Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea Magistrale in

Ingegneria Informatica

Implementazione e analisi delle prestazioni del protocollo di instradamento RPL e valutazione del suo utilizzo in presenza di mobilità

Relatore

Prof. Michele Zorzi

Laureando

Walter Berardi

Anno Accademico 2010/2011

Sommario

Le reti di sensori wireless stanno riscontrando un crescente interesse da parte di numerosi gruppi di ricerca a causa della grande varietà di scenari applicativi a cui esse si prestano. Tra gli aspetti più rilevanti vi è sicuramente lo sviluppo di protocolli di instradamento che si adattino ai vincoli che tali reti impongono. Questo lavoro di tesi è mirato allo studio di *Ipv6 Routing Protocol for Low power and lossy network* (RPL) discusso e in corso di standardizzazione dalla comunità IETF come protocollo di instradamento per reti con scarse capacità di trasmissione, risorse energetiche, computazionali e di memorizzazione come sono quelle di sensori wireless. L'implementazione del protocollo descritta nella tesi è stata verificata tramite un'approfondita analisi prestazionale nello scenario reale del testbed del dipartimento, distinguendosi quindi da altri lavori su RPL in ambiente simulato. È poi presentata una soluzione efficiente integrata in RPL che garantisca una corretta gestione dei guasti e della perdità di connettività dei nodi. Infine un rilevante contributo riguarda lo studio di una possibile estensione del protocollo in scenari di mobilità, evidenziando aspetti e limiti di cui è importante tenere conto. Questo studio è corredato da una implementazione adatta a situazioni in cui vi sia la presenza di un unico nodo mobile.

Abstract

Wireless sensor networks have been drawing the attention from many research groups due to their many application scenarios. One of the most relevant topics is the development of routing protocols which are able to meet all the wireless sensor networks constraints. This work focuses on the study and implementation of Ipv6 Routing Protocol for Low power and lossy network (RPL) presented by the IETF community and proposed as a standard for energy, memory and computational constrained resources networks. A performance evaluation is provided on a real scenario such as the Information Engineering Department testbed, unlike other works in which simulations are carried out to evaluate the protocol. Furthermore an efficient solution which allows our implementation to guarantee the correct execution in case of failure occurring to the nodes. But the most original contribute of this work is the study of an extension which allows the protocol to handle the presence of nodes mobility accounting for the constraints, the limitations and the critical factors. The thesis concludes by presenting an implementation of a mobility aware RPL version and validates it by means of a thorough experimental campaign.

Indice

Introduzione

Le reti di sensori wireless hanno attirato negli ultimi anni una grande attenzione da svariati gruppi di ricerca e aziende, soprattutto in ragione della loro flessibilità unita alla grande varietà di scenari applicativi in cui possono essere impiegate. Tuttavia esse sono caratterizzate da alcuni aspetti che le distinguono in maniera significativa dalle tradizionali reti di calcolatori: innanzitutto l'alimentazione dei nodi costituita principalmente da batterie necessita di un'attenzione particolare all'efficienza energetica, in secondo luogo la comunicazione wireless è gestita tramite antenne con limitata potenza e raggio di copertura che, insieme alla dislocazione nei più svariati ambienti, può comportare una probabilità maggiore di guasti o perdita di connettività e infine le scarse risorse di calcolo e di memorizzazione di cui dispongono i nodi. Lo sviluppo di protocolli e applicazioni che tengano conto di tali problematiche è oggetto di numerosi studi e ricerche e tra di essi occupano un ruolo di primo piano i protocolli di routing. La comunità IETF si è interessata a questo settore proponendo un nuovo protocollo di instradamento per reti di sensori denominato *Ipv6 Routing Protocol for Low power and lossy network* (RPL) e questo lavoro di tesi si pone come obiettivo il suo studio e la sua analisi da vari punti di vista. Partendo dal documento *draft* rilasciato da IETF si è proceduto con l'implementazione del protocollo in linguaggio NesC basandosi sul sistema operativo TinyOS. A differenza della grande maggioranza di lavori dedicati a RPL che validano lo studio tramite una simulazione, l'implementazione proposta è stata valutata in un scenario reale quale il testbed presente nell'edificio del dipartimento di ingegneria dell'informazione dell'università degli studi di Padova. Questo permette di porre l'accento su alcune problematiche di carattere pratico che non possono emergere da un ambiente simulato, in particolare riguardo la scelta dei valori di alcuni paramentri di RPL permettendo quindi di avere risultati non solo di carattere teorico, ma anche e soprattutto sull'effettiva utilizzabilità in uno scenario reale. Oltre all'implementazione e all'analisi prestazionale di RPL la tesi prende in considerazioni altri due aspetti critici per un protocollo di routing per reti di sensori. La prima riguarda la gestione dei guasti e della perdita di connettività proponendo una soluzione efficiente il cui funzio-

namento è sempre stato verificato nella rete del testbed. L'altro aspetto è il più originale della tesi e riguarda l'utilizzo di RPL in uno scenario che preveda la presenza di nodi mobili. Un'iniziale analisi teorica mette in luce gli aspetti fondamentali e i limiti che è necessario tenere in considerazione in tale scenario per poi presentare una possibile soluzione limitata alla presenza di un unico nodo mobile. La strategia è stata implementata e sottoposta a verifiche sperimentali sul testbed per ricavare i risultati presentati nella tesi.

La tesi è organizzata nel modo seguente:

- I capitoli 1 e 2 presentano un'introduzione generale alle reti di sensori e ai protocolli di routing. Il capitolo 2 si conclude ripercorrendo le motivazioni tecniche che hanno spinto IETF a sviluppare RPL
- Il capitolo 3 descrive le piattaforme hardware e software utilizzate per l'implementazione e l'analisi del protocollo.
- Il capitolo 4 descrive in maniera estremamente dettagliata le caratteristiche e il funzionamento di RPL così come presentato nel *draft* rilasciato dall'IETF.
- Il capitolo 5 è incentrato sugli aspetti più rilevanti dell'implementazione di RPL e sull'analisi delle prestazioni da questa ottenute.
- Il capitolo 6 descrive una soluzione per la gestione dei guasti integrata in RPL.
- Il capitolo 7 si concentra sulla possibilità di utilizzare il protocollo in uno scenario di mobilità, evidenziando gli aspetti fondamentali, i limiti, una possibile soluzione e la sua implementazione insieme ai risultati.
- Infine il capitolo 8 riassume gli obiettivi raggiunti dalla tesi proponendo possibili sviluppi futuri.

CAPITOLO 1

Wireless Sensor Networks

Indice

Svariati gruppi di ricerca, insieme ad aziende e compagnie operanti in diversi settori, si sono interessati negli ultimi anni alle potenzialità delle reti di sensori senza fili, *Wireless Sensor Network* (WSN) [1]. La prima causa di tale attenzione è da ricercare nello sviluppo e diffusione di tecnologie quali i sistemi micro-elettro-meccanici (MEMS) che, insieme alla sempre maggiore integrazione dei componenti digitali, hanno portato alla fabbricazione di sensori dai costi contenuti. I sensori sono generalmente dotati di una unità di calcolo, un'unità di comunicazione wireless a basso raggio, e componenti che permettono di rilevare grandezze fisiche. Un'insieme di nodi sensori, detti anche *mote*, che collaborano tra di loro nella raccolta, nell'elaborazione e nella diffusione di dati e misure constituiscono appunto le Wireless Sensor Network. Tali componenti possono essere utilizzati in ambienti molto diversi tra loro: dall'interno di un edificio ai dintorni di un vulcano e persino sotto il mare, oltre a poter essere equipaggiati con qualsiasi tipo di rilevatore di grandezze fisiche. Si può quindi facilmente immaginare come una tale flessibilità di utilizzo e di possibili applicazioni stimoli una ricerca attiva e sempre crescente. Nel seguito del capitolo saranno esposte le caratteristiche e i vincoli delle reti di sensori, insieme ad un breve approfondimento degli scenari e dei campi di applicazione.

1.1 Caratteristiche delle reti di sensori

Come già precedentemente accennato, le wireless sensor network sono costituite da un insieme di nodi sensore che collaborano tra di loro nel monitoraggio di un ambiente. Le informazioni raccolte sono tipicamente indirizzate verso una *stazione base* (o *sink*) la quale è collegata ad una unità di aggregazione ed elaborazione. Le reti di sensori si possono generalamente suddividere in due categorie, non strutturate e strutturate. Le prime hanno una elevata densità di sensori, i quali possono essere distribuiti seguendo determinati criteri oppure in maniera totalmente casuale. La rete è completamente autonoma nelle operazioni di misura dell'ambiente. L'elevata densità comporta evidentemente maggiori problemi di gestione sia della rete sia di eventuali rotture o comportamenti anomali dei nodi stessi. Le reti strutturate d'altra parte sono caratterizzate dall'avere i nodi sensori disposti in precise locazioni stabilite a priori. Il vantaggio di una tale disposizione è la possibilità di avere un numero minore di nodi posizionati in modo tale da garantire la comunicazione tra di essi.

Le reti di sensori si distinguono da quelle tradizionali per alcuni vincoli che le caratterizzano. Il primo e più importante vincolo è costituito dai nodi stessi. Innanzitutto la principale fonte di alimentazione è costituita da batterie, le quali forniscono una quantità di energia limitata nel tempo. Il fattore energetico costituisce un aspetto critico di tali reti e svariate ricerche sono state condotte con lo scopo di ottenere la massima efficienza possibile riducendo il consumo energetico. La mancanza di energia in un sensore comporta inevitabilmente il suo spegnimento, impattando sulle performance o perfino sulla capacità della rete di svolgere il compito per la quale è stata progettata. La componente energetica influisce quindi anche sul tempo di vita e di utilizzo della WSN. È possibile naturalmente equipaggiare i sensori con ulteriori dispositivi in grado di acquisire energia dall'ambiente. La tecnologia più sfruttata è quella dei pannelli solari [2], ma vi sono anche studi riguardanti l'utilizzo di celle combustibile o persino vibrazioni [3]. Al problema energetico si aggiunge il limitato raggio di copertura del segnale wireless, la scarsa capacità di calcolo e di memorizzazione. Tuttavia è comune trovare differenti tipologie di sensori all'interno di una WSN, in relazione al ruolo che essi devono ricoprire. I sensori generici, il cui scopo principale è quello di raccogliere misure dall'ambiente, sono equipaggiati con dispositivi capaci di rilevare umidità, temperatura, luce, pressione, accellerazione, posizione, sengali acustici ecc.. I sensori *gateway* invece hanno una capacità di calcolo e di autonomia energetica superiore che permette loro di svolgere operazioni di aggregazione dei dati e invio verso la stazione base. Oltre a citati limiti tecnologici, la progettazione delle reti di sensori è fortemente dipendente dal tipo di applicazione e dall'ambiente in cui essa si trova ad operare, che giocano un ruolo decisivo.

Entrando maggiormente nel dettaglio, la tecnologia sfruttata dalle reti di sensori comprende aspetti che possono essere classificati secondo tre categorie. Il primo aspetto da tenere in considerazione è il sistema. Esso comprende piattaforme che permettono alla WSN di supportare svariate tipologie di sensori con componenti radio, processori e capacità di storage diverse. I sistemi operativi devono essere progettati per supportare tali piattorme. Il protocollo di comunicazione è generalmente costituito da 5 livelli standard: applicazione, trasporto, rete, data-link, fisico. Tale stack di protocolli deve essere efficiente in termini energetici e in grado di operare correttamente in presenza di centinaia o migliaia di nodi. Infine la rete deve essere in grado di mettere a disposizione delle funzionalità che possano essere poi sfruttate dalle applicazioni a seconda della esigenze.

Negli anni sono stati sviluppati numerosi standard per reti di sensori con lo scopo di soddisfare i requisiti di basso consumo di energia. Tra di essi di trovano:

IEEE 802.15.4 [10] è lo standard proposto per regolare il livello MAC e fisico nelle reti che sfruttano comunicazioni wireless con basse capacità di trasferimento, basso raggio di copertura e scarsa autonomia energetica.

ZigBee [11] specifica un insieme di protocolli di comunicazione wireless di livello superiore operanti sopra IEEE 802.15.4. Essi sono progettati con l'obiettivo di rispettare vincoli di semplicità e basse rischieste energetiche e sono integrati in molti dispositivi con lo scopo di creare reti di sensori per monitoraggio ambientale.

6LoWPAN [12, 13] definisce un meccanismo di compressione e adattamento dell'header di un pacchetto IPv6 in modo da poterlo utilizzare con il protocollo IEEE 802.15.4. In questo modo è possibile stabilire comunicazioni tra i sensori e dispositivi che utilizzano protocolli basati su IP.

1.2 Tipologie di reti di sensori

Data la grande flessibilità delle reti di sensori, è possibile utilizzarle negli ambienti più disparati, dalla terra, al mare, all'interno degli edifici. Nel seguito vengono brevemente descritte le tipologie di WSN più diffuse in relazione all'ambiente in cui si trovano ad operare.

Reti di sensori terrestri Sono tipicamente costituite da centinaia o migliaia di nodi distribuiti in un'area secondo uno schema sia strutturato che non strutturato [4]. In ambienti difficilmente accessibili, i nodi possono

essere lanciati da un aereo e quindi distribuirsi in maniera casuale nell'area di interesse. Un'importante funzionalità che deve possedere la rete è quindi quella di costituirsi e configurarsi in maniera autonoma garantendo una costante comunicazine tra i nodi e la stazione base. I sensori possono essere equipaggiati con dispositivi secondari in grado di fornire energia, come le celle solari.

Reti di sensori sotterranee [5] Sono costituite da sensori installati sotto la superficie terrestre, per esempio una cava o una miniera, con lo scopo di monitorare le condizione di tale ambiente. A questi nodi se ne aggiungono alcuni in superficie che assumo il ruolo collegamento verso la stazione base. La creazione di tale tipologia di rete è generalmente più costosa della corrispettiva terrestre sia per quanto riguarda i nodi sia per la gestione stessa. Infatti i nodi devono essere in grado di comunicare attraverso rocce, acqua, terriccio e minerali che rendono la comunicazione wireless spesso difficile. Considerato l'ambiente in cui operano, le reti di sensori sotterranee sono generalmente di tipo strutturato, e la posizione dei singoli nodi deve essere calcolata con precisione. Per quanto riguarda l'aspetto energetico, è difficile ricavare energia supplementare sottoterra, quindi l'efficienza e la minimizzazione del consumo costituisce un fattore critico.

Reti di sensori sottomarine [6] Sono costituite da sensori distribuiti sotto la superficie del mare. Il loro numero è generalmente inferiore rispetto a quello delle reti terrestri e sono utilizzati veicoli sottomarini appositi per la raccolta delle informazioni e dei dati. La trasmissione avviene tramite onde acustiche, con i relativi problemi di banda limitata, elevato ritardo di trasmissione, e decandenza del segnale tipici di un ambiente acquatico. Anche in questo caso il fattore energetico costituisce un aspetto critico, in quanto la fonte di energia è costituita dalle sole batterie e l'operazione di sostituzione risulta spesso molto difficile e costosa. Inoltre la probabilità di riscontrare problemi di affidabilità e funzionamento dei nodi è molto alta proprio a causa delle condizioni in cui si trovano ad operare.

Reti di sensori multimediali [7] Questa tipologia di rete si adatta a quelle applicazioni il cui scopo è monitorare un ambiente registrando dati in formato video, audio o immagini, affiancando quindi ai sensori strumenti quali microfoni e videocamere. Si tratta di reti strutturate, in quanto è necessario garantire la copertura completa dell'ambiente da monitorare. Gli aspetti critici che riguardano tale tipologia di rete sono l'elevata richiesta di banda insieme all'elevato consumo di energia dovuto all'utilizzo di algoritmi di compressione ed eleborazione di dati complessi come quelli multimediali.

Reti di sensori mobili Consistono in un insieme di nodi che hanno la possibilità di muoversi in maniera autonoma e interagire con l'ambiente fisico. L'apetto critico è la capacità della rete di riconfigurarsi in seguito ai movimenti dei nodi. A questo di aggiunge la distribuzione dei dati, che deve essere coadiuvata da un protocollo di routing dinamico, in contrasto con quello statico utilizzato dalle tipologie di reti descritte precedentemente. Le reti mobili possono essere sfruttate per ottenere un maggiore grado di copertura dell'ambiente, oltre alla possibilità del nodo di aggirare ostacoli che gli impediscano di raggiungere la migliore posizione possibile per svolgere il suo compito.

1.3 Applicazioni

Due sono le categorie in cui possono essere classificate le applicazioni operanti sulle reti di sensori: il monitoraggio e il tracciamento. Nel seguito vedremo alcuni settori di utilizzo facendo esempi di progetti appartenenti a entrambe le categorie [8].

1.3.1 Applicazioni militari

Le applicazioni militari hanno un rapporto molto stretto con le reti di sensori, tanto che è difficile stabilire se esse sono nate per soddisfare necessità militari, oppure se il loro sviluppo è stato indipendente e solo in un secondo momento i reparti di ricerca tecnologica dell'esercito se ne sono interessati. Tuttavia le aree di maggiore interesse vanno dalla semplice raccolta di informazioni su una determinata area, al tracciamento di unità nemiche o target specifici fino alla sorveglianza di un campo di battaglia. Molte applicazioni pensate inizialmente per scopi militari possono essere utilizzate anche in ambito civile, basti pensare alla sorveglianza di un'abitazione o di un edificio pubblico.

Un esempio di applicazione relativa alla sorveglianza di aree specifiche è stata sviluppata dall'università della Virginia [9]. Migliaia di nodi sparsi possono individuare intrusioni di unità ostili avvertendo l'unità di difesa. Queste tipologie di applicazioni potranno in un futuro sostituire l'utilizzo delle mine come difesa di un territorio.

Per quanto riguarda il tracciamento delle unità nemiche, *PinPtr* [14] è stato sviluppato con lo scopo di individuare e localizzare cecchini. Il sistema utilizza una densa distribuzione di sensori in grado di misurare il tempo dell'onda d'urto insieme ad altre grandezze come la luce, il fumo o il calore generate dallo sparo. Le misurazioni vengono inviate alla stazione base per il computo della locazione del cecchino.

1.3.2 Monitoraggio di un ambiente

Le applicazioni di monitoraggio dell'ambiente possono essere distinte a seconda che si trovino ad operare in ambienti chiusi oppure aperti.

Applicazioni indoor Un esempio di applicazione indoor è quella di favorire l'integrazione tra il sistema di rilevamento di un incendio con quello di segnalazione delle uscite di sicurezza, in modo da condurre i residenti dell'edificio in fiamme verso la via di salvezza più sicura.

Applicazioni Outdoor Le WSN possono essere installate in un ambiente vulcanico per monitarne l'attività [15]. 16 nodi sensori sono stati distribuiti nel Volcàn Reventador situato nel nord dell'Ecuador ed equipaggiati con sismografo e un microfono. Ciascuno di essi campiona i dati catturati dagli strumenti salvandoli nella memoria flash interna. Quando si presenta un evento significativo il nodo comunica con la stazione base il verificarsi di tale evento.
Un altro esempio è presentato in [16]. 32 nodi sono stati installati in Great Duck Island nel golfo del Maine con lo scopo di monitorare l'habitat, dimora di molti uccelli, tramite misure di temperatura, pressione e umidità. I nodi sono stati posti anche nelle buche dove i volativi trovano solitamente rifugio e sono in grado rilevarne la presenza. I dati raccolti vengono trasferiti ad una stazione base che li accumula in un database accessibile da internet.

1.3.3 Applicazioni mediche

Le reti di sensori possono offrire un valido supporto al monitoraggio delle condizioni di salute di un paziente e avvisare se queste cambiano in maniera significativa e improvvisa [17]. Un ulteriore esempio di monitoraggio dei dati fisiologici di un paziente è costituito dal sistema Health Smart Home [18] realizzato dalla facoltà di medicina di Grenoble in Francia.

CAPITOLO 2

Routing per le reti di sensori

Indice

Come si è ampiamente discusso nel capitolo precedente, le reti di sensori permettono di raccogliere dati da un ambiente e trasferire tali informazioni verso una stazione base. Per raggiungere tale scopo, un ruolo fondamentale è ricoperto dal livello di networking dello stack di protocolli di comunicazione. Tale livello si occupa principalmente dell'instradamento delle informazione o routing. Esso è di fondamentale importanza in una qualsiasi tipologia di rete, dai calcolatori ai sensori. Un protocollo di routing ha la funzione di determinare un cammino tra due nodi della rete che non hanno la possibilità di comunicare direttamente tra loro. Il routing nelle reti di sensori [26] è un problema di grande interesse sia per l'importanza del suo ruolo nel corretto funzionamento delle applicazioni, sia perchè lo sviluppo di questi protocolli è impegnativo a causa delle caratteristiche proprie delle reti di sensori che le distinguono dalle comuni reti wireless. Innanzitutto non è sempre efficiente costituire un sistema di indirizzamento globale stile IP a causa dell'elevato numero di nodi sensori. Inoltre, le esigenze di efficienza energetica, computazione e capacità di memorizzazione limitata richiedono particolare attenzione anche nella progettazione del protocollo di routing. Infine, la grande varietà di possibili scenari in cui si trova ad operare una rete di sensori rende difficile individuare un protocollo di utilizzo generale.

2.1 Aspetti critici del routing

La progettazione di protocolli di routing per reti di sensori deve tenere conto di vari aspetti riassunti nei prossimi paragrafi.

Distribuzione dei nodi Nelle reti di sensori strutturate, essendo i nodi distribuiti in posizioni stabilite a priori, è relativamente semplice specificare dei percorsi. Nelle reti non strutturate i nodi devono essere in grado di organizzarsi in maniera autonoma e determinare le rotte verso la stazione base. Inoltre in tale tipologia di rete non è raro trovare una distribuzione dei nodi non uniforme che quindi suggerisce l'idea di formare cluster di nodi ciascuno con un *coordinatore* in modo da garantire la connettività di tutte le regioni della rete attraverso una struttura multi-hop. Compito del protocollo di routing sarà anche quello di formare nella maniera più efficiente possibile tale gerarchia di cluster.

Considerazioni energetiche La limitata disponibilità energetica dei nodi influenza notevolmente la durata del funzionamento della rete di sensori e il protocollo di routing deve contribuire a minimizzare il consumo di energia del nodo sensore. Inoltre in uno scenario multi-hop, ciascun nodo può assumere sia il compito di semplice monitoraggio dell'ambiente, sia il ruolo di router verso la stazione base e quindi una mancanza di energia di un nodo, con il suo conseguente spegnimento, può condurre alla necessità di dover calcolare nuovi cammini.

Limiti computazionali e di memorizzazione La limitata capacità di computazione e di memorizzazione del nodo impone al protocollo di routing una complessità non eccessiva accompagnata da una limitata necessità di informazioni da mantenere in memoria. Questi fattori devono inoltre essere scalabili rispetto alla dimensione della rete e quindi non crescere significativamente in funzione del numero di nodi.

Mezzo trasmissivo Le comunicazioni all'interno di una rete di sensori avvengono tramite componenti wireless con tutti i tradizionali problemi connessi a questo tipo di mezzo trasmissivo, quali fading, interferenze e alto tasso di errore oltre alla limitata potenza trasmissiva dei nodi.

Tecniche di richiesta e consegna delle informazioni I dati accumulati dai nodi possono essere inviati alla stazione base secondo tre fondamentali tecniche: *time-driven*, *event-driven*, *query-driven*. La prima tecnica prevede che i nodi raccolgano e trasmettano le loro informazioni secondo precisi intervalli di tempo mentre con la seconda tecnica i nodi reagiscono a particolari eventi tali da comportare significativi cambiamenti nei dati

misurati dai sensori. Infine la terza strategia assume che sia la stazione base, o qualsiasi altro nodo, a chiedere informazioni inviando una richiesta direttamente al nodo.

Fault Tolerance I nodi possono incorrere in comportamenti anomali, oppure smettere di funzionare correttamente per diverse cause come la mancanza di energia, il danneggiamento fisico, oppure la presenza di un ostacolo. Il protocollo di routing deve essere in grado di gestire queste situazioni calcolando strade alternative verso la stazione base.

Scalabilità Il numero di nodi che compongono una rete di sensori può variare dalle decine alle migliaia di unità. Tale paramentro della rete non dovrebbe influire sul funzionamento del protocollo di routing, che deve inoltre essere in grado di far fronte ad un improvviso aumento di traffico dovuto all'occorrere di un particolare evento che stimoli la trasmissione di dati da parte di un gran numero di nodi in precedenza silenti.

Dinamicità della rete Dalla sezione 1.2 si deduce come le reti di sensori siano costituite principalmente da nodi la cui posizione non cambia nel tempo, ma esistono applicazioni che necessitano di vari livelli di mobilità. Gestire questa situazione da parte del protocollo di routing garantendo sempre una strada verso la stazione base è un problema di complessità non indifferente, come si vedrà anche nel capitolo 7.

2.2 Protocolli di routing per reti di sensori

In relazione alla struttura della rete i protocolli di routing sviluppati per le wireless sensor network possono essere classificati secondo tre categorie principali: *data-centric*, gerarchico (*hierarchical*) e geografico (*geographic* o *location-based*). Queste categorie sono generali e costituiscono un paradigma di funzionamento del protocollo. A partire dalle prime versioni dei protocolli che specificavano tali paradigmi, nel corso degli anni sono stati sviluppati molte altre versioni modificate e migliorate per far fronte a diversi scenari di applicazione delle reti di sensori. Un'ulteriore possibile distinzione è tra protocollo proattivo, reattivo oppure ibrido, a seconda di come sono determinate le rotte nella rete. Un protocollo di routing proattivo calcola i cammini prima che questi siano effettivamente necessari. Al contrario un protocollo reattivo determina la strada ogni volta che questa viene richiesta. I protocolli ibridi utilizzano una combinazione delle due precedenti tecniche. Le descrizioni dei protocolli che seguono fanno riferimento a studi piuttosto datati, ma hanno l'unico scopo di inquadrare le diverse strategie secondo le loro caratteristiche principali. Nel corso degli anni sono stati pubblicati una

grande varietà di articoli che rielaborano queste idee sviluppando protocolli più performanti oppure specifici per un determinato scenario di applicazione.

2.2.1 Data-centric routing

Questa tipologia di routing si adatta alle reti di sensori non strutturate con un grande numero di nodi distribuiti nell'area da monitorare. A causa della grande quantità di nodi, non è efficiente creare un sistema di indirizzamento globale e inoltre la ditribuzione casuale e la densità della rete conducono spesso a una notevole ridondanza di informazione tra nodi appartenenti ad una stessa regione. Questi limiti hanno portato allo sviluppo di protocolli di routing dove il sink richiede i dati ai nodi posti in diverse regioni dell'ambiente da monitorare. Poichè i dati sono richiesti in maniera *query-driven*, è necessario costruire uno schema di nomi che identifichi le risorse messe a disposizione dai nodi. Sono stati sviluppati vari protocolli secondo il paradigma data-centric, nel seguito vengono descritti i più significativi.

SPIN [27] L'idea chiave dietro al *Sensor Protocols for Information via Negotiation* è quella di costruire uno schema di metadati di alto livello che identifichi i dati semplici. Prima che avvenga la trasmissione del dato vero e proprio, il nodo invia un pacchetto ADV che pubblicizza ai vicini il metadato corrispondente all'informazione di cui è in possesso. Una volta ricevuto il messaggio ADV, i nodi interessati a tale informazione inviano di ritorno una richiesta REQ affinchè il nodo invii tramite un pacchetto DATA il dato che aveva pubblicizzato in precedenza. A questo punto i nodi in possesso della nuova informazione possono a loro volta inviare un messaggio ADV in modo da diffondere il dato nella rete. Tuttavia la consegna di un dato ad un nodo non è garantita dal protocollo. Un semplice esempio di tale scenario si ha quando un nodo interessato ad un'informazione è distante, nella rete, da quello che effettivamente la possiede e i nodi intermedi non sono interessati all'informazione stessa. Nella sua relativa semplicità, SPIN è stato progettato con l'intento di migliorare le prestazioni di una classica inondazione della rete con i dati raccolti dai nodi. Il protocollo infatti permette di raggiungere una maggiore efficienza energetica diffondendo i metadati invece del contenuto effettivo dell'informazione, oltre a eliminare la ridondanza nella diffusione dei dati.

Directed Diffusion [28] Directed Diffusion utilizza uno schema di metadati costituito da una coppia attributo-valore. Una richiesta di informazione, detta *interest*, è costituita da una lista di tali coppie, per esempio nome dell'oggetto, intervallo, area geografica. Il sink, o chi è interessato ad un dato, effettua il broadcast di un *interest*, che viene diffuso da nodo a nodo fino a coprire tutta la rete. Durante la propagazione ciascun nodo calcola un gradiente bidirezionale costituito generalmente da un valore scalare e una

direzione rispetto al nodo da cui ha ricevuto la richiesta. La propagazione delle richieste e la costituzione dei gradienti permette di stabilire vari percorsi tra la sorgente dell'informazione e il sink e tra questi ne viene scelto uno o un numero ristretto di essi. Un secondo invio dell'*interest* lungo questi cammini preferiti permette di rinforzare il percorso e avviare la trasmissione effettiva del dato lungo lo stesso cammino. Su questo schema generale sono possibili miglioramenti tramite sistemi di caching degli *interest* e dei percorsi, ricostituzione di percorsi, controllo sulla formazione di loop e aggregazione dei dati. Directed Diffusion è un protocollo reattivo *query-driven* e quindi non si adatta ad applicazioni che richiedono un continuo invio di dati verso il sink. Una versione leggermente diversa di Directed Diffusion è stata proposta in [29] sotto il nome di *Gradient-based routing*. L'idea è quella di considerare il numero di hop come fattore di gradiente, in modo che ciascun nodo possa scoprire la sua distanza dal sink, detta altezza del nodo.

2.2.2 Routing gerarchico

Per far fronte a problemi di scalabilità della rete e possibile sovraccarico dei gateway nei casi di aumento della densità dei nodi o di presenza di un unico nodo gateway, un approccio di tipo *clustering* è applicato al routing. Nella rete vengono formati dei cluster di nodi che fanno riferimento a un nodo in particolare che assume il ruolo di *cluster-head* (CH). Tale nodo, che può disporre di risorse energetiche e di calcolo superiori, ha il compito di raccogliere le informazioni, svolgere operazioni di aggregazione e instradarle verso la stazione base. Si viene quindi a formare una struttura gerarchica organizzata in livelli, come raffigurato in figura 2.1. Il vantaggio di tale approccio è quello di ridurre il numero di pacchetti inviati al sink, garantendo una diminuzione del consumo energetico all'interno dei cluster e quindi aumentando il tempo di vita generale della rete, anche se si aggiunge la complessità della formazione della struttura cluster. A differenza del paradigma data-centric, il routing gerachico non è di tipo *query-driven*, ma le informazioni sono inviate dai nodi in maniera *time-driven* o *event-driven*.

LEACH [30] *Low-Energy Adaptive Clustering Hierarchy* è uno dei protocolli di router gerarchico più diffusi e popolari. I *cluster-head* sono scelti in maniera casuale e si occupano di comprimere i dati ricevuti dai nodi appartenti al cluster, aggregarli e inviarli alla stazione base. La raccolta dei dati avviene in maniera centralizzata e periodica, approccio adeguato per una applicazione che necessita di monitorare costantemente parametri e grandezze fisiche. Le operazioni svolte dal protocollo si possono dividere in due distinte fasi: la formazione della struttura cluster e la consegna dei dati aggregati alla stazione base. Per quanto riguarda la selezione dei *cluster-head*, ciascun nodo sceglie un numero casuale r tra 0 e 1, e lo valuta in relazione alla

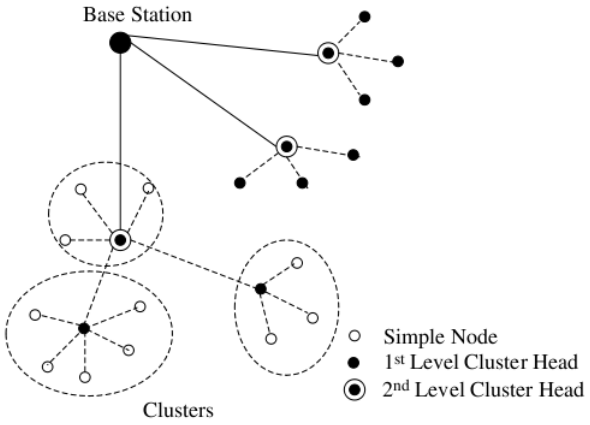


Fig. 2.1: Esempio di routing gerarchico

seguente soglia $T(n)$

$$T(n) = \frac{p}{1 - p(r \bmod(1/p))} \qquad n \in G$$

dove p è la percentuale desiderata di *cluster-head* e G l'insieme dei nodi coinvolti nell'elezione. Se si verifica la condizione $r < T(n)$, il nodo diventa *cluster-head* e comunica in broadcast la sua avvenuta elezione. I nodi non eletti selezionano a quale cluster agganciarsi sulla base della potenza del segnale del messaggio ricevuto comunicando al prescelto *cluster-head* la loro appartenenza. Dopo aver ricevuto le notifiche da parte di tutti i nodi, il CH assegna a ciascuno di essi un intervallo di tempo di trasmissione per ridurre al minimo le collisioni dovute a comunicazioni simultanee. Dopo un intervallo di tempo stabilito a priori viene ripetuto il processo di formazione dei cluster con la scelta di nuovi CH, in modo da portare ad un uniforme consumo di energia tra i nodi della rete. Sebbene questo protocollo sia in grado di aumentare il tempo di vita della rete, esistono delle limitazioni dovute ad alcune assunzioni sulla rete di sensori alla quale esso si applica. Innanzitutto si assume che i nodi abbiano una potenza di trasmissione tale da poter sempre raggiungere il sink, ipotesi che esclude l'utilizzo in reti distribuite in aree molto grandi. Inoltre non è garantita una scelta omogena nella rete dei *cluster-head*, quindi alcuni nodi potrebbero non avere alcun CH a cui agganciarsi.

PEGASIS [31] *Power Efficient GAthering in Sensor Information Systems* può essere considerato un miglioramento di LEACH. Il protocollo costituisce delle catene di nodi in modo tale che ciascuno di essi trasmetta e riceva pacchetti dal vicino più prossimo in termini di potenza di segnale. Una volta formata la catena, viene scelto in maniera casuale un nodo, diverso ad

ogni round di comunicazione, che assuma il ruolo di leader nelle trasmissioni verso la stazione base in modo da distribuire tra tutti i nodi il maggiore carico di lavoro richiesto dalle operazioni di leader. Per comprendere il funzionamento del protocollo, si consideri la figura 2.2.

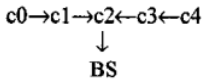


Fig. 2.2: Esempio di funzionamento del protocollo PEGASIS

Il ruolo di leader è assengato al nodo c2, il quale utilizza un piccolo messaggio token per far cominciare la trasmissione dall'estremo della catena. Ricevuto il token, c0 manda i suoi dati a c1, il quale li aggrega con quelli in suo possesso e infine li spedisce a c2. Il leader quindi manda il token verso c4, ricevendo da c3 i dati aggregati del lato destro della catena. Una volta ricevuti i dati da entrambi i lati della catena, c2 può aggregare i suoi e inviarli infine alla stazione base. PEGASIS si è dimostrato essere in grado di aumentare il tempo di vita della rete, principalmente grazie all'eliminazione della fase di creazione dei cluster. Tuttavia presenta limiti dovuti all'assunzione che ogni nodo possa comunicare direttamente con la stazione base, la conoscenza completa della topologia della rete, il ritardo introdotto per i nodi più distanti nella catena e infine il collo di bottiglia che potrebbe costituire il leader.

2.2.3 Routing Geografico

Questa tipologia di routing permette di identificare i nodi per mezzo della loro posizione. La distanza tra i nodi può essere stimata sulla base della potenza del segnale entrante e, scambiando questa informazione, è possibile ottenere delle coordinate in un sistema di riferimento relativo al nodo stesso. In alternativa la posizione può essere determinata in maniera più precisa tramite strumenti GPS montati sui nodi.

GEAR [32] *Geographic Energy Aware Routing* utilizza una selezione euristica dei nodi basata sulle informazioni geografiche e di consumo energetico con il fine di instradare un pacchetto verso una regione di destinazione. L'idea è quella di ridurre il numero di richieste di informazioni limitandole a determinate regioni invece di inondare tutta la rete, come avviene per esempio in Directed Diffusion. Ogni nodo che implementa GEAR memorizza due parametri di costo per raggiungere una destinazione: stimato e appreso. Il costo stimato è calcolato sulla base di una combinazione tra l'energia residua del nodo e la distanza dalla destinazione stessa. Il costo appreso invece è un raffinamento del costo stimato che tiene conto di eventuali buchi presenti

nella rete. Si ha un buco quando un nodo non ha vicini maggiormente prossimi all'area di destinazione che il nodo stesso, il quale naturalmente non si trova in tale regione. Se la rete non presenta buchi, il costo stimato coincide con quello appreso e quest'ultimo è propagato a ritroso ogni volta che un pacchetto raggiunge l'area di destinazione. Per quanto riguarda l'instradamento del pacchetto, esistono due scenari: il forward verso la regione target, oppure il forward all'interno della regione. Nel primo caso un nodo che riceve un pacchetto controlla se esiste un suo vicino più prossimo alla destinazione e, in caso positivo, il pacchetto è consegnato a tale vicino. Una risposta negativa al controllo potrebbe essere indice della presenza di un buco nella rete e quindi uno dei vicini è selezionato per il forward in accordo con il costo appreso, il quale sarà poi aggiornato seguendo il cammino del pacchetto. Nel secondo scenario invece, l'informazione può essere diffusa seguendo in maniera ricorsiva il procedimento appena descritto, suddividendo la regione in quadranti e inviando il pacchetto ad un nodo in ciascun quadrante, il quale opererà un'ulteriore partizione fino alla formazione di aree con un solo nodo. Un'altra tecnica, utile quando la rete non è molto densamente popolata, è un flooding dell'informazione all'interno della regione.

2.3 ROLL

La grande diffusione delle reti di sensori, chiamate in questa sede con il termine generale di *Low-power and Lossy Networks* (LLN), ha spinto la comunità *Internet Engeneering Task Force*[1] (IETF) ad interessarsi ad esse con progetti come 6LoWPAN ma anche all'aspetto del routing. Come si è visto nella sezione precedente, la problematica dell'instradamento nelle reti di sensori richiede molta attenzione nella progettazione, conducendo allo sviluppo di svariati protocolli senza però essere accompagnati da una specifica *Request For Comment* con la prospettiva futura di diventare standard. Scopo di IETF è invece quello di sviluppare un protocollo di routing che possa proporsi come uno standard per LLN. Il primo passo verso questa direzione è valutare la possibilità di utilizzare un protocollo esistente specificato da IETF nel contesto delle reti di sensori. Vengono presi in considerazione solo tale famiglia di protocolli perchè essi hanno una corrispondente descrizione in un RFC. I risultati di questa indagine sono esposti nel documento [34], in cui si conclude che nessun protocollo esistente specificato da IETF può essere utilizzato in una rete di sensori in quanto non rispetta uno o più degli aspetti critici descritti nella sezione 2.1. Per tale motivo all'interno di IETF si è costituito il *Routing Over Low-power and Lossy networks working group* (ROLL) [33] con l'obiettivo di definire un nuovo protocollo di routing specifico per LLN. Il risultato è *Ipv6 Routing Protocol for Low power and lossy network* (RPL) [35], che costituisce il soggetto principale di studio del

[1] `http://www.ietf.org/`

presente lavoro di tesi e sarà ampiamente descritto e analizzato nei capitoli seguenti. È interessante percorrere le considerazioni e le argomentazioni che hanno portato i ricercatori di IETF a bocciare l'utilizzo dei protocolli esistenti in LLN perché mettono in luce i motivi di alcune scelte progettuali di RPL.

Il documento [34] prende in considerazione cinque diversi criteri che un protocollo di routing deve soddisfare per poter essere utilizzato nelle reti di sensori. Tali criteri costituiscono dei vincoli di alto livello che devono essere necessariamente rispettati ma non rappresentano un elenco esauriente e completo e alcune applicazioni potrebbero richiedere ulteriori vincoli più stringenti. Essi, riprendendo quanto esposto nella sezione 2.1, sono:

1. **Routing State** Questo criterio si riferisce alla capacità del protocollo di essere scalabile rispetto alla quantità di memoria richiesta all'aumentare sia del numero di nodi, sia del numero di destinazioni a cui i dati raccolti devono essere consegnati. In particolare la relazione tra questi fattori non deve essere linerare ma sub-lineare.

2. **Loss Response** Poichè il mezzo trasmissivo wireless sfruttato nelle reti di sensori non è completamente affidabile, il protocollo di routing non deve richiedere l'utilizzo di molti link tra i nodi per propagarsi nella rete. È infatti preferibile che il routing richieda la trasmissione di informazioni solo quando queste vadano effettivamente ad incidere sui cammini verso destinazioni attive oppure servano per la scoperta di nodi vicini.

3. **Control Cost** Un protocollo di routing operante in una LLN non deve richiedere un elevato traffico di controllo sia per la scoperta che per il mantenimento delle rotte. Per dare un'idea numerica, una comunicazione ogni ora è di gran lunga più desiderabile di una ogni secondo, e in ogni caso il traffico di controllo deve essere limitato superiormente dal traffico dati. Questo implica che il protocollo deve essere in grado di diminuire il traffico generato quando necessario.

4. **Link Cost**: Il protocollo deve tenere in considerazione la qualità del link radio e includerla come metrica nella funzione che determina la scelta dei cammini, con l'obiettivo di minimizzare la latenza e massimizzare l'affidabilità della comunicazione.

5. **Node Cost**: Analogamente al criterio precedente, il protocollo deve tenere in considerazione lo stato di un nodo nel calcolo dei cammini. In particolare valori come la potenza, la memoria e il livello di energia residuo devono influire sulla rotta che il pacchetto segue nella rete.

Nelle sezioni seguenti vengono brevemente descritti i protocolli IETF e i corrispondenti limiti del loro utilizzo in una rete di sensori. La suddivisione è fatta tra protocolli basati sullo stato del collegamento (link-state) e sui vettori delle distanze (Distance Vector). All'interno di queste due categorie sono considerati protocolli di routing progettati per *mobile ad hoc network* (MANET), che comprendono reti mobili in cui i dispositivi hanno possibilità di muoversi liberamente in ogni direzione e le comunicazioni avvengono per mezzo del canale wireless.

2.3.1 Protocolli link-state

Il paradigma link-state prevede che ciasun router comunichi a tutti i nodi della rete lo stato dei suoi collegamenti adiacenti. Questo tipo di informazioni ricevute da tutti i partecipanti al protocollo permette di creare un database che rappresenta la topologia completa della rete. A questo punto, sfruttando algoritmi di tipo *shortest path*, è possibile calcolare un cammino verso la destinazione.

OSPF [37, 38] *Open Shortest Path First* è un protocollo link-state che permette la suddivione della rete in diverse aree stabilendo un struttura gerarchica tra di esse. Lo stato dei collegamenti adiacenti al router è ricavato scambiando messaggi di tipo *hello*, e il cammino è determinato considerando metriche di link come la larghezza di banda e il ritardo di propagazione. OSPF non rispetta il vincolo Routing State in quanto esso necessita della conoscenza di tutti i link della rete con conseguente dimensione linerare nel numero di router della tabella di routing. Il protocollo fallisce il criterio Control Cost in quanto necessita di un flooding delle informazioni nella rete sia nella fase di inizializzazione sia in caso di un cambiamento del link, anche se non utilizzato. Queste caratteristiche non permettono inoltre di rispettare il criterio Loss Response. Per quanto riguarda il vincolo Link Cost e Node Cost si può concludere che, viste le precedenti considerazioni sul funzionamento di OSPF, il primo è rispettato, mentre non si può dire altrettanto del secondo.

OLSR & OLSRv2 [39, 40] *Optimized Link State Routing* è un protocollo link-state proattivo sviluppato per reti MANET. Secondo il paradigma link-state, ogni nodo ottiene una mappa della topologia della rete tramite uno scambio di messaggi contenenti informazioni sullo stato dei link. Per limitare il traffico di controllo che si generebbe a causa delle continue variazioni dei collegamenti, OLSR impone un intervallo di tempo minimo tra due successive trasmissioni che comunque possono essere opzionali. Ciascun nodo mantiene inoltre un insieme di vicini distanti un hop chiamati *Multipoint Relays* (MPR), i quali garantiscono connetività rispetto a vicini distanti due hop. Un nodo MPR contiene la lista dei nodi, detta MPR *selectors*,

che lo hanno scelto come MPR. Essi inoltre sono gli unici che diffondono le informazioni link-state riguardati i nodi appartenenti alla MPR selectors. In questo modo il protocollo riesce a limitare la diffusione di pacchetti di controllo e ottenere un rapido adattamento ai cambiamenti topologici. Tuttavia la dimensione della tabella di routing aumenta in relazione al numero di nodi, non rispettando il vincolo Routing State. Per quanto riguarda il traffico di controllo, esistono tecniche come la *fisheye* routing che permette di diminuire il numero di aggiornamenti inviati da un nodo riducendo il carico del traffico di controllo a livelli accettabili per una LLN, tuttavia non esiste una descrizione di come questo possa essere integrato nel protocollo OLSR. Un cambiamento nella topologia necessita l'invio di informazioni link state attraverso la rete anche se tali link non sono utilzzati per il routing e OLSR indica che l'invio di tali informazioni è opzionale ma non descrive come gestire problemi di consistenza della topologia dovuti al mancato invio di queste informazioni. Quindi i criteri Loss Response e Control Cost necessitano di ulteriori specifiche per essere rispettati. Si può invece concludere che OLSR soddisfa i criteri Link e Node Cost.

2.3.2 Protocolli Distance Vector

Un protocollo di tipo Distance Vector scambia informazioni sui cammini piuttosto che informazioni topologiche sulla rete. Le informazioni sono scambiate tra nodi vicini e riguardano i collegamenti diretti. In questo modo i nodi non hanno una conoscenza della topologia dell'intera rete, ma semplicemente sui collegamenti attivi con i vicini. Per ogni destinazione appresa viene creata una voce in tabella che ne indica la distanza basata su una qualche metrica e il primo passo per raggiungerla. I nodi inoltre comunicano informazioni sulle rotte effettivamente utilizzate con la possibilità di stabilirle *on-demand*. Se da una parte i protocolli Distance Vector richiedono uno scambio di dati solo con i vicini più prossimi, dall'altra essi hanno in generale tempi di convergenza più alti e possono portare alla formazione di cicli nei cammini della rete.

RIP [41] *Routing Information Protocol* è un protocollo di tipo Distance Vector utilizzato nelle reti di calcolatori e come tale può incorrere nei problemi appena citati di loop e lento tempo di convergenza. Come metrica utilizza il numero di hop. La grandezza della tabella di routing, che tipicamente è lineare rispetto al numero di nodi della rete, può essere ridotta nel caso in cui si abbia una conoscenza a priori delle destinazioni da raggiungere senza quindi aumentare il numero delle righe in relazione al crescere della rete permettendo a RIP di soddisfare il criterio Routing State. La versione del protocollo denominata *Triggered RIP* innesca l'invio di aggiornamenti solo quando la topologia della rete effettivamente cambia, quindi anche il criterio Control Cost può dirsi soddisfatto. Il criterio Loss Response invece non

può essere considerato rispettato poichè la propagazione di aggiornamenti avviene anche se il cammino non è effettivamente utilizzato. Per quanto riguarda il Link Cost, l'RFC specifica la possibilità di utilizzare metriche più complesse del semplice hop-count, anche se questo potrebbe portare ad instabilità. Infine le proprietà del nodo non sono prese in considerazione dal protocollo e quindi il criterio Node Cost non risulta rispettato.

AODV [42] *Ad-hoc On-demand Distance Vector* è un protocollo progettato per reti MANET che costruisce i cammini solo quando questi sono effettivamente richiesti. A tale scopo, AODV inonda la rete con un pacchetto Route Request (RREQ) alla cui ricezione un nodo può memorizzare un percorso che permetta di raggiungere il nodo origine del pacchetto RREQ. Il destinatario della richiesta, risponde con un pacchetto Route Replay (RREP) che, grazie alla informazioni acquisite dai nodi durante la propagazione della richiesta, giunge alla sorgente creando e memorizzando una rotta di comunicazione tra i due nodi. AODV utilizza tecniche che permettono di controllare l'esplosione dei messaggi di richiesta nella rete in modo da indirizzarli verso il nodo destinazione insieme ad un meccanismo di numero di sequenza che previene la formazione di loop. La metrica di scelta del cammino è basata sul numero di hop. Quando un collegamento non risulta essere più valido, avviene una procedura di riparazione che permette di modificare i percorsi in modo da evitare il link rotto. Questo avviene tramite la propagazione di messaggi Route Error (RERR) in tutta la rete. Il numero di righe della routing table è proporzionale al numero di nodi necessari a raggiungere i capi della comunicazione e non alla densità totale della rete, caratteristica che permette di soddisfare il criterio Routing State. Analogamente anche il criterio Control Cost è soddisfatto in quanto il protocollo genera traffico di controllo solo quando è necessario un cammino per instradare i dati. AODV invece fallisce il criterio Loss Response poichè la rottura di un link provoca la propagazione di messaggi necessari alla modifica dei cammini che utilizzano tale link anche se questo in realtà non è attivo per la comunicazioni di dati. I criteri Link Cost e Node Cost non vengono soddisfatti poichè nè proprietà del link nè dei nodi vengono prese in considerazione per la scelta di percorsi.

CAPITOLO 3

Piattaforme hardware e software

Indice

In questo capitolo vengono descritti in dettaglio le piattaforme hardware e software utilizzate per l'implementazione e l'analisi delle prestazione del protocollo RPL come descritto nei capitoli seguenti. La struttura hardware è costituita dal testbed sviluppato e costituito all'interno del dipartimento di ingegneria dell'informazione dell'università degli studi di Padova (DEI). È formato da 340 nodi di tipo TelosB distribuiti in tutto l'edificio e ha finalità di ricerca e test di protocolli e applicazioni per reti di sensori. Gli strumenti software comprendono il sistema operativo TinyOS, sviluppato dall'università di Berkley in California specificamente per rispettare tutti i vincoli e i requisiti propri delle reti di sensori, e la piattaforma TinyNET, sviluppata dal gruppo di ricerca SIGNET del dipartimento di ingegneria dell'informazione di Padova. Quest'ultima permette una veloce e semplice integrazione di protocolli e applicazioni nel sistema operativo TinyOS.

3.1 Testbed

Il testbed sviluppato e costruito all'interno del DEI rientra nel contensto del progetto *WIreless SEnsor network for city-Wide Ambient Intelligence* (WISE-WAI) [19] fondato dalla cassa di risparmio di Padova e Rovigo con il

principale scopo di condurre test e ricerche sull'effettiva possibilità di utilizzare una rete di sensori largamente distribuita per tipologie di applicazioni tipiche come localizzazione, monitoraggio e rilevamento di eventi pericolosi.

Il testbed è costituito da 340 nodi distribuiti in tutto il dipartimento come mostra la figura 3.1 dove i quadrati colorati rappresentano i singoli nodi.

Fig. 3.1: Testbed del DEI

I nodi sono di tipo TelosB [20] e ciascuno di essi è connesso ad una dorsale (*backbone*) di rete tramite via Universal Serial Bus che permette di fornire energia senza quindi problemi di esaurimento di batterie con conseguente continuo e dispendioso rimpiazzamento. Le comunicazioni tra i nodi avvengono tuttavia tramite il canale wireless. L'alimentazione energetica dei nodi non è l'unico scopo della USB *backbone*: essa mette a disposizione uno strumento per recuperare dati di log dai nodi per operazioni di debugging. Questo aspetto è di fondamentale importanza durante le fasi di test, in quanto permette la raccolta dei dati in senza dover fisicamente scaricarli da ogni singolo nodo. Il testbed è progettato in maniera gerarchica e tutti i nodi, tramite USB hub, sono collegati ad un computer integrato che agisce come gateway ed è chiamato *Node Cluster Gateway* (NCG). Il componente NCG gestisce il flusso dati da e verso i nodi, permettendo così di ricevere informazioni di debug ma anche di operare la programmazione, lo spegnimento, l'accensione e il reset dei nodi stessi. I nodi gateway sono a loro volta connessi tramite una dorsale Ethernet ad un *remote access gateway* che si occupa di gestire le comunicazioni locali della porzione di rete a lui assegnata e sono collegati tra di loro tramite un tunnel VPN. Il principale punto di accesso alla struttura è costituito da un server che è in grado di gestire e controllare tutte le comunicazioni e gli aspetti dell'intero testbed. La figura 3.2 schematizza quanto appena descritto.

La rete di sensori così progettata risulta essere scalabile e facilmente gestibile da remoto in ogni aspetto, caratteristiche che la rendono particolarmente adatta per lo sviluppo e il test di applicazioni e protocolli.

3.1.1 TelosB Mote

Come già più volte ricordato, i nodi sensore che costituiscono il testbed sono di tipo TelosB /Tmote SKY, figura 3.3.

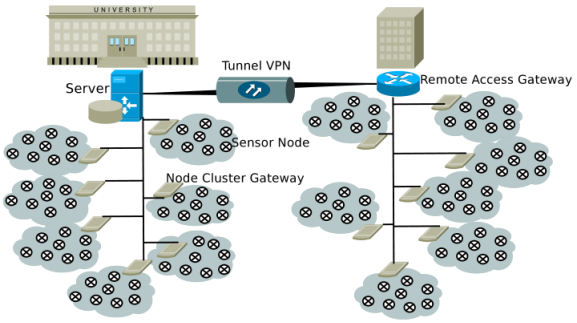


Fig. 3.2: Struttura del testbed

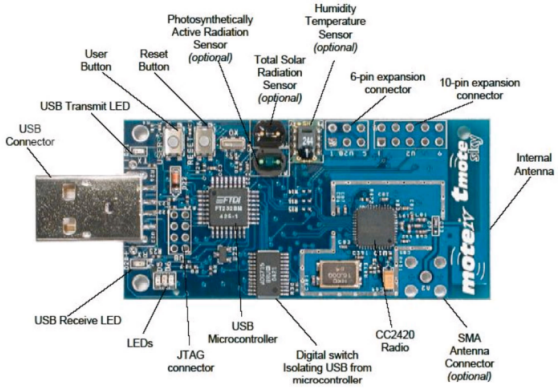


Fig. 3.3: TelosB

La scelta di questa tipologia di sensori è dovuta principalmente alla loro diffusione e al costante supporto e aggiornamento. Il TelosB è un nodo sensore wireless a basso consumo energetico progettato specificatamente per wireless sensor network equipaggiato con tre componenti in grado di misurare umidità, temperatura e intensità luminosa e una porta USB che permette il collegamento diretto ad un altri dispositivi. La gestione della comunicazione wireless è affidata al componente radio CC2420 [21] che implementa lo standard IEEE 802.15.4 e lo stack di protocolli ZigBee. Tra i vari aspetti concernenti il CC2420 è importante evidenziare la disponibilità di determinare due grandezze relative alla qualità del link: il *Received Signal Strength Indicator* (RSSI) e il *Link Quality Indicator* (LQI). L'RSSI è una grandezza espressa in dBm calcolata, dato il valore `RSSI_VALUE` fornito dal componente, tramite la formula:

$$\texttt{RSSI} = \texttt{RSSI\_VALUE} + \texttt{RSSI\_OFFSET}$$

dove `RSSI_OFFSET` è un valore determinato in maniera empirica e valutato in -45 dBm. L'LQI invece è uno scalare che identifica la forza o la qualità del link ed è compreso tra 0 e 255. Tipicamente, un valore intorno a 110 indica

la migliore qualità di link mentre un valore intorno a 50 indica la peggiore qualità di link misurabile dal componente CC2420. È compito del software estendere il valore ricevuto dal componente nella scala tra 0 e 255.

L'integrazione del componente CC2420 nel sistema TinyOS mette a disposizione la possibilità di utilizzare un meccanismo di controllo di ricezione dei pacchetti tramite degli ACK denominato *Packet Link*. È' possibile impostare il numero di ritrasmissioni del pacchetto per il quale non è stato ricevuto l'ACK al termine delle quali il pacchetto viene scartato. Infine, anche se di default il componente non processa pacchetti destinati al nodo, esso permette di impostare un ascolto in modalità promiscua consentendo quindi di processare tutti i pacchetti ricevuti entro il raggio di copertura del segnale wireless.

3.2 TinyOS

I sistemi operativi per le reti di sensori si distinguono in maniera significativa da quelli tradizionali a causa dei numerosi vincoli che queste impongono. Infatti devono essere in grado di compiere le loro operazioni incidendo il meno possibile sul consumo energetico e avendo a disposizione poca capacità di calcolo e di memoria. Il sistema operativo deve quindi garantire il corretto funzionamento di un'applicazione incidendo il meno possibile sulle già limitate risorse che questa ha a disposizione. Il capitolo precedente ha già messo in evidenza la grande varietà di dispositivi che costituiscono una rete di sensori e gli ambienti in cui questa si trova ad operare, motivi per i quali il sistema operativo dovrebbe essere in grado di adattarsi facilmente a cambiamenti hardware e software oltre a garantire affidabilità e robustezza di esecuzione di un'applicazione. Storicamente, sono stati seguiti due approcci nello sviluppo di sistemi operativi per reti di sensori:

1. L'applicazione viene compilata assieme ai componenti del sistema operativo.

2. Il sistema operativo è caratterizzato dai classici strati software modificati per rispettare i vincoli.

Il secondo approccio è quello seguito dal sistema Mantis [22] e presenta un difficile controllo e gestione dei consumi e delle risorse anche se permette un'elevata versatilità consentendo l'esecuzione contemporanea di più applicazioni. Di contro il primo approccio consente di avere consumi bassisimi, anche se è possibile eseguire una sola applicazione. Questa strategia è quella seguita dal sistema TinyOS [23], sviluppato dall'Università della California Berkley in collaborazione con il centro di ricerche Intel con lo scopo di raggiungere una efficiente gestione energetica, basso carico computazionale e

ridotte dimensioni in memoria. Come ogni altro sistema operativo si occupa di gestire le risorse hardware e software mettendo a disposizione dell'utente un'astrazione dello strato hardware. Generalmente le applicazioni sviluppate per TinyOS occupano qualche decina di KB, dei quali solo 400 byte sono dedicati al sistema operativo.

Lo sviluppo di TinyOS ha seguito due principi fondamentali: è specifico per una particolare applicazione ed è basato sugli eventi. La prima caratteristica si riferisce al fatto che, poichè il codice del sistema operativo e quello dell'applicazione sono compilati assieme, tutti i componenti non utilizzati non vengono compilati e installati ottenendo un eseguibile finale specifico per ciascuna applicazione. Il modello di programmazione basato sugli eventi è utilizzato da tinyOS per gestire la concorrenza. I sistemi operativi per PC utilizzano un approccio basato sui processi tale per cui uno scheduler si occupa di selezionare all'interno di opportune code e secondo una opportuna politica un processo a cui allocare le risorse della CPU. Il passaggio della CPU richiede la fase di *context switch* in cui è salvato il contesto del processo attualmente in esecuzione e il caricamento di quello nuovo. Questo approccio risulta essere inadeguato per le reti di sensori in quanto il *context switch* implica un significativo carico computazionale oltre allo spazio in memoria necessario al salvataggio del contesto. L'approccio basato sugli eventi utilizzato da TinyOS prevede la presenza di un gestore che si occupa di interrompere il normale processamento del codice all'occorrenza di eventi significativi, eseguendo le procedure relative all'evento occorso.

La programmazione di applicazioni per TinyOS avviene tramite il linguaggio NesC [24], derivato dal C. Si tratta di un linguaggio ad eventi basato su componenti legati tra di loro. Ogni componente chiede servizi per mezzo di comandi ed risponde ad eventi segnalati da altri componenti. I comandi e gli eventi sono definiti all'interno di interfacce e ciascun componente può utilizzare o fornire tali interfacce. Nel primo caso vengono richiesti dati da un componente che fornisce l'interfaccia e quindi devono essere implementate le procedure di gestione degli eventi, nel secondo caso invece il componente deve implementare le operazioni svolte dai comandi.

3.3 TinyNET

TinyNET [25] è una piattaforma modulare sviluppata all'interno del gruppo di ricerca SIGNET[1] dell'università di Padova con lo scopo di permettere una rapida integrazione in TinyOS di protocolli e applicazioni. TinyOS infatti non presenta una struttura a livelli che consenta di collegare facilmente i protocolli, per esempio secondo il modello ISO/OSI. Il program-

[1] `http://telecom.dei.unipd.it/labs/read/3/`

matore quindi non ha la possibilità di scegliere dove posizionare il proprio protocollo nello stack, ma è costretto a sviluppare tutti i livelli, da quello di accesso al canale, al routing fino all'applicativo. TinyNET tenta di risolvere questo inconveniente proponendosi come un framework modulare in grado di facilitare lo sviluppo di applicazioni, fornire protocolli e applicazioni con un'interfaccia di rete stratificata, permettere una veloce configurazione delle applicazioni con nuovi protocolli e funzionalità trasparenti nello stack.

La figura 3.4 rappresenta i componenti principali costituenti l'architettura di TinyNET, che può essere divisa in due livelli, quello applicativo e quello di rete. Il livello applicativo è simile a quello standard di TinyOS centralizzando tuttavia il controllo della radio in modo da intercettare qualsiasi richiesta di controllo. Il livello di rete contiene tutti i moduli che consentono l'accesso ad ogni pacchetto inviato o ricevuto dall'interfaccia radio. Svolge inoltre il ruolo di punto di ingresso per nuovi protocolli che risulteranno essere completamente trasparenti al livello applicativo.

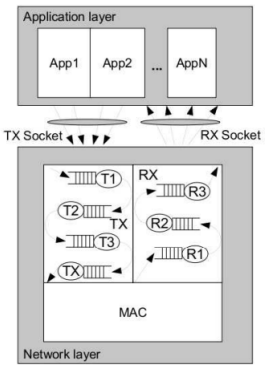


Fig. 3.4: Architettura di TinyNET

3.4 Integrazione del protocollo di routing

TinyNET ha un modulo che implementa un semplice protocollo di routing grazie al quale è possibile raggiungere il sink e il protocollo RPL sviluppato e presentato in questa tesi si propone come suo sostituto all'interno della piattaforma. È quindi necessario analizzare come avviene questa integrazione e come TinyNET gestisce il processo di inoltro dei pacchetti ricevuti. Nell'header di un pacchetto trasmesso sono memorizzati 4 diversi indirizzi: la sorgente e la destinazione originale definiti dal nodo che trasmette il pacchetto originale, la sorgente e la destinazione a singolo hop dei nodi intermedi. Questi indirizzi sono specificati tramite dei valori numerici definiti in fase di

programmazione del nodo e accessibili tramite la variabile `TOS_NODE_ID`. Il protocollo di routing deve fornire l'interfaccia `Route` così definita:

```
1 interface Route {
2   command bool forward(rxring_buffentry_t* rxbuf);
3   command bool isForMe(rxring_buffentry_t* rxbuf);
4 }
```

Alla ricezione di un pacchetto, tramite l'invocazione del comando `isForMe`, viene verificato se il nodo è la destinazione originale del pacchetto. In caso positivo viene segnalato l'evento di ricezione di un pacchetto. In caso negativo viene controllato, tramite l'invocazione del comando `forward`, se il nodo ha il compito di effettuare l'inoltro avendo `TOS_NODE_ID` uguale a quello di destinazione a singolo hop. L'indirizzo del nexthop è fornito dal protocollo di routing tramite l'implementazione dell'interfaccia `RouteGet` così definita:

```
1 interface RouteGet {
2   command am_addr_t get( am_addr_t dst );
3 }
```

Essa controlla se l'indirizzo di destinazione originale è presente nella tabella di instradamento, nel qual caso viene restituito il nexthop verso quel nodo, altrimenti se la destinazione è il sink viene restituito il nexthop verso di esso.

CAPITOLO 4

Protocollo di routing RPL

Indice

Come già anticipato nella sezione 2.3 il presente lavoro di tesi è incentrato sull'implementazione e sull'analisi delle prestazioni del protocollo di routing IETF RPL. Si tratta di un protocollo proattivo a vettori di distanza inquadrabile nel contesto del routing gerarchico, anche se non in senso stretto in quanto non prevede l'elezione e la formazione di cluster, ma forma una particolare struttura di grafo in cui è presente una radice rappresentata dal nodo sink o dalla stazione base. RPL è compatibile con un indirizzamento IPv6 dei sensori, caratteristica che consente di relazionare la LLN su cui opera con reti di calcolatori basate sullo stack TCP/IP. Lo sviluppo di RPL da parte del gruppo di lavoro IETF ROLL è documentato dai documenti *draft* che vengono regolarmente pubblicati con una elevata frequenza di aggiornamento. A questo proposito, la descrizione del protocollo è basata sul documento [35] datato luglio 2010, mentre al momento della stesura finale di questo documento di tesi, la versione più aggiornata è [36]. Tuttavia le modifiche tra i vari documenti sono minimi e non influiscono sulla struttura fondamentale del protocollo. In questo capitolo viene presentato quindi in dettaglio il funzionamento di RPL come specificato nel documento [35].

4.1 Topologia e parametri

4.1.1 DODAG

Un *Directed Acyclic Graph* (DAG) è un grafo orientato in cui non esistono cicli e gli archi formano dei cammini verso i nodi radice, in cui tale cammino termina. I nodi radice hanno la caratteristica di non avere archi uscenti ma solo entranti. Un DAG con un unico nodo radice che costituisce la destinazione di tutti i percorsi del grafo è detto *Destination Oriented DAG* (DODAG). Il protocollo RPL crea tra i nodi che lo eseguono una struttura di tipo DODAG, che può essere vista come un albero la cui radice è detta *DODAG root* (figura 4.1). Per tale motivo ciascun nodo può ricoprire il ruolo di genitore, figlio, ancestore, e foglia.

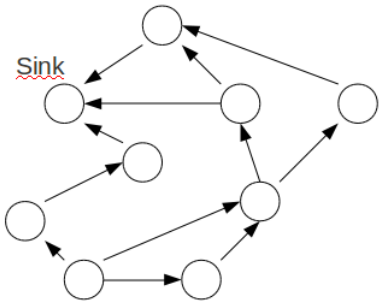


Fig. 4.1: DODAG

Una rete su cui opera RPL può comprendere diversi DODAG a seconda di varie esigenze. Per mantenere coerenza tra di essi, il protocollo fornisce quattro paramentri:

- `RPLInstanceID`. Questo valore identifica un'istanza di RPL. Essa comprende uno o più DODAG che condividono metriche e funzioni per calcolare il `Rank` (si veda il quarto parametro). Una rete può avere più istanze di RPL, con differenti `RPLInstanceID`, ciascuna delle quali identifica DODAG indipendenti e non correlati basati su metriche e funzioni diverse. Un nodo può appartenere ad un unico DODAG all'interno della stessa istanza di RPL.

- `DODAGID`. Scalare unico all'interno di una istanza RPL e identifica un DODAG root e il DODAG corrispondente. La coppia (`RPLInstanceID, DODAGID`) identifica in maniera univoca un DODAG nella rete.

- `DODAGVersionNumber`. Il nodo radice ha la possibilità di decidere la ricostruzione di un DODAG sulla base di certi eventi incrementando lo scalare `DODAGVersionNumber`. La tripla (`RPLInstanceID, DODAGID, DODAGVersionNumber`) identifica una versione del DODAG.

- `Rank`. Il `Rank` stabilisce un ordinamento all'interno di una versione del DODAG e identifica la posizione di ciascun nodo rispetto alla radice. Esso cresce o diminuisce in maniera strettamente monotona a seconda, rispettivamente, che ci si allontani o avvicini al nodo root. Il suo valore è determinato da una funzione dipendente dalla distanza dalla radice, da metriche di link, di nodo o da altri vincoli. Il valore `MinHopRankIncrease` è il minimo incremento possibile tra un nodo e il suo genitore nel grafo. Nel momento in cui è necessario effetturare un confronto tra i rank di due nodi, il valore da comparare dovrebbe essere derivato dalla formula floor(`Rank`/`MinHopRankIncrease`). Per una efficiente implementazione `MinHopRankIncrease` dovrebbe essere una potenza di due e il suo valore influenza il numero di incrementi possibili prima di raggiungere il limite `INFINITE_RANK`, quantità che rappresenta una distanza infinita rispetto alla radice e può essere utile per contrastare il problema del *count-to-infinity*.

RPL supporta tre diversi tipi di flusso di traffico: da molti a uno, multipoint-to-point (MP2P); da uno a molti, point-to-multipoint (P2MP); da uno a uno, point-to-point (P2P). Il caso MP2P è quello più comunemente richiesto da applicazioni operanti su LLN e comprende l'invio di dati da molti nodi verso uno specifico. Tale obiettivo può essere facilmente raggiunto sfruttando il traffico diretto verso la radice del DODAG.

4.1.2 Metriche

RPL non specifica esplicitamente le metriche da utilizzare per calcolare i cammini all'interno del grafo, poichè queste sono fortemente dipendenti dal tipo di applicazioni. Il documento [43] presenta in maniera generale quali tipologie di misurazioni e di vincoli dovrebbero essere seguiti da RPL nella scelta dei percorsi tra i nodi.

4.2 Messaggi di controllo del protocollo RPL

I messaggi di controllo di RPL sono definiti all'interno di un pacchetto ICMPv6, figura 4.2, e sono costituiti da un'intestazione ICMPv6 e da un corpo che comprende il messaggio stesso e una serie di possibili opzioni.

Il campo `Type` assume il valore 155, mentre il campo `Code` assume uno dei seguenti codici a seconda del tipo di messaggio di controllo contenuto:

- `0x00`: DODAG Information Sollecitation (DIS)
- `0x01`: DODAG Information Object (DIO)
- `0x02`: Destination Advertisement Object (DAO)
- `0x03`: Destination Advertisement Object Acknowledgment (DAO-ACK)

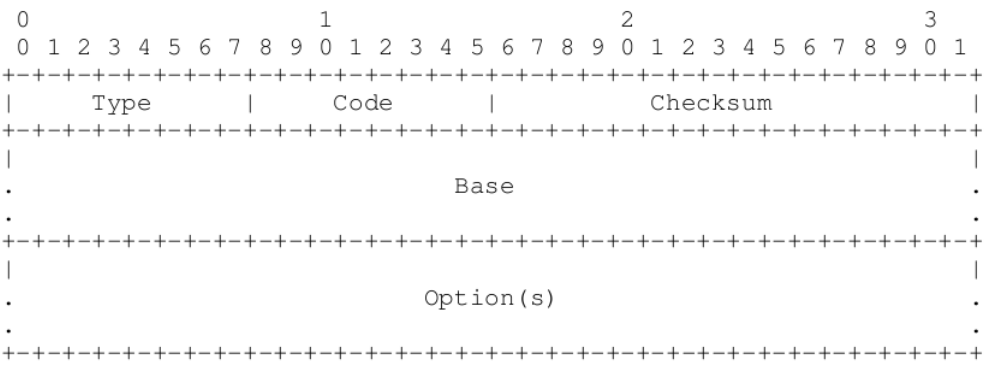


Fig. 4.2: Struttura del pacchetto ICMPv6

- `0x80`: Secure DODAG Information Sollecitation
- `0x81`: Secure DODAG Information Object (DIO)
- `0x82`: Secure Destination Advertisement Object (DAO)
- `0x83`: Secure Destination Advertisement Object Acknowledgment (DAO-ACK)
- `0x8A`: Consistency Check

Dall'elenco precedente si può notare come i pacchetti di controllo di RPL siano quattro: DIS, DIO, DAO e DAO-ACK. Nelle sezioni seguenti se ne descriveranno in dettaglio le strutture, le funzionalità e gli utilizzi. Tuttavia si omette la trattazione di tutti gli aspetti di sicurezza del protocollo in quanto non obiettivo del lavoro di tesi.

4.2.1 DODAG Information Object (DIO)

Il messaggio DIO è utilizzato per la formazione e il mantenimento dei cammini verso il nodo radice (rotte verso l'alto). Le informazioni contenute al suo interno permettono ad un nodo di scoprire l'esistenza di una istanza di RPL e apprenderne i corrispondenti parametri di configurazione. Inoltre esso permette l'identificazione di un insieme di nodi all'interno del DODAG tra i quali il nodo può scegliere il genitore diventando così parte del DODAG. I paramentri di configurazione sono generalemente inizializzati dal nodo che crea l'istanza RPL, tipicamente il sink, assumendo il ruolo di root nel DODAG.

Il formato del pacchetto raffigurato in figura 4.3 e il significato di ciascun campo è così definito:

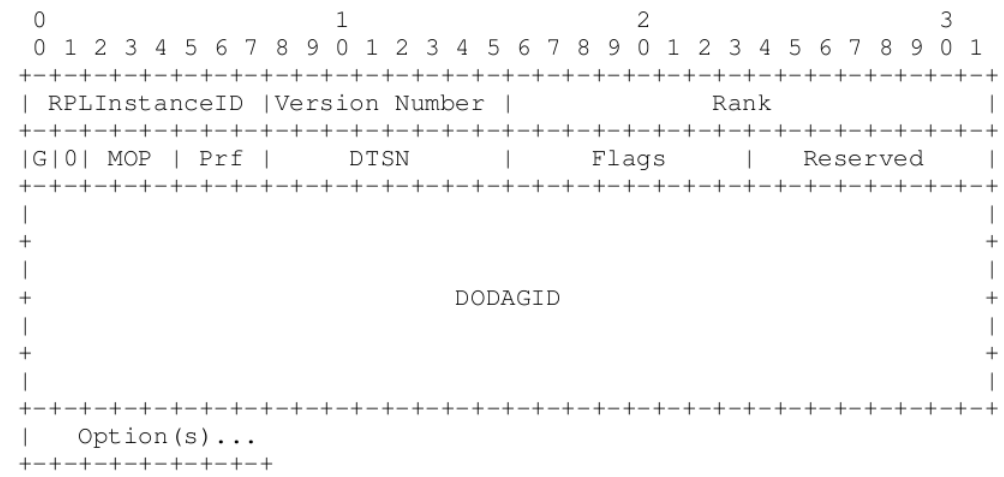


Fig. 4.3: Formato del pacchetto DIO

- `RPLInstanceID`: campo di 8 bit impostato dal DODAG root che definisce l'identificativo dell'istanza di RPL di cui fa parte il DODAG.
- `Version`: campo intero di 8 bit senza segno impostato dal DODAG root che identifica la versione del DODAG.
- `Rank`: campo intero di 16 bit che identifica il rank di un nodo.
- `Grounded G`: questo bit impostato dal root definisce se il DODAG è *grounded* oppure no. Un DODAG si definisce *grounded* se esso offre connettività a nodi necessari all'applicazione per realizzare il suo compito. I nodi non *grounded* sono detti *floating* e spesso il loro unico scopo è quello di fornire connetività tra nodi che non svolgono funzioni nell'ambito dell'applicazione in esecuzione nella rete.
- `Mode Of Operation (MOP)`: campo di tre bit definito dal nodo radice che stabilisce la tecnica utilizzata per costruire i cammini dal DODAG root verso gli altri nodi. Ciascun elemento del DODAG deve essere in grado di adempiere al ruolo di router come specificato dal campo MOP, altrimenti può agganciarsi al DODAG solo come foglia. I valori possibili del campo sono i seguenti:

  000: nessun cammino dal nodo radice è mantenuto da RPL.

  001: modalità di non memorizzazione, *non storing mode*

  010: modalità di memorizzazione senza supporto per il multicast, *Storing mode without multicast*

  011: modalità di memorizzazione con supporto per il multicast, *Storing mode with multicast*

Gli altri valori sono considerati riservati. La modalità *non storing* sfrutta la tecnica del *source routing* per costruire i cammini, mentre la modalità *storing* utilizza le tabelle di routing. La descrizione dettagliata di queste tecniche è presente nella sezione 4.4.

- `DODAGPreference (Prf)`: campo intero di 3 bit senza segno impostato dal root che definisce il livello di preferenza (in ordine crescente da `0x00` a `0x07`) del DODAG rispetto ad un altro all'interno della stessa istanza RPL.
- `DSTN`: il campo intero di 8 bit senza segno *Destination advertisement Triggered Sequence Number* (DTSN) è utilizzato per il mantenimento delle rotte dal nodo radice e sarà descritto nella sezione 4.4.
- `DODAGID`: campo intero di 128 bit senza segno impostato dal root che identifica in maniera univoca il DODAG all'interno dell'istanza di RPL.

Gli altri bit non assegnati sono considerati riservati. Il messaggio DIO può includere le seguenti opzioni (il dettaglio di tutte le opzioni è presente nella sezione 4.2.5):

- Pad1
- PadN
- Metric Container
- Routing Information
- DODAG Configuration
- Prefix Information

4.2.2 DODAG Information Solicitation (DIS)

Il messaggio DIS può essere utilizzato con lo scopo di sollecitare l'invio di un pacchetto DIO da un altro nodo al fine di scoprire o verificare la presenza di nodi vicini. Il formato del pacchetto è il seguente

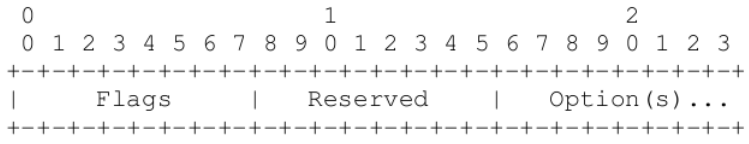


Fig. 4.4: Formato del pacchetto DIS

I bit non assegnati sono considerati riservati e le opzioni che possono essere incluse sono:

- Pad1
- PadN
- Solicited Information

4.2.3 Destination Advertisement Object (DAO)

Il messaggio DAO è utilizzato per la costruzione dei cammini dal nodo radice verso gli altri nodi del DODAG ed è mandato in modalità *unicast* da un figlio ad uno (o più) genitore. Il DAO può essere opzionalmente confermato dal genitore inviando in risposta un pacchetto DAO-ACK al corrispondente figlio. Il formato del pacchetto è

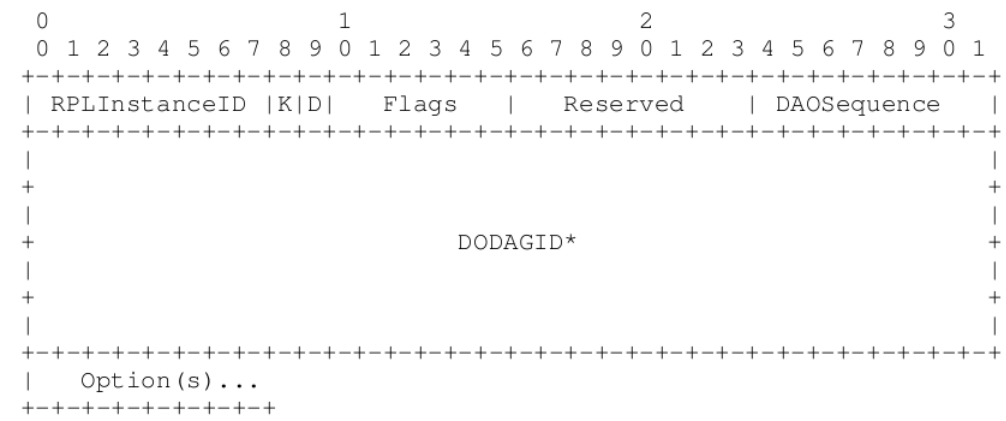


Fig. 4.5: Formato del pacchetto DAO

con corrispondente significato dei campi:

- `RPLInstanceID`: campo intero di 8 bit che identifica un'istanza di RPL. Questo valore è appreso dal messaggio DIO
- `K`: bit che indica se la sorgente si aspetta un pacchetto DAO-ACK in risposta al DAO inviato
- `D`: bit indicante la presenza o meno del valore `DODAGID` nel pacchetto. Esso deve essere presente nel caso in cui vi siano più DODAG appartenenti alla stessa istanza di RPL.
- `DAOSequence`: valore incrementato a ciascun invio del DAO, e ricevuto uguale come risposta in un DAO-ACK.
- `DODAGID`: campo opzionale intero di 16 bit senza segno che identifica univocamente il DODAG all'interno di un'istanza RPL. Il campo è presente solo se il bit `D` è attivo.

I rimanenti bit non assegnati sono considerati riservati e le opzioni che possono essere incluse in un messaggio DAO sono:

- Pad1
- PadN
- RPL Target
- Transit Information

4.2.4 Destination Advertisement Object Acknowledgement (DAO-ACK)

Il messaggio DAO-ACK è mandato dal nodo genitore al nodo figlio in risposta ad un DAO. Il formato del pacchetto è il seguente

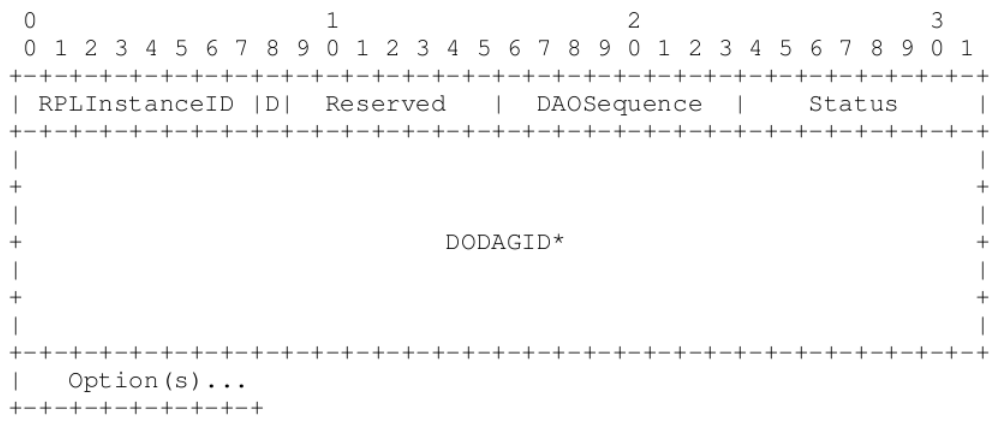


Fig. 4.6: Formato del pacchetto DAO-ACK

- `RPLInstanceID`: campo intero di 8 bit che identifica un'istanza di RPL. Questo valore è appreso dal messaggio DIO.
- `D`: bit indicante la presenza o meno del valore `DODAGID` nel pacchetto. Esso deve essere presente nel caso in cui vi siano più DODAG appartenenti alla stessa istanza di RPL.
- `DAOSequence`: valore corrispondente allo stesso campo del DAO ricevuto.
- `Status`: un valore sopra il 128 indica che il nodo dovrebbe selezionare un genitore alternativo.
- `DODAGID`: campo opzionale intero di 16 bit senza segno che identifica univocamente il DODAG all'interno di un'istanza RPL. Il campo è presente solo se il bit `D` è attivo.

I rimanenti bit sono considerati riservati e il messaggio DAO-ACK non prevede la presenza di opzioni al suo interno.

4.2.5 Opzioni

Le opzioni che possono essere incluse nei pacchetti di controllo di RPL sono nove:

0x00 Pad1: permette di inserire un ottetto di padding nel messaggio con il fine di ottenere un allineamento delle opzioni seguendo la convenzione adottata per il protocollo IPv6.

0x01 PadN: permette di inserire N ottetti di padding al messaggio sempre con fini di allineamento.

0x02 Metric Container: permette di diffondere e riportare valori di metrica lungo il DODAG.

0x03 Route Information: è usata per indicare se è presente connettività dal nodo root verso una destinazione specificata tramite un prefisso.

0x04 DODAG Configuration: è utilizzata per diffondere parametri di configurazione necessari per il funzionamento del protocollo RPL. Queste variabili devono essere specificate dal nodo radice e solo lui deve avere il permesso di modificarle.

0x05 RPL Target: é utilizzata per indicare l'indirizzo IPv6 di una particolare destinazione. Può identificare una particolare risorsa che il nodo radice sta cercando di raggiungere.

0x06 Transit Information: é utilizzata da un nodo per indicare gli attributi di un cammino per una o più destinazioni specificate dalle opzioni RPL Target che precedono le opzioni Transit Information. Sono utilizzate principalmente quando il protocollo lavora in modalità *non storing*.

0x07 Solicited Information: è utilizzata da un nodo per richiedere l'invio di un messaggio DIO da uno o più nodi vicini.

0x08 Prefix Information: permette di distribuire prefissi di indirizzamento utilizzati nel DODAG.

e hanno un'intestazione comune rappresentata in figura 4.7 i cui campi identificano:

- `Option Type`: campo intero di 8 bit che identifica il tipo di opzione.

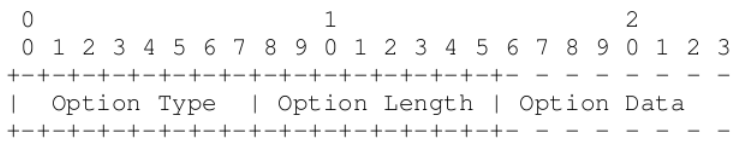


Fig. 4.7: Formato comune delle opzioni

- `Option Length`: campo intero di 8 bit senza segno che rappresenta la lunghezza in ottetti dell'opzione escludendo i campi `Option Type` e `Option Length`.
- `Option Data`: campo di lunghezza `Option Length` che contiene le informazioni specificate dall'opzione.

Nel seguito verranno descritte in dettaglio le opzioni DODAG Configuration e Solicited Information in quanto sono state utilizzate nella versione del protocollo implementata. I dettagli delle altre sono lasciate al documento draft.

4.2.5.1 DODAG Configuration Option

Le *DODAG Configuration Option* possono essere presenti all'interno di un messaggio DIO e hanno il seguente formato:

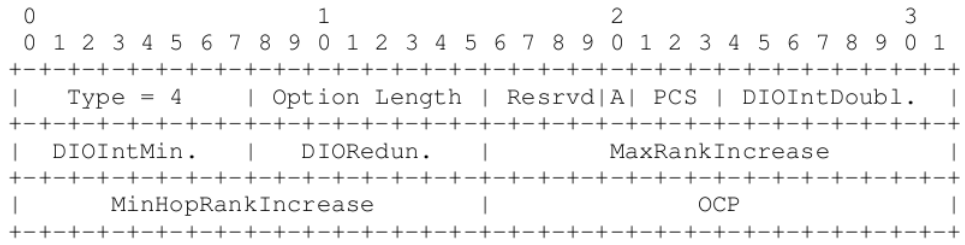


Fig. 4.8: Formato DODAG Configuration Option

Come già accennato esse sono utilizzate per distribuire informazioni di configurazione ai nodi del DODAG, principalmente riferite all'utilizzo del *Trickle Timer* (sezione 4.3.3) e agli intervalli di incremento del rank di ciascun nodo. Queste informazioni devono essere incluse in un pacchetto DIO in risposta ad un messaggio DIS unicast. In dettaglio:

- `Option Type`: 0x04
- `Option Length`: 8 byte
- `Authentication Enabled (A)`: bit indicante la modalità di sicurezza utilizzata nella rete.

- `Path Control Size (PCS)`: campo intero di 3 bit senza segno che permette di definire un valore utilizzato nelle opzioni Target Information al fine di accettare da un nodo richieste di cammini multipli.
- `DIOIntervalDoublings`: campo intero di 8 bit senza segno usato per configurare il valore `Imax` utilizzato dall'algoritmo Trickle Timer.
- `DIOIntervalMin`: campo intero di 8 bit senza segno usato per configurare il valore `Imin` utilizzato dall'algoritmo Trickle Timer.
- `DIORedundancyConstant`: campo intero di 8 bit senza segno usato per configurare il valore `K` utilizzato dall'algoritmo Trickle Timer.
- `MaxRankIncrease`: campo intero di 16 bit senza segno usato per configurare il valore `DAGMaxRankIncrease` utilizzato per specificare il massimo incremento possibile durante un'operazione di riparazione locale (si faccia riferimento al capitolo 6).
- `MinHopRankInc`: campo intero di 16 bit senza segno usato per configurare il valore `MinHopRankIncrease`.
- `Objective Code Point (OCP)`: campo intero di 16 bit senza segno usato per identificare la funzione obiettivo.

4.2.5.2 Solicited Information Option

Le opzioni *Solicited Information* possono essere incluse in un pacchetto DIS con lo scopo di sollecitare l'invio di un pacchetto DIO e hanno il seguente formato:

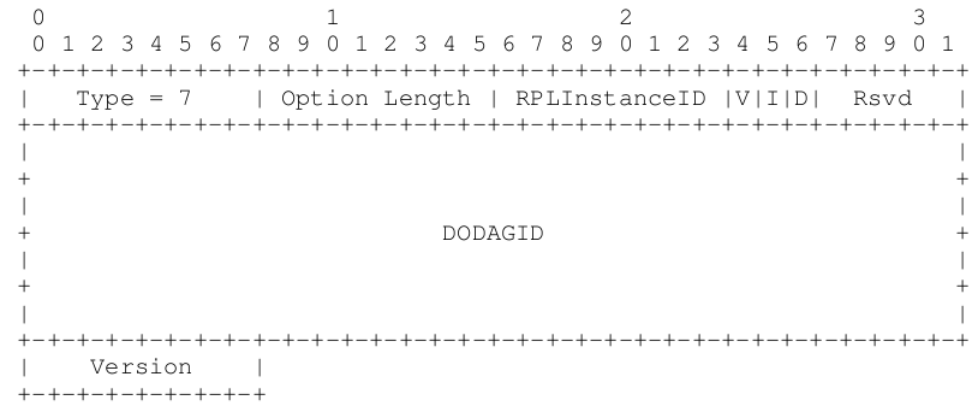


Fig. 4.9: Formato Solicited Information Option

- `Option Type`: 0x07
- `Option Length`: 19 byte

- `V`: se tale bit è impostato, allora il campo `Version` contiene un valore valido di versione del DODAG e il nodo che riceve il pacchetto DIS deve controllare se il valore riportato in tale campo e il proprio coincidono. Se `V` non è impostato, il nodo ricevente deve ignorare il campo `Version`.

- `I`: bit che stabilisce se il campo `RPLInstanceID` contiene un valore valido di istanza di RPL. Il comportamento del nodo ricevente rispetto a tale bit è uguale a quello definito per il campo `V`.

- `D`: bit che stabilisce se il campo `DODAGID` contiene un valore valido. Anche in questo caso il comportamento del nodo ricevente è uguale a quello definito per il campo `V`.

- `RPLInstanceID`: campo intero di 8 bit senza segno valido solo se il campo `I` è impostato e rappresenta un'istanza di RPL.

- `Version`: campo intero di 8 bit senza segno valido solo se il campo `V` è impostato e rappresenta il valore di una versione di DODAG.

- `DODAGID`: campo intero di 128 bit senza segno valido solo se il campo `D` è impostato e rappresenta il valore DODAGID.

4.3 Creazione dei cammini verso il nodo radice

I cammini verso il nodo radice sono costruiti e mantenuti da RPL tramite l'invio di un pacchetto DIO tra i nodi della rete. I campi del pacchetto DIO `G, MOP, Prf, Version, RPLInstanceID` e `DODAGID` sono impostati dal nodo radice e i nodi che ricevono il messaggio devono adottare una configurazione coerente con tali paramentri e inoltrarli inalterati. Ciascun nodo invece può aggiornare i campi `Rank` e `DSTN`.

4.3.1 Scoperta di nodi vicini, selezione dei nodi genitori e del DODAG

Un nodo scopre l'esistenza di una versione di un DODAG tramite la ricezione di un messaggio DIO da parte di altri nodi. L'origine di tale messaggio è il DODAG root e il DIO è propagato attraverso tutti i nodi della rete secondo le regole definite in seguito. Un nodo quindi riceve un messaggio DIO relativo a un DODAG da un numero arbitrario di nodi. Esso seleziona tra di essi un insieme di nodi vicini, *neighbor*, e un insieme di nodi genitori, *parent*. L'esatta specifica con cui vengono definiti tali insiemi dipende dall'implementazione, tuttavia vi sono delle regole da seguire. In particolare i neighbor sono nodi raggiungibili direttamente tramite il mezzo trasmissivo wireless. I nodi parent costituiscono un sottoinsieme dei nodi neighbour il cui rank è strettamente maggiore di quello del nodo appena affiliato al

DODAG, il cui valore è determinato, incrementato e poi propagato a partire dal rank letto nel pacchetto DIO ricevuto e non può essere maggiore di `L + DAGMaxRankIncrease`, dove `L` è il valore più basso di rank ascoltato. Tra i nodi parent è selezionato un unico genitore preferito, *preferred parent*, che rappresenta il nodo di collegamento nel cammino verso la radice il quale avrà un insieme di nodi genitore vuoto. Dalle regole appena descritte si deduce che il rank descresce di passo in passo nel cammino verso il nodo root. Esso propaga un rank pari `ROOT_RANK` che costituisce il valore minimo di rank all'interno del DODAG. Nel caso in cui siano presenti più DODAG all'interno dell'istanza di RPL, un nodo sceglie quello a cui agganciarsi sulla base delle funzioni specificate nell'implementazione che tuttavia deve tenere conto della variabile `Prf`.

4.3.2 Gestione delle versioni di un DODAG

Una versione di DODAG è definita dalla tripla (`RPLInstanceID, DODAGID, DODAGVersionNumber`) stabilita dal nodo radice. Ogni elemento dell'insieme dei genitori di un nodo deve appartenere alla stessa versione del nodo che non può a sua volta comunicare all'interno del DIO un valore di versione diverso da quello di appartenenza. Quando il root incrementa il valore `DODAGVersionNumber` una nuova versione del DODAG viene diffusa nella rete. Non appena un nodo riceve un DIO con `DODAGVersionNumber` maggiore del suo attuale, deve migrare verso la nuova versione e comunicarlo nei DIO trasmessi.

4.3.3 Processamento e trasmissione del pacchetto DIO

Quando un nodo riceve un DIO, il cui contenuto non è corrotto, da parte di un nodo vicino, questo viene processato con il fine di stabilire se tale nodo può essere posto nell'insieme dei genitori e, in caso, essere eletto preferred parent. Da notare che il rank dei nodi all'interno dell'insieme dei parent può essere minore o uguale a quello del genitore preferito.

Seguendo quanto descritto nella sezione 2.3, uno dei criteri fondamentali di progettazione del protocollo è la possibilità di diminuire il traffico di controllo a seconda delle necessità. Per raggiungere tale scopo, RPL sfrutta un algoritmo detto *Trickle Timer* [44]. Esso necessita di tre parametri comunicati nel pacchetto DIO tramite le DODAG Configuration Option:

- `Imin`: rappresenta un tempo e costituisce l'intervallo di partenza dell'algoritmo.

- `Imax`: il suo valore rappresenta il massimo numero di raddoppi del valore `Imin` ed è un numero naturale.

- `K`: numero naturale maggiore di zero che rappresenta la costante di ridondanza.

Ad esse si aggiungono altre 3 variabili necessarie al funzionamento dell'algoritmo.

- `I`: rappresenta il corrente intervallo di tempo.
- `t`: valore di tempo all'interno del corrente intervallo.
- `c`: variabile contatore.

L'inizializzazione dell'algoritmo avviene assegnando `I = Imin` e `c = 0`. All'inizio di un intervallo, che termina dopo `I` unità di tempo, `t` viene scelto in maniera casuale all'interno del range [`I`/2, `I`). Ogni volta che viene rilevata una trasmissione *consistente*, la variabile `c` è incrementata di una unità e quando termina l'intervallo definito da `t` l'algoritmo permette la trasmissione del DIO solo se è verificata la condizione `c < K`. Infine, al termine dell'intervallo definito da `I`, il suo valore è raddoppiato fino a raggiungere il valore `Imax`, nel qual caso rimane `I = Imax` e viene impostato `c = 0`. Quando viene rilevata una trasmissione *inconsistente*, l'algoritmo effettua un reset delle variabili impostando nuovamente `I = Imin` e `c = 0`. Nell'ambito di RPL possiamo definire *consistente* la ricezione di DIO che non comporta cambiamenti del genitore preferito o del rank. Si ha invece una trasmissione *inconsistente* alla ricezione di un DIS multicast senza l'opzione Solicited Information oppure, nel caso in cui sia presente tale opzione, il nodo verifica i parametri definiti al suo interno prima di giudicarla inconsistente. Anche l'agganciamento ad una nuova versione di un DODAG deve essere considerata inconsistente. Il nodo non dovrebbe fare il reset del Trickle Timer alla ricezione di un pacchetto DIS diretto esplicitamente a lui, ma inviare semplicemente un pacchetto DIO con le opzioni DODAG Information Option in unicast al nodo origine e, nel caso in cui sia presente l'opzione Solicited Information, l'invio è vincolato alla coincidenza dei parametri presenti nell'opzione stessa.

4.4 Creazione dei cammini dal nodo radice

La gestione e la creazione dei percorsi dal nodo radice verso gli altri nodi (rotte verso il basso), che permette di gestire il traffico P2MP e P2P, avviene tramite i pacchetti DAO inviati verso il nodo radice. Questo è possibile tramite la selezione di uno o più *next-hop*, detti *DAO parent*, all'interno dell'insieme dei DODAG parent che può coincidere o meno con il preferred parent. I campi `RPLInstanceID` e `DODAGID` del pacchetto DAO devono avere gli stessi valori presenti nel DIO, mentre il campo `DAOSequence` è incrementato al massimo di una unità per ogni nuovo DAO inviato. Infine l'abilitazione del bit `K` e la conseguente richiesta di un DAO-ACK è opzionale, ma

un nodo può comunque inviare un DAO-ACK in risposta a un DAO senza `K` impostato con il fine di segnalare degli errori. Come già precedentemente accennato, è possibile disabilitare la creazione delle rotte verso il basso, oppure decidere la modalità con cui queste vengono mantenute, impostando il campo `MOP` del pacchetto DIO. Nel caso in cui tale valore sia 0, nessun nodo agganciato al DODAG deve inviare messaggi DAO e ignorarli in ricezione.

Poichè i pacchetti DAO sono diretti verso il nodo radice, la loro ricezione da parte di un nodo potrebbe richiedere a sua volta la trasmissione di un DAO. Questo invio non dovrebbe essere immediato, ma ritardato in modo da permettere l'aggregazione di dati provenienti da altri nodi. È possibile da parte di un nodo costringere il corrispondente sub-DODAG (sottoalbero con radice il nodo in questione) a inviare messaggi DAO incrementando il campo `DSTN` del messaggio DIO.

4.4.1 Non-storing Mode

In questa modalità operativa, i cammini vengono costruiti utilizzando la tecnica *source routing* senza mantenere tabelle di instradamento nei nodi. Le rotte sono popolate dal nodo radice e vengono costruite sfruttando DAO contenenti le opzioni Transit information e RPL Target. In particolare il campo `Parent Field` dell'opzione Transit information specifica uno o più DAO parent di un nodo figlio, il cui indirizzo è specificato dalla RPL Target information che deve precedere l'altra opzione. I messaggi DAO risalgono il DODAG tramite forward successivi fino al nodo radice, il quale aggrega le informazioni per calcolare i cammini verso ciascun nodo.

4.4.2 Stroring Mode

La modalità Storing mode sfrutta le tabelle di routing per determinare i cammini verso verso il basso. Il messaggio DAO non contiene informazioni sul percorso, ma questo è derivato implicitamente dal pacchetto stesso tramite l'indirizzo sorgente e l'indirizzo destinazione verso cui il pacchetto è stato instradato. Anche in questo caso i pacchetti sono indirizzati verso il nodo radice, ma ciascun nodo che riceve il DAO memorizza in una tabella l'informazione su chi ha generato tale DAO e il nodo vicino che permette di raggiungerlo. Quindi ciascun DAO parent avrà una tabella relativa ai nodi del proprio sub-DODAG.

CAPITOLO 5

Implementazione e prestazioni di RPL

Indice

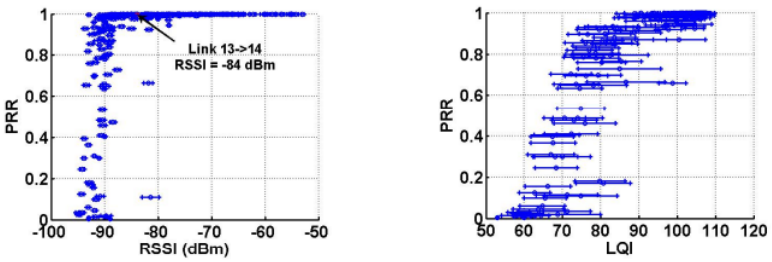
In questo capitolo viene descritta dettagliatamente l'implementazione del protocollo RPL in linguaggio NesC e le sue prestazioni di utilizzo sul testbed del dipartimento di ingegneria dell'informazione (DEI) dell'università degli studi di Padova. Le diverse scelte implementative adottate sono pensate per un protocollo operante su tale tipologia di struttura hardware e sul framewok TinyNET. Altre implementazioni del protocollo RPL pubbliche sono [45] e [46]. La prima si riferisce ad una versione di RPL inserita all'interno del progetto Contiki, un sistema operativo alternativo a TinyOS per le reti di sensori. La seconda invece costiuisce un'integrazione del protocollo nel sistema operativo TinyOS. La descrizione che segue ripercorre la struttura del capitolo 4 evidenziando e motivando le scelte implementative adottate.

5.1 Considerazioni iniziali

L'idea alla base del progetto è quella di realizzare una versione base funzionante del protocollo. Quindi molte funzionalità aggiuntive non sono state prese in considerazione, ma lasciate ad un successivo sviluppo. Seguendo questa linea guida si considera la presenza di un'unica istanza di RPL

con la creazione di un unico DODAG con un nodo radice, il sink. Non sono quindi previste funzionalità relative alla gestione di diversi DODAG, con conseguente mancanza di distinzione tra grafi *grounded* e *floating*. L'implementazione inoltre non consente al nodo root di cambiare versione del DODAG, che rimane sempre la stessa. Il nodo sink ha il compito di costruire il primo DIO stabilendo i parametri necessari alla creazione del DODAG e trasmetterlo in broadcast. I nodi che ricevono il DIO impareranno le configurazioni e trasmetteranno a loro volta, sempre in broadcast, il DIO in modo da raggiungere tutti i nodi della rete. Il rank di un nodo è definito come un intero senza segno rappresentante il numero di salti necessario a raggiungere il nodo radice (*hopcount*). Esso è incrementato di una unità scendendo in profondità nel grafo a partire dal valore 1 che identifica in maniera univoca il rank del root. Per quanto riguarda la modalità di costruzione delle rotte verso il basso, si è scelto di utilizzare quella storing senza multicast.

Il componente CC2420 montato sui nodi Tmote sky utilizzati nel testbed del DEI mette a disposizione due grandezze per misurare la qualità del link, LQI e il RSSI. Questi due valori quindi possono essere utilizzati come metrica per valutare la qualità di un link. L'articolo [47] presenta i risultati di uno studio condotto proprio con il fine di valutare la corrispondenza tra la qualità del link e valore di RSSI e LQI fornito dal componente CC2420. Per raggiungere tale scopo trenta nodi sono stati programmati in modo tale da inviare duecento pacchetti, ciascuno ad intervalli di 10ms, verso tutti gli altri nodi, uno alla volta. Una volta terminata la trasmissione, vengono raccolti il valore di RSSI, LQI e il numero di pacchetto ricevuti, *packet reception rate* (PRR). I risultati, sono raffigurati in figura 5.1 per un livello di potenza pari a 0 dBm, lo stesso utilizzato nel testbed del DEI durante i test condotti usando RPL. I grafici raffigurano la media e la deviazione standard di RSSI e LQI sulle 200 prove dei dati raccolti su ogni link.



(a) PRR vs RSSI (b) PRR vs LQI

Fig. 5.1: Risultati ottenuti dai test in [47]

La grandezza di RSSI dimostra di avere un'ottima stabilità e varia molto poco rispetto al suo valore medio. I valori superiori a una soglia intorno ai

-85 dBm sono indice di una buona qualità del link avendo PRR sempre superiore all'85%. Quando si supera tale soglia invece il parametro PRR può assumere un qualsiasi valore tra 0 e 100%, rendendo impossibile dare un giudizio sull'affidabilità del collegamento. Prove simili condotte sul testbed del dipartimento di ingegneria a Padova hanno ottenuto i medesimi risultati, evidenziando tuttavia una soglia ottimale pari a -80 dBm. Discorso diverso per quanto riguarda LQI. Come si può vedere dal corrispondente grafico, essa dimostra avere una deviazione standard di gran lunga superiore rispetto all'RSSI, oltre ad una difficile identificazione di un range di valori che stabilisca con ragionevole certezza un link affidabile. Anche in questo caso, esperimenti condotti sul testbed hanno confermato questi risultati, rendendo difficile stabire una corrispondenza diretta tra valore di LQI e qualità del collegamento. Per tali motivi, si è scelto di utilizzare solo la grandezza RSSI come metrica di link per l'implementazione di RPL. Infine, poichè i nodi del testbed sono tutti uguali e hanno un'alimentazione costante tramite USB, non sono state prese in considerazione metriche relative allo stato del nodo, come per esempio il suo livello di energia residuo.

5.2 Creazione dei cammini verso il sink

5.2.1 Messaggio di controllo DIO

Il messaggio di controllo DIO è definito dalla seguente struttura:

```
1  typedef nx_struct dio_msg
2  {
3    nx_uint8_t  type;
4    nx_uint8_t  code;
5    nx_uint16_t cksum;
6
7    nx_uint8_t  RPLInstanceID;
8    nx_uint8_t  version;
9    nx_uint16_t rank;
10   nx_uint8_t  G:1;
11   nx_uint8_t  O:1;
12   nx_uint8_t  MOP:3;
13   nx_uint8_t  Prf:3;
14   nx_uint8_t  DTSN;
15   nx_uint16_t reserved;
16   nx_struct in6_addr_nx DODAGID;
17   nx_uint8_t  subopt[0];
18 } dio_msg_t;
```

dove i primi tre campi si riferiscono all'header standard ICMPv6, mentre le altre variabili corrispondono ai paramentri definiti nella sezione 4.2.1. Tra le varie opzioni che possono essere integrate all'interno del pacchetto DIO, viene utilizzata solo la DODAG configuration option, necessaria per definire i paramentri dell'algoritmo Trickle Timer. Il sink si occupa della creazione e della trasmissione del primo DIO con la DODAG configuration

option. I valori dei vari parametri rispecchiano le considerazioni svolte precedentemente, tuttavia è importante ricordare che `rank` assume valore 1, mentre a `DODAGID` è assegnato un numero esadecimale corrispondente al TOS_NODE_ID del nodo root.

5.2.2 Scelta dei nodi neighbour, parent e preferred parent

Il nodo sorgente di un pacchetto DIO ricevuto da un altro nodo è posto in una tabella che costituisce l'insieme dei nodi vicini. La cardinalità della tabella è un paramentro del protocollo, ed è impostata ad una valore tra cinque e otto valore adeguato a dare un'idea precisa dei neighbor senza incidere eccessivamente sulla quantità di memoria richiesta. Una volta che la tabella è stata riempita, eventuali nodi da cui ha avuto origine la trasmissione di un DIO vengono semplicemente ignorati senza aggiornamento della tabella stessa. Insieme al TOS_NODE_ID, viene memorizzato il rank e il valore di RSSI. Un neighbor può essere eletto genitore se viene verificata la seguente relazione:

```
1 if(nb.hopcount < hopcount && nb.rssi >= RSSI_THRESHOLD)
```

dove `nb` è una struttura contenente le informazione riferite al vicino, `hopcount` è il corrente valore di rank del nodo (la cui inizializzazione è 0xFF) e `RSSI_THRESHOLD` è un valore di RSSI che, viste le precedenti considerazioni, è uguale a -80 dBm. La formazione dell'insieme dei genitori è quindi vincolata ad un confronto sul rank e sulla qualità del link. Per quanta riguarda la prima condizione, si vuole che il genitore abbia hopcount strettamente minore di quello attuale del nodo. Da notare che alla ricezione del primo DIO questa condizione è sempre verificata. Il vincolo sulla qualità del link impone invece superamento di una soglia di RSSI. Il parent set è formato al più da tre nodi ed è rappresentato da un array di strutture `parent` di cardinalità tre. Le informazioni memorizzate sono anche in questo caso l'indirizzo, il valore di rssi e il rank. Poichè vi è una sostanziale equivalenza tra i parent, il ruolo di genitore preferito è assegnato al nodo posto nella prima posizione dell'array, che coincide con il primo nodo il cui rank publicizzato nel DIO è risultato strettamente minore di quello del nodo ricevente. Il preferred parent così eletto diventa il *next hop* del nodo e il rank è impostato uguale a quello di tale genitore aumentato di una unità. Ad ogni DIO ricevuto, viene effettuato un controllo nel parent e neighbor set in modo da aggiornare il valore di rank e di rssi corrispondente.

5.2.3 Trasmissione del messaggio DIO

La trasmissione del messaggio DIO segue fedelmente quanto esposto nella sezione 4.3.3. Il nodo apprende i paramentri dai valori presenti nell'opzione DODAG configuration option e avvia il Trickle Timer non appena è eletto il

genitore preferito. Allo scadere del tempo `t`, viene trasmesso il primo DIO. Infine il paramentro `c` dell'algoritmo è incrementato ad ogni ricezione di un DIO successivo al primo.

5.3 Creazione dei cammini verso il basso

5.3.1 Messaggio di controllo DAO

La struttura che definisce il messaggio di controllo DAO è la seguente:

```
1  typedef nx_struct dao_msg {
2    nx_uint8_t   type;
3    nx_uint8_t   code;
4    nx_uint16_t cksum;
5
6    nx_uint8_t   RPLInstanceID;
7    nx_uint16_t K:1;
8    nx_uint16_t D:1;
9    nx_uint16_t reserved:14;
10   nx_uint16_t DAOSeq;
11   nx_struct in6_addr_nx DODAGID;
12   nx_uint8_t   subopt[0];
13 } dao_msg_t;
```

Come nel caso del DIO i primi tre campi indentificano l'header ICMPv6. Poichè la modalità di costruizione dei cammini è *storing*, le informazioni necessarie per la creazione delle tabelle possono essere ricavate dal messaggio DAO stesso e quindi non è necessario aggiungere alcun tipo di opzione. Ai campi del pacchetto DAO sono assegnati gli stessi valori per tutti i nodi, `RPLInstanceID` è ricavato dall'omonimo campo del DIO e `D` è posto uguale a 0 in quanto vi è un unico DODAG nella rete.

5.3.2 Trasmissione del messaggio DAO

Seguendo la medesima filosofia dell'algoritmo Trickle Timer, anche l'invio del messaggio DAO avviene tramite l'utilizzo di un timer la cui durata cresce con il tempo di esecuzione del protocollo fino a stabilizzarsi ad un valore fisso. Infatti dopo la prima fase in cui è necessario trasmettere i DAO con un certa frequenza in modo da riuscire a popolare le tabelle di routing, queste, essendo la rete del testbed statica, non richiedono un aggiornamento frequente. Il comportamento in caso di un'eventuale rottura del collegamento è descritto nel capitolo 6. Il DAO è un messaggio con destinazione sempre il nodo root del DODAG.

Appena un nodo sceglie il suo genitore preferito, viene inizializzato un timer `DAO_Timer` di durata pari a un iniziale valore di default

$$\texttt{DAO\_Timer} = 2^{\texttt{DEFAULT\_DAO\_INTERVAL+DAO\_Counter-1}} + \texttt{dao\_offset}$$

dove `DEFAULT_DAO_INTERVAL` è un parametro del protocollo, `DAO_Counter` rappresenta il numero di DAO trasmessi ed è inizializzato ad 1, mentre `dao_offset` è un tempo casuale scelto nell'intervallo [0, 4) secondi. Quando il timer termina, viene trasmesso un DAO con conseguente incremento della variabile `DAO_Counter` e inizializzato nuovamente il timer stesso secondo la formula precedente. Questo prosegue fino a quando la quantità `DEFAULT_DAO_INTERVAL+DAO_Counter` non raggiunge il paramentro `MAX_DAO_INTERVAL` che costituisce il limite superiore di incremento del timer, raggiunto il quale il DAO viene trasmesso costantemente con un intervallo pari a

$$\texttt{DAO\_Timer} = 2^{\texttt{MAX\_DAO\_INTERVAL}} + \texttt{dao\_offset}$$

Tutte le unità di tempo appena descritte, compreso `dao_offset`, sono da considerarsi in millisecondi.

5.3.3 Creazione delle tabelle di routing

Il messaggio DAO è indirizzato sempre verso il nodo radice tramite il *next-hop* dei nodi che ne fanno il forward. Durante questa operazione, un nodo memorizza in una tabella l'indirizzo sorgente originale del pacchetto DAO e l'indirizzo sorgente del nodo che ha effettivamente inoltrato il messaggio. Entrambe queste informazioni sono presenti nell'header di un messaggio gestito da framework TinyNET. Il nodo radice, destinatario di tutti i pacchetti DAO, avrà una tabella completa di tutti i nodi della rete a cui è associato il nodo hop direttamente raggiungibile tramite il mezzo trasmissivo. Quindi in conclusione ciascun nodo che assume il ruolo di preferred parent per un altro nodo crea e mantiene le tabelle di instradamento.

5.4 Messaggi di controllo DIS

I messaggi di controllo DIS hanno lo scopo di innescare l'invio di un DIO e sono definiti dalla struttura

```
1 typedef nx_struct dis_msg
2 {
3 nx_uint8_t  type;
4 nx_uint8_t  code;
5 nx_uint16_t cksum;
6
7 nx_uint16_t reserved;
8 nx_uint8_t  opt[0];
9 } dis_msg_t;
```

Nell'implementazione del protocollo sviluppata, questa possibilità è prevista in due casi.

Il primo caso riguarda una situazione in cui un nodo viene acceso in una rete in cui tutti gli altri nodi eseguono un'applicazione che sfrutta il protocollo RPL da un tempo sufficientemente lungo per cui la trasmissione dei DIO gestita dall'algoritmo Trickle avviene ad intervalli molto ampi. Per evitare che il nodo rimanga inattivo per molto tempo, nella sue fase di inizializzazione viene attivato un timer il cui intervallo è definito dal paramentro `DIS_MULTI_TIMER`. Se in tale intervallo il nodo rileva un DIO da cui può definire un genitore preferito, allora il timer viene rimosso. Se questo non avviene, allo scadere del timer viene inviato un DIS in broadcast. Tale evento costituisce un trasmissione *inconsistente* che ha l'effetto di un reset del Trickle Timer per tutti i nodi che ricevono il DIS. In questo modo il nodo riceverà immediatamente i pacchetti DIO che gli permettono di agganciarsi al DODAG.

Il secondo scenario riguarda un controllo sul link tra il nodo e il suo preferred parent quando non vi sono trasmissioni di informazioni tra i due. Nel caso in cui vi sia traffico tra di essi, il controllo sulla qualità del canale è ampiamente descritto nel capitolo 6. Alla scelta del genitore preferito, viene attivato un timer il cui intervallo è posto a `DIS_UNICAST_TIMER` $= 2^{Imax}$ che corrisponde al massimo intervallo possibile di trasmissione di due DIO consecutivi. Se prima dello scadere del timer il nodo riceve un DIO dal genitore preferito, il timer viene semplicemente reinizializzato con lo stesso valore. Se questo non avviene e il timer scade, viene generato un messaggio DIS unicast compreso della Solicited Information option diretto verso il genitore preferito e contemporaneamente inizializzato un ulteriore timer con valore `DIS_MULTI_TIMER`. La ricezione di un DIS unicast non è un evento *inconsistente* e innesca solo l'invio di un pacchetto DIO. Se il messaggio viene ricevuto prima dello scadere del `DIS_MULTI_TIMER`, allora questo timer viene rimosso e il `DIS_UNICAST_TIMER` nuovamente attivato con il valore iniziale. In caso contrario vengono eliminate tutte le informazioni presenti nelle tabelle dei vicini e dei genitori e generato un DIS in broadcast in modo da ricevere DIO da tutti i nodi vicini e popolare nuovamente le tabelle. Questa operazione è in contrasto con il criterio *Loss Response* definito nella sezione 2.3 in quanto il cammino non è effettivamente sfruttato dall'applicazione. Tuttavia si tratta di un traffico di controllo ridotto al minimo lungo un canale inutilizzato ed è un buon compromesso rispetto alla perdita delle informazioni nel caso in cui tale canale sia nuovamente necessario all'applicazione ma uno dei due estremi non è attivo per un qualche motivo. Prima dell'invio del DIS multicast il nodo invia un DIO con rank infinito in modo da innescare l'operazione di riparazione locale nei figli (capitolo 6).

5.5 Analisi delle prestazioni

In questa sezione vengono presentati i risultati di performance ottenuti dal protocollo RPL implementato come descritto precedentemente. Altri lavori riguardanti l'analisi delle prestazioni del protocollo RPL sono [48] e [49]. Entrambi però sfruttano un ambiente simulato, in particolare il primo utilizza Omnet++[1], mentre il secondo NS2[2]. I test i cui risultati sono presentati di seguito sono invece realizzati in uno scenario reale quale il testbed del DEI. Questo comporta una verifica pratica dell'implementazione e non solo teorica come avverrebbe in uno scenario simulato, in cui per esempio la comunicazione tra i nodi sfrutta un particolare modello di propagazione. D'altra parte risulta difficile analizzare con precisione alcuni dettagli del protocollo a causa di problematiche di origine diversa. Innanzitutto l'intrinseca complessità del debug che caratterizza il sistema TinyOS e le reti di sensori in generale. Inoltre operazioni di test in scenari reali devono tenere conto di inconvenienti e interferenze indipendenti dal funzionamento del protocollo, situazioni abbastanza comuni in un ambiente ampiamente distribuito come le reti di sensori.

Poichè i lavori precedentemente citati sono basati su un ambiente simulato totalmente diverso rispetto allo scenario del testbed, non è possibile comparare i risultati ottenuti. L'articolo [50] presenta invece l'analisi di un'applicazione di monitoraggio basata sul protocollo RPL implementata nel testbed del DEI. Il lavoro espone alcuni risultati prestazionali ottenuti dalla prima versione di RPL sviluppata dal gruppo di ricerca SIGNET dell'università di Padova. Le analisi che seguono approfondiscono notevolmente questi risultati, coprendo aspetti tralasciati nell'articolo.

La metodologia di esecuzione dei test comprende una ripetizione degli stessi in modo da avere media dei valori di interesse. Gli aspetti presi in considerazione riguardano il comportamento del protocollo al variare dei paramentri del trickle timer, la scalabilità, il traffico di controllo generato e il troughput di un'applicazione basata su RPL. Nel resto della sezione sono specificati i parametri utilizzati per ciascun test, tuttavia i valori utilizzati per la trasmissione dei pacchetti DAO sono sempre gli stessi. In particolare, `DEFAULT_DAO_INTERVAL` assume il valore 12 corrispondente a 4096 millisecondi, mentre `MAX_DAO_INTERVAL` è impostato a 18 corrispondente a circa 4 minuti. Alcuni grafici presentano risultati dipendenti dal tempo di convergenza del protocollo che equivale al tempo in cui il sink riceve i DAO da tutti i nodi partecipanti al test ed è calcolato a partire dall'invio del primo DIO.

[1] http://www.omnetpp.org/
[2] http://www.isi.edu/nsnam/ns/

5.5.1 Parametri del trickle Timer

La campagna di test presentata in questa sezione analizza il comportamento del protocollo al variare dei parametri `Imin` e `K` del trickle timer. Il range di valori di `Imin` è 7, 10, 12 e rappresenta un esponente. I corrispondenti tempi sono quindi rispettivamente 128, 1024, 4096 millisecondi. Per quanto riguarda `K`, questo assume valori 1, 2, 5, 255. Il valore 255 corrisponde a disattivare il meccanismo di non trasmissione dei pacchetti DIO in relazione al numero di eventi consistenti rilevati, inviandolo sempre allo scadere del tempo `t`. Il valore di `Imax` è pari a `Imin + 7`, con un raddoppio di 7 volte del valore di `Imin` prima di raggiungere il valore definitivo rispettivamente di 16,384, 131,072, 524,288 secondi. I test di questa sezione considerano uno scenario di 52 nodi e un sink, con massima distanza da esso pari a 6 hop.

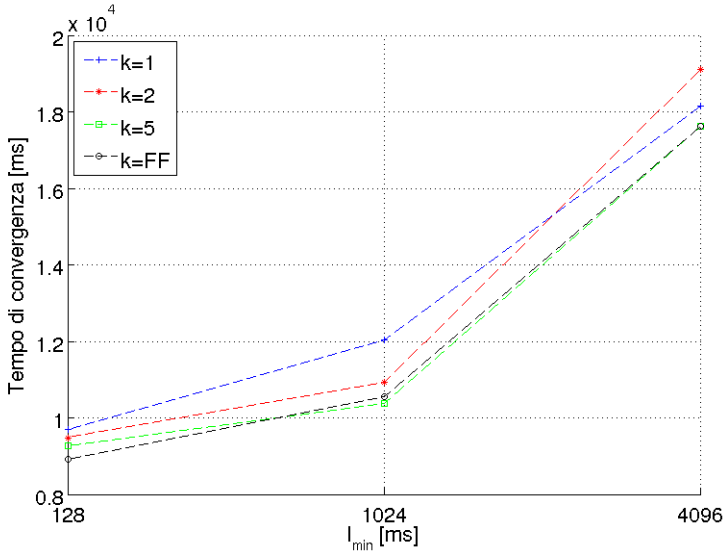


Fig. 5.2: Tempo di congergenza vs. `Imin`

Il grafico in figura 5.2 rappresenta il tempo di convergenza del protocollo in relazione al valore di `Imin` per ciascun valore di `K`. Si nota come tale valore cresce con l'aumentare di `Imin`, e valori bassi di `K`, 1 e 2, hanno risultati peggiori rispetto a 5 oppure al meccanismo disattivato. L'andamento crescente del tempo di convergenza rispetto a `Imin` è dovuto all'algoritmo trickle timer. Si ricorda infatti che il primo DIO viene inviato da ciascun nodo allo scadere del tempo `t` e alla prima esecuzione dell'algoritmo tale tempo è scelto in maniera casuale nell'intervallo [`Imin`/2, `Imin`). La propagazione del DIO nel DODAG quindi aumenta all'aumentare di `Imin` e questo risultato è confermato dal grafico di figura 5.3 in cui sono presentati i tempi di

propagazione del DIO nei livelli del grafo per `K` pari a 5.[3]

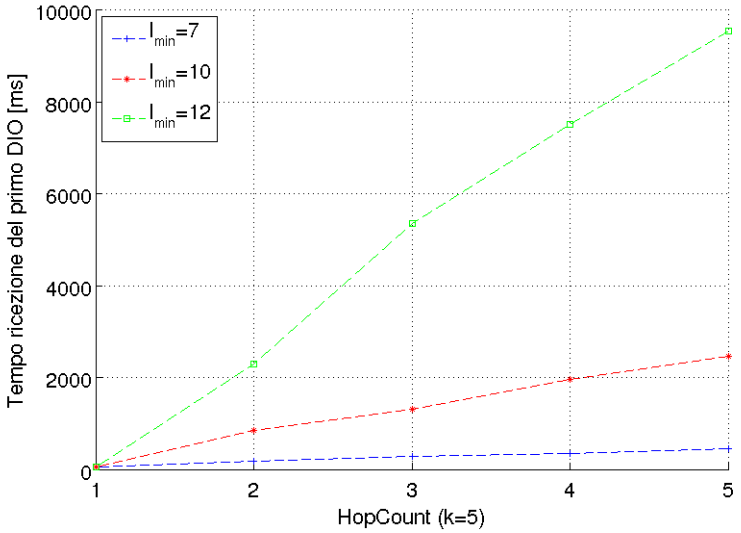


Fig. 5.3: Ricezione del primo DIO vs. hopcount per K=5

Si noti come il tempo di ricezione relativo ad hopcount uguale a 1, corrispondente ai nodi che hanno come genitore preferito il sink, è lo stesso per tutti i valori di `Imin` e altro non è se non il tempo di propagazione del pacchetto nel mezzo trasmissivo tra due nodi entro il proprio raggio di copertura rispetto al limite di -80 dbm di RSSI e varia circa tra i 40 e 70 ms. Una possibile modifica al protocollo descritto nelle pagine precedenti è imporre l'invio immediato del DIO non appena si è selezionato un genitore preferito. Questo comporterebbe una propagazione del DIO nel DODAG indipendente dal parametro `Imin` in un tempo variabile tra i 200 e i 350 ms per 5 hop, valore confermato dalla retta corrispondente a `Imin=7` in cui la ricezione del DIO a hopcount 6 è circa di 450 ms (si ricorda che in questo caso oltre al tempo di propagazione va sommato il tempo `t` selezionato all'interno dell'intervallo [64, 128) ms per ciascun hop). Altro fatto che influenza notevolmente il tempo di convergenza del protocollo è l'istante di invio del primo DAO che si ricorda essere determinato da un tempo causale nell'intervallo (4096, 8192) ms. Si consideri quindi come esempio la spezzata riferita `K=5` per `Imin=10` (1024 ms) nel grafico 5.2, che specifica un tempo di convergenza di poco superiore ai 10 secondi. Tale risultato è determinato dai 2,5 secondi necessari al pacchetto DIO per raggiungere i nodi con hopcount 6 e proprio uno di essi genera il DAO dopo circa 7,5 secondi[4], a cui bisogna

[3] Grafici dallo stesso andamento si ottengono per gli altri valori di `K`

[4] Il sistema operativo TinyOS determina una sequenza di numeri pseudocasuali utilizzando come seme il TOS_NODE_ID del nodo, per cui a parità di indice, la sequenza generata è sempre la stessa. Per tale motivo, nelle diverse esecuzione del test, il tempo di

ulteriormente sommare il tempo necessario alla propagazione del DAO fino al sink nel DODAG superando di poco quindi gli 10 secondi totali. Infine, dal grafico si evince come sia bastato l'invio di un DAO da parte di tutti i nodi per avere la completa formazione del grafo, senza quindi alcuna perdita dei pacchetti. Questo è indice di una buona metrica di link su cui è basata la scelta del next-hop, anche se la rete selezionata è relativamente piccola e quindi con pochi problemi di congestione.

Tornando al grafico di figura 5.2, si è già evidenziato come impostando `K=5` o disattivando il meccanismo di *drop* dei DIO, si tendono ad avere risultati leggermente migliori rispetto al caso di `K=1` o `K=2`. L'interpretazione di tale comportamento è da ricercare nel fatto che un limitato traffico di DIO rallenta la loro propagazione nel grafo, in particolare verso quei nodi relativamente isolati in cui la mancata trasmissione del DIO da parte di un nodo è sufficiente per non permettere ad altri nodi di apprendere l'esistenza del DODAG. Tali risultati sono raffigurati nel grafo 5.4, in cui è particolarmente evidente come il tempo di ricezione del DIO aumenti sensibilmente per `K=1`, motivo per cui il tempo di convergenza riportato nel grafo 5.2 per `K=1` e `Imin=10` risulta più alto rispetto a quello registrato per gli altri valori di `K`.

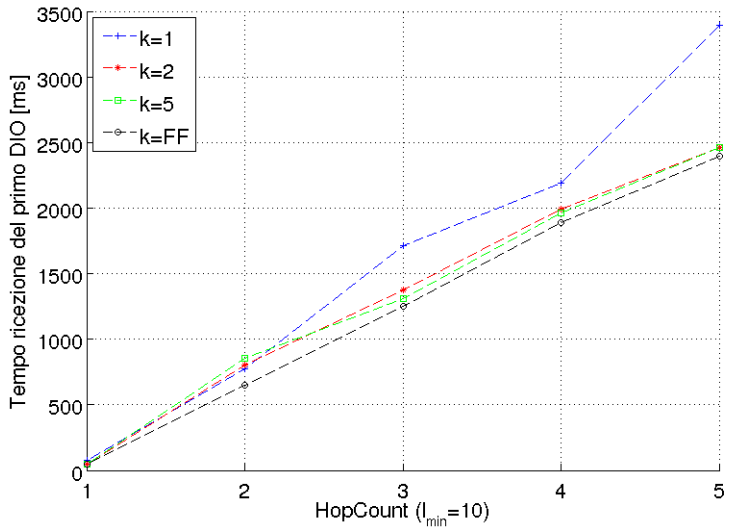


Fig. 5.4: Tempo di ricezione del primo DIO vs. `Imin` per diversi valori di `K`

D'altra parte però disattivare complemetamente il meccanismo di *drop* dei DIO non è la scelta migliore, come evidenzia il grafico 5.5, in cui è raffigurato il numero di DIO trasmessi complessivamente dalla rete entro un intervallo di 30 secondi a partire, per ciascun nodo, dalla ricezione del primo

invio del DAO è sempre uguale per lo stesso nodo.

DIO. Nonostante l'intervallo di tempo preso in considerazione sia contenuto, il traffico dei DIO raggiunge le 150 trasmissioni per `K=255` e `Imin=12`, con una media di 3 trasmissioni per nodo. Per altri valori di `Imin` il risultato è ancora naturalmente peggiore e nel caso di `Imin=7` il numero di trasmissioni triplica tra il caso `K=5` e `K=255`.

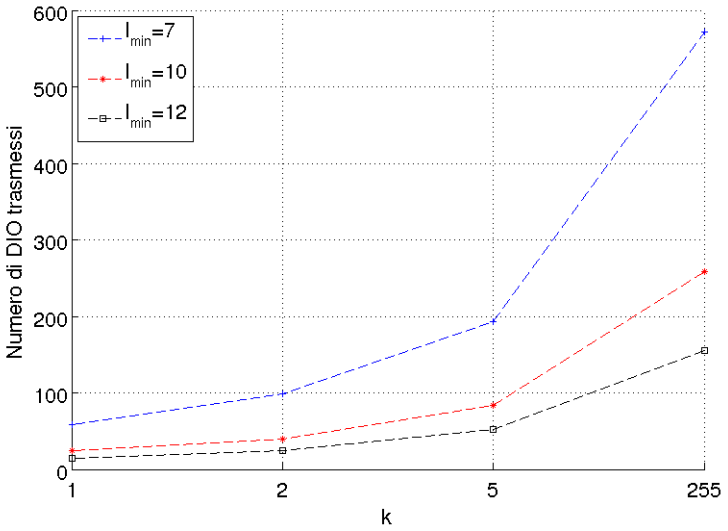


Fig. 5.5: Numero di DIO tramessi vs. `K` per i diversi valori di `Imin`

Un ultimo dato interessante riguarda l'effettiva percentuale di DIO non trasmessi a seconda del valore di `K` ed è presentato nel grafico di figura 5.6. Esso è naturalmente indipendente dal particolare valore di `Imin` e si può notare come `K=1` comporti la non trasmissione di oltre il 90% dei DIO, valore decisamente elevato. Il dato è leggermente inferiore per `K=2`, mentre per `K=5` si raggiunge un buon compromesso con circa il 67% di DIO scartati.

5.5.2 Scalabilità

In questa sezione viene analizzato il comportamento del protocollo in diversi scenari di rete tra i quali cambia la densità dei nodi e la profondità del DODAG. Dai dati presentati nella sezione precedente si può concludere che i valori `K=5` e `Imin=10` consituiscono un buon compromesso tra traffico di controllo generato e tempo di converenza e quindi scelti come paramentri del Trickle Timer per i test descritti nel seguito. Per quanto riguarda i tempi di trasmissione dei DAO, i corrispondenti valori dei paramentri rimangono inalterati rispetto a quelli già descritti.

Gli scenari presi in considerazione sono tre a seconda della profondità del DODAG e ciascuno è ulteriormente suddiviso secondo la densità dei nodi.

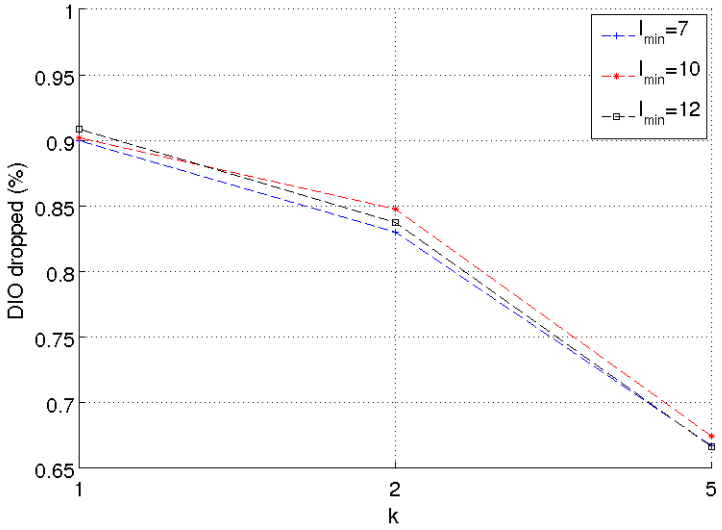


Fig. 5.6: Percentuale di DIO non tramessi vs. `K` per i diversi valori di `Imin`

Scenario 1 : massimo valore di hopcount: 8.

A : 75 nodi.

B : 52 nodi.

C : 15 nodi.

Scenario 2 : massimo valore di hopcount: 5.

A : 46 nodi.

B : 23 nodi.

C : 9 nodi.

Scenario 3 : massimo valore di hopcount: 14.

A : 165 nodi.

B : 64 nodi.

C : 37 nodi.

Il grafico di figura 5.7 presenta il tempo di convergenza in relazione alla densità dei nodi per gli scenari **1** e **2**. Da esso si nota come il tempo di convergenza aumenti all'aumentare del numero di nodi, ma questo incremento è molto contenuto. Per esempio considerando lo scenario **1A** e **1B**, dove vi è una differenza di 50 nodi, l'aumento del tempo di convergenza è di appena 600ms.

Il grafico di figura 5.8 presenta un altro risultato relativo al tempo di convergenza rispetto al valore di hopcount e quindi alla profondità del DODAG.

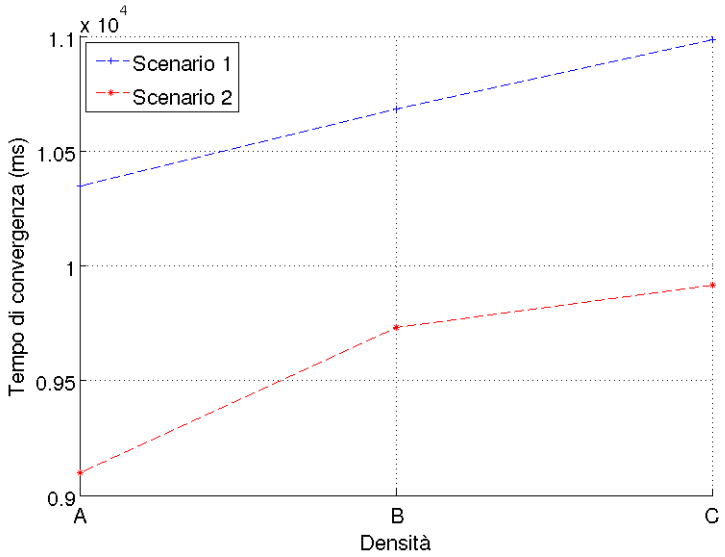


Fig. 5.7: Tempo di convergenza vs. densità dei nodi per gli scenari 1 e 2

Fino ad una profondità di 10 hop, il tempo di convergenza aumenta in maniera regolare passando dai circa 9 secondi per un hop ai circa 14 secondi per dieci hop. A questo punto vi è netto incremento arrivando ai 30 secondi per un DODAG di profondità 15. Il motivo di una così netta differenza è dovuto alla topologia della rete in quanto la notevole profondità del DODAG può comportare una maggiore probabilità sia di perdita dei DAO sia dei DIO, come mostra il grafico di figura 5.9 dove sono raffigurati i risultati di dieci esperimenti per lo scenario **3B** con il corrispondente valore medio (retta azzurra) riportato poi anche nel grafico 5.8. Sette prove su dieci hanno avuto un tempo di convergenza intorno ai 17 secondi indice del fatto che non vi è stata perdita di pacchetti. In tre esperimenti tuttavia questo non è accaduto, in particolare la propagazione dei DIO ha richiesto più trasmissioni portando il tempo di ricezione del DIO oltre i 40 secondi.

5.5.3 Traffico di controllo

Un'ulteriore dato interessante riguarda l'evoluzione del traffico di controllo di RPL nel tempo. Mantenendo i paramentri `K=5` e `Imin=10` con `MAX_DAO_INTERVAL=18` e `Imax=17` si è lasciato eseguire il protocollo RPL nello scenario **1B** per 25 minuti misurando a intervalli di 10 secondi il numero complessivo di DIO trasmessi da tutti i nodi della rete e il numero di DAO ricevuti dal sink. Il grafico 5.10 mostra il numero di DIO in funzione del tempo ed esprime in maniera chiara l'influenza dell'algoritmo Trickle Timer sulle trasmissioni. Nei primi 2 minuti infatti si può notare l'incremento da poco più di 50 a oltre 110 messaggi trasmessi. Si tratta della prima fase del-

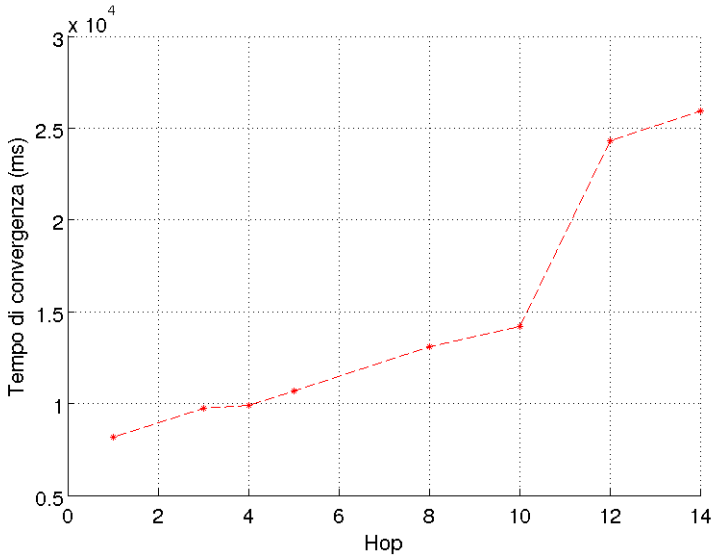


Fig. 5.8: Tempo di convergenza vs. hopcount

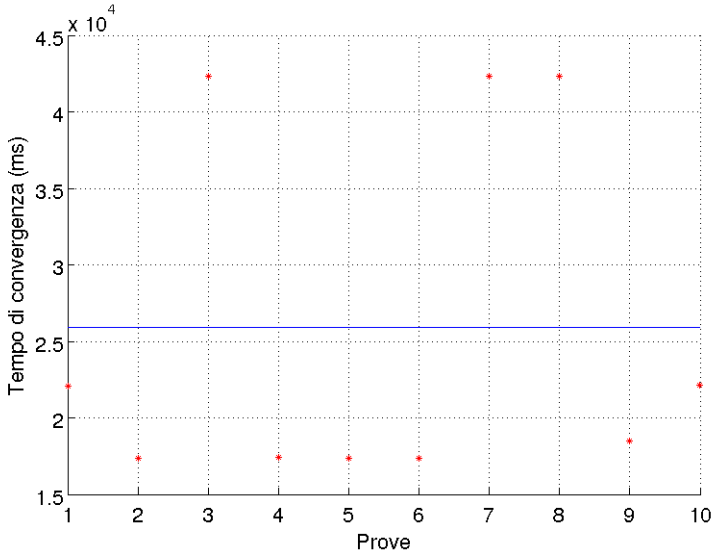


Fig. 5.9: Tempo di convergenza vs. prove effettuate

l'algoritmo in cui `I` viene raddoppiato fino a raggiungere il valore di `Imax`. Nelle fasi successive si raggiunge la stabilità, con un incremento del numero di trasmissioni in un periodo di circa 2 minuti, proprio uguale al valore di `Imax`. L'esatto numero di trasmissioni tra due intervalli consecutivi dipende dal valore di `K` e, con `K=5` e `Imin=10`, si hanno circa 20 DIO inviati dai nodi.

Il grafico di figura 5.11 mostra il numero di DAO ricevuti dal sink in

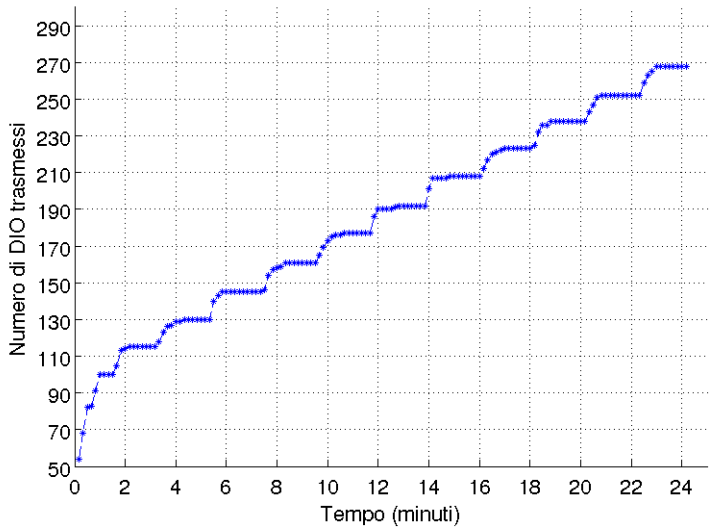


Fig. 5.10: Numero di DIO trasmessi vs. tempo

funzione del tempo. Si nota che l'andamento è simile al precedente, in quanto il procedimento è il medesimo, con la differenza che il limite è raggiunto a circa 4 minuti, valore corrispondente a `MAX_DAO_INTERVAL=18`. Da notare inoltre come l'incremento del numero di DAO sia di 50 tra ogni intervallo ed è dovuto all'assenza di un meccanismo di inibizione delle trasmissioni come avviene per DIO.

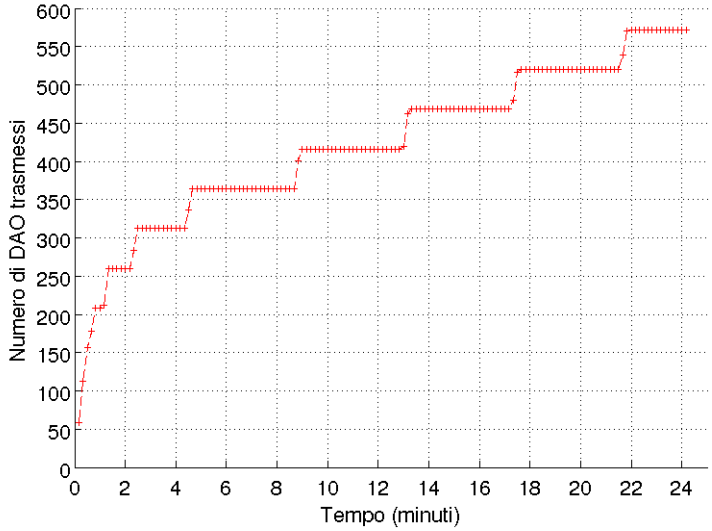


Fig. 5.11: Numero di DAO trasmessi vs. tempo

5.5.4 Troughput

Come già più volte ricordato, il traffico verso il sink è quello più comune all'interno di una rete di sensori e la campagna di test presentata di seguito mira a valutare la capacità della rete del testbed del DEI di gestirlo utilizzando RPL come protocollo di routing. Per tale valutazione sono stati considerati tre scenari, detti per comodità **1**, **2**, **3** rispettivamente di 30, 62 e 89 nodi con profondità 3, 6 e 8 hop. Per ciascuno di essi i nodi inviano 20 pacchetti applicazione verso il sink con tempi di generazione di 500 ms, 1 s, 2 s, 5 s, 10 s ed è stato successivamente verificato il numero effettivo di dati ricevuti dal sink. Intuitivamente più il numero di nodi e la profondità dell'albero cresce, maggiore è il tempo di generazione necessario per avere una ricezione da parte del sink di tutti i pacchetti e i risultati presentati di seguito confermano quest'ipotesi. Da notare che in questo caso RPL ha l'unica funzione di determinare le rotte verso il sink e tutti i test hanno confermato una corretta generazione del DODAG.

Il grafico di figura 5.12 rappresenta la percentuale di pacchetti ricevuti in funzione dell'hopcount e del tempo di generazione per lo scenario **3**. Si nota subito come un tempo di generazione di 5 s, e di conseguenza anche 10 s, garantisce la ricezione da parte del sink di tutti i pacchetti da tutti i nodi. Diminuendo il tempo di generazione la situazione peggiora in particolare per i nodi a profondità maggiore che non riescono a consegnare i loro dati.

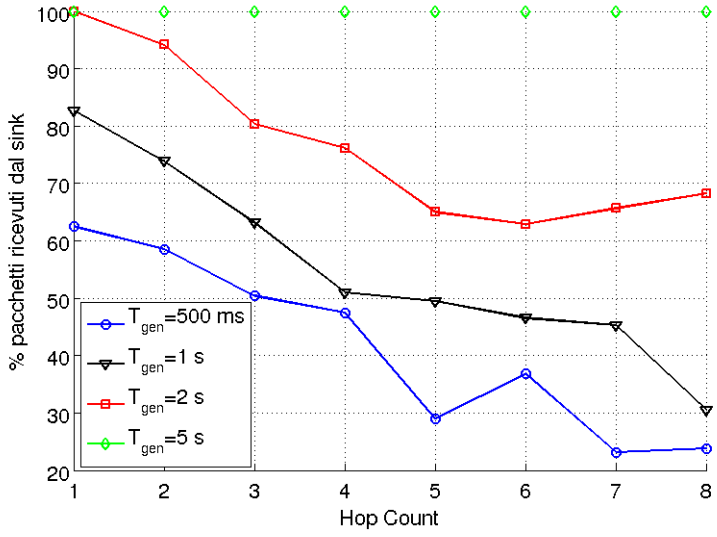


Fig. 5.12: Percentuale di pacchetti ricevuti in funzione dell'hop count e del tempo di generazione per lo scenario **3**

Il grafico di figura 5.13 raffigura la stesso risultato precedente per lo scenario **2**. Si nota subito come il tempo di generazine di 2 s garantisce il 100% di pacchetti ricevuti dal sink, di fatto dimezzato rispetto alla situazione di figura 5.12 e in generale anche i risultati ottenuti per tempi di generazione inferiori sono migliori. Per quanto riguarda infine lo scenario **1**, il limite di abbassa fino a 500 ms (a parte perdite occasionali di al massimo uno o due dati).

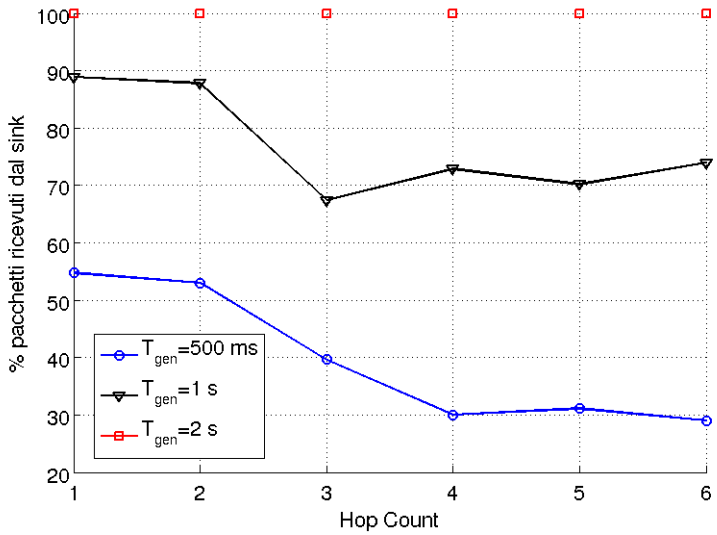


Fig. 5.13: Percentuale di pacchetti ricevuti in funzione dell'hop count e del tempo di generazione per lo scenario **2**

CAPITOLO 6

Gestione dei guasti

Indice

Una delle caratteristiche più importanti di un protocollo di routing è quella di riuscire a gestire situazioni in cui un collegamento viene interrotto per cause diverse, dalla rottura o spegnimento di un nodo alla presenza improvvisa di un ostacolo. La risoluzione di una tale problematica deve mantenere coerente la struttura del DODAG senza l'introduzione di cicli. Questo capitolo si occupa di presentare le strategie che permettono al protocollo RPL di affrontare situazioni di questo tipo. Mentre nella prima parte sono descritte le tecniche presentate nel draft [35], la seconda descrive un approccio leggermente differente e più specifico per reti di sensori statiche come il testbed del DEI.

6.1 Tecniche di gestione di cicli

La struttura DODAG su cui si basa RPL ha come caratteristica intrinseca quella di essere aciclica. Infatti, il vincolo per cui un nodo assume rank strettamente maggiore dei nodi appartenenti al suo parent set impedisce la formazione di un ciclo che porterebbe ad uno scambio di un messaggio tra lo stesso gruppo di nodi senza poter mai raggiungere la destinzazione o il sink. Quindi se i nodi mantengono sempre lo stesso valore di rank nel tempo, vi è la certezza dell'assenza di cicli nel DODAG. Tuttavia non è raro il caso in

cui un nodo necessiti di modificare il proprio rank con lo scopo di poter includere altri nodi all'interno dell'insieme dei genitori. Le ragioni di una tale operazione possono essere diverse e dovute principalmente alla valutazione di una qualche metrica che induca il nodo a includere un genitore nell'insieme oppure addirittura cambiare il preferred parent. Queste considerazioni sono strettamente collegate ad uno scenario in cui un genitore preferito smetta di funzionare e il protocollo debba quindi calcolare una strada alternativa per mantenere la connettività tra i nodi funzionanti. Anche in questo caso, dovendo selezionare un altro preferred parent, è possibile che vi sia la necessità di modificare il proprio rank.

Un nodo può diminuire il suo rank in qualsiasi momento senza che questo provochi dei cicli nel grafo. L'unico accorgimento necessario è quello di eliminare dal parent set tutti quei nodi che, dopo l'aggiornamento, hanno rank superiore o uguale a quello attuale del nodo. Discorso diverso per quanto riguarda la volontà di aumentare il rank, operazione che può portare alla formazione di un ciclo. Infatti si consideri un nodo A che voglia incrementare il suo rank dal valore 3 al valore 5 per includere all'interno del proprio parent set i nodi con rank 3 e 4. Si consideri poi il nodo B il cui rank è 4 e il preferred parent è proprio il nodo A. Risulta quindi evidente che se A sceglie come genitore preferito il nodo B (operazione consentita in quanto il rank di A è strettamente maggiore del rank di B) si ottiene la formazione di un ciclo. Da un'esempio così semplice si evince che la problematica della formazione dei cicli nel DODAG è legata ad un incremento del valore del rank con conseguente aggiunta nel parent set di nodi che precedentemente facevano parte del sub-DODAG del nodo in questione. Un'altra situazione che si può verificare è quella *count-to-infinity*, evento probabile quando una coppia di nodi è collegata al DODAG tramite lo stesso preferred parent, ma senza altre alternative in quanto essi si trovano isolati rispetto agli altri nodi. Supponiamo che i nodi A e B siano nella situzione appena descritta e abbiano come genitore preferito C, che costituisce anche l'unico elemento del parent set. A e B vogliono aumentare il proprio rank per includere un altro genitore in quanto, per esempio, si sono accorti che C ha smesso di funzionare. A incrementerà il suo rank per includere B, ma anche B effettuerà la stessa operazione per includere A il quale, venendo avvisato che B ha aggiornato il rank al medesimo suo valore, lo aumenterà nuovamente e così anche B. Questa situazione prosegue fino al raggiungimento del massimo valore di rank possibile.

Diverse sono le possibilità che RPL mette a disposizione per evitare l'insorgere di loop e possono essere classificate secondo tre strategie: prevenzione, riconoscimento, impedimento. Un'implementazione del protocollo potrebbe sfruttare una combinazione delle tre strategie.

Loop Prevention Seguendo tale strategia non è concesso ai nodi di incrementare il proprio rank finchè il nodo radice non pubblicizza nel proprio DIO un nuovo `DODAGVersioneNumber`. Tale evento forza i nodi ad unirsi alla nuova versione del DODAG con la possibilità di scegliere un nuovo valore di rank assolutamente non vincolato a quello precedente. Inoltre, poichè la nuova versione del DODAG è originata dal sink e si propaga attraverso i preferred parent fino alle foglie, un nodo può collegarsi a qualsiasi vicino che publicizza la nuova versione senza timore che questo appartenga al proprio sub-DODAG. Tale situazione può essere sfruttata da tutti i nodi che hanno bisogno di modificare il proprio insieme di genitori ed è chiamata in quest'ottica riparazione globale del DODAG, *global repair*. I limiti di tale strategia sono evidenti. Innanzitutto non è reattiva, quindi un nodo deve aspettare di ricevere un pacchetto DIO con un nuovo numero di versione prima di poter modificare il suo rank, soluzione non accettabile soprattutto nel caso in cui la modifica è conseguente al rilevamento di un guasto perchè comporterebbe una mancanza di connettività per un tempo prolungato.

Loop Detection RPL prevede la possibilità di rilevare la presenza di un ciclo aggiungendo informazioni al pacchetto dati. In particolare viene aggiunto un campo `senderRank` posto a 0 dalla sorgente e modificato dai nodi che si trovano ad operare come router. Considerando per esempio un pacchetto diretto verso la radice, se un un nodo trova il campo `senderRank` con un valore più basso del proprio allora è probabile che vi sia un'inconsistenza nel cammino dovuta alla presenza di un ciclo. Il nodo router potrebbe decidere comunque di inoltrare il pacchetto, a meno di non rilevare nuovamente tale situazione, nel qual caso il pacchetto andrebbe scartato. Un'altra soluzione prevede di avviare i procedimenti di riparazione locale descritti in seguito.

Loop Avoidance Questa strategia differisce rispetto alla *loop prevention* poichè permette ad un nodo di incrementare in qualsiasi momento il suo rank. Al fine di impedire l'insorgere di cicli devono però essere effettuate operazioni di riparazione locale, *local repair* la cui specifica dipende dall'implementazione, ma in generale si basano su tre passi: staccare il nodo dal DODAG, segnalare che un eventuale cammino passante attraverso tale nodo non è più disponibile, riagganciare il nodo al DODAG dopo l'aggiornamento del rank. Poichè la modifica del rank ha come scopo principale l'inserimento di altri nodi all'interno del parent set e selezionare quindi un altro genitore preferito, il primo passo da compiere è proprio quello di eliminare completamente o in parte i nodi appartenenti all'attuale insieme dei genitori, operazione che di fatto consiste in un distacco dal DODAG. L'incremento

del rank da parte di un nodo impone che tutti i figli non possano più essere considerati tali senza violare la struttura del grafo e a loro volta devono quindi scegliere un diverso genitore preferito come next-hop. Affichè ciò sia possibile, il nodo deve informare i propri figli dell'incremento di rank e può farlo tramite la trasmissione di un DIO con rank pari a `INFINITE_RANK`, evento che provoca un *avvelenamento* dei cammini. Infatti la ricezione di un DIO con massimo valore impone l'eliminazione immediata di tale nodo dall'insieme dei genitori. Se questo era il genitore preferito è necessaria la selezione di un altro nodo che possa sostituirlo. Dopo la segnalazione e l'aggiornamento del rank, il nodo può scegliere il proprio parent set e preferred parent secondo le esigenze riagganciandosi così al DODAG.

6.2 Local Repair

Un meccanismo che permetta ad un nodo di cambiare il proprio genitore preferito è un aspetto fondamentale nell'implementazione di RPL in quanto permette di gestire situazioni di spegnimento di un nodo o interruzione del collegamento wireless, eventi comuni nelle reti di sensori. Dalle considerazioni precedenti la strategia più efficace risulta essere la riparazione locale, in quanto permette un gestione reattiva dei guasti. Le altre due strategie possono costiuire un supporto, ma non la soluzione al problema.

6.2.1 Avvio del meccanismo di riparazione locale

Il meccanismo di *local repair* implementato in RPL è attivato quando un nodo si accorge di non poter più contattare il suo next-hop. Questa condizione non è verificata tramite l'utilizzo di pacchetti di controllo, ma sfruttando il traffico applicazione, permettendo così di non violare il criterio loss response. Infatti, se un nodo genera del traffico, ma questo non può essere correttamente consegnato a causa di un guasto, è necessario trovare un percorso alternativo che permette all'applicazione di continuare a funzionare. D'altra parte se il nodo non genera traffico non serve controllare che il collegamento con il next-hop sia effettivamente attivo poichè tale strada non è utilizzata. Il componente CC2420 mette a disposizione un meccanismo di ritrasmissione dei pacchetti detto *packet link* che se abilitato richiede per ogni trasmissione una conferma tramite l'invio di un ACK. Se l'ACK non è ricevuto dalla sorgente, il dato viene ritrasmesso fino ad un limite massimo di 3 volte, raggiunto il quale il pacchetto viene scartato. A questo livello si inserice il meccanismo di riparazione locale, rilevando se la trasmissione ha avuto effettivamente successo o meno. Una variabile contatore viene incrementata ogni volta che viene segnalato un insuccesso nella trasmissione di un pacchetto, mentre viene decrementato

se la trasmissione ha avuto successo. Quando la variabile contatore raggiunge la soglia `FAILURE_TRASMISSION_COUNTER` viene attivato il meccanismo di riparazione locale. Una simile strategia implica la perdita di almeno `FAILURE_TRASMISSION_COUNTER` pacchetti applicazione prima di iniziare la ricerca di un percorso alternativo. Tuttavia questo è un buon compromesso rispetto ad una strategia di riparazione globale che potrebbe comportare un numero maggiore di informazioni perse, senza contare il traffico di controllo generato per la ristrutturazione di tutto il DODAG. Il parametro `FAILURE_TRASMISSION_COUNTER` non deve essere impostato ad un valore troppo piccolo poichè il fallimento della trasmissione potrebbe essere dovuto semplicemente alla momentanea presenza di un ostacolo nel collegamento, oppure ad una temporanea congestione del nodo e non ad una effettivo guasto. D'altra parte un valore troppo elevato comporterebbe una perdita eccessiva di pacchetti dati e quindi l'esatto valore dipende dalle esigenze dell'applicazione. Un buon compromesso, utilizzato nell'implementazione, è tre. Una strategia di riparazione locale come quella appena descritta richiede un collegamento tra il livello networking del routing e il livello sottostante datalink. Questa condizione non è ideale, in quanto limita la portabilità dell'implementazione da un hardware all'altro. Tuttavia le modifiche sono minime, e per quanto riguarda il nesC consistono semplicemente nella creazione di un evento segnalato dal modulo che si occupa della trasmissione e ricevuto dal modulo di routing.

Se l'origine dell'avvio della riparazione locale è l'impossibilità di contattare il next-hop, l'operazione deve essere effettuata anche dai nodi che ricevono un DIO con rank pari a `INFINITE_RANK`, poichè questo evento li costringe a eliminare il nodo dal proprio parent set e, se questo coincide con il preferred parent, è evidente la necessità della ricerca di un next-hop alternativo.

6.2.2 Ricerca del percorso alternativo

Si consideri innanzitutto le operazione effettuate dal nodo che rileva l'impossibilità di comunicare con il proprio genitore preferito. L'idea della ricerca di un percorso alternativo sfrutta l'insieme dei genitori e quello dei vicini. Si ricorda infatti che la rete di sensori per cui è sviluppata questa implementazione di RPL è statica, senza cambiamenti significativi di topologia, quindi le informazioni presenti nelle tabelle sono da considerarsi attendibili nel tempo, a meno naturalmente di guasti.
Quando un nodo rileva di non poter comunicare con il suo next-hop, la prima operazione da compiere è la ricerca di un genitore alternativo nella tabella dei genitori. Questi nodi hanno rank strettamente inferiore e valore di RSSI maggiore della soglia `RSSI_THRESHOLD`. Se il parent set contiene almeno un nodo oltre al preferred parent, questo ne assume il ruolo diventando next-hop. Il valore di rank del nodo non è cambiato e quindi non

è necessario affettuare alcuna operazione aggiuntiva. Si noti come questa operazione è completamente a carico del nodo e non necessita di ulteriore traffico di controllo rispetto a quello normale di RPL.

Se non esiste un genitore alternativo all'interno del parent set, la ricerca viene fatta all'interno dell'insieme dei nodi vicini. Ricordiamo che tale insieme è costituito da tutti nodi con qualsiasi valore di rssi e rank da cui è stato ricevuto almeno un pacchetto DIO. La ricerca di un genitore alternativo in tale insieme è condotta cercando un nodo con uguale valore di rank e con RSSI maggiore della soglia `RSSI_THRESHOLD` selezionando quindi un nodo allo stesso livello del DODAG. L'elezione di tale nodo a genitore impone naturalmente l'aumento del rank ad un valore superiore rispetto a quello precedente ed è quindi necessario avvisare gli eventuali figli di tale cambiamento inviando un DIO con rank pari a `INFINITE_RANK`. Coloro che ricevono tale DIO dal genitore preferito affettuano le medesime operazione descritte in quanto devono cercare un percorso alternativo. Nel caso in cui il nodo sorgente del DIO faccia solo parte del parent set, questo viene semplicemente eliminato. Dopo aver compiuto l'operazione di sganciamento il nodo può selezionare il vicino come genitore preferito e resettare il proprio trickle timer per avvisare del cambiamento. L'azione di riparazione locale appena descritta richiede per ciascun nodo che la intraprende l'invio di un DIO con rank infinito oltre al reset del trickle timer. Il traffico di controllo è quindi ridotto al minimo possibile come specificato nel documento draft. Se anche questo controllo nella tabella dei vicini fallisce e non viene trovato nessun altro candidato possibile il nodo non può fare altro che cancellare tutti gli elementi del parent e del neighbor set e aspettare la ricezione di un DIO da un nodo per agganciarsi, evento che può essere indotto tramite l'invio di un DIS in multicast. Infatti non è possibile affidarsi a nodi con rank superiore al proprio, in quanto essi potrebbero far parte del proprio sub-DODAG e quindi, a causa dell'invio del DIO con rank infinito, aver modificato la proprio posizione nel grafo rendendo l'informazione presente nella tabella non corretta.

6.2.3 Analisi

Nel seguito viene analizzato il comportamento della soluzione in alcune situazioni di esempio verificate sul testbed presentando poi alcuni risultati ottenuti. Nelle figure 6.1, 6.2, 6.3, il colore rosso degli archi indica il genitore preferito e next-hop, il colore verde indica un nodo presente nel parent set mentre il colore nero indica un nodo presente nell'insieme dei vicini. Nodi allo stesso livello hanno lo stesso valore di rank. La parte sinistra della figura rappresenta la situazione originale, mentre la parte destra la situazione dopo la riparazione locale, in cui sono evidenziati solamente i collegamenti con i preferred parent. Si consideri sempre un guasto del nodo **A**.

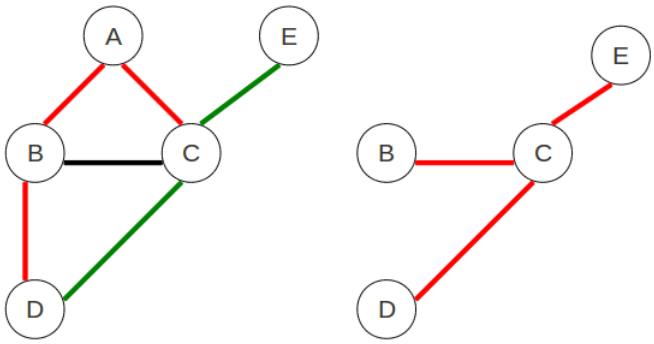


Fig. 6.1: Scenario 1

Nello scenario di figura 6.1 il nodo **A** ha due figli **B** e **C**. Quando il nodo **C** rileva il fallimento del collegamento con **A**, controlla nel proprio parent set trovando il nodo **E** che quindi assume il nuovo ruolo di genitore preferito. Il nodo **B** invece non trova alcun nodo alternativo nel proprio insieme dei genitori, quindi seleziona il vicino **C** come genitore preferito. Questa operazione implica un incremento del valore di rank e quindi l'invio di un DIO con `INFINITE_RANK`. Alla ricezione di tale DIO il figlio **D** elimina il nodo **B** dal proprio insieme dei genitori in cui però è trovato **C** che assume quindi il ruolo di nuovo genitore preferito. Il nuovo percorso alternativo all'originale attraverso **A** è quindi stato costituito con l'invio di un pacchetto DIO da parte di **B**, oltre al reset del trickle timer.

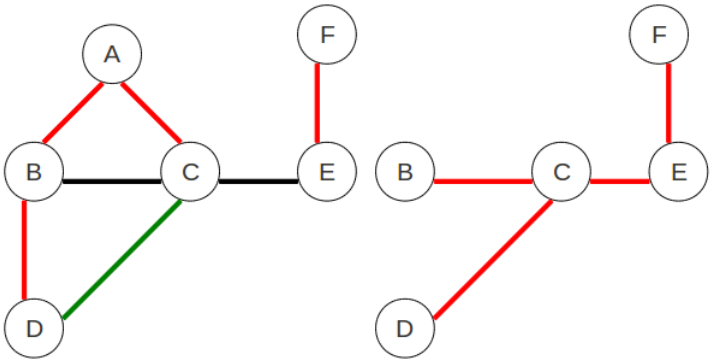


Fig. 6.2: Scenario 2

Nello scenario di figura 6.2, in cui il guasto si verifica sempre nel nodo **A**, sono possibili due diversi casi di creazione del cammino alternativo, a seconda di chi tra il nodo **B** e **C** si accorge prima del guasto.

1. Il nodo **C** si accorge prima di **B** del guasto di **A**. Per rendere più chiara la successiva spiegazione si assegnino il valore di rank 2 ai nodi **A** e **F**, 3 ai nodi **B**, **C**, **E** e 4 al nodo **D**. **C**, non avendo elementi

nel parent set oltre ad **A**, seleziona **E** come nuovo preferred parent, invia un DIO con rank infinito, imposta il proprio rank a 4 e avvia il reset del trickle timer. Poichè **C** non ha figli, l'invio del DIO con rank infinito provoca la sua eliminazione dal parent set del nodo **D**. D'altra parte però, il reset del trickle timer e l'invio di DIO con rank pari a 4 da parte di **C**, provoca l'aggiornamento nella tabella dei vicini di **B** e un nuovo inserimento nella tabella di **D**. A questo punto **B** ha nel neighbour set **C** con rank 4 e **D** ha il parent set vuoto e **C** nell'insieme dei vicini sempre con rank 4. **B** si accorge del fallimento di **A**, ma non avendo nodi a cui agganciarsi invia il DIO con rank infinito e rimane in ascolto di un pacchetto DIO valido da parte di un nodo a cui agganciarsi. **D**, ricevuto da **B** il DIO con rank infinito, seleziona il nodo **C** come genitore preferito, aggiorna il suo rank al valore 5, invia a sua volta un DIO con rank infinito, e procede con il reset del trickle timer. **B**, sentendo i DIO di **D** e **C**, selezionerà quest'ultimo cone preferred parent avendo rank inferiore. Il cammino alternativo risulta quindi formato.

2. Il nodo **B** si accorge prima di **C** del guasto di **A** e si assumano gli stessi valori iniziali di rank del punto precedente. In questa situazione, seguendo un procedimento analogo a quello dello scenario 1, si arriva ad una configurazione in cui i nodi **B** e **D** hanno **C** come genitore preferito e rank pari a 4. Quando **C** si accorge del guasto di **A**, selezionerà il nodo E come genitore preferito, provocando con l'invio di un DIO con rank infinito, il distacco di **B** e **D**. Se **B** e **D** sono a vicenda nei rispettivi neighbor set, si crea la situazione descritta nello scenario 3. Altrimenti, essi selezionano **C** come genitore preferito con rank 5 alla ricezione di DIO da parte di **C**.

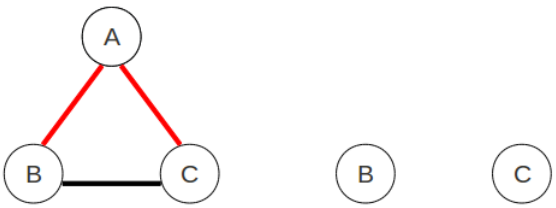
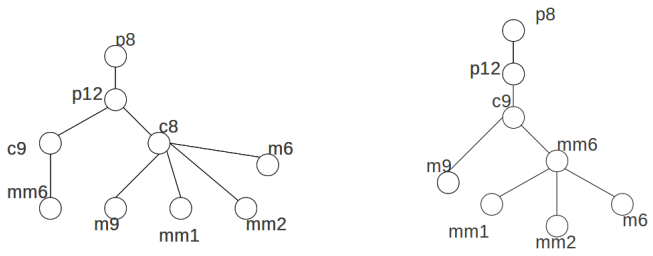


Fig. 6.3: Scenario 3

Lo scenario ritratto in figura 6.3 raffigura dei nodi collegati al medesimo genitore preferito e isolati da tutti gli altri nodi. Questa situazione potrebbe dare origine al problema del *count to infinity*, ma la soluzione descritta in precedenza ne evita l'insorgere. Infatti si ipotizzi che **B** rilevi il guasto di **A** selezionando **C** come genitore preferito. A sua volta **C** rilevando il

guasto di **A**, cerca di collegarsi a **B** inviando un DIO con rank infinito con conseguente distacco di **B** da **C**. A questo punto **B**, non avendo alcun altro elemento nelle proprie tabelle invierà un ulteriore DIO con rank infinito provocando il distacco di **C**. A questo punto entrambi i nodi rimangono in una situazione in cui aspettano di ricevere un pacchetto DIO da un nodo valido a cui agganciarsi, senza incorrere in una situazione di tipo *count to infinity*. Naturalmente la stessa cosa avviene scambiando i ruoli tra **B** e **C**.

La figura 6.4 rappresenta la porzione di un DODAG formato da 20 nodi, in particolare il sottoalbero relativo al nodo **c8**. Nel test viene spento il nodo **c8** e tutti i nodi del sottoalbero, seguendo le procedure descritte precedentemente, riescono a trovare un genitore alternativo e quindi ricostruire un DODAG consistente senza la presenza di **c8** come evidenziato dalla parte destra della figura stessa. In questo caso il nodo **m9** trova un genitore alternativo nella tabella dei genitori mentre i nodi **mm1**, **mm2** e **m6** selezionano il nodo **mm6** trovato nella tabella dei vicini effettuando quindi un incremento del proprio rank con conseguente reset del Trickle Timer. Un dato da valutare è l'incremento di traffico di controllo dovuto alla procedura di riparazione locale, rappresentato nel grafico di figura 6.5. I parametri del trickle timer sono `Imin=10` e `K=5`.



(a) Prima della rottura di **c8** (b) Dopo della rottura di **c8**

Fig. 6.4: Esempio di riparazione locale in un DODAG con guasto al nodo **c8**

Il guasto si verifica dopo circa 100 secondi dall'inizio del test e coinvolge i tre nodi prima citati. L'incremento è di circa 30 DIO generati soprattutto a causa del reset del trickle timer. Quindi mediamente ogni nodo coinvolto nel guasto invia 10 pacchetti DIO in più rispetto alla situazione senza anomalie.

La figura 6.6 raffigura un altro scenario evidenziando la porzione di un DODAG, formato da 27 nodi, riferita al sottoalbero del nodo **r21**. Anche in questo caso è raffigurata la situazione prima e dopo il guasto occorso al nodo **r21** confermando ancora la correttezza delle operazioni di riparazione locale che portano alla formazione di un nuovo DODAG consistente. Si nota che in questo caso i nodi **e11**, **e4**, **e7** selezionano il nuovo preferred parent

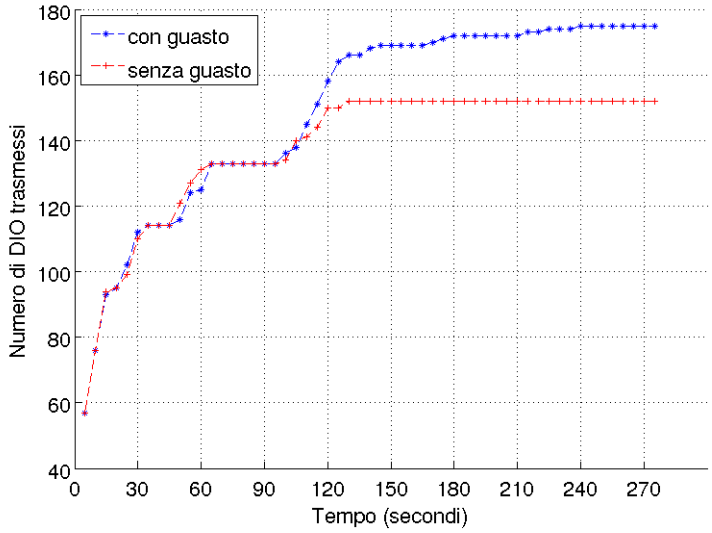
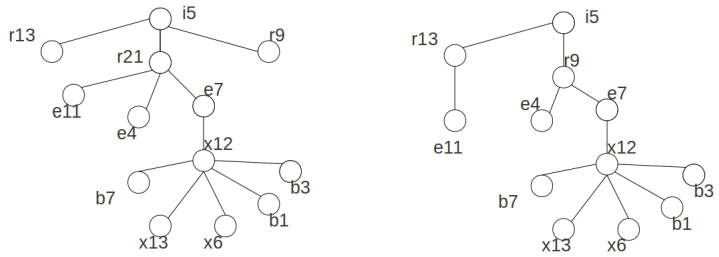


Fig. 6.5: Incremento del traffico di controllo in seguito all'operazione di local repair

dalla tabella dei genitori senza quindi la necessità di staccare i rispettivi sottoalberi, confermato dalla situazione inalterata prima e dopo il guasto del sottoalbero di **e7**. In questo scenario, ciascun nodo invia al sink un pacchetto applicazione ogni 10 secondi. La tabella 6.1 mostra il numero di pacchetti applicazione persi a causa del guasto per i nodi facenti parte del sottoalbero di **r21**.



(a) Prima della rottura di **r21** (b) Dopo della rottura di **r21**

Fig. 6.6: Esempio di riparazione locale in un DODAG con guasto al nodo **r21**

Nodo	e11	e4	e7	x12	b7	x13	x6	b1	b3
Pacchetti persi	2	3	1	0	0	1	0	0	1

Tabella 6.1: Numero di pacchetti persi da parte dei nodi nel sottoalbero di **r21**

Ricordando che il limite di attivazione della riparazione locale è tre

trasmissioni fallite, si nota come i nodi figli di **r21** senza discendenti hanno una perdita di pacchetti che raggiunge proprio tre nel caso di **e4** oppure due per **e11** in cui vi è anche la perdita di un DAO. Discorso diverso per il nodo **e7**, il quale inoltrando i pacchetti provenienti dal suo sottoalbero si accorge prima del guasto permettendo a lui stesso e ai discendenti di perdere un numero inferiore di dati applicativi. Infatti **e7** si accorge del guasto perdendo un suo pacchetto, uno inoltrato da **x13** e uno da **b3** per un totale di tre. Gli altri nodi non perdono alcun pacchetto applicazione evidenziando quindi una buona reattività della soluzione.

6.3 Considerazione finali

L'articolo [51] propone un'analisi approfondita del comportamento del protocollo RPL in situazioni di riparazione locale. L'implementazione della procedura è leggermente differente rispetto a quella appena descritta. L'analisi è tuttavia compiuta tramite l'utilizzo di un ambiente simulato (nello specifico NS2), che permette uno studio più attento e dettagliato del comportamento del protocollo in questi scenari. Infatti risulta relativamente complesso realizzare un'estesa campagna di test di questa tipologia sul testbed del dipartimento. Tuttavia, uno sviluppo futuro di questo lavoro di tesi è proprio la migrazione dell'implementazione di RPL in un ambiente simulato. Una approfondita analisi della tecnica di gestione dei guasti appena descritta è quindi lasciata per il momento in sospeso.

È possibile tuttavia evidenziare subito alcuni limiti della strategia. La più evidente è il mancato aggiornamento delle cammini dal nodo root verso i singoli nodi. Infatti, se da un lato l'evento di reset del trickle timer potrebbe coincidere con un analogo reset del DAO timer permettendo così l'aggiornamento dei cammini, nel caso in cui il nuovo genitore preferito è selezionato nell'insieme dei parent set, non è prevista la possibilità di informare il sub-DODAG del cambiamento. Quindi l'aggiornamento dei cammini è lasciato all'invio di un pacchetto DAO secondo le modalità standard del protocollo, evento che potrebbe essere molto prolungato nel tempo. La soluzione quindi non è indicata per applicazioni che richiedono traffico P2P e P2MP. Un ulteriore limite, come già ricordato, riguarda l'assunzione di una rete statica in cui le informazioni contenute nel parent e neighbor set non diventano obsolete con il tempo.

CAPITOLO 7

Mobilità

Indice

Questo capitolo si occupa di valutare il comportamento del protocollo RPL in scenari con la presenza di uno o più nodi mobili. Il supporto alla mobilità non è un criterio di progetto del protocollo, e il fatto di essere proattivo a vettori di distanza lo dimostra. Questa tipologia di routing, calcolando e mantenendo costantemente i cammini per raggiungere tutte le destinazioni della rete, necessita di un scambio di pacchetti per la riconfigurazione della rete non appena un nodo cambia la sua posizione portando ad un traffico di controllo chiaramente eccessivo e inefficiente. Protocolli di routing reattivi on-demand sono più adatti per reti mobili, basti pensare per esempio a AODV. RPL tuttavia si propone di diventare lo standard dei protocolli di routing per LLN e per tale motivo è interessante valutarne con precisione i limiti e l'impatto delle modifiche necessarie per un suo utilizzo in scenari mobili. Questo tipo di analisi è abbastanza originale, in quanto allo stato attuale non esistono studi specifici. Nella mailing list della gruppo di sviluppatori di RPL[1] sono pochissime le discussioni in questo ambito e senza conclusioni significative. Il lavoro [52] valuta una strategia che utilizzi RPL in presenza di un sink mobile con il fine di aumentare il tempo di vita

[1] http://www.ietf.org/mail-archive/web/roll/current/maillist.html

della rete. La strategia sfrutta la possibilità da parte del sink di generare diverse versioni del DAG e ad ogni nuova versione viene determinata la foglia più lontana dal root, dove la lontananza è una funzione del numero di hop, dell'energia residua del nodo e del numero di vicini. Il sink quindi si muove verso tali foglie, con lo scopo principale di aumentare il tempo di vita della rete. Questo lavoro non valuta direttamente RPL in scenari mobili, tuttavia può essere considerata un primo tentativo di analisi in questa direzione.

7.1 Scenario e vincoli

Lo scenario in analisi prevede innanzitutto la presenza di un unico nodo mobile che cambi la sua posizione nella rete in qualsiasi momento senza alcun vincolo stabilito a priori. Considerazioni su un maggiore grado di mobilità sono presentati alla fine del capitolo. Il nodo mobile si può spostare in un punto qualsiasi della rete e rimanere fermo per un periodo di tempo arbitrario per poi muoversi nuovamente in un altro punto. Anche se si considera la presenza di un unico nodo mobile in un determinato istante, la sua identità non è stabilita a priori quindi potenzialmente qualsiasi nodo sensore ha la possibilità di muoversi, senza che tale ruolo sia ricoperto sempre da uno in particolare. Infine la possibilità di muoversi è prevista solo per i nodi sensori, non per il sink che è considerato sempre fermo.

Nello sviluppo di una versione RPL per lo scenario appena descritto, si sono tenuti in considerazione i seguenti vincoli, non considerando quindi altre situazioni:

- Limitare il più possibile le modifiche al protocollo come descritto nei capitoli precedenti, senza aggiungere campi al pacchetto o addirittura nuovi messaggi di controllo diversi da DIO, DAO e DIS con relative opzioni.

- Il protocollo deve essere in grado di operare senza richiedere nuove funzionalità ai livelli di rete sottostanti, data-link e fisico.

- Il protocollo con supporto della mobilità deve poter operare indifferentemente nei casi in cui la rete presenti effettivamente nodi mobili oppure sia completamente statica, senza la necessità di operare alcuna modifica. Inoltre non vi è alcuna definizione a priori dei nodi mobili.

- I nodi non hanno informazioni della loro posizione e non sono dotati di strumenti in grado di calcolarla. Le uniche informazioni sui collegamenti sono forniti dal componente CC2420, quindi RSSI e LQI.

7.2 Aspetti critici

Nello sviluppo di un adattamento di RPL ad uno scenario mobile sono emersi alcuni aspetti critici che devono essere tenuti in considerazione.

Stabilità del DODAG Poichè il nodo mobile può essere qualsiasi elemento della rete, esiste la possibilità che esso ricopra il ruolo di genitore preferito. Quando esso si sposta, i figli devono essere in grado di identificare un nuovo preferred parent mantenendo cosi la connettività nel DODAG formato dai nodi stabili.

Connettività del nodo mobile Il nodo mobile deve essere in grado di comunicare i propri dati al sink, e in secondo luogo deve poter essere raggiunto dagli altri nodi della rete.

Metrica di mobilità Poichè il nodo mobile non è identificato a priori, deve esistere la possibilità di identificarlo, sia dalla rete se necessario, ma soprattutto il nodo stesso deve rendersi conto di essere mobile. Per tale motivo deve essere prevista un metrica di mobilità che permetta di stabilire se un nodo è mobile oppure o no.

7.2.1 Stabilità del DODAG

Il mantenimento della connettività del DODAG nel caso di spostamento del nodo mobile può essere considerato un caso particolare della situazione analizzata nel capitolo 6 dedicato alla gestione dei guasti. È sufficiente quindi che il nodo mobile invii un DIO con rank infinito per innescare il meccanismo di riparazione locale creando tra i nodi stabili percorsi alternativi. Questa situazione quindi non prevede modifiche al protocollo RPL, tuttavia è fondamentale che un nodo sia in grado di identificare la propria condizione di mobilità per sapere quando inviare il DIO con rank infinito, problematica affrontata nella sezione 7.2.3.

7.2.2 Connettività del nodo mobile

Il protocollo deve consentire al nodo mobile di poter comunicare i propri dati al sink o a qualsiasi altro nodo della rete poichè evidentemente questa mancanza renderebbe il nodo mobile praticamente inutile. Altra caratteristica utile è la capacità da parte degli elementi della rete di comunicare con il nodo mobile, anche se essa è meno importante rispetto alla precendente in quanto la maggior parte delle applicazioni richiedono un traffico verso il sink piuttosto che un traffico P2P. RPL richiede l'agganciamento al DODAG e l'identificazione di un genitore preferito affinchè sia possibile determinare un cammino tra i nodi della rete. Questa situazione è evidentemente un limite in uno scenario di mobilità, in quanto il nodo mobile dovrebbe essere sempre

a conoscenza di un nodo facente parte del DODAG a cui inoltrare i propri dati, in evidente contrasto con la condizione di mobilità stessa.

La situazione ideale è quella in cui il nodo mobile non necessiti di identificare un preferred parent per inoltrare i propri pacchetti, ma sfrutti invece un invio in broadcast dei dati. Il problema diventa quindi la gestione da parte dei nodi agganciati al DODAG dei pacchetti broadcast tramite un meccanismo che elegga uno o più nodi designati all'inoltro verso il sink del dato. Infatti una gestione banale tale per cui ciascun nodo che sente un pacchetto broadcast lo inoltri al proprio next-hop non è percorribile in quanto un numero potenzialmente molto elevato di nodi invierebbe lo stesso pacchetto generando un notevole volume di traffico ridondante. Una rete poco densa potrebbe limitare questo problema, ma la soluzione resta non efficiente rendendo quindi necessario un meccanismo di elezione. Una strategia potrebbe sfruttare l'uso di timer nel seguente modo: quando un nodo riceve un pacchetto broadcast avvia un timer casuale o proporzionale a una qualche metrica allo scadere del quale inoltra, sempre in broadcast, il dato a meno che in tale intervallo non riceva il medesimo pacchetto, evidentemente inoltrato da un altro nodo, e quindi procede a scartarlo senza inviarlo. Il procedimento continua fino a quando il dato non raggiunge il sink. La situazione appena descritta presenta un notevole inconveniente: il dato viene inoltrato verso tutte le direzioni del DODAG, non solo nella direzione del sink facendo rimbalzare il pacchetto nella rete per un tempo indefinito. Questo problema è derivato fondamentalmente dal fatto che i nodi, compreso quello mobile, non hanno alcuna consapevolezza della propria posizione geografica rispetto al sink. Se questa informazione fosse disponibile, allora il procedimento limiterebbe la partecipazione all'inoltro ai soli nodi più vicini al sink rispetto alla posizione del nodo mobile, riducendo o addirittura eliminando completamente la ridondanza. Alcuni lavori che sfruttano questa tecnica di inoltro sono [53], [54] e [55]. Tuttavia i vincoli esposti nella sezione 7.1 rendono anche questa strategia non perseguibile vista la totale mancanza di informazioni geografiche a disposizione dei nodi. L'articolo [56] propone di sfruttare unicamente il valore di RSSI per calcolare la posizione dei nodi relativa al sink. Tuttavia si suppone che il sink sia dodato di una potenza trasmissiva tale da raggiungere tutti i nodi della rete, condizione fortemente limitante e comunque non disponibile nella rete testbed del DEI.

Dall'analisi appena esposta emerge l'impossibilità di sfruttare una diffusione in broadcast dei dati da parte del nodo mobile che quindi deve necessariamente agganciarsi al DODAG selezionando un preferred parent a cui fare l'inoltro dei propri dati. Questo costituisce un limite notevole all'utilizzo di RPL in scenari di mobilità.

7.2.3 Metrica di mobilità

Come detto in precedenza la non identificazione a priori del nodo mobile necessita della presenza di una metrica di mobilità la cui valutazione permetta al nodo di stabilire la sua condizione di movimento o di staticità. Poichè non è prevista la presenza di accelerometri o GPS montati sui nodi, la metrica si basa esclusivamente su informazioni relative ad uno scambio di pacchetti con altri nodi, in particolare queste si riducono ad un elenco di nodi vicini e grandezze come la potenza ricevuta. Questo è un ulteriore limite in quanto presuppone l'esistenza di un livello di traffico nella rete sufficiente a rendere effettivamente utilizzabile la metrica. Vi sono diversi lavori dedicati al problema sotto questi vincoli e nel seguito ne viene data una breve descrizione insieme una valutazione di utilizzo rispetto scenario e in analisi.

L'articolo [57] utilizza una metrica basata sulla potenza del segnale ricevuto suddivisa in due fasi che permette di determinare una alta o bassa mobilità, informazione poi tramessa nel pacchetto HELLO scambiato tra i nodi della rete. Si suppone inoltre di essere in grado di calcolare la distanza relativa tra due nodi tramite la potenza del segnale ricevuto che in prima approssimazione è inversamente proporzionale al quadrato della distanza. La prima fase prevede di determinare la differenza di distanza tra due successive trasmissioni del pacchetto HELLO di un nodo vicino. Un nodo può avere diversi vicini, e sono presi in considerazione solo quelli che publicizzano un basso grado di mobilità. Il valore di differenza di distanza è poi comparato ad una particolare soglia C_m. Si supponga che un nodo abbia ricevuto n trasmissioni di cui solo k da nodi con bassa mobilità e a loro volta solo i tali per cui la differenza di distanza è risultata superiore alla soglia C_m. La fase due prevede di verificare la condizione $(i/k) > C_v$ dove $0 < C_v < 1$ rappresenta una soglia di maggioranza. Se la condizione è verificata il nodo si identifica come altamente mobile, altrimenti come poco mobile. Da questa descrizione, le informazioni necessarie ad un protocollo che si basi su questa metrica sono una tabella dei vicini e la potenza del segnale ricevuto, entrambe presenti nello scenario di utilizzo di RPL. Per quanto riguarda il messaggio HELLO, la sua funzione è quella di permettere il calcolo della distanza, cosa che può essere fatta da qualsiasi pacchetto: DIO, DAO o anche applicazione. Diverso il discorso sulla trasmissione dello stato di mobilità che richiederebbe una significativa modifica del pacchetto, oltre alla necessità di inviarlo molto frequentemente. Nel nostro scenario tuttavia si considera la presenza di un solo nodo mobile rendendo questa informazione superflua. Gli articoli [58] e [59] utilizzano un approccio simile basandosi entrambi sulla differenza di potenza tra due trasmissioni consecutive dei nodi vicini. In particolare il primo lavoro considera il rapporto in scala logarimica tra la potenza dell'ultima trasmissione rispetto alla prece-

dente per tutti i nodi vicini. Successivamente viene calcolata la varianza rispetto a zero di tutte le misure e sulla base di questo dato viene determinata la mobilità del nodo: limitata per bassi valori di varianza, maggiore per valori più alti. Il secondo lavoro sfrutta invece una formula in cui la differenza di distanza relativa ad un vicino è pesata secondo un coefficiente che tiene in considerazione l'indice di mobilità del nodo stesso, informazione trasmessa nel pacchetto di controllo. L'articolo [61] è un esempio di protocollo per reti di sensori mobili che sfrutta la metrica di mobilità definita in [59].

L'articolo [60] propone una metrica di mobilità basata sulla variazione degli elementi della tabella dei vicini. In particolare il traffico di controllo è costituito da un pacchetto HELLO che i nodi inviano periodicamente. La tabella dei vicini è costituita dall'indirizzo sorgente del pacchetto HELLO, un contatore e un timer il cui intervallo è maggiore di quello di invio del messaggio HELLO. Per ciascun HELLO ricevuto i nodi inseriscono o aggiornano l'entry della tabella dei vicini relativa all'indirizzo sorgente impostando il counter a 0 e avviando o resettando il timer. Se questo scade, viene incrementato il contatore e se questo supera una determinata soglia, allora quel particolare nodo viene eliminato dalla tabella. A intervalli regolari vengono controllati i cambiamenti nella tabella e se questi sono superiori ad una soglia, il nodo viene considerato mobile. Anche l'articolo [62] propone un meccanismo simile basato sull'invio periodico di un messaggio HELLO che permetta di valutare i cambiamenti nella tabella dei vicini. Applicare l'approccio descritto al nostro scenario presuppone di avere un messaggio di controllo, per esempio il DIO, inviato in maniera periodica, eliminando quindi l'uso del trickle timer. Si è più volte sottolineata l'importanza di ridurre al minimo il traffico di controllo dei protocolli operanti su reti di sensori, quindi un invio periodico di messaggi è una condizione non accettabile rendendo l'approccio poco appropriato.

La valutazione della metrica di mobilità può essere fatta a intervalli periodici, oppure ad ogni evento significativo, come la ricezione di un nuovo messaggio. La prima strategia permettere di svolgere le operazioni necessarie al calcolo della metrica solo a precisi istanti di tempo, mentre la seconda strategia richiede il calcolo, potenzialmente oneroso, con una frequenza notevolmente maggiore. D'altra parte però l'utilizzo di un timer richiede di valutare con molta attenzione la grandezza dell'intervallo al fine di rendere affidabile la metrica stessa.

7.3 Implementazione

In questa sezione si ripercorrono le considerazione teoriche svolte nella sezione precedente evidenziando le strategia implementative adottate.

7.3.1 Considerazioni sul RSSI

L'aspetto più importante da definire nell'adattamento di RPL in uno scenario mobile è la specifica di una precisa metrica di mobilità. Dalla sezione 7.2.3 è emerso che l'approccio migliore si basa sulla differenza di distanza di tra due ricezioni consecutive ricavata dalla potenza del segnale ottenuto. La grandezza a disposizione per tale misurazione è il valore di RSSI. La relazione tra RSSI e distanza presuppone la definizione di un modello di propagazione del segnale che tenga conto della dissipazione della potenza emessa dal trasmettitore e degli effetti di attenuazione introdotti dal canale (*path loss*). A questo si aggiunge la presenza di ostacoli tra il trasmettitore e il ricevitore che producono un disturbo del segnale (*shadowing*) dovuto ad assorbimento, riflessione, diffusione e diffrazione. Esistono molti modelli di diversa complessità a seconda delle componenti di disturbo che vengono prese in considerazione e una dettagliata analisi può essere trovata in [63]. Per gli scopi introduttivi del progetto ci si è affidati ad un modello di propagazione semplificato, *simplified path loss model*, che non tiene in considerazione gli effetti di disturbo dovuto ad ostacoli; ma, per quanto semplice, cattura l'essenza della propagazione del segnale e costiuisce la base di molti modelli più complessi che in ogni caso sono una approssimazione del canale reale. Il modello si basa sulla seguente formula:

$$P_r = P_t K \left[\frac{d_0}{d}\right]^{\gamma} \tag{7.1}$$

con attenuazione dB

$$P_r \text{ dBm} = P_t \text{ dBm} + K \text{ dB} - 10\gamma \log_{10} \left[\frac{d}{d_0}\right]^{\gamma} \tag{7.2}$$

dove P_r è la potenza ricevuta, P_t la potenza trasmessa, K una costante che dipende dalle caratteristiche dell'antenna e dall'attenuazione media del canale, d_0 è la distanza riferita al campo lontano dell'antezza e infine γ è un fattore esponenziale.

Il grafico di figura 7.1 mostra il valore di P_r (RSSI) dato dal componente CC2420 in relazione alla distanza in condizioni ideali di propagazione senza ostacoli e disturbi secondo l'andamento (7.1). Considerando $d_0 = 0.1$ metri, il valore di K è calcolato secondo l'espressione $K = 20 \log_{10} \frac{\lambda}{4\pi d_0}$, valida per la propagazione in spazio libero da ostacoli di diffrazione e riflessione, come l'aria. In accordo con questa ipotesi di idealità è posto $\gamma = 2$ e la potenza trasmessa è quella massima di 0 dBm. Le linee verticali del grafico mostrano

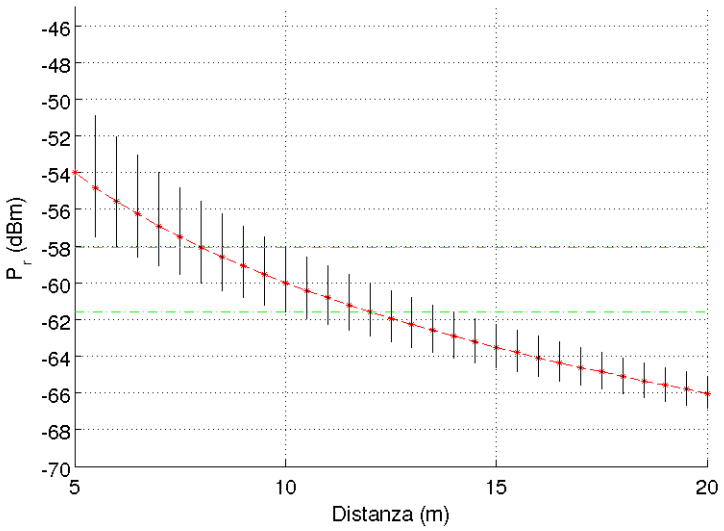


Fig. 7.1: P_r vs. distanza (passo 0.5 metri)

il valore di differenza di P_r che indichi uno spostamento di due metri dalla distanza considerata verso e dal ricevitore, rispettivamente parte superiore e inferiore della curva rossa. Prendendo per esempio il valore di RSSI -60 dBm relativo ad una distanza di 10 metri, le rette in verde nel grafico permettono di verificare che la barra superiore intercetta la linea rossa nel valore 8 metri, mentre quella inferiore a 12 metri. In condizioni di spazio libero quindi, dato il valore di P_r uguale a -60 dBm, una differenza pari a $(-58-(-60)) = 2$ dBm indica uno spostamento di 2 metri vero la sorgente da una distanza di 10 metri, mentre una differenza di $(-61.6 - (-60)) = -1.6$ dBm indica uno spostamento dalla sorgente di 2 metri. Un altro importante elemento da notare nel grafico riguarda l'ampiezza delle linee verticali che, a causa dell'andamento logaritmico della curva, hanno un'ampiezza minore al diminuire del valore di RSSI. Questo vuol dire per piccole potenze ricevute, una piccola differenza è sufficiente a indicare uno spostamento notevolmente maggiore rispetto alla stessa differenza considerata per valori di RSSI più alti.

Anche se l'analisi svolta finora è stata basata su un modello notevolmente semplificato di propagazione la sua validità generale è confermata anche nell'ambiente del testbed dove sono presenti numerosi fattori di disturbo, basti solo pensare ai muri stessi dell'edificio.

7.3.2 Metrica di mobilità

La metrica di mobilità scelta segue la linea guida del lavoro [57], basandosi quindi sulla valutazione della differenza di distanza rispetto a due

trasmissioni dei nodi vicini. Tuttavia la particolare condizione della presenza di un unico nodo mobile permette di aggiungere un controllo della tabella dei vicini mirato all'identificazioni di nuovi nodi apparsi nel raggio di copertura, evento che potrebbe indicare uno spostamento verso una regione diversa della rete. La metrica di mobilità è controllata tramite un timer periodico di intervallo `MOBILITY_TIMER`, che costituisce un paramentro critico dell'algoritmo. Se la verifica della metrica identifica il nodo come mobile, vengono eliminati tutti i valori all'interno della tabella dei vicini e dei genitori, avviate le procedure di stabilità del DODAG e connettività del nodo mobile riportati nelle sezioni 7.3.4 e 7.3.3 e impostata a TRUE la variabile booleana `MOBILITY_STATE` che assume invece il valore FALSE all'inizializzazione e nel caso di nodo stabile. Sia che il nodo sia mobile o meno, tutte le variabili elencate nel seguito e utilizzate per il calcolo della metrica sono impostate ai rispettivi valori di default ogni volta che `MOBILITY_TIMER` scade. Un'ulteriore condizione che potrebbe indicare la presenza di mobilità del nodo è la perdita di connettività con il proprio genitore preferito. Pertanto, al rilevamento di tale situazione, viene controllata la metrica di mobilità, e in caso di valutazione positiva avviate le stesse procedure appena descritte. In caso invece di valutazione negativa prende luogo l'operazione di riparazione locale. Poichè la metrica è basata sulla ricezione di pacchetti da parte di altri nodi, nel caso in cui non vi sia traffico nella rete il risultato sarà sempre negativo. L'unico indizio di mobilità quindi è dovuto alla perdita del collegamento con il genitore e in questo caso, se il traffico rilevato tramite un contatore è inferiore alla soglia C_p, si considera il nodo mobile e non viene attivato il meccanismo di riparazione locale. Questa scelta è dovuta al fatto che nel caso in cui la perdita di comunicazione sia dovuta alla mobilità la riparazione locale comporterebbe una sicura perdita di pacchetti dati a causa della non validità delle informazioni contenute nella tabelle. Il traffico di controllo dalla condizione di mobilità è comunque limitato e non incide sulla rete visto il basso livello di traffico applicativo.

7.3.2.1 Metrica basata sulla differenza di RSSI

Rispetto all'articolo [57] più volte citato, la metrica scelta si basa su due qualsiasi trasmissioni originate da un particolare nodo vicino. Non vengono quindi considerati solo i pacchetti DIO, ma, grazie alla possibilità offerta dal componente CC2420 di processare pacchetti la cui destinazione è diversa dal nodo in questione, anche i pacchetti applicazione contribuiscono al calcolo della differenza di RSSI. Questo approccio permette di sfruttare la metrica anche in una fase avanzata del protocollo dove l'invio dei messaggi di controllo è notevolmente dilazionato nel tempo.

Dato il valore di RSSI fornito dal CC2420, la distanza corrispondente dipende dal modello di propagazione utilizzato. Tuttavia questo calcolo è

superfluo, in quanto l'obiettivo non è individuare la distanza del nodo sorgente, ma identificare il movimento e per tale scopo è sufficiente il valore di RSSI. In accordo con quanto descritto nella sezione 7.3.1, sono stati individuati dei valori di differenza rispetto ad un RSSI di riferimento. Tale valore di riferimento è calcolato come la media aritmetica tra le due misurazioni di RSSI, e, a seconda che questo sia più o meno grande, vengono utilizzate diverse soglie di differenza. Quando questa operazione viene eseguita, in contemporanea è impostata a TRUE la variabile booleana `DETECTED` (inizializzata a FALSE) relativa a quel particolare nodo che permette, in una fase successiva, di calcolare il numero di nodi vicini coinvolti nel calcolo della metrica. Nel caso in cui tale differenza sia superiore alla soglia, viene impostata a TRUE una ulteriore variabile booleana `OVERTHRESHOLD` (inizializzata a FALSE) che ne notifica il superamento. Al controllo della metrica sono contati i nodi con `DETECTED=TRUE` (k) e `OVERTHRESHOLD=TRUE` (i) e valutata la condizione $i/k > C_v$ dove $0 < C_v < 1$ è un valore di soglia.

La strategia di metrica appena descritta non necessita di alcuna modifica al protocollo, però impone l'introduzione di un vincolo sulla tabella dei vicini: sono considerati vicini solo quei nodi il cui valore di RSSI è superiore alla soglia `RSSI_THRESHOLD` -80 dBm. Questo è necessario in quanto l'introduzione di elementi con valori inferiori di RSSI renderebbe meno precisa la metrica a causa della scarsa qualità del collegamento.

7.3.2.2 Metrica basata sul riconoscimento di nuovi nodi

L'utilizzo di una strategia che individui nuovi nodi all'interno del raggio di copertura potrebbe portare notevoli benefici all'identificazione dello stato del nodo. Tuttavia la sua introduzione necessita di alcuni accorgimenti. Innanzitutto la definizione di *nuovo nodo*: è evidente che questa deve essere riferita ad una precedente condizione di stabilità in cui si ha una conoscenza precisa dell'identità di tutti i nodi vicini. Questo implica un aumento delle dimensioni della tabella dei vicini che permetta di memorizzare tutti i nodi all'interno del raggio di copertura. Prima di rendere valida la metrica è quindi necessario aspettare il riempimento della tabella dei vicini, in particolare durante le prime fasi di esecuzione di RPL in cui viene creato il DODAG. Un modo semplice per riconoscere tale situazione è valutare il numero di DIO trasmessi dal nodo e rendere valido il controllo della metrica solo dopo il superamento di un certo numero di invii dato dal parametro `NUM_DIO_TRANS`. Superato tale istante, per ogni nuovo pacchetto, applicazione o di controllo, ricevuto viene vericato se la sorgente è presente nella tabella dei vicini. In caso negativo, l'indirizzo è memorizzato in una tabella predisposta a contenere tutti i nuovi vicini, e viene incrementato un contatore `newNeighbor`. La metrica di mobilità si basa sulla condizione `newNeighbor` $> C_n$ dove $C_n > 1$ è una soglia che identifica il numero di nuovi vicini necessari per considerare

mobile il nodo. Il suo valore deve necessariamente essere maggiore di uno per evitare falsi riconoscimenti di mobilità da parte di nodi stabili nel cui raggio di copertura appaia il nodo effettivamente mobile.

Quando un nodo si riconosce mobile vengono eliminati tutti i valori della tabella dei vicini e dei genitori. Come si vedrà nella sezione 7.3.3 il mantenimento della connettività del nodo mobile richiede un aumento del traffico di DIO al fine di identificare nuovi vicini e genitori. Questa situazione è analoga alla fase di inizializzazione del protocollo RPL e quindi non è possibile basarsi sul riconoscimento di nuovi nodi in quanto non ci si trova in una situazione stabile. Per tale motivo, ogni volta che la variabile `MOBILITY_STATE` è uguale a TRUE, la metrica viene resa inattiva per essere successivamente utilizzata una volta che questa è nuovamente impostata a FALSE.

7.3.2.3 Integrazione delle due metriche

La metrica basata sull'identificazione dei nuovi vicini costituisce un supporto rispetto a quella basata sulla differenza di RSSI, in quanto non può essere sempre utilizzata. Infatti, nel caso in cui il nodo si nelle prime fasi di inizializzazione del protocollo, oppure prosegua il suo movimento per un intervallo di tempo superiore a `MOBILITY_TIMER`, ci si deve affidare solamente sulla differenza di RSSI per riconoscere lo stato di mobilità. D'altra parte però il valore di potenza trasmessa può essere falsato da numerosi elementi di disturbo e rendere quindi imprecisa la sua valutazione. Per questo motivo si è deciso di valutare prima la condizione di nuovi vicini quando possibile, per poi passare, in caso negativo, alla valutazione dell'altra metrica. La sezione 7.4 confermerà questa scelta con dei test pratici sul testbed.

7.3.3 Connettività del nodo mobile

Dall'analisi svolta nella sezione 7.2.2 è emerso come il nodo mobile possa inviare i propri dati verso il sink solo dopo aver identificato un genitore preferito. Poichè la condizione di mobilità comporta la cancellazione degli elementi delle tabelle dei genitori e dei vicini è necessario ricevere pacchetti DIO dai nodi per riempirle nuovamente e selezionare un genitore preferito. Tuttavia a causa dell'algoritmo Trickle Timer, in fasi avanzate di esecuzione di RPL il traffico di controllo è ridotto al minimo rendendo non soddisfacente l'attesa, potenzialmente molto lunga, di ricezione di un DIO. Risulta quindi necessario stimolare l'invio di DIO dai nodi nel raggio di copertura del nodo mobile. Una prima idea è quella di inviare un DIS in broadcast, causando il reset del Trickle Timer e il conseguente invio di DIO. Tale scelta però non sembra ideale, in quanto provocherebbe un significativo aumento del traffico di controllo con il solo scopo di far agganciare il nodo mobile al DODAG. Esso infatti richiede solamente il riconoscimento di un nodo a cui poter inviare

i propri dati. Si è quindi optato per l'invio di un DIS leggermente modificato che inneschi in chi lo riceve l'invio di un solo messaggio DIO senza il reset del Trickle Timer. Il pacchetto DIS contiene alcuni campi riservati adatti a questo scopo, in particolare il campo `reserved`, inizializzato a zero, presente nella struttura definita nella sezione 5.4. Il nodo mobile invia in broadcast un messaggio con tale campo impostato a 1 e alla sua ricezione viene inviato un singolo messaggio DIO senza modificare la normale evoluzione del Trickle Timer. Questa modifica non comporta alcuna aggiunta di nuovi campi al messaggio rispettando i vincoli definiti in precedenza.

7.3.4 Stabilità del DODAG

La stabilità del DODAG richiede innanzitutto che gli evenutali figli del nodo mobile ricerchino un percorso alternativo verso il sink. Come già ricordato, un modo semplice per raggiungere questo scopo è l'invio di DIO con rank infinito non appena un nodo si riconosce come mobile. Oltre a questo, si vuole evitare che i nodi stabili scelgano come genitore preferito il nodo mobile, a causa della sua evidentente instabilità. Quindi per tutto il tempo in cui `MOBILITY_STATE` assume il valore TRUE è inibita la trasmissione di pacchetti DIO che riprende solo dopo il riconoscimento di una condizione stabile tramite il reset del Trickle Timer. Un nodo in uno stato di mobilità è quindi sempre una foglia del DODAG.

7.4 Analisi

L'adattamento di RPL descritto rispetta i vincoli della sezione 7.1 non introducendo nuovi messaggi di controllo o campi in quelli standard. Inoltre il suo utilizzo in una situazione di completa stabilità non modifica il comportamento usuale del protocollo poichè la periodica valutazione della metrica di mobilità restituirà sempre un risultato negativo. È evidente tuttavia che l'aspetto più importante riguarda proprio l'affidabilità di tale metrica, i cui paramentri critici sono riassunti di seguito:

- Paramentri generali:
  - `MOBILITY_TIMER`: intervallo di valutazione della metrica.
- Metrica basata sulla differenza di RSSI:
  - Soglie di differenza tra due valori di RSSI.
  - C_v: soglia del rapporto tra i nodi coinvolti nel calcolo e quelli con valore oltre la soglia
- Metrica basata sul riconoscimento dei nuovi nodi:
  - `NUM_NEIGH`: cardinalità della tabella dei vicini.

– `NUM_DIO_TRANS`: numero di DIO tramessi prima di rendere valida la metrica

– C_n: soglia relativa al numero di nuovi nodi rilevati.

La definizione dei valori dei parametri dipende notevolmente dalla situazione in cui la rete di sensori si trova ad operare. La maggior parte dei test hanno quindi avuto lo scopo di individuare tali valori che si basano su misurazioni empiriche. Lo scenario utilizzato per i test è raffigurato in figura 7.2 e consiste in un insieme di 24 nodi fissi del testbed disposti lungo un corridoio. Tutti i nodi, compreso quello mobile sono programmati con la versione mobile di RPL appena descritta e i valori del Trickle Timer sono stati impostati a `Imin=10` e `K=5`. Inoltre i nodi si scambiano pacchetti applicazione ogni 10 secondi. Il movimento del nodo mobile è lungo il corridoio.

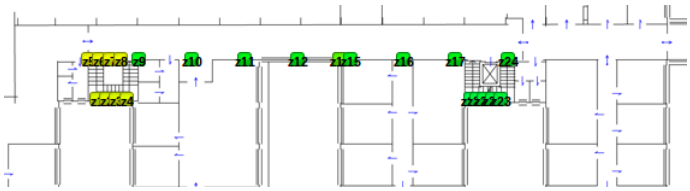


Fig. 7.2: Scenario di prova di RPL in uno scenario mobile

7.4.1 Definizione dei parametri

Il valore del parametro `MOBILITY_TIMER` è molto importante: se troppo elevato si rischia di non identificare la condizione di mobilità fino a quando non viene rilevata l'impossibilità di comunicare con il genitore preferito, con conseguente perdita di pacchetti di informazione. D'altra parte un breve intervallo non consente di raccogliere una quantità di informazioni sufficienti a rendere affidabile la valutazione della metrica. Per individuare un valore di compromesso tra queste situazioni è necessario avere un'idea a grandi linee del traffico generato dalla rete: più questo è elevato più l'intervallo di tempo può essere ridotto. Nello scenario in esame, un buon valore per `MOBILITY_TIMER` è 10 secondi.

Le soglie di RSSI costituiscono un ulteriore parametro critico del protocollo in quanto specificano, insieme a C_v, la sensibilità della metrica. In una prima fase si è utilizzata un'unica soglia fissa indipendente dal valore di RSSI di riferimento. Dati empirici hanno mostrato che una buon valore di differenza è tra i 15 e i 10 dBm (valore assoluto). Successivamente si sono introdotte soglie differenti a seconda dell'RSSI di riferimento. Anche in questo caso valutazioni empiriche hanno portato all'individuazione delle seguenti soglie, dove il *average RSSI* è il valore medio di RSSI misurato nelle

due ricezioni successive e *differential threshold* è la soglia, in valore assoluto, legata alla differenza tra le due misurazioni:

Average RSSI	Differential Threshold
> -65 dBm	14 dBm
> -75 && ≤ -65 dBm	11 dBm
≤ -75 dBm	8 dBm

Ulteriori test hanno dimostrato che l'utilizzo di più soglie rende la metrica più sensibile portando a riconoscere la mobilità anche in presenza di spostamenti di uno o due metri, spesso ininfluenti sulla connettività del nodo nel DODAG. Nel caso di un unico valore invece lo spostamento richiesto per riconoscere la mobilità è notevolmente superiore, oltre i cinque metri. Queste osservazioni mettono in evidenza l'importanza di specificare valori precisi di soglia, in particolare quando se ne vogliono usare più di una.

Il parametro C_v è legato alla densità della rete: un alto numero di vicini permette di diminuirne il valore. Nello scenario in esame i nodi hanno tra i 5 e i 10 neighbor, e impostando $C_v = 0.4$ si richiede un numero di nodi che superino la soglia di RSSI variabile tra 2 e 4. Inoltre, poichè la rete ha un livello di traffico relativamente alto, la valutazione della metrica è basata su trasmissioni che coinvolgono sempre la maggior parte se non tutti i vicini e quindi il valore di 0.4 risulta adeguato.

Il numero dei vicini, come già ricordato, è in media 8 per nodo. Tuttavia per avere la sicurezza di memorizzarli tutti si è utilizzata una tabella di capacità 15. Non si tratta di valori molto elevati, e quindi non incidono in maniera significativa sulle limitate risorse del nodo. È evidente tuttavia che valutazioni diverse sono necessarie in casi di densità maggiore, fino alla possibile eliminazione di una metrica basata sul numero di nuovi vicini.

L'intervallo di non utilizzo delle metrica nelle fasi iniziali del protocollo RPL può essere stimato sui 2 o 3 DIO trasmessi, che, per `Imin=10`, corrisponde circa a 8 o 16 secondi.

Infine il parametro C_n dipende anch'esso dalla densità e dal traffico di pacchetti nella rete. Viste le caratteristiche dello scenario in esame si è imposta la condizione $C_n \geq 4$ per considerare il nodo mobile.

7.4.2 Risultati dei test

Il movimento del nodo mobile nello scenario di figura 7.2 è rappresentato in figura 7.3 dove il sink è il nodo **z10**. In una prima fase di inizializzazione il nodo rimane fermo nel punto **A** per 40 secondi, terminati i quali esso si

muove verso il punto **B** con un intervallo di 20 secondi. Nel punto **B** il nodo rimane fermo per altri 40 secondi per poi spostarsi verso il punto **C**. In questo scenario sono stati effettuati varie prove, apportando alcune modifiche per verificare il comportamento del protocollo in situazioni particolari.

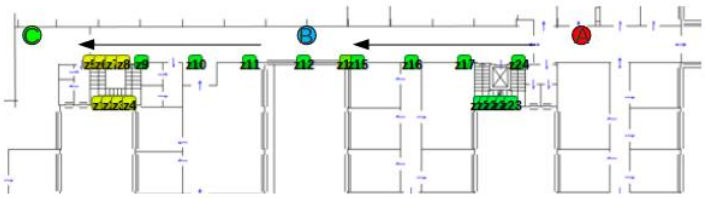


Fig. 7.3: Scenario di prova di RPL in uno scenario mobile

1	2	3	4	5	6
STOP A					
S(z24)	**S**(z17)	**S**(z24)	**S**(z17)	**S**(z17)	**S**(z17)
S(z24)	**S**(z17)	**S**(z24)	**S**(z17)	**S**(z17)	**S**(z17)
S(z24)	**S**(z17)	**S**(z24)	**S**(z17)	**S**(z17)	**S**(z17)
S(z24)	**M**(T)(z17)	**S**(z24)	**S**(z17)	**S**(z17)	**S**(z17)
MOVIMENTO A ——> B					
M(8N/T)(z10)	**S**(z17)	**M**(4N/T)(z10)	**S**(z17)	**M**(5N/T)(z17)	**M**(4N)(z10)
S(z10)	**M**(9N)(z10)	**M**(4N/T)(z10)	**M**(6N)(z11)	**M**(T)(z10)	**M**(4N/T)(z10)
STOP B					
S(z10)	**S**(z10)	**S**(z17)	**S**(z11)	**M**(T)(z10)	**S**(z10)
S(z10)	**S**(z10)	**S**(z17)	**S**(z11)	**S**(z10)	**S**(z10)
S(z10)	**S**(z10)	**S**(z17)	**S**(z11)	**S**(z10)	**M**(T)(z15)
S(z10)	**S**(z10)	**S**(z17)	**S**(z11)	**S**(z10)	**S**(z15)
MOVIMENTO B ——> C					
S(z10)	**M**(T)(z10)	**S**(z10)	**S**(z11)	**S**(z10)	**S**(z15)
M(5N/T)(z10)	**M**(T)(z14)	**M**(4N/T)(z10)	**M**(6N/T)(z10)	**M**(5N)(z10)	**M**(4N/T)(z5)
STOP C					
S(z10)	**M**(T)(z10)	**S**(z10)	**S**(z10)	**S**(z10)	**S**(z5)
S(z10)	**S**(z6)	**S**(z10)	**S**(z10)	**S**(z10)	**S**(z5)
S(z10)	**S**(z6)	**S**(z10)	**S**(z10)	**S**(z10)	**S**(z5)
S(z10)	**S**(z6)	**S**(z10)	**S**(z10)	**S**(z10)	**S**(z5)

Tabella 7.1: Risultati mobilità per lo scenario di figura 7.3

La tabella 7.1 rappresenta schematicamente i risultati di un campione di sei prove nello scenario di figura 7.3 ottenuti dal controllo periodico (ogni 10 secondi) della metrica di mobilità. La lettera **S** indica che la metrica ha rilevato la stabilità del nodo e il valore tra parentesi specifica il genitore preferito relativo a quella situazione. La stringa **M**(nN/T)(nodo) vuole significare il rilevamento di una condizione di mobilità dovuta all'individuatione di n nuovi vicini (N) e/o variazione di RSSI (T). L'ultimo valore tra parentesi indica il nodo scelto come genitore preferito tra quelli che hanno risposto al DIS mobile.

La prima osservazione, non rappresentata in tabella, riguarda il comportamento dei nodi statici: in nessuna prova qualcuno tra di essi si è riconosciuto

come mobile, evidenziando quindi l'ininfluenza delle modifiche in condizioni statiche.

La condizione di stabilità nei tre punti è correttamente rilevata in tutte le prove eccetto un caso occorso nella seconda e nella sesta prova, in cui, rispettivamente, il nodo si è riconosciuto falsamente mobile durante la sosta nel punto **A** e nel punto **B**. L'errore è dovuto sempre alla valutazione della metrica basata sulle soglie che risulta quindi sensibile a interferenze ambientali. La condizione di mobilità è rilevata correttamente in tutte le prove tranne nella prima durante lo spostamento da **A** a **B**. Si noti infatti come stabilità o mobilità durante le fasi di transizione non sono da considerare valutazioni sbagliate poichè esse dipendono dalla raccolta di dati in un intervallo di 10 secondi e quindi, se per esempio lo stop avviene a ridosso del termine del `MOBILITY_TIMER` è probabile che la metrica valuti mobilità avendo raccolto la maggior parte dei dati in una condizione di movimento. Invece non è corretto il caso prima citato nella prima prova, in cui è rilevata stabilità in una evidente situazione di mobilità. Poichè la valutazione precedente ha avuto come risultato la condizione di movimento, l'errore è dovuto anche in questo caso ad un'errato responso del controllo sulla variazione di RSSI evidenziando ancora una volta come sia difficile trovare precise soglie che garantiscano un risultato sempre affidabile. La metrica basata sui vicini invece si dimostra più affidabile dando esito positivo nelle situazioni corrette, anche se non in tutti i casi di movimento. Il campione di prove presentato in tabella mostra tuttavia come la combinazione delle due tecniche porta a risultati più che buoni a patto di accettare alcuni falsi riconoscimenti dovuti a tutta quella serie di fattori esterni e ambientali discussi precedentemente.

Un dato importante da evidenziare riguarda la ricezione da parte del sink di tutti i pacchetti inviati dal nodo mobile dimostrando quindi che le strategia di mantenimento della connettività del nodo mobile sono valide. Infine ulteriori considerazioni possono essere fatte sui cammini dal sink al nodo mobile, imponendo a quest'ultimo di inviare un DAO non appena passa da una condizione di mobilità a quella di stabilità. L'aggiornamento dei cammini avviene in maniera corretta ed è possibile un'immediata comunicazione tra il sink e il nodo. Si potrebbe imporre l'invio di un DAO anche in condizioni di mobilità, tuttavia, considerando la relativa brevità del periodo di controllo della metrica, si può supporre che, se il movimento non è eccessivo, il nuovo preferred parent selezionato sia il medesimo anche nella successiva condizione di stabilità, mentre nel caso di movimenti prolungati l'invio di DAO a tutti i genitori preferiti selezionati non ha alcun vantaggio visto il loro continuo cambiamento.

Un'ultima considerazione riguarda lo scenario in cui la rete non abbia un elevato traffico. Si è creata tale situazione impedendo ai nodi di trasmettere pacchetti applicazione, ma lasciando operare solo il protocollo RPL. Il movimento del nodo è stato fatto lungo lo stesso percorso per un tempo di 5 minuti. I risultati hanno evidenziato che la condizione di mobilità è stata riconosciuta solo nel primo spostamento da **A** a **B** e da **B** a **C**, sempre tramite l'individuazione di nuovi vicini. Nei successivi spostamenti la valutazione della metrica ha sempre evidenziato stabilità. Ogni qual volta il nodo perdeva il contatto con il rispettivo genitore, visto che il numero di pacchetti ricevuti era inferiore alla soglia C_p (impostato in questo caso a 4), non veniva attivato il processo di riparazione locale, ma la ricerca di un nuovo genitore tramite l'invio di un DIS con `reserved=1`.

7.5 Conclusioni e Maggiore livello di mobilità

Lo sviluppo di un adattamento di RPL ad uno scenario mobile ha messo in evidenza una serie di limiti e problematiche. L'aspetto più controverso riguarda la difficoltà di individuare una metrica di mobilità con un livello di affidabilità sufficientemente elevato che tuttavia la differenza di potenza ricevuta tra due trasmissioni non consente sempre di raggiungere. All'interno del laboratorio SIGNET è allo studio un progetto di localizzazione di un nodo nel dipartimento tramite il confronto dei valori di RSSI calcolati rispetto ai nodi fissi del testbed. L'operazione di localizzazione avviene tramite la consultazione di un database contenente misure campione. Il valore di RSSI del nodo da localizzare rispetto a quelli fissi è determinato tramite una serie di misurazioni svolte su diversi canali di comunicazioni in modo da ottenere un valore il più preciso possibile. Da questo esempio risulta evidente come solo due misurazioni non siano assolutamente sufficienti per stabilire con precisione la distanza tra due nodi. D'altra parte però effettuare più misurazioni vorrebbe dire affidarsi ad un'applicazione dedicata e richiederebbe un tempo di calcolo notevolmente superiore che poco si adatta alle esigenze di reattività e mantenimento della connettività del nodo mobile. A questo si aggiunge il problema di una precisa individuazione delle soglie che possono essere valide per un determinato scenario, ma non adattarsi ad un altro, imponendo quindi ulteriori test per il calcolo dei nuovi valori. A queste considerazioni di carattere pratico, si aggiungono quelle relative alle prestazioni del protocollo così come è stato descritto. Infatti, condurre test in uno scenario reale comporta l'impossibilità di effettuare un numero consistente di prove nelle situazioni più diverse al fine di verificare la correttezza e le prestazioni del protocollo, cosa notevolmente più semplice da ottenere in uno scenario simulato. Un possibile sviluppo del presente lavoro di tesi consiste appunto nell'implementazione del protocollo in un simulatore. Sarebbe

così possibile effettuare una serie di test al fine di capire quali siano i migliori valori per quei paramentri dipendenti dalla densità e dal traffico della rete.

Infine è interessante interrogarsi sull'estensione di tale tecnica alla presenza di un numero arbitrario di nodi mobili. In uno scenario simile la complessità del problema cresce notevolmente, in particolare in riferimento alla ricerca di una metrica che riesca a distinguere tra gruppi di nodi mobili e nodi stabili. Si pensi per esempio al caso di un nodo fisso verso cui si muovono diversi nodi mobili. La rilevazione di nuovi nodi porterebbe inevitabilmente al riconoscimento di una falsa condizione di mobilità. Ma anche la metrica basata sulla differenza di RSSI potrebbe portare all'individuazini di false mobilità, per esempio nel caso in cui un gruppo di nodi si sposti rispetto ad un nodo fisso. Risolvere questo tipo di problematiche mantenendo tutti i vincoli presentati nella sezione 7.1 è molto complesso e forse senza soluzione. Un'indagine in questo senso richiederebbe la presenza di un ambiente simulato in cui poter effettuare svariate prove al fine di trovare la strategia migliore.

CAPITOLO 8

Conclusioni e sviluppi futuri

I protocolli di routing sono di fondamentale importanza per qualsiasi tipologia di rete di comunicazioni e nel caso specifico delle reti di sensori questi richiedono un'attenzione particolare al fine di rispettare tutti i vincoli che esse impongono. In questa tesi è stato sviluppato il protocollo RPL descritto nel *draft* rilasciato dalla comunità IETF poichè esso si propone come standard per le reti di sensori. Il suo sviluppo è stato mirato non ad uno studio teorico di funzionamento, ma ad un utilizzo pratico nello scenario del testbed del DEI consentendo quindi di osservarne le prestazioni effettive, rispetto a tutte le ricerche svolte in ambiente simulato. Inoltre è stato possibile mettere in luce le problematiche relative ad aspetti critici come le metriche di qualità del link basata su RSSI, la selezione dei vicini e dei genitori preferiti motivando le scelte dei diversi paramentri del protocollo al fine di ottenere i migliori risultati possibili di funzionamento nel testbed. Poichè lo sviluppo di RPL è ancora in corso, i risultati presentati nella tesi possono contribuire ad avere un riscontro sulla sua utilizzabilità in uno scenario reale. Il corretto e buon funzionamento dell'implementazione presentata è confermata dal suo utilizzo in applicazioni sviluppate dal gruppo di ricerca SIGNET. Tuttavia essa si presenta come una versione base del protocollo e il draft presenta numerose altre funzionalità alcune delle quali piuttosto rilevanti come la possibilità di utilizzare la modalità non-storing per costruire i cammini, fondamentale nel caso in cui la memoria occupata nel sensore sia un aspetto critico dell'applicazione. Per tale motivo il lavoro di implementazione dovrebbe proseguire includendo queste ulteriori funzionalità per giungere infine ad una versione completa di RPL. Per quanto riguarda l'analisi delle prestazioni, un aspetto non considerato è il consumo energetico sul nodo da parte del protocollo. Questo tipo di analisi

non è indispensabile in un ambiente come il testbed dove i nodi sono alimentati tramite il collegamento USB, tuttavia per un utilizzo in ambienti dove i nodi sono alimentati a batteria diventa estremamente importante. Al consumo energetico è collegata anche l'integrazione di metriche sul nodo che considerino il livello di energia residuo in un nodo come parametro di scelta dei vicini e dei genitori. Tutti questi aspetti costituiscono un proseguo del presente lavoro di tesi indirizzato ad avere non solo una versione base funzionante di RPL, ma anche completa e il più possibile generale.

La tesi ha poi presentato una integrazione di RPL con una strategia che consenta la gestione dei guasti e la perdita di connettività. La soluzione descritta mira a ridurre al minimo il traffico di controllo necessario alla ricostruzione del DODAG e il numero di pacchetti applicazione persi. Il *draft* lascia all'implementazione la scelta di come effettuare le operazioni di local repair e la tesi contribuisce a chiarire ed evidenziare aspetti e problematiche di cui è necessario tenere conto fornendo per ciascuno di essi una possibile soluzione. Anche in questo caso però alcune funzionalità importanti non sono state implementate, principalmente quelle riguardo il controllo della presenza di cicli all'interno del DODAG. Inoltre la soluzione è stata verificata in scenari relativamente ristretti e quindi una completa validazione della stessa richiederebbe una più estesa campagna di test. A tale scopo risulterebbe molto utile l'aiuto di un simulatore, in grado di mettere in luce aspetti prestazionali ed comportamentali della soluzione in diverse situazioni.

L'aspetto più originale della tesi riguarda la valutazione di RPL in presenza di nodi mobili. L'obiettivo è sviluppare una strategia che non modifichi, se non in minima parte, il comportamento del protocollo in modo da avere una soluzione generale adatta sia in condizioni di stabilità che di mobilità. Seguendo tale filosofia sono stati messi in evidenza numerosi limiti, in particolare sulla metrica di mobilità e sull'effettiva possibilità di estenderla ad una situazione generale con più di un nodo mobile. Per valutare meglio questi aspetti sarebbe necessario sfruttare anche in questo caso un ambiente simulato principalmente perchè i test di mobilità risultano estremamente complessi nello scenario del testbed e non adatti alla ricerca e allo sviluppo di una versione mobile di RPL. Tuttavia la tesi è un primo inizio di ricerca in questa direzione e potrebbe costituire un punto di riferimento per ulteriori studi oltre a presentare una soluzione funzionante nel caso sia sufficiente la presenza di un solo nodo mobile.

Ringraziamenti

Il raggiungimento di un obiettivo così importante non sarebbe stato possibile senza i miei genitori Teresa e Mario che, oltre ad offrirmi l'opportunità, mi hanno supportato nei momenti difficili e condiviso le gioie e le soddisfazioni dei risultati ottenuti. Per questo e altri motivi meritano il più grande dei ringraziamenti.

Questi cinque anni non sono stati solo di studio ma anche e soprattutto di condivisione e amicizia, vorrei quindi ringraziare innanzitutto i miei amici trentini, in particolare Joshua, Simone e Luca. Il percorso di studi a Padova mi ha poi permesso di conoscere molte altre persone sia fuori che dentro l'ambiente universitario e tra i primi non posso che ringraziare Michele, Alessandro e Vincenzo. Tra i compagni di studio e di fatica un ringraziamento particolare va a Dario, Federica e Nicholas. Infine un grazie generale a tutte quelle altre persone che hanno fatto parte della mia vita in questo lungo percorso verso la laurea.

Per l'opportunità concessami di svolgere questo lavoro di tesi ringrazio il prof. Michele Zorzi ma soprattutto Nicola Bressan e Nicola Bui per il costante supporto dimostrato in questi mesi. Un grazie inoltre a tutti i ragazzi del laboratorio SIGNET per la calorosa accoglienza riservatami.

Bibliografia

[1] J. Yick, B. Mukherjee, and D. Ghosal, *Wireless Sensor Network Survey*, Elsevier Computer Networks, 52(12): 2293-2330, 2008.

[2] V. Raghunatham, A. Kansai, J. Hse, J. Friedman, M. Srivastava, *Design considerations for solar energy harvesting wireless embedded systems*, Proceedings of the ISPN, 2005, pp. 457-462

[3] S.Roundy, J.M Rabaey, P.K Wright, *Energy scavenging for wireless sensor network*, Springer-Verlag, New York, LLC, 2004

[4] S. Toumpis, T. Tassiulas, *Optimal deployment of large wireless sensor networks*, IEEE Transaction on Information Theory 52 (2006), pp. 2935-2953

[5] I.F. Akyildiz, E.P Stuntebeck, *Wireless underground sensor networks: research challenge*, Ad-Hoc Networks 4 (2006), pp. 669-686

[6] J.Heidemann, Y. Li, A.Syed, J. Wills, W. Ye, *Underwater sensor networking: research challenges and potential application*, Proceedings of Technical Report ISI-TR-2005-603, USC/Information Sciences Institute, 2005

[7] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, *A survey on wireless multimedia sensor networks*, Computer Networks Elsevier 51 (2007) 921-960

[8] Th. Arampatzis, J. Lygeros, S. Manesis, *A survey of application of wireless sensor and wireless sensor networks*, Proceeding of the 13th Mediterranean Conference on Control and Automation at Limassol, Cyprus, 2005

[9] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, *An energy-efficient surveillance system using wireless sensor networks*, MobiSys'04, Boston, MA, 2004

[10] I. Howitt, J.A. Gutierrez, *IEEE802.15.4 low rate-wireless personal area network coexistence issues*, Wireless Communications and Networking 3 (2003) 1481–1486.

[11] ZibBee Alliance `http://www.zigbee.org/Home.aspx`

[12] G. Mulligan, L.W. Group, *The 6LoWPAN architecture*, Proceedings of the EmNets, Cork, Ireland, 2007.

[13] 6LowPAN Working Group `http://datatracker.ietf.org/wg/6lowpan/charter/`

[14] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, K. Frampton, *Sensor network-based countersniper system*, Proceedings of the Second International Conference on Embedded Networked Sensor Systems (Sensys), Baltimore, MD, 2004

[15] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, J. Lees, *Deploying a wireless sensor networks on an active volcano*, IEEE Internet Computing 10 (2006) 18-15

[16] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, *Wireless sensor networks for habitat monitoring*, ACM inter. Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, GA, 2002

[17] C.R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A.D. Minassians, G. Dervisoglu, L. Gutnik, M.B. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E.S. Leland, T. Pering, P.K. Wright, *Wireless sensor networks for home health care*, AINAW, Ontario, Canada, 2007.

[18] A. M. Tabar, A. Keshavarz, H Aghajan, *Smart home care network using sensor fusion and distributed vision-based reasoning*, In VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks, pages 145–154, New York, NY, USA, 2006. ACM Press.

[19] `http://cariparo.dei.unipd.it/`

[20] `http://www.willow.co.uk/TelosB_Datasheet.pdf`

[21] `http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf`

[22] S. Bhatti, J. Carlson, Hui Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R Han, *Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms* Mobile Network Application 10(4), pp 563–579, 2005.

[23] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. *TinyOS: An operating system for wireless sensor networks*, Ambient Intelligence by W. Weber, J. Rabaey, and E. Aarts. 2005. Springer; 1 edition (April 19, 2005).

[24] D. Gay at al., *The NesC language: a holistic approach to networked embedded systems*, in Proc. of ACM PLDI, San Diego, CA, 2003

[25] A. Castellani, P. Casari, M. Zorzi, *TinyNET: A Tiny Network framework for TinyOS*, in Proc. of the 2009 International Conference on Wireless Communications and Mobile Computing, 2009

[26] J.N: Al-Karaki, A.E. Kamal, *Routing Techniques in wireless sensor networks: a survey*, IEEE Wireless Comunication vol.11, no.6, pp.6-28, 2004

[27] W. Heinzelman, J. Kulik, H. Balakrishnan, *Adaptive protocols for information dissemination in wireless sensor networks*, Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Net-working (MobiCom99), Seattle, WA, August 1999

[28] C. Intanagonwiwat, R. Govindan, D. Estrin, *Directed diffusion: a scalable and robust communication paradigm for sensor networks*, Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom00), Boston, MA, August 2000.

[29] C. Schurgers, M.B. Srivastava, *Energy efficient routing in wireless sensor networks*, on Communications for Network-Centric Operations: Creating the Information Force, McLean, VA, 2001.

[30] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, *Energy-efficient communication protocol for wireless sensor networks*, Proceeding of the Hawaii International Conference System Sciences, Hawaii, January 2000.

[31] S. Lindsey, C.S. Raghavendra, *PEGASIS: power efficient gathering in sensor information systems*, Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, March 2002.

[32] Y. Yu, D. Estrin, R. Govindan, *Geographical and energy-aware routing: a recursive data dissemination protocol for wireless sensor networks*,

UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, May 2001.

[33] IETF Routing Over Low-power and Lossy networks `https://datatracker.ietf.org/wg/roll/charter/`

[34] P.Levis, A.Tavakoli, S. Dawson-Haggerty *Overview of existing routing protocols for low power and lossy networks*, IETF, Internet-Draft draft-ietf-roll-protocols-survey-07, 2009

[35] T. Winter, P. Thuber, *RPL: Ipv6 Routing Protocol for Low power and lossy network*, IETF, Internet-Draft, draft-ietf-roll-rpl-10, 2011

[36] T. Winter, P. Thuber, *RPL: Ipv6 Routing Protocol for Low power and lossy network*, IETF, Internet-Draft, draft-ietf-roll-rpl-18, 2011

[37] J. Moy, *OSPF Version 2*, STD 54, RFC 2328, April 1998.

[38] R. Coltun, D. Ferguson, J. Moy, *OSPF for IPv6*, RFC 2740, December 1999.

[39] T. Clausen, P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, RFC 3626, October 2003.

[40] T. Clausen, C. Dearlove, P. Jacquet, *The Optimized Link State Routing Protocol version 2*, draft-ietf-manet-olsrv2-07, July 2008.

[41] G. Malkin, *RIP Version 2*, STD 56, RFC 2453, November 1998.

[42] Perkins, C., Belding-Royer, E., and S. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, RFC 3561, July 2003

[43] J. Vasseur, M. Kim, K. Pister, N. Dejean, D. Barthel, *Routing Metrics used for Path Calculation in Low Power and Lossy Networks*, draft-ietf-roll-routing-metrics-17 (work in progress), January 2011.

[44] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, *The Trickle Algorithm*, draft-ietf-roll-trickle-08 (work in progress), January 2011.

[45] ContikiRPL `http://www.sics.se/contiki/tutorials/contikirpl-the-new-default-contiki-ipv6/6lowpan-routing-protocol.html`

[46] `http://code.google.com/p/tinyos-main/source/browse/?r=5326#svn%2Ftrunk%2Ftos%2Flib%2Fnet%2Frpl`

[47] K. Srinivasan P. Levis, *Rssi is under appreciated*, In Proceedings of the Third ACM Workshop on Embedded Networked Sensors (EmNets 2006), May 2006.

[48] J. Tripathi, J. C. de Oliveira, J. P.Vasseur, *A performance evaluation study of RPL: Routing Protocol for Low power and Lossy Networks*, Information Sciences and Systems (CISS), March 2010

[49] T. Clausen, U. Herberg, *Multipoint-to-Point and Broadcast in RPL* Network-Based Information Systems (NBiS), Sept. 2010

[50] N. Bressan, L. Bazzaco, N. Bui, P. Casari, L. Vangelista, M. Zorzi, *The Deployment of a Smart Monitoring System using Wireless Sensors and Actuators Networks*, 2010 First IEEE International Conference on Smart Grid Communication (SmartGridComm), October 2010.

[51] W. Xie, M. Goyal, H. Hosseini, J. Martocci, Y. Bashir, E. Baccelli, A. Durresi, *Routing Loops in DAG-Based Low Power and Lossy Networks*, Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference

[52] L. B. Saad, B. Tourancheau, *Sinks Mobility Strategy in IPv6-based WSNs for Network Lifetime Improvement*, International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2011

[53] M. Zorzi, R.R. Rao, *Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance* IEEE Transactions on Mobile Computing, vol.2, no.4, pp. 337- 348, Oct.-Dec. 2003

[54] Xianchang Li; Jun Zhang; Xuan Shi, *CGR: Contention-Based Geographic Routing Protocol for Mobile Ad Hoc Networks* 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08, 12-14 Oct. 2008

[55] T. Melodia, D. Pompili, I.F. Akyildiz, *Optimal local topology knowledge for energy efficient geographical routing in sensor networks* INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies , vol.3, no., pp.1705-1716 vol.3, 7-11 March 2004

[56] A. Boukerche, H.A.B. Oliveira, E.F. Nakamura, A.A.F. Loureiro, *A Novel Location-Free Greedy Forward Algorithm for Wireless Sensor Networks* IEEE International Conference on Communications, ICC '08, 19-23 May 2008

[57] Chun-Hung Chen; Ho-Ting Wu; Kai-Wei Ke; *The design and application of mobile awareness mechanism in wireless ad hoc networks without GPS* The 9th Asia-Pacific Conference on Communications, APCC 2003, pp. 148- 152 Vol.1, 21-24 Sept. 2003

[58] P. Basu, N. Khan, T.D.C. Little, *A mobility based metric for clustering in mobile ad hoc networks* International Conference on Distributed Computing Systems Workshop, 2001, pp.413-418, Apr 2001

[59] Ho-Ting Wu; Kai-Wei Ke; Chun-Hung Chen; Chen-Wei Kuan; *Mobile awareness based cluster selection mechanisms in wireless ad hoc networks* Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th , vol.4, no., pp. 2774- 2778 Vol. 4, 26-29 Sept. 2004

[60] Hyun Yu; Sanghyun Ann; *Node Movement Detection to Overcome False Route Failures in Mobile Ad Hoc Networks* International Conference on Information Science and Security, 2008. ICISS., pp.137-140, 10-12 Jan. 2008

[61] A. Raja, Xiao Su, *A Mobility Adaptive Hybrid Protocol for Wireless Sensor Networks* 5th IEEE Consumer Communications and Networking Conference, CCNC 2008, pp.692-696, 10-12 Jan. 2008

[62] Bo Gu, Xiaoyan Hong, *Mobility identification and clustering in sparse mobile networks* IEEE Military Communications Conference, 2009. MILCOM 2009, pp.1-7, 18-21 Oct. 2009

[63] A. Goldsmith, *Wireless Communications*, Cambridge Univ. Press, 2004