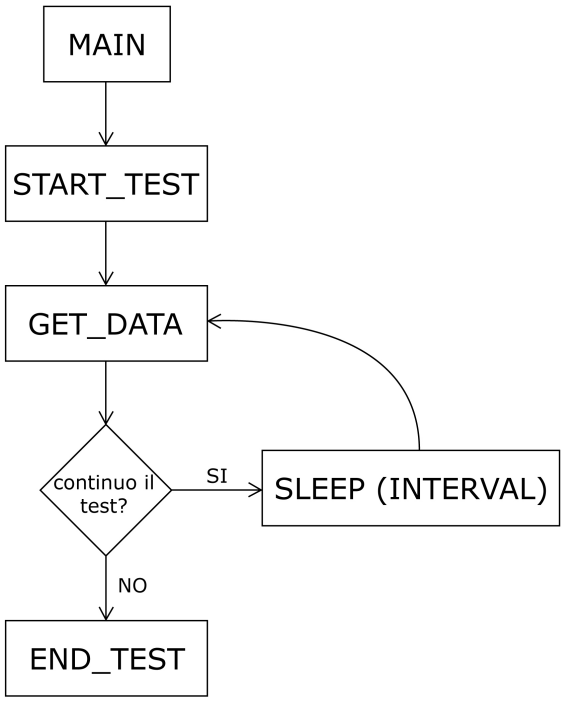
- SEC e USEC indicano in secondi e millisecondi quando è stato inviato il pacchetto; attualmente non viene utilizzato, ma ipotizzando una perfetta sincronizzazione della data di sistema tra i nodi della rete, si potrebbero utilizzare per calcolare il delay del link in esame;

- Il PADDING serve a generare pacchetti di dimensione arbitraria ed è riempito con 0. Si potrebbe utilizzare in futuro se si ha la necessità di inviare altri dati.

Per ora è possibile lanciare il tool su un nodo per una sola interfaccia di rete. Non si è presentata la necessità di implementarlo per nodi con più 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 già accennato, tale server si occupa di memorizzare i dati ricevuti dai nodi della rete mesh su cui è in esecuzione il tool. Il flusso di esecuzione è quello illustrato in figura.



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. È 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. È consigliabile che possa comunicare con più 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 è del tipo mostrato in FIGURA5a

Pacchetto start_test

0	SEQ	UNINTERVAL	BROADINTERVAL	EPOCHE	TIMEEPOCA	FINESTRA

Pacchetto get_data

1

Pacchetto end_test

2

- 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 è di ipo FIFO.

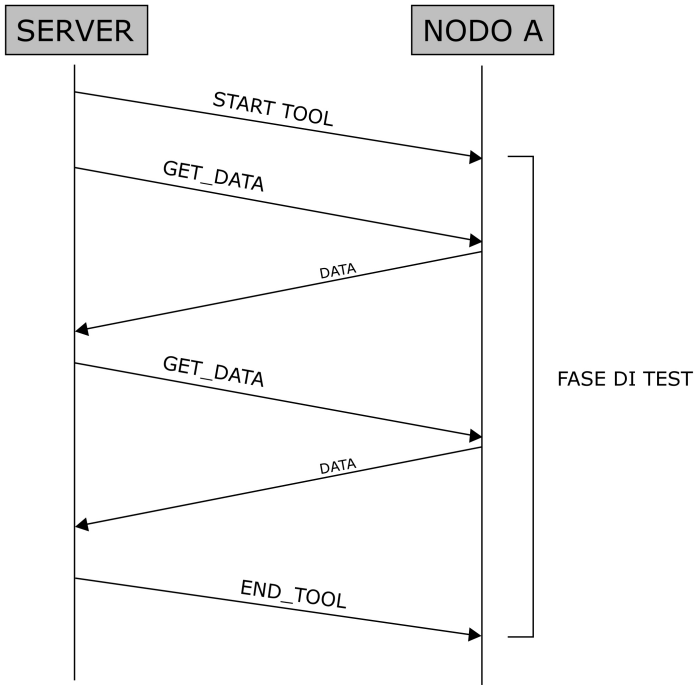
- il campo FINESTRA indica quanti secondi deve durare la fase attiva di invio pacchetti di broadcast. È strettamente legato a BROADINTERVAL e al SEQ e lo andiamo ad illustrare con un esempio. Ipotizziamo che FINESTRA sia di 30 secondi; se BROADINTERVAL fosse 2 invieremmo un pacchetto di broadcast ogni 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 sarà il tempo che trascorrerà 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 può essere inviato un numero arbitrario di volte.

La comunicazione tra client e server avviene per mezzo dell'instaurazione di una connessione TCP/IP classica. Di seguito si può osservare un esempio di scala temporale relativa a tale comunicazione.

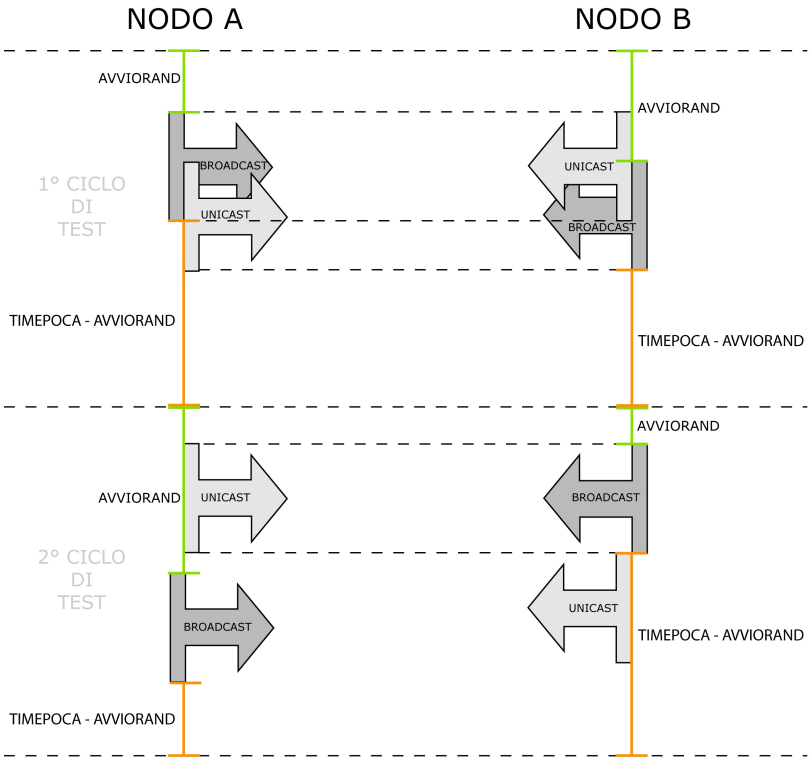


4.1.4 comunicazione tra i nodi

Come già 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 è di 30 secondi e UNIINTERVAL è pari a 1 secondo, il nodo invierà 30 pacchetti di unicast con numero di sequenza incrementale da 1 a 30 ogni secondo.

Per la comunicazione tra i vari nodi abbiamo implementato una SOCKET RAW. Tale scelta è stata obbligata dalla necessità 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 è 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.



La figura illustra un esempio di applicazione del test non eseguita in continuo. È possibile modificare i parametri del tool affinché i test vengano fatti in continuo senza intervalli di inattività tra un'epoca e l'altra. Basta impostare AVVIORAND pari a 0, e FINESTRA uguale a TIMEEPOCA.

livello2 - livello3

Per avere le informazioni dal livello fisico della scheda di rete abbiamo castato i pacchetti ricevuti al livello fisico tramite il PRISM HEADER . C'è da dire che il PRISM HEADER non è universale per tutte le schede di rete, poiché è stato concepito per schede con chipset ATHEROS, BROADCOM e PRISM. Del PRISM HEADER tratteremo più specificatamente nei prossimi paragrafi. Per le schede con chipset INTEL quali ipw2200 e ipw3386 c'è 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 è stato implementato nessun meccanismo di monitoring del canale. Per tali schede si è 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. Ciò 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 è parte del file prism.h. Si possono notare le voci RSSI, Data_Rate e Frame_Control all'interno del quale è 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 è 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 è 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* è una sockaddr_in contenente l'IP del nodo da cui si è ricevuto il pacchetto;
- *type* 'u' se il pacchetto ricevuto è di unicast e 'b' se il pacchetto ricevuto è di broadcast;
- *check_t_unicast* indica se c'è già un thread che sta inviando pacchetti di unicast verso il nodo da cui si è ricevuto il pacchetto di broadcast;

- *epoca* indica per quel nodo in quale ciclo di test siamo. Inizialmente è 0 per tutti i nodi. Con il susseguirsi dei cicli di test tale valore viene incrementato di una unità;

- *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 è 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 è 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 è stato ricevuto il pacchetto inviato dal nodo X con numero di sequenza i;

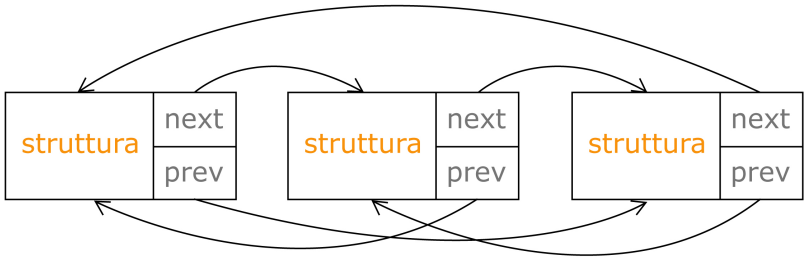
- in *uretry* viene memorizzato in posizione i il numero di retry relativi al pacchetto i inviato dal nodo X e ricevuto dal nostro nodo. Tale tipo di buffer è presente soo nella struttura unicast poiché il broadcast non in implementa il meccanismo di ritrasmissione dei paccheti;

- L'ultimo campo è la *list_head* di cui tratteremo più avanti.

struct unicast{

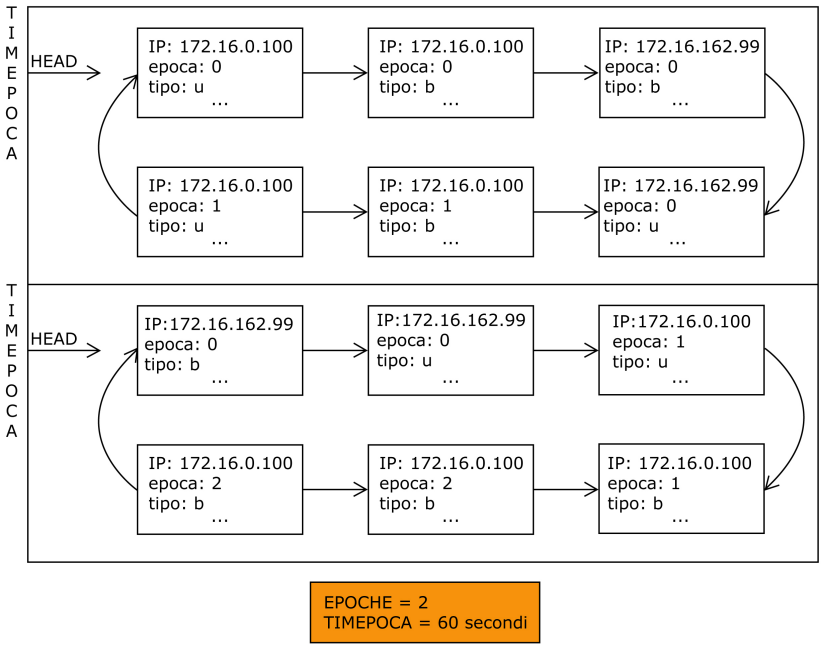
```
char ubuffer[FINESTRA];
short urate[FINESTRA];
short usize[FINESTRA];
short urssi[FINESTRA];
short uretry[FINESTRA];
};
struct broadcast{
char bbuffer[FINESTRA];
short brate[FINESTRA];
short bsize[FINESTRA];
short brssi[FINESTRA];
short padding[FINESTRA];
};
struct neigh_node{
char type;
int check_t_unicast;
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 è scelto di gestire la memorizzazione dei dati tramite le LIST HEAD di Linux per la loro efficienza e provata affidabilità. Sono infatti implementate nel kernel di Linux e utilizzate dallo scheduler dei processi. Si tratta di liste circolari doppiamente collegate[riferimento].



Nel tool vengono utilizzate per memorizzare i risultati dei probe sui vari link adiacenti al nodo in esame. Ogni entry relativa ad un'epoca,un ip e un tipo di traffico. Non è possibile avere due o più entry con stessa epoca, ip e tipo. Forse i "databasari" 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 più vecchi vengono cancellati per far spazio a quelli più 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 può trovare la voce :

struct list_head neigh_list;

che non è 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 è possibile utilizzare il PRISM HEADER è 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 è la struttura neigh_node già utilizzata sui nodi che serve per castare i dati ricevuti e poi stamparli a video.

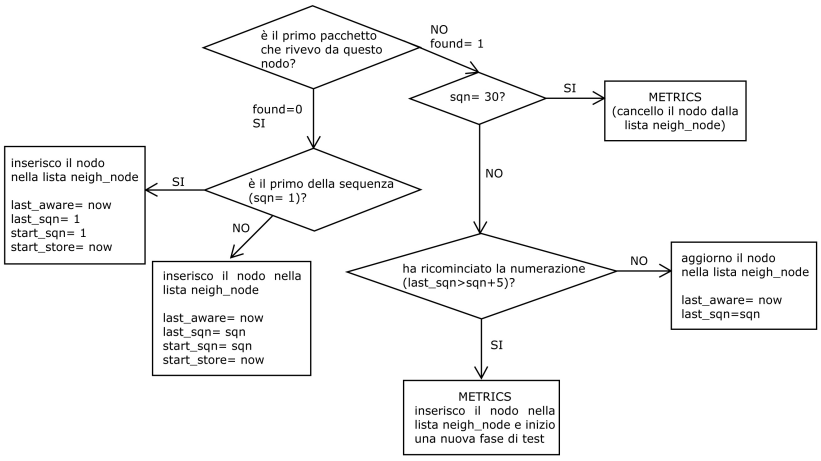
La struttura from_server viene utilizzata per preparare le informazioni che verranno spedite ai nodi della rete.

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 grande la finestra di attivit
};

4.3 algoritmi

4.3.1 Ricezione dei pacchetti

Di seguito viene illustrato come un nodo gestisce un pacchetto, unicast o broadcast, appena ricevuto.



4.3.2 epoche

Un'epoca è identificata temporalmente dalle variabili *start_aware* e *last_aware*. La numerazione delle epoche relative a sessioni di broadcast è 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 risponderà con una serie di pacchetti di unicast e creerà 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 risponderà con una serie di pacchetti unicast; anche in tal caso A creerà una nuova entry per la lista neigh_node con ip di B, epoca 0 ma tipo u. Quindi è nella lista neigh_node saranno presenti due entry con lo stesso ip, stessa epoca ma tipo differente.

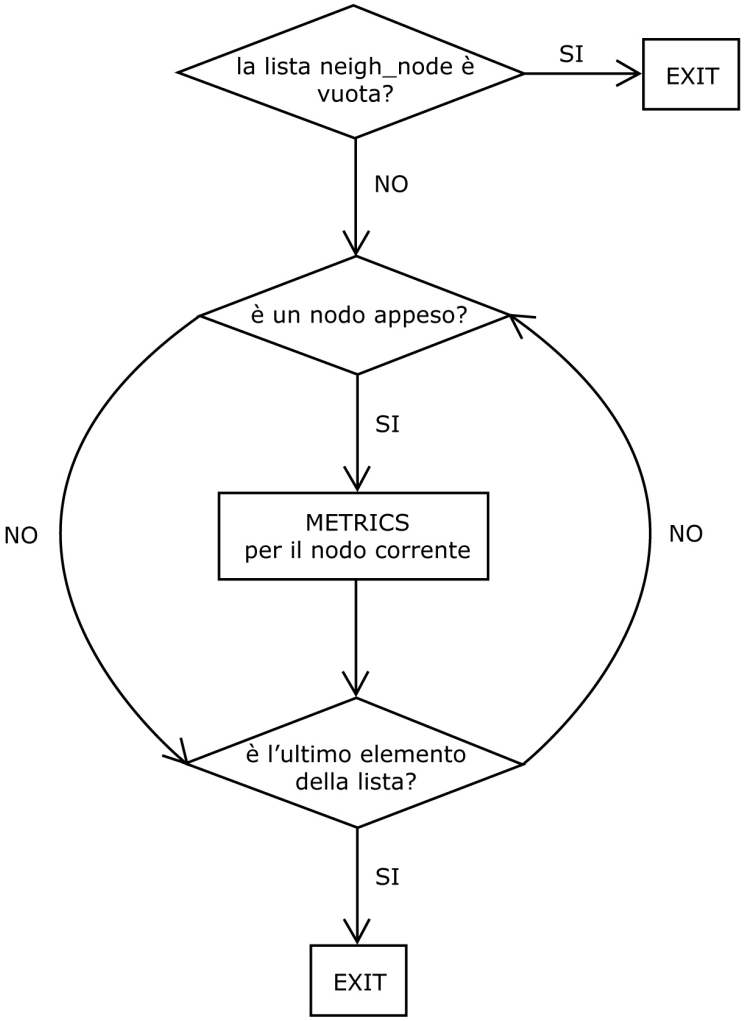
Ogni epoca ha la durata massima di TIMEPOCA secondi e si possono memorizzare sino ad un massimo di 255*2(broadcast e unicast) epoche per ogni nodo a discapito della memoria. È quindi possibile memorizzare

l'andamento di un link per 255*TIMEPOCA secondi. È ovvio che sta al buon senso dell'utente che utilizzerà il tool il compito di dimensionare a modo i parametri di test.

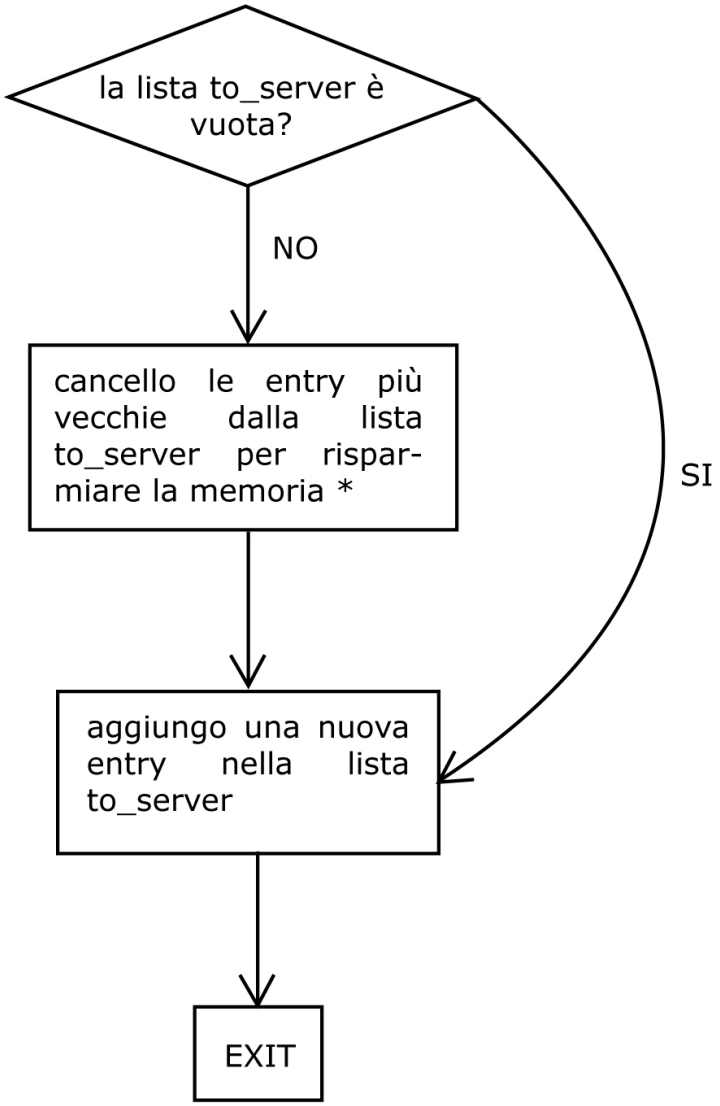
4.3.3 purge e metrics

La funzione sui nodi che rende disponibili i dati per le richieste del server è metrics. Tale funzione provvede a cancellare una entry dalla lista neigh_node e inserirli in un'altra lista di tipo LIST HEAD che verrà spedita al server. È 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.



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 relative a tale probe rimarrebbero "appese" senza mai essere preparate e spedite al server. È stato quindi implementato un meccanismo di rintracciamento delle entry "appese" che viene lanciato ogni fine sessione di test prima di iniziare una nuova sessione.



* Serve a cancellare le statistiche relative ai test più vecchi nel caso in cui il server di memorizzazione non riesca a raggiungere il nodo per lunghi periodi di tempo.

4.4 tunabilità

Uno dei punti forti del tool è la sua alta personalizzabilità a seconda delle esigenze di chi vuole testare la rete. Agendo sui paramentri di configurazione

è 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 così la possibilità di costruire dei test specifici.
Il tool può 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 più dettagliate.

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

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 è ricevuto almeno un pacchetto di broadcast. Notare che la lunghezza di SEQ è divisore di (FINESTRA/BROADINTERVAL) e di (FINESTRA/UNIINTERVAL). Tale tipo di test può essere utile per stressare continuamente la rete e vedere come reagisce il protocollo di routing.

Un altro tipo di test continuo un pò 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 può 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.

è possibile agire sulla dimensione dei pacchetti e sulla sequenza SEQ per simulare un determinato tipo di traffico. Ad esempio è 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

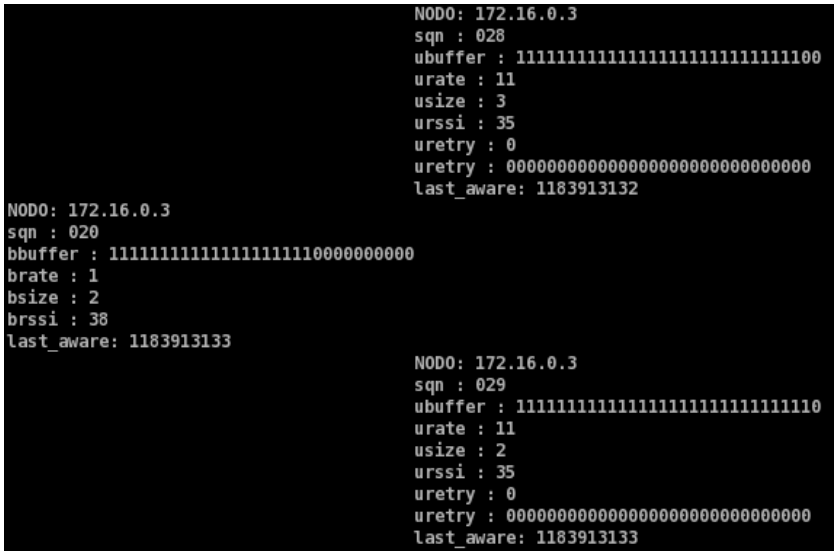
Con tale approccio si può simulare sulla rete mesh un qualunque tipo di traffico.

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 è un pò diverso e più 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 è 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 più efficienti.

4.5 rilettura dei dati

Sia sui nodi è possibile impostare 2 livello di debug. Il livello 1 stampa ad ogni ricezione di pacchetto inviato dal nodo X tutti i buffer relativi al nodo X dell'epoca corrente. Il livello 2 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 è però sconsigliato utilizzare il debug di livello 1, per questioni di memoria.

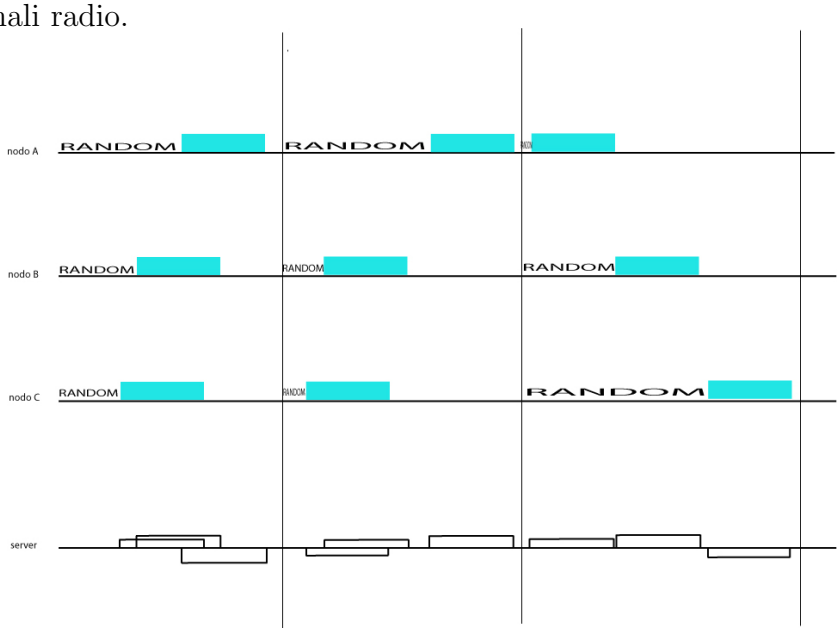


Sul server è previsto un solo livello di debug. Ad ogni ricezione di pacchetto stampa tutti i dati di tutti i nodi di cui è in possesso.

```
nodo_unicast_ip:172.16.0.100    last_aware:1183913135   start_aware:1183913114  epoca:8         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    0    0    0    0    0    0    0    0    0
urate :    001  001  001  001  001  001  001  001  001  001  001  001  001  001  001  002  001  001  001  001  001  000  000  000  000  000  000  000  000  000
usize :    003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  002  001  003  000  000  000  000  000  000  000  000
urssi :    -60  -57  -52  -44  -46  -63  -66  -73  -64  -61  -60  -72  -61  -66  -66  -69  -63  -63  -58  -60  -61  -64  000  000  000  000  000  000  000  000
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:172.16.0.100  last_aware:1183913134  start_aware:1183913105  epoca:8         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 :    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
brssi :    -61  -60  -64  -61  -54  -54  -58  -72  -57  -58  -61  -49  -48  -49  -64  -66  -69  -64  -66  -64  -73  -70  -61  -66  -66  -73  -66  -72  -75  -76
nodo_unicast_ip:172.16.0.100   last_aware:1183917000  start_aware:1183916971  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    1    1
urate :    054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054  054
usize :    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
urssi :    -66  -66  -66  -69  -67  -70  -72  -67  -64  -64  -65  -64  -65  -64  -64  -63  -64  -64  -65  -65  -65  -66  -65  -64  -64  -65  -64  -65  -64  -64
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:1183916998  start_aware:1183916971  epoca:0         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    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    1    1    1
brate :    000  000  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 :    000  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  003  002  001
brssi :    000  000  036  036  035  034  035  031  033  035  038  037  037  037  035  036  037  037  038  038  035  038  036  037  037  035  036  038  037  037
nodo_broadcast_ip:172.16.0.100  last_aware:1183917007  start_aware:1183916978  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    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 :    -73  -78  -58  -58  -58  -58  -59  -58  -62  -58  -72  -59  -56  -72  -58  -70  -58  -58  -61  -56  -77  -61  -58  -66  -70  -58  -72  -70  -70  -70
nodo_unicast_ip:172.16.0.3     last_aware:1183917007  start_aware:1183916977  epoca:0         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  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  018  024  024  024  024  024  024  024
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 :    035  036  033  033  032  033  032  033  033  033  035  033  034  033  031  030  034  033  032  031  032  032  033  031  030  031  030  031  031  031
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
```

4.6 esempio scala temporale

Di seguito viene mostrato un esempio di successione temporale di esecuzione del tool su più nodi contemplando contemporaneamente gli invii e le ricezioni simultanee. Come si nota dalla figura seguente l'avvio random associato alla presenza di più nodi consente di analizzare situzioni differenti del carico dei canali radio.



Capitolo 5

Test

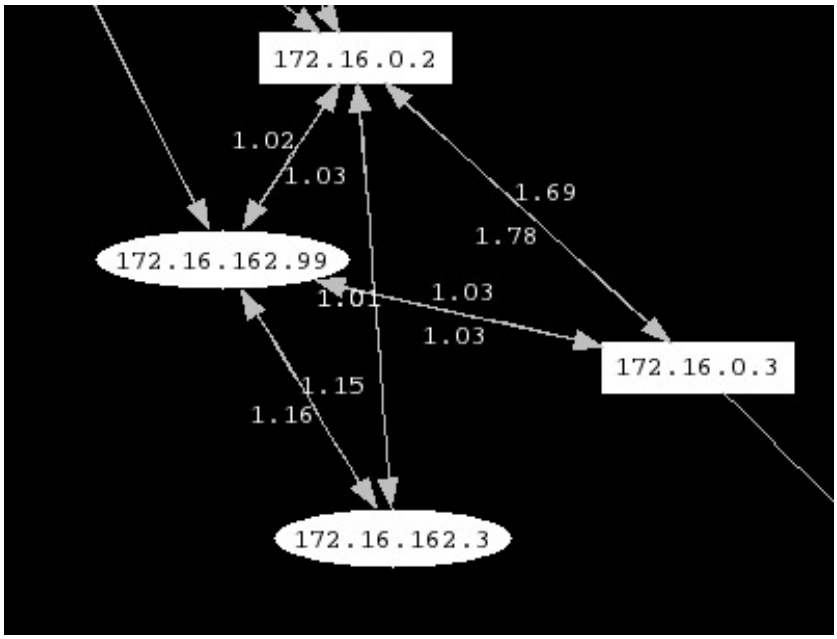
In questo capitolo si analizzeranno i dati raccolti durante differenti sessioni di test; il primo test che verrà effettuato prenderà in esame la rilevazione del fenomeno chiamato Gray Zone, gli altri due, invece, mostrano come i probe utilizzati dai protocolli di routing esistenti per reti mesh potrebbero valutare meglio lo stato dei link rispetto a quanto non facciano.
Ad illustrazione degli esiti delle prove, in questa trattazione sono riportati alcuni *screenshot* dei dati raccolti dal server; si ricordano, perciò, i significati delle sigle che compaiono:

nodo_broadcast_ip o nodo_unicast_ip	il nodo che ha spedito la sequenza di pacchetti
last_aware	il timestamp dell'arrivo dell'ultimo pacchetto della sequenza ricevuto
start_aware	il timestamp dell'arrivo del primo pacchetto della sequenza ricevuto
epoca	il numero dell'epoca della finestra di test
type	tipo di test (b=broadcast u=unicast)
NODO_SORGENTE	nodo che ha ricevuto i pacchetti e che ha storato le relative statistiche
n.pack	numero di sequenza pacchetto
bbuffer o ubuffer	sequenza di ricezione pacchetti (1=arrivato, 0=non arrivato)
brate o urate	*data* rate nominale
bsize o usize	dimensione pacchetto (1=81 byte, 2=749 byte, 3=1449 byte)
brssi o brssi	RSSI ricezione pacchetto
uretry (solo per gli unicast)	numero ritrasmissioni n-esimo pacchetto

I *testbed* (ambiente e condizioni del test) utilizzati per condurre i test sono stati collocati in porzioni della rete OLSR della wireless community network di Roma, nota con l'identificativo di ninux.org (www.ninux.org); è per questo che i confronti saranno valutati solamente con la metrica ETX di OLSR (implementato su questa rete) e non con quella di B.A.T.M.A.N, anche se le valutazioni avrebbero carattere del tutto analogo.

5.1 Rilevamento della Gray Zone

In questa sessione di test si vedrà come il tool è capace di raccogliere informzioni che consentono di stabilire quando e dove si è verificato il fenomeno della Gray Zone (della quale si già in precedenza trattato).
Il testbed a cui si farà riferimento è illustrato nella seguente figura che rappresenta parte della rete ninux.org, per mezzo della topologia generata da OLSR.



5.1.1 Condizioni di test

I nodi utilizzati nel test sono di due tipi specifici: Access Point Linksys WRT54GL (172.16.162.3, 172.16.0.2, 172.16.0.3) e laptop Pentium3 con Slackware Linux (172.16.162.99). All'ultimo di questi nodi è stato attribuito il ruolo di server per immagazzinare i dati e per rilevare le topologie OLSR.

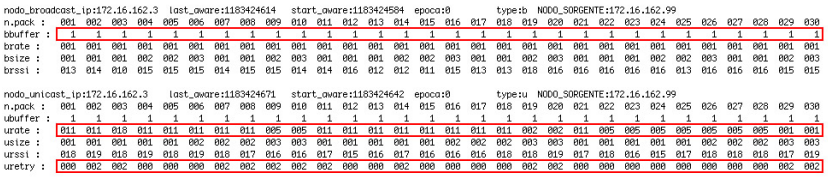
5.1.2 Il test

Durante lo sviluppo del tool sono state effettuati sessioni di test per controllare il funzionamento del software elaborato. Tra le prove di cui si sono conservati i dati, si riportano due casi emblematici in cui si è rivelato il fenomeno della Gray Zone.

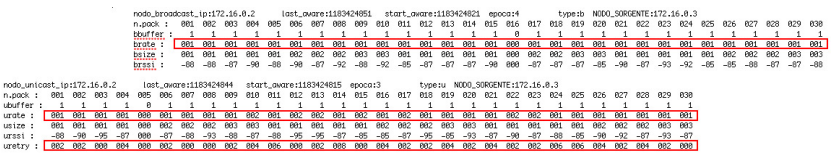
Come è stato detto nella relativa sezione, ci si aspetterebbe che il fenomeno della Gray Zone si manifesti in una determinata zona della rete in cui i pacchetti spediti da un nodo in broadcast arrivino senza alcun impedimento e che i corrispettivi pacchetti unicast trovino, invece, difficoltà a transitare.

5.1.3 Risultati

Il primo caso isolato mostra un collegamento, quello tra il nodo 172.16.162.3 e il nodo 172.16.162.99, in cui il fenomeno della Gray Zone si manifesta in maniera poco evidente: i pacchetti spediti in uniscat con rate alti arrivano a destinazione solamente perché sono stati ritrasmessi, conservando tuttavia, un valore del rate più alto di 1 Mb/s. Il che significa che Il nodo 172.16.162.99 si trova sul limite della Gray Zone del nodo 172.16.162.3 perché i pacchetti ricevuti, conservano ancora il rate alto, ma devono essere necessariamente ritrasmessi per poter giungere a destinazione. Questa condizione è riscontrabile nel seguente *screenshot*:



Il secondo esempio di Gray Zone, invece, prende in esame due nodi in cui uno dei due si trova evidentemente nella Gray Zone dell'altro.
Come è chiaro dall'analisi dei dati (vedi prossimo *screenshot*), gli invii di pacchetto contemporanei dal nodo 172.16.0.2 al nodo 172.16.0.3 nel caso dei pacchetti broadcast sono giunti a destinazione senza alcun problema, quelli unicast si sono persi ai rate più alti, costringendo il meccanismo di riadattamento del rate a ripetere l'invio a rate minimo di 1Mb/s. Il driver delle schede wireless quando spedisce pacchetti unicast abbassa il rate solamente perché non ha ricevuto pacchetti di riscontro (ACK) ai pacchetti inviati ai rate più alti.



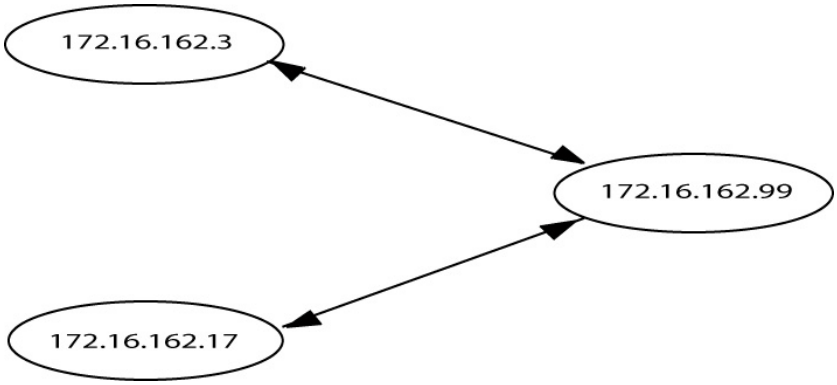
5.2 Stima della metrica nelle stesse condizioni (broadcast) di OLSR e B.A.T.M.A.N ma con pachetti di probing di dimensioni diverse

Nella sessione di test che segue ci si concentra sull' utilizzo di pacchetti di probe di grandezza variabile per valutare la reattività di un collegamento wireless in una rete mesh. Ciò consente di svantaggiare, nella valutazione di una metrica, i collegamenti non in grado di trasportare correttamente i pacchetti di grosse dimesioni.

Facendo così si valutano le proprietà dei collegamenti più dettagliatamente di quanto non si possa fare con pacchetti di dimensione fissa.
Se un collegamento è in grado di trasportate pacchetti da 32 byte, ma non è in grado di veicolare pacchetti da 1500 byte, esso è penalizzato nel calcolo della relativa metrica a causa della più alta percentuale di pacchetti persi.
In questo test ci si aspetta, quindi, di isolare un caso in cui il probing di OLSR e di B.A.T.M.A.N, emulato, utilizzando i pacchetti di probe tutti di broadcast e della dimensione di 32 byte, cotruisce metriche imprecise; di conseguenza esso sceglierebbe impropriamente un certo path come ottimo, mentre il tool, utilizzando pacchetti di broadcast di dimensione variabile, fa notare come sullo stesso collegamento i pacchetti di più grosse dimensioni vadano persi, e quindi come la scelta di un altro path potrebbe risultare più opportuna.
Volendo analizzare quanto più efficaci, ai fini del test, siano i pacchetti di probe di dimenzioni variabili rispetto a quelli di dimensione fissa, saranno effettuate prove sullo stesso link con una *distribuzione* di pacchetti aventi dimensioni diverse

5.2.1 Condizioni di test

In concreto, prendiamo in esame la seguente figura che ci mostra due link del nodo 172.16.162.99 con i nodi 172.16.162.3 e 172.16.162.17. Per simulare il grado di percezione di un protocollo di routing che, per costruire una metrica, utilizza solamente dati di *Livello Rete*, si fa riferimento nella rilettura dei dati solamente alle informazioni disponibili a *Livello Trasporto* e *Livello Rete*; nella fattispecie si consulta solamente la statistica sulle ricezione dei pacchetti e sulla dimensione di essi.



Si riportano di seguito le impostazioni di OLSR significative di tutti i nodi utilizzati per effettuare il test:

Parametri	valore	significato
TcRedundancy	2	Nei pacchetti TC (Topology Control) ogni nodo inserisce le informazioni di tutti i suoi vicini
MprCoverage	3	Il numero minimo di MPR che deve essere eletto da ogni nodo
LinkQualityFishEye	0	Scaling Fish Eye disabilitato
LinkQualityWinSize	30	Finestra di pacchetti sulla quale è calcolata la metrica ETX
LinkQualityLevel	2	ETX utilizzato per la scelta dei MPR e per il routing

Nella tabella è da notare che la metrica ETX è calcolata su una *finestra* statistica (LinkQualityWinSize) degli ultimi 30 pacchetti: questo ci consente un confronto con i dati del tool che utilizza finestre temporali di test della consistenza di 30 pacchetti.

Durante la fase di test si è applicato il tool ai tre nodi in questione e si sono immagazinate le informazioni ricavate nel server installato sul nodo 172.16.162.99. Contemporaneamente è stato utilizzato uno *script* per generare una topologia OLSR per ogni ciclo di test.

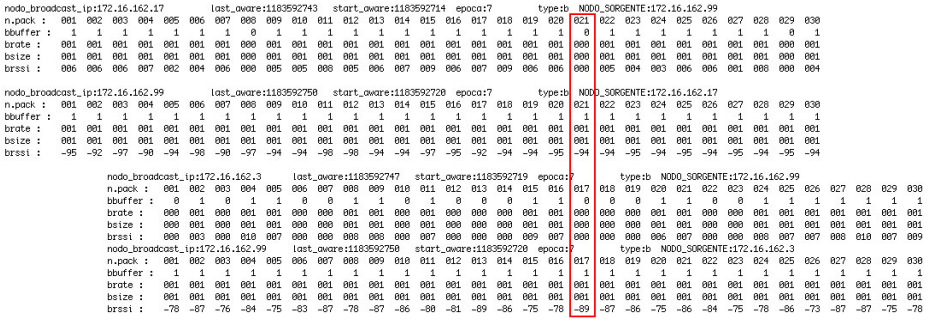
Il confronto con le scelte operate da OLSR ci serve per capire quale percezione ha il protocollo della bontà dei link, anche se esso non utilizza probe di tipo evoluto.

5.2.2 I test

Si sono eseguite sessioni di test differenti sugli stessi link, utilizzando per ciascuna sessione una serie (o finestra) di 30 pacchetti broadcast: in essa la dimensione di ogni pacchetto spedito varia secondo un a sequenza (*pattern*) configurata dall'utente. Nel suddetto *pattern* l'invio un *1* indica la spedizionne di un pacchetto di 32 byte di dati, un *2* quella di un pacchetto da 700 byte e un *3* indica un pacchetto da 1400 byte.
È stato eseguito per prima il test con il pattern più semplice, che meglio simula l'uso sulla rete dei probe propri dei protocolli di routing OLSR e B.A.T.M.A.N:

111111111111111111111111111111.

OLSR, infatti usa come probe pacchetti di 28byte per inviare i suoi segnali di sonda, (*HALLO message*), mentre B.A.T.M.A.N usa per probe pacchetti di 56 byte totali, compresi gli overhead dovuti agli header IP e UDP.
Come è possibile notare dal sequente *screenshot* (riquadro rosso) dei dati raccolti sul server nel primo test,



le perdite dei pacchetti sono casuali. Non è possibile notare alcuna fluttuazione caratteristica dovuta al tipo di traffico: il ventunesimo pacchetto della sequenza, per esempio, è stato perso in entrambi i collegamenti, sia tra 172.16.162.17 e 172.16.162.99 che tra 172.16.162.3 e 172.16.162.99, a causa una interferenza momentanea che ha interessato ambedue le comunicazioni verso il nodo 172.16.162.99.

Con questo tipo di probe, perciò, i protocolli di routing non mostrano di avere una piena percezione della capacità del link di trasportare pacchetti di grosse dimensioni come quelli in uso, per esempio, nei servizi di trasporto file.

La decisione di routing dei detti protocolli non tiene quindi conto della effettiva possibilità di veicolare comunicazioni differenti a seconda del tipo di servizio implementato (Es. ftp, telnet, ecc.) .

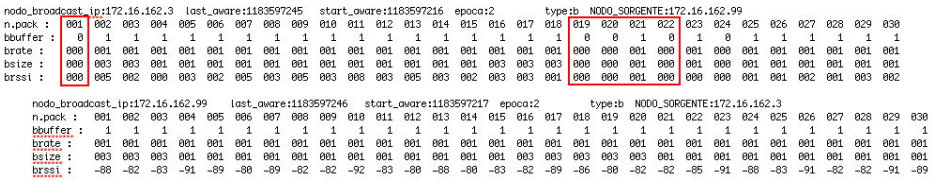
Ci si concentra ora solamente su uno dei due link, quello che ha dimostrato maggiore instabilità .

Nel test successivo è stato utilizzato il pattern:

3331111111111113333311111111111.

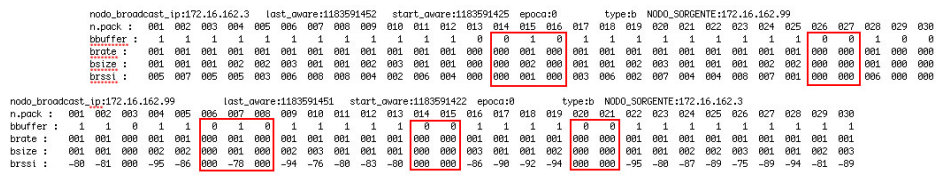
Qui, come si vede dai simboli, si introducono pacchetti della dimensione massima trasportabile da una connessione 802.11b/g; su questa tecnologia di solito il valore dell' *MTU (Unit'a Massima di Trasporto o Maximum Trans-*

port Unit) è fissata a 1500 byte. Il nostro tool, utilizza pacchetti di 1400 byte di dati più l'overhead di 28 byte per gli header IP e UDP e 21 per l'header del tool, al fine di sovraccaricare i collegamenti. Questa metodologia consente di portare alla luce la reale capacità di ogni link di trasportare dati.
Come si può notare nei riquadri evidenziati in rosso nello *screenshot* del server, l'invio di pacchetti di 1400 byte (i primi della sequenza e poi quelli dal quindicesimo al ventesimo) stressa il collegamento fino a far perdere alcune delle trasmisioni.



è da notare in particolare come la variazione della dimensione dei pacchetti spediti influisca negativamente anche sulla trasmissione dei pacchetti successivi a quellii di grossa dimensione.
Il prossimo esempio mostra, inoltre, che la perdita dei pacchetti è maggiore quando, nella sequenza, la dimensione dei pacchetti inviati varia repentinamente; il terzo pattern utilizzato per condurre il test fornisce una prova di questo comportamento.
Il pattern che si è utilizzato è conformato secondo lo schema:

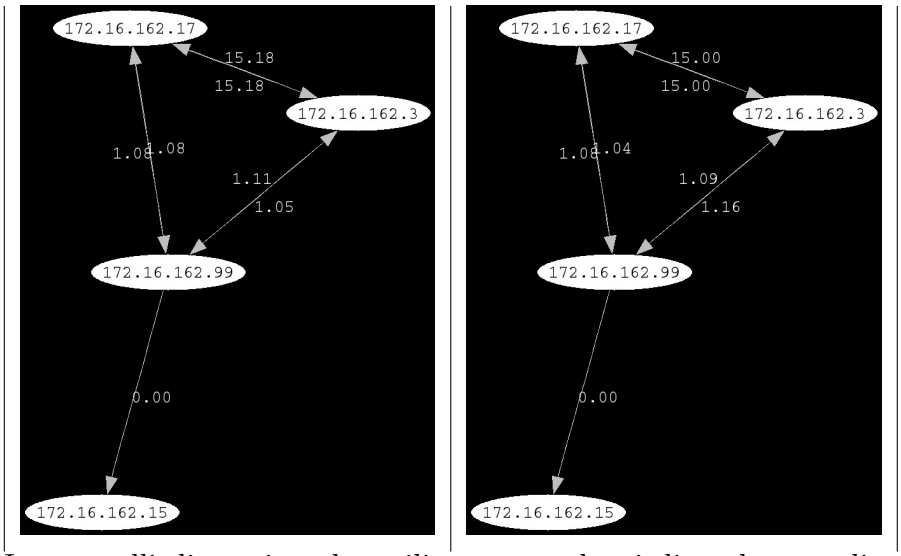
1112231123111223112311122311 23.



Come è evidenziato dal riquadro, sul link tra il nodo 172.16.162.99 e il nodo 172.16.162.3, la comunicazione è inibita chiaramente dalla presenza di pacchetti di 1449 byte. È possibile notare come la perdita di pacchetti non dipenda da una fluttuazione momentanea del collegamento ma solamente dalla *quantità* di dati spediti. Essendo aumentati i pacchetti di grosse dimensioni rispetto al test precedente, nella finestra si rileva in corrispondenza un numero minore di pacchetti giunti a destinazione.

5.2.3 Risultati

Come si può vedere, i collegamenti non si dimostrano stabili allo stesso modo se utilizzati per inviare pacchetti di dimensioni diverse. Inoltre, la variazione repentina della dimensione dei pacchetti spediti compromette significativamente la trasmissione dei pacchetti stessi su link instabili: perciò, valutare la qualità dei collegamenti solo sulla base di pacchetti di uguale dimensione, come usa fare con i protocolli di routing *attuali*, costituisce un approssimazione grossolana del reale stato della rete.
Le stime della qualità del collegamento posto in esame, costruite sulla base delle statistiche delle ricezioni dei probe del protocollo OLSR, evidenziano infatti, tranne qualche momentanea fluttuazione, sempre lo stesso stato del link: come si vede nelle seguenti topologie della rete salvate nella prima e nella terza sessione di test.



I protocolli di routing che utilizzano pacchetti di probe con dimensione fissa non sono quindi in grado di stabilire se un link è migliore di un altro, a menocché i due collegamenti non differiscano fortemente per capacità di trasportare pacchetti di piccole dimensioni.

Le decisioni di routing sarebbero pertanto molto più accurate e consentirebbero un impiego più efficiente delle risorse di rete se venissero prese sulla base delle statistiche raccolte relativamente all'invio di pacchetti di dimensione diversa; i collegamenti più capaci sarebbero usati per trasportare gran parte del traffico, mentre a quelli meno efficenti si farebbe ricorso solo nel caso non ci fosse un percorso migliore. Con l'aumento dell'estensione della rete mesh, i protocolli OLSR e B.A.T.M.AN. non sempre utilizzano i percorsi dotati di un'effettiva maggiore capacità.

5.3 Stima della metrica nelle stesse condizioni (broadcast) di OLSR e B.A.T.M.A.N con pacchetti di probing di dimensioni diverse e con informazioni di Livello *Data Link*

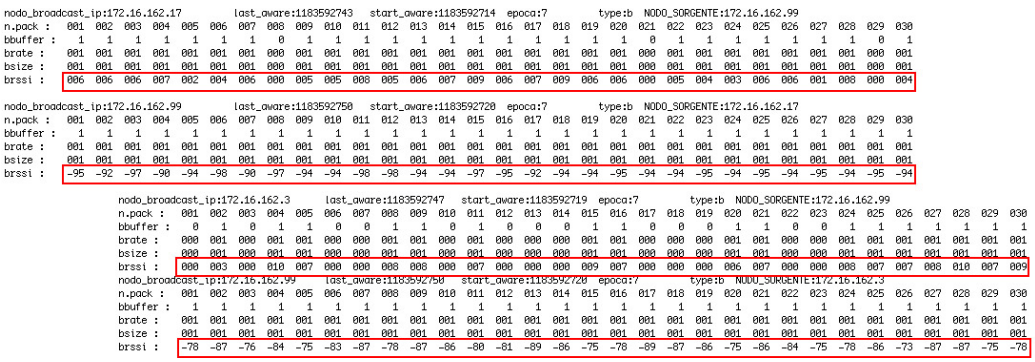
5.3.1 Condizioni di test

Il testbed utilizzato è lo stesso descritto precedentemente: le condizioni rimangono invariate, vengono quindi analizzati gli stessi dati raccolti in precedenza, ma ora in aggiunta si prendono in considerazione anche i dati di *Livello Data Link* e di *Livello Fisico*. In questa analisi si vede come l'utilizzo di questi ultimi dati di basso livello non aggiunga molte più informazioni rispetto all'uso di dati di *Livello Rete*. Nel caso dell'invio di pacchetti broadcast i driver delle schede wireless -così come avviene nello standard 802.11b/g- *forgiano* pacchetti al rate più basso (ossia al rate base) e non sono previsti, come si è visto nel sezione dedicata alla Gray Zone, meccanismi di riscontro e di ritrasmissione. In questo test, quindi, si considererà in più solamente il dato RSSI (*Received Signal Strength Indication*), che fornisce un indice della qualità del segnale radio su cui sono state ricevute le informazioni.
Si ritiene opportuno evidenziere in prposito le seguenti circostanze: ogni driver registra le informazioni relative all' RSSI su una scala diversa da quella degli altri driver; inoltre in questi test sono stati utilizzati nodi con due differenti tipi di driver: Broadcom nel caso degli access point Linksys WRT54GL (172.16.162.17 e 172.16.162.3) ed Atheros con MadWiFi sul computer che ha fatto da server (172.16.162.99). Il primo di questi utilizza una scala da -100

a 0 che ne esprime il segnale in dbm(decibel per mW), l'altro invece adotta una scala positiva da 0 a 100. Negli esempi, perciò, un valore del segnale di -94 su un nodo Linksys corrisponde ad un segnale di 6 nel nodo Atheros.
Si ricorda (come del resto è indicato nella sigla) che RSSI costituisce solamente un indicazione piuttosto che una rilevazione quantitativamente precisa.

5.3.2 Il test

Prendiamo ora in esame il primo gruppo di dati analizzati:



In questa sessione ci limiteremo solamente all'analisi dei dati raccolti nella precedente sessione, questa volta, però, si prendono in considerazione anche i dati raccolti dal tool dai livelli più bassi della connessione.
Da notare come il link tra il nodo 172.16.162.3 e 172.16.162.99 goda di un segnale radio più forte di di quello del link tra il nodo 172.16.162.17 e il 172.16.162.99: il livello del segnale radio medio ricevuto dal nodo 172.16.162.99 rispetto al nodo 172.16.162.3 è infatti -84dbm, mentre quello ricevuto dal 172.16.162.3 è -89dbm.
L'RSSI è anche un buon indice del grado di asimmetria di un collegamento. Vediamo, per esempio, come nel precedente screenshot, il segnale dal nodo 172.16.162.17 al nodo 172.16.162.99 sia più forte (RSSI medio -90dbm) che

nel senso contrario (RSSI medio -94dbm).

5.3.3 Risultati

Analizzando, quindi, anche i dati messi a disposizione nel PRISM HEADER è possibile affinare ulteriormente le conoscenze delle capacità di un collegamento. Il segnale RSSI potrebbe, perciò, risultare valido per la costruzione di una metrica -ancora più precisa rispetto a quella formulata in base alle sole statistiche sulla ricezione dei pacchetti Livello Rete- che tenga conto anche di questo fattore per la valutazione della stabilità dei collegamenti tra i nodi. Come si è visto, sarebbe possibile utilizzare questo dato, sia per valutare la capacità di un collegamento, che per stabilire in che misura esso sia dotato della proprietà di simmetria. Questo perché, come più volte accennato, il grado di asimmetria di un collegamento wireless influisce notevolmente sulle prestazioni della rete nella trasmissione dei dati.

Capitolo 6

Conclusioni

Era nell'intedimento di questo lavoro, data l'importanza che vanno via via assumendo le reti mesh in termini di diffusione, di flessibilità e di prospettive di sviluppo, elaborare un metodo e uno strumento (tool) che potesse fornire elementi di valutazione, quanto più possibile significativi e realisici, della funzionalità di una qualsiasi rete mesh. Il tool qui presentato può rivelarsi utile anche ai fini della scelta di un protocollo di routing e all'eventuale elaborazione di miglioramenti del medesimo.

Si è pertanto ritenuto opportuno procedere alla scelta e alla classificazione delle caratteristiche ritenute più salienti degli elementi base di una rete, ossia dei collegamenti wireless (link) tra nodi vicini.

Le caratteristiche delle connesioni senza fili individuate sono la *stabilità*, con la sottocategoria della *simmetria*, e la *velocità*. Queste due categorie sono ritenute di particolare interesse inquanto da esse dipende l'efficienza, la flessibilità, la scalabilià, l'ottimizzazione della fruizione e la sostenibilià, proprie delle reti mesh. La rete, infatti, può essere riguardata come uno strumento che trasporta le informazioni in un'area geografica, a disposizione di tutti, e come tale deve poter fornire i servizi nella maniera più efficiente possibile. Per

far ciò è indispensabile che la scelta di percorsi per il migliore instradamento delle informazioni venga fatta sulla base di elementi oggettivi di valutazione. Il tool intende quindi offrire la possibilità di sondare con quali metodologie conviene valutare la qualità dei collegamenti tra i vari nodi di un percorso sotto l'aspetto delle caratteristiche sopra evidenziate. In quest'ordine di idee, chi volesse, per esempio, migliorare un protocollo di routing per reti mesh esistente o volesse sviluppare un nuovo protocollo di routing, avrebbe modo di decidere quale metodologia di probe implementare al posto di quella adottata da tutti gli attuali protocolli di routing, perché possano essere individuati i percorsi realmente più capaci. I protocolli attualmente sviluppati (come OLSR e B.A.T.M.A.N) non fanno altro che sondare la rete con le stesse modalità e con pacchetti di probe uniformi. Il tool invece consente di far generare ad ogni nodo della rete pacchetti di sondaggio non più uniformi, ma modulati secondo le intenzioni del progettista e, tramite l'acquisizione dei dati significativi di tutti i nodi della rete in osservazione e relativi alle ricezione dei pacchetti stessi, valutare quale metodologia di probe consenta un uso migliore della rete stessa. (rispetto alle altre.)

In definitiva il tool fa essenzialmente due cose. Comanda i nodi di una rete mesh perché generino probe diversi, (broadcast e unicast), di dimensione variabile e seguendo un schema appositamente formulato che consente di focalizzare il sondaggio sugli aspetti ritenuti di volta in volta di magiore interesse. La seconda funzionalità consiste nel collezionare, raggruppandole, le ricezioni provenienti da ogni nodo, immagazzinarle su uno stesso server come dati relativi a più livelli della comunicazione. Un utilizzatore, o una macchina a ciò deputata, potrebbe accedere a tali dati e valutare quali tipi di pacchetti di probe abbiano messo in luce, meglio di altri, le caratteristiche dei collegamenti che formano una rete mesh. Quel che verrebbe fuori sarebbe

un'immagine fedele delle peculiarità delle metodologie di probe capaci di favorire così la scelta progettuale di una metrica più calibrata allo svilupppo di un protocollo efficiente.

I risultati ottenuti sperimentalmente per mezzo del tool, con il vincolo di sfruttare solamente i pacchetti broadcast ed eventualmente con l'acquisizione anche dei dati di *Livello Data Link* e di *Livello Fisico*, dimostrano che l'utilizzo di pacchetti di probe di dimensioni non uniformi mette in luce conportamenti dei collegamenti che possono essere diversi da quelli rilevati mediante l'invio di pacchetti di probe di dimensioni uniformi.
Si è visto infatti che con la nuova modalità adottata, alcuni collegamenti, che per i probe di OLSR appaiono di uguale capacità, risultano di fatto reattivi in misura diversa se attraversati da traffico di controllo di dimesioni variabili. La scelta, quindi, di un appropriato pattern per l'invio di un pacchetto di probe influisce significativamente sulla percezione della qualità dei collegamenti.
Questa metodologia potrebbe essere anche agevolmente implementata sugli attuali protocolli di routing come estensione della valutazione della metrica da parte dei protocolli stessi, senza per questo dover stravolgere l'intero algoritmo di misura.
Il secondo test effettuato rivela che l'uso anche dei dati di *Livello Data Link* e di *Livello Fisico* può aggiungere ulteriori informazioni utili alla valutazione della stabilità e della asimmetria di un link (nel caso specifico del probe broadcast si tratterebbe solamente dell'informazione aggiuntiva sul livello del segnale radio o RSSI), ma questo comporterebbe la complicazione dello sviluppo di una versione del protocollo per ogni particolare driver di scheda wireless. Per questo motivo, nei casi di dispositivi hardware con tecnologie eterogenee, potrebbe risultare troppo macchinosa l'adozione di questo ap-

proccio che si risolverebbe in una condizione limitativa della portatilità.
La scelta dei pacchetti broadcast può essere riguardata come primo passo da fare verso il superamento della modalità di sondaggio attualmente adottata dai protocolli di routing per reti mesh. Questa modalità, infatti, consente di esplorare una rete nel modo meno invasivo possibile e nello stesso tempo costituisce la base per la determinazione dei nodi vicini e la loro raggiugibilità, senza la quale nessun ulteriore sondaggio avrebbe sigificato. Si è usciti però dalla limitazione di sondaggi fatti esclusivamente con pacchetti uniformi e di piccole dimensioni e dalla percezione di dati provenienti dal solo *Livello di Rete*.

Una delle condizioni di svantaggio sulle reti mesh è il verificarsi del fenomeno della Gray Zone, che inganna i probe dei protocolli di routing, ai quali il fenomeno sfugge. In merito a ciò è stato visto, invece, come, tramite l'impiego di pacchetti di unicast, si possano rilevare le anolalie connesse con tale fenomeno e quindi prenderne atto ai fini della valutazione della qualità dei link.

Gli attuali protocolli di routing fanno delle scelte in base a valutazioni approssimative della qualità dei link. Il tool aggiunge ulteriori elementi per migliorare le scelte in relazione a una percezione più raffinata delle caratteristiche intriseche delle connessioni, rilevate mediante probe di diversa natura. In prospettiva futura, quali caratterizzazioni privileggiare per ottenere scelte in assoluto migliori? La risposta a questa domanda potrebbe venire dall'applicazione dei così detti algoritmi genetici. Questi algoritmi dovrebbero riuscire, utilizzando ognuno *cromosomi* in cui le scelte di routing sono associate con un certo peso ad ogni caratteristica dei collegamenti, attraverso un meccanismo di riproduzione e di confronto, a rispondere all'interrogativo in modo sperimentale. Si otterrebbe euristicamente, alla fine, una combinazione

di pesi, tale da fornire la soluzione migliore al problema del routing.

6.1 To Do

6.1.1 snmp

Nella prima revisione, il tool, per la comunicazione tra i nodi e il collezionatore dei dati , si avvale, di un protocollo non standard e per questo necessita di un proprio raccoglitore di informazioni. Un naturale sviluppo del tool, potrebbe essere quello di inserire, invece, le informazioni dei risultati dei test in un albero SNMP, che è il protocollo standard per la raccolta di dati degli apparati di rete. Il tool, infatti, per questo sviluppo futuro è stato concepito, in modo chè sia un server a chiedere ciclicamente il risultato dei test ai nodi e non viceversa. Così facendo qualsiasi client SNMP potrebbe interrogare i nodi permettendo di sfruttare la varietà di software già presenti sul mercato.

6.1.2 tool grafico GIS e GPS

Attualmente il tool raccoglie le informazioni che riguardano le prove e le visualizzano in forma testuale; la naturale evoluzione del software porterebbe alla costruzione di una GUI (Graphic User Interface) che, *a colpo d'occhio*, consentirebbe di individuare le caratteristiche della rete anche nella sua evoluzione temporale e fornire un'immagine complessiva della topologia della maglia. Inoltre l'utilizzo di un software grafico GIS (Geographic Information System) -come il software opensource Mapserver- accoppiato all'uso di dispositivi GPS, renderebbe disponibili informazioni preziose sulla locazione geografica degli elementi della maglia e sulle relazioni tra capacit‘a del link e distanza tra i nodi della rete mesh, anche al loro variare con la mobilitİn

questo caso, grazie al sistema GPS, si potrebbe anche godere anche di una perfetta sincronizzazione degli orologi di tutti i nodi.

6.2 Ringraziamenti

Ringrazio Eleonora, Salvatore Santalucia (Sal della community ninux.org), Saverio Proto (ZioProto della community ninux.org), Cristina ed Angelo per il supporto datomi a vario titolo.
Alla variabile più variabile di tutte: pippo

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] Clausen Jacquet INRIA, RFC 3626 Optimized Link State Routing Protocol, October 2003

[2] Gray Zone